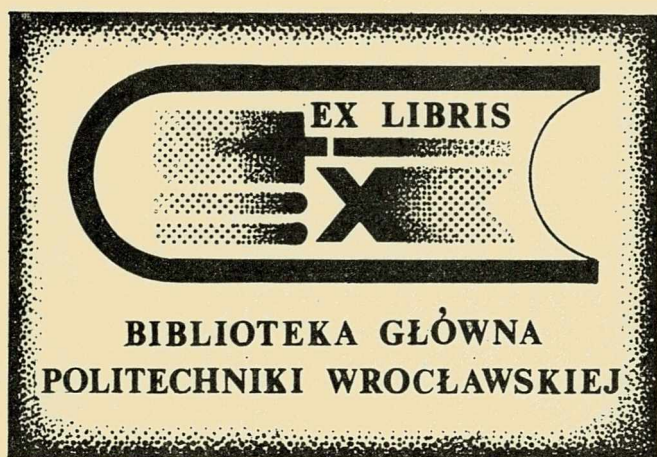


Hanna Mazur

**Typy danych
w programowaniu strukturalnym**

**Zbiór zadań z programowania
w języku Pascal**





EX LIBRIS

BIBLIOTEKA GŁÓWNA
POLITECHNIKI WROCŁAWSKIEJ

Politechnika Wrocławska



Hanna Mazur

Typy danych w programowaniu strukturalnym

Zbiór zadań z programowania
w języku Pascal

Wrocław 1990

Wydawnictwo dydaktyczne do przedmiotów:
Podstawy informatyki, Zaawansowane metody programowania
dla studentów wszystkich wydziałów Politechniki Wrocławskiej

Opiniodawcy:
Helena KRUPICKA
Jerzy KUCHARCZYK

Opracowanie redakcyjne
Teresa JARMAKOWICZ

Korekta
Teresa JARMAKOWICZ



263892 D/2

© Copyright by Wydawnictwo Politechniki Wrocławskiej, Wrocław 1990

WYDAWNICTWO POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

Biblioteka Główna i OIMT
Politechniki Wrocławskiej



000846856

Nakład 3000 + 70 egz. Ark. wyd. 13 1/2. Ark. druk. 15 1/4. Papier offset, kl. III, 70 g, B1.
Oddano do druku w listopadzie 1989 r. Druk ukończono w styczniu 1990 r.
Zakład Graficzny Politechniki Wrocławskiej. Zam. nr 1612/89. Cena zł 2500,-

Alu 58/4/90

SPIS TREŚCI

WSTĘP	5
1. WPROWADZENIE	7
2. ZSTĘPUJĄCA METODA KONSTRUOWANIA PROGRAMÓW	9
3. KOMPILACJA I WYKONYWANIE PROGRAMÓW	13
3.1. Zadania	15
4. STRUKTURA PROGRAMU W JĘZYKU TURBO PASCAL 3.0	16
4.1. Instrukcje	17
4.2. Procedury i funkcje	21
4.3. Typy danych	25
4.4. Proste i strukturalne typy danych	26
4.4.1. Typy standardowe	27
4.4.2. Typy wyliczeniowe i okrojone	30
4.5. Definicje stałych	31
4.6. Deklaracje zmiennych	32
4.7. Zadania	40
5. TYP TABLICOWY	61
5.1. Zadania	68
6. TYP ŁAŃCUCHOWY	104
6.1. Operacje na łańcuchach	105
6.2. Zadania	110
7. TYP REKORDOWY	115
7.1. Instrukcja wiążąca	117
7.2. Rekordy z wariantami	120
7.3. Zadania	126
8. TYP ZBIOROWY	131
8.1. Zadania	138
9. TYP PLIKOWY	141
9.1. Pliki tekstowe	144
9.2. Pliki bez typu elementów	146
9.3. Zadania	147
10. TYP WSKAŹNIKOWY	152
10.1. Zadania	165
11. ZADANIA RÓŻNE	170

12. ROZWIĄZANIA ZADAŃ	188
Do rozdz. 4	188
Do rozdz. 5	198
Do rozdz. 6	209
Do rozdz. 7	211
Do rozdz. 8	215
Do rozdz. 9	218
Do rozdz. 10	219
Do rozdz. 11	220
 DODATEK A. SKŁADNIA JĘZYKA TURBO PASCAL 3.0 (DIAGRAMY SYNTAKTYCZ-	
NE)	225
DODATEK B. ZESTAWIENIE OBIEKTÓW STANDARDOWYCH W JĘZYKU TURBO	
PASCAL 3.0	233
DODATEK C. TABLICA KODU ASCII	238
LITERATURA	239
SKOROWIDZ	240

WSTĘP

Niniejszy skrypt jest przeznaczony zarówno dla osób uczących się programowania w języku Turbo Pascal wersji 3.0 (w skrócie TP 3.0) jak i dla zaawansowanych w programowaniu. Skrypt ten, ze względu na dużą liczbę zamieszczonych w nim zadań, może być również wykorzystywany przez osoby prowadzące zajęcia związane z programowaniem w TP 3.0.

W skrypcie omówiono typy danych występujące w języku Turbo Pascal wersji 3.0. Rozdział 1 zawiera uwagi ogólne na temat metodologii programowania. W rozdziale 2 przedstawiono zstępującą metodę konstruowania programów. Rozdział 3 dotyczy kompilacji i wykonywania programów. W rozdziałach 4-10 scharakteryzowano poszczególne typy oraz podano przykłady ilustrujące ich użycie. Po każdym rozdziale zamieszczono zestaw zadań do samodzielnego opracowania. Z myślą o Czytelniku początkującym umieszczono po kilka zadań tego samego typu. Zakłada się, że Czytelnik zna podstawowe pojęcia z dziedziny programowania (np. wyrażenie arytmetyczne).

Przyjęty układ rozdziałów (opis typu, przykłady, zadania) ma sugerować, że zadania umieszczone na końcu danego rozdziału można rozwiązać za pomocą dotychczas omówionych typów danych, ale wybór odpowiedniego typu pozostawia się Czytelnikowi.

Znaczna część studentów jest zainteresowana dodatkowymi tematami zadań. Jedni z nich pragną zadań bardzo prostych, na których mogliby samodzielnie przećwiczyć podstawowe techniki programowania i nabrać większej wprawy w pisanie programów, inni natomiast poszukują zadań bardziej złożonych. Dlatego też stopień trudności zamieszczonych zadań jest bardzo zróżnicowany. Celowo nie wyróżniono zadań trudniejszych, ażeby nie zniechęcać Czytelników nie wierzących we własne siły, tym bardziej, że pojęcie "zadanie trudne" jest względne. Na końcu skryptu znajdują się rozwiązania niektórych zadań. Programy zamieszczone w tym skrypcie zostały napisane w języku Turbo Pascal wersji 3.0 i uruchomione na komputerze IBM PC/XT/AT. Ze względu na ograniczoną objętość skryptu w niektórych rozwiązaniach zadań pominięto komentarze lub drukowanie tekstów objaśniających a skupiono się na samym algorytmie. Jako ćwiczenie dla Czytelnika pozostawia się dopisanie odpowiednich komentarzy oraz instrukcji wyjścia.

Wszystkie zwykłe znaki przestankowe (kropka, przecinek, średnik i in.) są symbolami języka Pascal. Z tego też powodu w niektórych zdaniach zawierających przykłady w tym języku zasady interpunkcji nie są przestrzegane.

W celu zwiększenia czytelności zamieszczonych w skrypcie przykładów i programów w tekstach napisanych w języku TP 3.0 używa się polskich liter, które w alfabecie TP 3.0 nie występują.

Koniec przykładu jest oznaczany symbolem ■.

Życzymy wszystkim Czytelnikom wytrwałości w pisaniu programów, co często nie jest rzeczą łatwą ani małą. Najlepszy efekt w nauce programowania można osiągnąć uruchamiając napisane programy. Dlatego też w celu sprawdzenia swoich umiejętności radzimy, w miarę możliwości dostępu do sprzętu, uruchamiać opracowane programy.

Pragniemy również zwrócić uwagę Czytelników, że często programy realizujące ten sam algorytm, ale pisane przez różne osoby mogą się różnić w zapisie, w użytych typach danych, liczbie zmiennych, wykorzystanych instrukcjach itd. Z wielu możliwych (i co najważniejsze, poprawnych) należy wybrać ten, który jest najbardziej czytelny, optymalny (np. pod względem czasu obliczeń), wykorzystuje jak najmniej pamięci itp. Zagadnienie to jest omówione w rozdz. 1.

1. WPROWADZENIE

Przystępując do rozwiązania dowolnego zadania należy dokładnie sformułować jego treść, tzn. określić, co jest dane, a co należy obliczyć i podać jako wynik. Następnie należy dokonać wyboru metody rozwiązania danego problemu, czyli podać algorytm.

Do precyzyjnego zapisu algorytmów służą języki programowania. Program napisany w danym języku programowania powinien dawać poprawne wyniki. Na ogół proces tworzenia programu przebiega w trzech etapach:

1. opracowanie algorytmu,
2. wybór odpowiedniej reprezentacji danych związanych z tym algorytmem,
3. napisanie programu w danym języku programowania.

Jedną z metod tworzenia poprawnych programów jest programowanie strukturalne. Opracowując algorytm w programowaniu strukturalnym dzielimy problem na podproblemy a następnie zajmujemy się ich rozwiązaniem. Jeśli podproblemy te są złożone, to znowu dzielimy je na podproblemy itd. Taką metodę konstruowania programów nazywa się metodą zstępującą i jest ona opisana w rozdz. 2.

W praktyce na ogół etapy 1 + 3 łączą się ze sobą. Dwa pierwsze etapy, czyli opracowanie algorytmu i wybór reprezentacji danych, będziemy nazywali szkicem programu. Często właśnie opracowanie szkicu programu sprawia największą trudność, natomiast napisanie programu w odpowiednim języku i dla konkretnego komputera przez osobę znającą ten język i komputer nie przedstawia większej trudności, jest czynnością wręcz mechaniczną.

Pisząc program należy zadbać o to, aby po napisaniu był on:

- a) poprawny,
- b) uniwersalny,
- c) zupełny,
- d) efektywny,
- e) czytelny.

Pokrótkie scharakteryzujemy własności a + e.

ad a) Program jest poprawny, jeśli dla każdego możliwego zbioru wartości danych otrzymuje się poprawny wynik.

ad b) Program jest uniwersalny, jeśli jest niezależny od konkretnego zestawu danych, tzn. łatwo można go zmodyfikować, aby wprowadzić inne dane. W celu osiągnięcia uniwersalności powinno się parametryzować program za pomocą zmiennych, a nie stałych. Na przykład, jeśli chcemy napisać program, który wczytuje 100 liczb rzeczywistych i oblicza ich średnią, to powinniśmy napisać

```
SR:=0.0;
readln(n);
for i:=1 to n do
begin
  read(liczba);
  SR:=SR+liczba
end {i};
SR:=SR/n
```

a nie

```
SR:=0.0;
for i:=1 to 100 do
begin
  read(liczba);
  SR:=SR+liczba
end {i};
SR:=SR/100
```

ad c) Program powinien przewidywać możliwość wprowadzenia złych danych i wyprowadzać odpowiedni komunikat informujący o błędzie. Nie powinien wykonywać dalszych obliczeń na błędnych danych i podawać wyników według zasady "błędne dane - błędne wyniki".

ad d) Obliczenia wykonywane przez program powinny być wykonalne w dostępnej pamięci komputera i być skończone w czasie.

ad e) Program powinien być zrozumiały i czytelny. Niektórzy nazywają tę zasadę zasadą BUZI (Bez Udzivnień Zapisuj Idioto - ang. Keep It Simple, Stupid).

Oczywiście, często spełnienie jednego warunku odbywa się kosztem innego, np. można napisać bardzo zrozumiały program ale nieoptymalny, tzn. wykorzystujący zbyt dużo pamięci i wykonujący wiele zbędnych obliczeń. Modyfikacja takiego programu może doprowadzić do oszczędniejszej gospodarki pamięcią i czasem, ale program straci na przejrzystości. Programista musi sam wybrać te kryteria, na których spełnieniu zależy mu bardziej.

2. ZSTĘPUJĄCA METODA KONSTRUOWANIA PROGRAMÓW

W rozdziale 1 wprowadziliśmy pojęcie metody zstępującej. Posługując się nią próbujemy rozłożyć dany problem na ciąg prostszych podproblemów. Dla danego zadania P próbujemy znaleźć zbiór mniej złożonych zadań $P_1, P_2, \dots, P_n$ taki, że rozwiązując zadanie P_1 , następnie P_2 itd. otrzymamy rozwiązanie wyjściowego zadania P . To podejście jest stosowane również do każdego podproblemu $P_1, P_2, \dots, P_n$, dając w ten sposób zbiór podpodproblemów, których rozwiązaniem jest rozwiązaniem zadania P . Proces ten kontynuujemy tak długo, aż podproblemy będą tak małe, że dadzą się bezpośrednio rozwiązać.

W każdym kroku metody musimy mieć na uwadze dwa warunki:

a) poprawność - czy poprawne rozwiązanie podproblemów $P_1, P_2, \dots, P_n$ da nam poprawne rozwiązanie problemu P ?

b) wykonalność - czy rozwiązanie podproblemów $P_1, P_2, \dots, P_n$ będzie wykonalne w danym języku programowania i na danym komputerze?

Każdy błąd wykryty podczas procesu konstruowania programu powinien być poddany pełnej analizie i po kolei należy sprawdzać:

- jego przyczynę i zasięg,
- niewłaściwe zrozumienie lub niespełnienie założeń doprowadzające do powstania błędu.

Jeśli przyczyna błędu jest nie znana, programista musi cofnąć się do pierwszego bezbłędnego kroku metody zstępującej. Tylko w ten sposób będzie zapewniona poprawność szkicu końcowego.

Umiejętność ustalenia przyczyny błędu jest jedną z cech odróżniających wprawnego programistę od niespecjalisty.

W praktyce na ogół do opracowania nowego algorytmu stosuje się metodę wstępującą. Znane są jeszcze inne metody programowania, np. wstępująca. W metodzie tej z prostych instrukcji tworzy się podprogramy (podrozdz. 4.2), z których otrzymuje się program będący rozwiązaniem danego zadania. Metodę wstępującą wykorzystuje się np. w celu dostosowania znanego algorytmu (programu) do nieco zmienionego zadania.

Aby lepiej zrozumieć ideę programowania zstępującego, prześledźmy następujący przykład.

Przykład 2.1

Danych jest n liczb całkowitych ($n \leq 20$). Napisać program, który czyta te liczby, porządkuje je w ciąg niemalejący i drukuje ten ciąg przed i po uporządkowaniu.

Rozwiązanie

Z zadania wynikają trzy podstawowe operacje do wykonania:

- czytanie ciągu,
- drukowanie ciągu,
- porządkowanie ciągu.

Pierwszy szkic programu ma postać:

```
begin
  czytaj_ciąg;
  drukuj_ciąg;
  porządkuj_ciąg;
  drukuj_ciąg
end.
```

Poprawność i wykonalność tego rozwiązania nie ulega wątpliwości. Musimy się teraz zastanowić za pomocą jakiej struktury reprezentować ciąg liczb całkowitych. Odpowiednią strukturą dla tego zadania będzie tablica jednowymiarowa, n -elementowa ($n \leq 20$), ponieważ do każdego elementu mamy bezpośredni dostęp, dzięki czemu możemy te elementy wczytywać, drukować i sortować. Możemy więc teraz zapisać drugi szkic:

```
program sortuj;
const nmax=20;
type tablica=array[1..nmax] of integer;
var a: tablica;
    i,n: integer; {n-liczba elementów ciągu}
begin
  {czytaj ciąg}
  write('Podaj n: '); readln(n);
  writeln('Podaj elementy ciągu:');
  for i:=1 to n do read(a[i]);

  {drukuj ciąg}
  writeln;
  writeln('Elementy ciągu przed uporządkowaniem:');
  drukuj_ciąg a1,a2,...,an oraz liczbę elementów ciągu a
  porządkuj_ciąg a1,a2,...,an;
  {drukuj ciąg}
```

```

writeln;
writeln('Elementy ciągu po uporządkowaniu:');
drukuj ciąg  $a_1, a_2, \dots, a_n$  oraz liczbę elementów ciągu a
end.

```

Ponieważ w programie dwukrotnie chcemy drukować ciąg, więc możemy w tym celu napisać następującą procedurę:

```

procedure drukuj(x: tablica; m: integer);
var i: integer;
begin
  writeln;
  for i:=1 to m do write(x[i]:3);
  writeln;
  writeln('Liczba elementów ciągu wynosi: ', m)
end {drukuj}

```

Teraz należy zastanowić się nad wyborem metody porządkowania dowolnego n -elementowego ciągu a . Możemy przyjąć następujący algorytm:

```

for i:=1 to n-1 do
begin
  znajdź minimum ciągu  $a_i, a_{i+1}, \dots, a_n$  oraz
  wskaźnik k elementu minimalnego;
  zamień  $a_i$  z  $a_k$ 
end

```

W następnym kroku metody zstępującej musimy sprecyzować sposób szukania minimum ciągu oraz zamiany a_i z a_k . W efekcie otrzymujemy algorytm szukania minimum:

```

min:=a[i]; k:=i;
for j:=i to n do
  if a[j]<min then
    begin
      min:=a[j];
      k:=j
    end {a[j]<min}

```

Kładąc ze sobą wszystkie fragmenty, otrzymamy następujący program:

```

program sortuj;
const nmax=20;
type tablica=array[1..nmax] of integer;
var   a: tablica;   {tablica liczb całkowitych}
      n: integer;   {liczba elementów w tablicy a}

```

```

        min: integer;      {minimalna wartość w tablicy a}
        i,j,k: integer;    {indeksy elementów w tablicy a}

procedure drukuj(x: tablica; m: integer);
var i: integer;
begin
    writeln;
    for i:=1 to m do write(x[i]:3);
    writeln;
    writeln('Liczba elementów ciągu wynosi: ', m)
end {drukuj};

begin
    {czytaj ciąg}
    write('Podaj n: ');readln(n);
    writeln('Podaj elementy ciągu:');
    for i:=1 to n do read(a[i]);

    {drukuj ciąg}
    writeln;
    writeln('Elementy ciągu przed uporządkowaniem:');
    drukuj(a,n);

    {porządkuj ciąg a[1],a[2],...a[n]}
    for i:=1 to n-1 do
    begin
        min:=a[i];
        k:=i;
        for j:=i to n do
            if a[j]<min then
                begin
                    min:=a[j];
                    k:=j
                end {a[j]<min};
        a[k]:=a[i];
        a[i]:=min
    end {i};

    {drukuj ciąg}
    writeln;
    writeln('Elementy ciągu po uporządkowaniu:');
    drukuj(a,n)
end.

```


3. KOMPILACJA I WYKONYWANIE PROGRAMÓW

Prawidłowe uruchomienie programu napisanego w języku Turbo Pascal wersji 3.0 przebiega w dwóch etapach:

- a) kompilacja,
- b) wykonanie.

Kompilacja polega na przetłumaczeniu programu napisanego w języku Turbo Pascal na program zapisany binarnie (w postaci ciągu zer i jedynek) w języku wewnętrznym komputera. Programy, których zadaniem jest tłumaczenie na język wewnętrzny komputera nazywamy *kompilatorami* danego języka. Program wykorzystywany do tłumaczenia programów napisanych w języku Pascal na kod wynikowy (czyli postać binarną) nazywa się *kompilatorem Pascala*.

Na etapie kompilacji sprawdzana jest poprawność syntaktyczna programu, tzn. czy dany program został napisany zgodnie z regułami języka. Jeśli nie, to są sygnalizowane wykryte błędy, jeśli natomiast program jest syntaktycznie poprawny, to jest generowany kod wynikowy i można przejść do etapu wykonania. Gdy nie ma żadnej sygnalizacji błędów wykonania i zostają wyprowadzone wyniki, wówczas należy dokładnie sprawdzić, czy są one zgodne z wynikami oczekiwanymi. Jeśli nie, to znaczy, że mamy do czynienia z błędami nie sygnalizowanymi.

Każdy język programowania, kompilator i komputer przyczynia się do popełniania błędów określonego typu. Rozróżnia się trzy najważniejsze typy błędów programowania:

- a) błędy kompilacji, np. brak słowa `end` dla odpowiedniego słowa `begin`; są to tzw. błędy syntaktyczne (składniowe),
- b) błędy wykonania, np. jeśli jako drugi argument operatora `div` pojawi się zero, to nastąpi sygnalizacja błędu ponieważ dzielenie przez zero jest niewykonalne; są to tzw. błędy semantyczne,
- c) błędy nie wykrywane przez komputer, np. jeśli w pewnym miejscu programu zamiast znaku `+` napiszemy znak `-` lub zamiast funkcji `sqrt` napiszemy `sqr`, to obliczenia zostaną wykonane i program zakończy swoje działanie podając błędne wyniki bez jakiejkolwiek sygnalizacji.

Jest jeszcze wiele innych możliwości powstania błędów w programie, np. wprowadzenie błędnych danych. Jeśli zamiast danej 10 wprowadzimy

daną 11, to uzyskamy zły wynik. Dlatego też w celu sprawdzenia poprawności danych wejściowych program powinien zawierać instrukcje drukowania danych.

Do błędów nie sygnalizowanych należy również często popełniany błąd złego umieszczenia słów kluczowych begin i end lub ich brak. Rozważmy na przykład następujące instrukcje:

```

1) if x>0 then begin
           k:=k+1;
           s:=s+x;
           p:=p*x
       end

2) begin
   if x>0 then k:=k+1;
   s:=s+x;
   p:=p*x
end

3) begin
   if x>0 then begin
           k:=k+1;
           s:=s+x
       end;
   p:=p*x
end

```

Instrukcje 1 + 3 mają zupełnie różne znaczenie, ale wszystkie są syntaktycznie i semantycznie poprawne.

Aby ustrzec się błędów wykonania, należy napisać np.

```

if x<>0 then z:=y/x
else stop

```

gdzie stop będzie procedurą drukującą informację o przyczynie błędu.

Wszystkie te nie zamierzone zatrzymania opóźniają zarówno kompilację, jak i poprawne wykonanie programu. Najczęściej jednak czas komputera można zaoszczędzić dzięki dobrej diagnostyce błędów.

3.1. Zadania

1. Odpowiedzieć na pytania dotyczące stylu programowania:

- a) dlaczego programy powinny być czytelne?
- b) jak często umieszczać komentarze?
- c) gdzie dogodnie jest umieścić komentarz (wymień niektóre miejsca)?
- d) jak należy wybierać nazwy zmiennych?
- e) jakimi regułami można się posłużyć w celu utworzenia czytelnych skrótów nazw zmiennych?
- f) gdzie powinno się stosować puste linie?
- g) dlaczego jest pożądane stosowanie odstępów?
- h) dlaczego nie powinno się umieszczać wielu instrukcji w jednej linii?
- i) podaj kilka przykładów sytuacji, w których nawiasy zwiększają czytelność programów.

2. Jakie są cechy dobrego programu?

3. Napisz krótki program (do 50 wierszy) tak, aby kompilator sygnalizował jak największą liczbę błędów.

4. Napisz programy powodujące błędy wykonania i sprawdź, jak odpowiednie błędy są sygnalizowane:

- a) dzielenie przez zero,
- b) przekroczenie zakresu tablicy,
- c) użycie w zmiennej tablicowej indeksu mniejszego (lub większego) od dopuszczalnego,
- d) błędna dana na wejściu (np. znak lub liczba rzeczywista, gdy powinna to być liczba całkowita).

5. Podać kilka rodzajów błędów nie wykrywanych przez kompilator.

6. Jaka jest różnica między błędem syntaktycznym i błędem semantycznym? Wymienić kilka błędów syntaktycznych i semantycznych.

4. STRUKTURA PROGRAMU W JĘZYKU TURBO PASCAL 3.0

Struktura programu napisanego w języku Turbo Pascal 3.0 jest następująca:

nagłówek programu; blok.

lub

blok.

gdzie *nagłówek programu* ma postać:

program nazwa_programu(lista_nazw_plików)

natomiast *blok* składa się z części definiującej i deklaracyjnej (mogą być pominięte) oraz wykonawczej:

{definicje i deklaracje}

begin

{instrukcje}

end

Występująca w nagłówku *nazwa_programu* jest dowolnym ciągiem liter, cyfr oraz znaku podkreślenia (ale zawsze musi zaczynać się literą) i nie ma znaczenia w tekście programu. Podobnie *lista_nazw_plików*, która może być pusta, nie ma żadnego znaczenia i jest pomijana. Znak kropki na końcu programu jest obowiązkowy. Kolejność definicji i deklaracji jest dowolna, ale musi być sensowna (zob. diagramy syntaktyczne zamieszczone w dodatku A). Instrukcje w programie oddziela się znakiem średnika.

Najprostszym programem w TP 3.0 jest

begin

end.

W dalszej części rozdziału pokrótce omówimy następujące konstrukcje języka Turbo Pascal 3.0:

- instrukcje,
- procedury,
- funkcje.

4.1. Instrukcje

Instrukcje opisują czynności wykonywane na ustalonym, określonym w programie zbiorze obiektów, zwanych danymi. Składnia języka Turbo Pascal wersji 3.0 (tzn. zbiór zasad budowy wszystkich konstrukcji języka) jest przedstawiona za pomocą diagramów syntaktycznych w dodatku A.

W języku TP 3.0 można wyróżnić następujące rodzaje instrukcji:

- przypisania

nazwa_zmiennej:=wyrażenie

nazwa_funkcji:=wyrażenie

Przykład 4.1

```
znak:='k';
x:=a+b;
q[i]:=i<j;
rok:=1989
```

- złożona

```
begin
  instrukcja1;
  instrukcja2;
  ...
  instrukcjan
end
```

Przykład 4.2

```
begin
  x:=sin(y);
  k:=k+1;
  y:=y+h
end

begin
  readln(x);
  writeln('sin(',x:6:2,')=',sin(x):8:5)
end
```

- pusta

Instrukcja pusta nie powoduje wykonania żadnej czynności i została wprowadzona do języka w celu ujednolicenia opisu pewnych konstrukcji.

Przykład 4.3

Miejsce wystąpienia instrukcji pustej zaznaczono komentarzem.

```
program nic;
begin
  repeat
    {instrukcja pusta}
  until KeyPressed;
  {instrukcja pusta}
end.
```

- warunkowe

```
if wyrażenie_logiczne then instrukcja

if wyrażenie_logiczne then instrukcja1 else instrukcja2
```

Przykład 4.4

```
if n=0 then silnia:=1
      else silnia:=n*silnia(n-1);
if q then w:=w+1
```

- iteracyjne

```
for nazwa_zmiennnej:=wyrażenie1 to wyrażenie2 do instrukcja

for nazwa_zmiennnej:=wyrażenie1 downto wyrażenie2 do instrukcja

while wyrażenie_logiczne do instrukcja

repeat ciąg_instrukcji until wyrażenie_logiczne
```

Przykład 4.5

```
repeat
  i:=i+1;
  read(c[i])
until c[i]='.'
```

```

for i:=1 to n do s:=s*i
for k:=n downto 1 do P:=P*x[k]
while n>0 do n:=n-m

```

- wyboru

```

case wyrażenie of
  lista_stałych_wyboru1: instrukcja1;
  lista_stałych_wyboru2: instrukcja2;
  ...
  lista_stałych_wyborun: instrukcjan
end

```

```

case wyrażenie of
  lista_stałych_wyboru1: instrukcja1;
  lista_stałych_wyboru2: instrukcja2;
  ...
  lista_stałych_wyborun: instrukcjan
else instrukcja
end

```

Jeśli dowolna instrukcja_k ma być wykonana dla ciągu kolejnych elementów typu porządkowego, to można napisać

pierwszy..ostatni: instrukcja_k

wyszczególniając pierwszy i ostatni element tego ciągu.

Przykład 4.6

```

case miesiąc of
  1..4,9..12: liczba_wyjazdów:=2;
  5,7,8: liczba_wyjazdów:=3;
  6: begin
    liczba_wyjazdów:=1;
    opłata:=3250
  end
end {case}

case znak of
  '0'..'9': writeln('CYFRA');
  'a'..'z': writeln('MAŁA LITERA');
  'A'..'Z': writeln('WIELKA LITERA')
  else      writeln('INNY ZNAK')
end {case}

```

- wiążąca

with lista_nazw_rekordów do instrukcja

Przykład 4.7

```
with auto[1] do
begin
  nrsilnika:=32703;
  rok_produkcji:=1978;
  nr_rejestracyjny:='WRA1234'
end

with student[1], data_ur do
begin
  writeln(nazwisko);
  writeln(wydział);
  writeln('Data urodzenia: ',dzień,'.',miesiąc,'.',rok)
end
```

- wywołania procedury

nazwa_procedury(lista_parametrów_aktualnych)
nazwa_procedury

Przykład 4.8

ClrScr	{czyszczenie ekranu}
DellLine	{usunięcie wiersza wskazanego przez kursor}
Delay(500)	{wstrzymanie wykonania programu na okres 500 milisekund}
GoToXY(3,5)	{ustawienie kursora w 5 wierszu i w 3 kolumnie}

- skoku

goto etykieta

gdzie *etykieta* jest liczbą całkowitą z przedziału [1,9999] lub nazwą.

Przykład 4.9

```
goto 3;
goto koniec
```


4.2. Procedury i funkcje

Podczas opracowywania programu możemy zauważyć, że pojawiają się w nim pewne takie same ciągi instrukcji operujące na różnych wartościach, w różnym miejscu programu. Do zapisu takich właśnie ciągów instrukcji służą podprogramy, wśród których rozróżniamy procedury i funkcje. Definicja funkcji (procedury) składa się z:

- nagłówka,
- bloku, tzn. części definiującej i deklarującej różne obiekty (tzw. zmienne lokalne) na użytek tej funkcji (procedury) oraz instrukcji złożonej (zob. dodatek A).

Nagłówek procedury ma postać

```
procedure nazwa(lista_parametrów_formalnych)
```

przy czym *lista_parametrów_formalnych* może być pusta.

Nagłówek funkcji różni się tym od nagłówka procedury, że dodatkowo określa się typ wyniku funkcji

```
function nazwa(lista_parametrów_formalnych): typ_wyniku
```

Zasadnicze różnice między procedurami a funkcjami to:

- w nagłówku procedury piszemy słowo **procedure**, a w nagłówku funkcji słowo **function**,
- w nagłówku funkcji określa się typ wyniku funkcji, a w nagłówku procedury nie,
- w bloku funkcji musi wystąpić instrukcja przypisania odpowiedniej wartości pod nazwę funkcji,
- procedurę wywołujemy w instrukcjach wywołania procedury (pisząc po prostu jej nazwę z parametrami aktualnymi), a funkcję wywołujemy w wyrażeniach.

W dalszej części skryptu terminy "procedura" i "podprogram" należy rozumieć jako "procedura lub funkcja", jeśli z kontekstu nie będzie wynikało inaczej.

Aby podprogram zdefiniowany w danym programie został wykonany musi nastąpić jego wywołanie z parametrami aktualnymi (jeśli występowały parametry formalne), które muszą występować w takiej kolejności jak odpowiadające im parametry formalne. Liczba parametrów aktualnych musi być równa liczbie parametrów formalnych danego podprogramu, a typy parametrów formalnych i aktualnych muszą być zgodne (zob. [2]).

Dotychczasowe uwagi zilustrujemy przykładami.

Przykład 4.10

Chcemy wydrukować w kilku miejscach programu n pustych wierszy. W tym celu możemy napisać procedurę $\text{PiszLinie}(n)$, gdzie n jest dowolną liczbą całkowitą określającą liczbę wyprowadzanych pustych wierszy.

Definicja tej procedury może mieć postać

```
procedure PiszLinie(n: integer);
var i: integer;    {i-zmienna lokalna procedury}
begin
    for i:=1 to n do writeln
end {PiszLinie};
```

Wywołaniem takiej procedury mogą być instrukcje

```
PiszLinie(10);
PiszLinie(k+j);
```

gdzie k, j są zmiennymi typu `integer`.

Przykład 4.11

Zapiszemy funkcję sgn wyznaczającą znak danej liczby całkowitej x , gdzie

$$\text{sgn}(x) = \begin{cases} -1 & \text{jeśli } x < 0, \\ 0 & \text{jeśli } x = 0, \\ 1 & \text{jeśli } x > 0. \end{cases}$$

Definicję funkcji $\text{sgn}(x)$ w można zapisać następująco:

```
function sgn(x: integer): integer;
begin
    if x < 0 then sgn:=-1
    else if x=0 then sgn:=0
    else sgn:=1
end {sgn};
```

Funkcję $\text{sgn}(x)$ wywołujemy w wyrażeniach arytmetycznych pisząc, np.

```
k:=sgn(z)*n1;
writeln('znak liczby a= ',sgn(a):2)
```

W nagłówkach procedur i funkcji określamy typy parametrów formalnych oraz sposób komunikowania się z otoczeniem. W języku Turbo Pascal wersji 3.0 parametry mogą być przekazywane przez:

1. wartość,
2. zmienną.

ad 1. W przypadku przekazywania parametru przez wartość parametr ten jest traktowany jako lokalna zmienna programu. W chwili wywołania podprogramu przypisuje się jej wartość wyrażenia stanowiącego odpowiedni parametr aktualny.

ad 2. Parametr formalny poprzedzony słowem kluczowym **var** jest przekazywany przez zmienną i wówczas odpowiedni parametr aktualny musi być nazwą zmiennej. Wszystkie instrukcje w danym podprogramie, wykorzystujące parametry formalne poprzedzone słowem **var**, są w rzeczywistości wykonywane na odpowiadających im parametrach aktualnych, tzn. na nielokalnych zmiennych.

Należy wspomnieć, że w języku standardowym Pascal jest jeszcze możliwość przekazywania parametrów przez funkcję i przez procedurę, natomiast w języku Turbo Pascal 3.0 taki sposób przekazywania parametrów jest niedopuszczalny.

Przykład 4.12

Napiszmy procedurę, która dla danego n-elementowego ciągu liczb rzeczywistych podaje liczbę oraz sumę elementów większych od zera. Niech

```
type tablica=array[1..20] of real;
```

Możemy zdefiniować następującą procedurę:

```
procedure Suma(var x:tablica; n: integer; var SumaDodatnich: real;
               var LiczbaDodatnich: integer);
var i: integer;
begin
  SumaDodatnich:=0.0;
  LiczbaDodatnich:=0;
  for i:=1 to n do
    if x[i] > 0 then
      begin
        SumaDodatnich:=SumaDodatnich+x[i];
        LiczbaDodatnich:=LiczbaDodatnich+1
      end {x[i]>0}
    end {Suma};
```

W nagłówku procedury występują:

- trzy parametry przekazywane przez zmienną:
x, SumaDodatnich, LiczbaDodatnich,
- jeden parametr przekazywany przez wartość: n.

Gdyby w treści procedury wystąpiła instrukcja zmieniająca wartość parametru formalnego nie poprzedzonego słowem **var**, a więc zmiennej n, np.


```

while n > 0 do
begin
  if x[n] > 0 then
    begin
      SumaDodatnich:=SumaDodatnich+x[n];
      LiczbaDodatnich:=LiczbaDodatnich+1
    end {x[n]>0};
    n:=n-1
  end {n>0};

```

to parametr aktualny m występujący w instrukcji wywołania

Suma(a,m,S,liczba)

w miejscu n, miałby po zakończeniu działania procedury taką samą wartość jak przed wywołaniem procedury.

Na zakończenie tego podrozdziału omówimy jeszcze możliwość deklarowania funkcji i procedur napisanych w wersji rekurencyjnej. W procedurach rekurencyjnych występuje bezpośrednie lub pośrednie wywołanie tej samej procedury.

Przykład 4.13

Daną procedurę iteracyjną drukowania liczb naturalnych od n do 1 napisaną w następujący sposób:

```

procedure drukuj1(n: integer);
var i: integer;
begin
  for i:=n downto 1 do
    write(i:4)
  end {drukuj1}

```

możemy napisać w postaci rekurencyjnej:

```

procedure drukuj2(n: integer);
begin
  write(n:4);
  if n>1 then drukuj2(n-1)
end {drukuj2}

```

Rekurencyjna definicja procedury często skraca jej zapis, ale towarzyszy temu na ogół wielokrotne jej wywoływanie oraz obliczanie tych samych wartości po kilka razy, co w sumie pogarsza efektywność algorytmu.

Należy podkreślić, że pojęcie procedury jest podstawowym pojęciem programowania strukturalnego i często procedurę pisze się nawet wówczas, gdy występuje w niej tylko jedna instrukcja.

4.3. Typy danych

Opracowując szkic programu, należy zastanowić się nad wyborem typów danych, jakie będą wykorzystane w programie.

W programie napisanym w języku Turbo Pascal 3.0 każda zmienna musi być zadeklarowana, tzn. musi mieć określony typ. Typ zmiennej jest to zbiór wartości, jakie ta zmienna może przyjmować.

Typy danych występujące w języku Turbo Pascal 3.0 zestawiono w tabelce:

Typ	Standardowy	Porządkowy	Prosty	Strukturalny
całkowity	+	+	+	
bajtowy	+	+	+	
rzeczywisty	+		+	
logiczny	+	+	+	
znakowy	+	+	+	
wyliczeniowy		+	+	
okrojony		+	+	
tablicowy				+
łańcuchowy				+
rekordowy				+
zbiorowy				+
plikowy				+
wskaźnikowy				

4.4. Proste i strukturalne typy danych

Wśród typów danych można wyróżnić:

- typy proste,
- typy strukturalne (złożone),
- typ wskaźnikowy.

Do prostych w języku Turbo Pascal 3.0 zaliczamy typy: całkowity, bajtowy, rzeczywisty, logiczny, znakowy, wyliczeniowy i okrojony. Podstawowymi typami są typy proste. Za ich pomocą określa się m.in. złożone struktury danych. Zbiór wartości każdego typu prostego jest zbiorem skończonym i uporządkowanym. Wartości dowolnego typu prostego można ze sobą porównywać za pomocą następujących operatorów relacyjnych:

<, <=, =, >, >=, >.

Wśród tych typów wyróżnia się typy standardowe, do których zaliczamy typy: całkowity, bajtowy, rzeczywisty, logiczny i znakowy. Są one opisane w p. 4.4.1. Dwa pozostałe typy proste, nie będące typami standardowymi, tzn. wyliczeniowy i okrojony, zostały omówione w p. 4.4.2.

Spśród typów prostych wszystkie typy, za wyjątkiem rzeczywistego (real) są typami porządkowymi. Dla danej x dowolnego typu porządkowego są określone funkcje:

- ord(x) - której wartością jest liczba porządkowa elementu x w tym typie,
- pred(x) - wyznaczająca element bezpośrednio poprzedzający x ,
- succ(x) - wyznaczająca element bezpośrednio następujący po x .

Uwaga. Funkcja $\text{pred}(x)$ nie jest określona dla pierwszego elementu natomiast funkcja $\text{succ}(x)$ dla ostatniego elementu danego typu.

Typy strukturalne są wykorzystywane do opisu obiektów złożonych. Do strukturalnych w TP 3.0 należą typy:

- tablicowy,
- łańcuchowy,
- rekordowy,
- zbiorowy,
- plikowy.

W języku Turbo Pascal 3.0 zmienne wszystkich typów danych z wyjątkiem zmiennych typu wskaźnikowego są zmiennymi statycznymi. Dla zmiennej typu statycznego miejsce w pamięci komputera jest przydzielane podczas kompilacji deklaracji tej zmiennej. Natomiast dla zmiennych typu wskaźnikowego miejsce w pamięci komputera jest przydzielane i zwalniane w trakcie wykonywania programu w chwili napotkania odpowiedniego żądania. Z tego względu zmienne typu wskaźnikowego nazywa się zmiennymi dynamicznymi. Typem wskaźnikowym zajmiemy się w rozdz. 10.

4.4.1. Typy standardowe

Typ całkowity o nazwie `integer` zawiera liczby z przedziału `(-maxint..maxint)`,

gdzie `maxint` jest stałą standardową o maksymalnej wartości całkowitej dostępnej na danym komputerze i w TP 3.0 wynosi 32 767. Przykłady liczb całkowitych: 17, -109, 0.

Każda dana typu całkowitego jest reprezentowana w 2 bajtach pamięci (1 bajt=8 bitów).

Na wielkościach typu całkowitego są określone operacje:

dodawanie (+), odejmowanie (-), mnożenie (*),
 dzielenie (/), dzielenie całkowite (div), modulo (mod),
 przesunięcie w lewo (shl), przesunięcie w prawo (shr),
 negacja (not), iloczyn logiczny (and),
 suma logiczna (or), różnica symetryczna (xor).

Przykład 4.14

<u>Wyrażenie</u>	<u>Wynik</u>
3/4	0.75
3 div 4	0
3 mod 4	3
3 shl 1	6
3 shr 1	1
not 3	-4
not 5	-6
5 and 2	0
5 and 3	1
5 or 2	7
5 or 3	7
5 xor 2	7
5 xor 3	6

Dla elementów dowolnego typu porządkowego (a więc i dla typu integer) są określone funkcje standardowe

ord(k)

pred(k)

succ(k)

Przykład 4.15

ord(-2)=-2,
ord(0)=0,
ord(5)=5,
pred(4)=3,
succ(6)=7.

Typ bajtowy o nazwie byte jest to podzbiór liczb całkowitych z przedziału [0,255]. Każda dana typu byte zajmuje 1 bajt pamięci.

W języku Turbo Pascal 3.0 dane typu całkowitego i bajtowego można zapisywać za pomocą cyfr szesnastkowych, tzn. cyfr od 0 do 9 oraz liter od A do F. Takie liczby poprzedzamy znakiem \$, np.

\$A, \$A2, -\$BC (odpowiednie wartości dziesiętne 10, 162, -188).

Typ rzeczywisty oznaczany jako real, jest to skończony podzbiór liczb rzeczywistych zawierający liczbę 0 oraz liczby o wartości bezwzględnej z przedziału $[10^{-38}, 10^{38}]$. Przykładami liczb rzeczywistych są 5.0, -7.25, 0.0, 4E-3 (czyli 0.004). Zapis 4E-3 oznacza 4 razy 10^{-3} . Pisząc liczby rzeczywiste zamiast przecinka używamy kropki.

Każda dana typu rzeczywistego jest reprezentowana w 6 bajtach pamięci z dokładnością ok. 11 cyfr znaczących.

Na wielkościach typu rzeczywistego są określone działania:

+, -, *, /.

Ponadto są określone standardowe funkcje przejścia z typu rzeczywistego na typ całkowity:

trunc(x) - odrzucenie części ułamkowej z x,

np. trunc(7.4)=7, trunc(-7.8)=-7.

round(x) - zaokrąglenie do najbliższej liczby całkowitej,

np. round(3.75)=4,
round(-3.15)=-3,
round(3.5)=4,
round(-3.5)=-4.

Za pomocą funkcji trunc funkcję round można zdefiniować następująco:

$$\text{round}(x) = \begin{cases} \text{trunc}(x+0.5) & \text{jeśli } x \geq 0, \\ \text{trunc}(x-0.5) & \text{jeśli } x < 0. \end{cases}$$

Typ logiczny o nazwie `boolean` jest określony tylko przez dwie wartości:

`false` (fałsz), oznaczaną umownie przez 0,

`true` (prawda), oznaczaną umownie przez 1.

Każda dana typu logicznego jest pamiętana w 1 bajcie pamięci. Dla zmiennych typu `boolean` są określone cztery operatory:

<u>Operator</u>	<u>Operacja</u>
<code>and</code>	koniunkcja
<code>or</code>	alternatywa
<code>xor</code>	różnica symetryczna
<code>not</code>	negacja

W tabelce poniżej zestawiono operacje na zmiennych typu logicznego.

p	q	not p	p or q	p and q	p xor q
1	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	0	1
0	0	1	0	0	0

Funkcje `ord(p)`, `pred(p)` oraz `succ(p)` dla danej `p` typu `boolean` przyjmują następujące wartości:

`ord(false)=0`, `ord(true)=1`,
`pred(true)=false`, `succ(false)=true`,
`pred(false)`, `succ(true)` – nieokreślone.

Typ znakowy o nazwie `char` obejmuje uporządkowany zbiór znaków zewnętrznych zależny od komputera. Zawsze zawiera on:

- cyfry od 0 do 9
`'0'<'1'<...<'9'`,
- łacińskie litery małe i wielkie alfabetycznie uporządkowane
`'a'<'b'<...<'z'` oraz `'A'<'B'<...<'Z'`,
- znak spacji `' '`,
- symbole specjalne, np. `'.'`, `'.'`, `'?`, `';`, `'/'`, `'*'`.

Wartości typu znakowego zapisujemy w apostrofach. Każda dana typu znakowego jest pamiętana w 1 bajcie pamięci.

Dla wartości typu `char` są określone funkcje standardowe:

`upcase(z)` – której wartość jest wyznaczana następująco:
 jeśli argumentem jest mała litera, to wynikiem jest odpowiednia wielka litera, w przeciwnym razie wynikiem jest ten sam znak,

`ord(z)` - której wartością jest numer porządkowy znaku `z`,

`chr(i)` - której wartością jest znak o numerze porządkowym `i`, jeśli taki znak istnieje.

Prawdziwe są równości:

`ord(chr(i))=i`, jeśli `chr(i)` jest określone,

`chr(ord(z))=z`, dla `z` typu `char`.

Przykład 4.16

`upcase('t')='T', upcase('T')='T', upcase('+')='+',`
`ord('a')=97, chr(97)='a', chr(ord('A'))='A'.`

4.4.2. Typy wyliczeniowe i okrojone

Typ wyliczeniowy definiujemy poprzez wyliczenie jego elementów, które muszą być unikalnymi nazwami, tzn. różnymi ciągami liter, cyfr i znaku podkreślenia zaczynającymi się od litery (nazwa w TP 3.0 może zawierać maksymalnie 127 znaków, przy czym wszystkie znaki są istotne). Na elementach typu wyliczeniowego nie wykonuje się żadnych operacji arytmetycznych.

Przykład 4.17

```
type pomieszczenie=(aula, klasa, gabinet, korytarz, szatnia);
budynek=(D_1, D_2, C_1, C_2, T_18);
zmienne=(a1, a2, a3, a4);
waluta=(dolar, funt, marka);
```

Elementy typu wyliczeniowego są uporządkowane i można je ze sobą porównywać, np.

```
dolar < funt,
funt < marka,
```

oraz mają przypisane numery porządkowe

```
ord(dolar)=0,
ord(funt) =1,
ord(marka)=2.
```

Dwa różne typy wyliczeniowe nie mogą mieć wspólnych wartości. Znacznym ograniczeniem w stosowaniu typu wyliczeniowego jest fakt, że jego wartości nie można wczytywać ani drukować.

Typ okrojony jest niepustym podzbiorem dowolnego typu porządkowego (integer, byte, char, boolean, wyliczeniowego).

Przykład 4.18

```
type pomieszczenie=(aula, klasa, gabinet, korytarz, szatnia):
    zakres=0..9;           {typ okrojony typu integer}
    WielkaLitera='A'..'Z'; {typ okrojony typu char}
    cyfra='0'..'9';        {typ okrojony typu char}
    sala=aula..gabinet;    {typ okrojony typu wyliczeniowego}
    logiczny=false..true;  {typ okrojony typu boolean}
```

Numery porządkowe dla wartości należących do typu okrojonego pierwotnego typu T są równe numerom porządkowym w typie T.

Przykład 4.19

Niech

```
type cyfra=(zero, jeden, dwa, trzy, cztery);
var n: 15..30;
    c: cyfra;
    w: dwa..cztery;
    ...
    n:=15;
    c:=zero;
    w:=dwa;
```

Wówczas $\text{ord}(n)=15$, $\text{ord}(30)=30$, $\text{ord}(c)=0$, $\text{ord}(w)=2$, $\text{ord}(\text{cztery})=4$,
 $\text{pred}(\text{dwa})=\text{jeden}$, $\text{succ}(\text{dwa})=\text{trzy}$,
 $\text{pred}(\text{zero})$, $\text{succ}(\text{cztery})$ - nieokreślone.

4.5. Definicje stałych

W celu zwiększenia czytelności programu oraz umożliwienia jego modyfikacji często używa się stałych, których definicja w języku Turbo Pascal 3.0 wygląda następująco:

```
const nazwa1=T1;
    ...
    nazwaK=Tk;
```

gdzie $T_1, \dots, T_k$ - liczba, łańcuch lub nazwa stałej, która może być poprzedzona znakiem + lub -.

Przykład 4.20

```

const m=10;
      minusm=-m;
      plusm=+m;
      szablon='*****';
      nl=^J;      {znak sterujący oznaczający nową linię}
      cr=^M;      {znak sterujący oznaczający powrót karetki}
      znakA=#65;   {znak o kodzie 65}
      znakN=#$4E;  {znak o kodzie szesnastkowym 4E}

```

W TP 3.0 standardowo są predefiniowane dwie stałe:

```

maxint=32 767;
pi=3.1415926536.

```

4.6. Deklaracje zmiennych

W języku Turbo Pascal 3.0 zmienne można deklarować dwoma sposobami:

```

var z1,z2,...,zk: T1;
    ...
    p1,p2,...,pt: Tn;

```

lub

```

type nazwa_typu=T;
var z1,z2,...,zk: nazwa_typu;

```

gdzie:

z1,z2,...,zk,...,p1,p2,...,pt - różne nazwy zmiennych,
T, T1,...,Tn - nazwy lub opisy typów.

Przykład 4.21

```

type odbiornik=(radio, tv);
      WielkaLitera='A'..'Z';
      var MałaLitera: 'a'..'z';
          l: WielkaLitera;
          S: odbiornik;
      drzewo: (klon, lipa, sosna, wierzba);
      nrdomu: 1..20;

```

Zakresem deklaracji zmiennej jest blok, w którym ta zmienna została zadeklarowana.

W języku Turbo Pascal 3.0 istnieje możliwość przypisywania zmiennym danych początkowych. Deklaracji zmiennych, którym nadajemy wartości początkowe, nie poprzedzamy słowem `var` tylko `const`.

Przykład 4.22

```
const godz: integer=60;
      głoski: set of char=['a','e','i','o','u','y'];
      jest: boolean=true;
      hasło: string[10]='PARASOL';
      sen: array[1..4] of integer=(5,7,6,9);
```

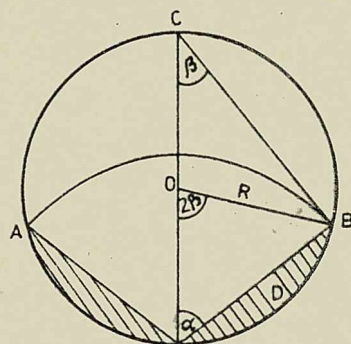
Jeśli wartości tak zadeklarowanych zmiennych ulegną zmianie w trakcie wykonywania programu, to przy ponownym uruchomieniu programu znajdującego się w pamięci operacyjnej zmienne mają wartość z ostatniego wykonywania. W związku z tym zaleca się nadawanie wartości początkowych tym zmiennym, które są traktowane w programie jak stałe.

Na zakończenie tego podrozdziału podamy przykład, w którym dokładnie precyzujemy problem oraz szukamy metod jego rozwiązania. Następnie podajemy treści programów napisanych w języku Turbo Pascal 3.0 oraz wyniki uruchomienia tych programów na IBM PC.

Przykład 4.23

Do palika wbitego na okręgu o promieniu R ograniczającego łukę, sznurkiem długości D jest przywiązana koza. Jaka powinna być długość sznurka, żeby koza pasła się tylko na połowie łuki?

Analizując zadanie będziemy wykorzystywać oznaczenia z rys. 4.1.



Rys. 4.1

Niech $\alpha = \frac{1}{2} \angle AKB$. Wówczas

$$\operatorname{tg} \alpha = \frac{BC}{KB}, \text{ gdzie } BC = \sqrt{4R^2 - D^2}$$

czyli

$$\operatorname{tg} \alpha = \frac{\sqrt{4R^2 - D^2}}{D}.$$

Stąd

$$\alpha = \arctg \frac{\sqrt{4R^2 - D^2}}{D}.$$

Pole S wypasu kozy jest sumą pola zakreskowanego S_1 oraz pola S_2 będącego wycinkiem koła o promieniu D , środku K i łuku AB , czyli

$$S = S_1 + S_2, \quad \text{gdzie } S_2 = \alpha D^2.$$

Aby obliczyć pole S_1 , obliczamy pole wycinka koła o promieniu R , środku w punkcie O oraz łuku AB , od którego odejmujemy pole czworokąta $AKBO$, czyli

$$S_1 = 2\beta R^2 - 2P_{\Delta OKB},$$

gdzie

$$P_{\Delta OKB} = \frac{R \cdot D \cdot \sin \alpha}{2}, \quad \beta = \frac{\pi}{2} - \alpha.$$

Stąd

$$S = \alpha D^2 + (\pi - 2\alpha) \cdot R^2 - R \cdot D \cdot \sin \alpha.$$

Dalej zadanie można rozwiązać dwiema metodami.

Metoda 1

Chcemy znaleźć takie D , by

$$S = \frac{1}{2} \pi R^2.$$

Z rysunku 4.1 widać, że interesuje nas $D > R$. W związku z tym musimy stabilizować funkcję

$$S(D) = \alpha D^2 - R \cdot D \cdot \sin \alpha + (\pi - 2\alpha) \cdot R^2$$

w przedziale $[D_{\min}, D_{\max}]$ z krokiem h , przy czym

$$\alpha = \arctg \frac{\sqrt{4R^2 - D^2}}{D},$$

$D_{\min}$ – minimalna długość sznura,

$D_{\max}$ – maksymalna długość sznura.

Na początek można wybrać:

$$D_{\min} = R, \quad D_{\max} = 2R, \quad h - \text{dowolne}$$

a następnie zawęzić przedział poszukiwań $[D_{\min}, D_{\max}]$ oraz zmniejszyć krok h .

Metodę tę można zapisać następująco:

- (1) Podaj $D_{\min}, D_{\max}, h$.
- (2) Stablicuj funkcję $S(D)$ w przedziale $[D_{\min}, D_{\max}]$ z krokiem h .
- (3) Przeprowadź analizę otrzymanych wyników.
- (4) Czy otrzymane wyniki są wystarczająco dokładne, tzn. czy

$$|S(D) - \frac{1}{2}\pi R^2| < \epsilon ?$$

Jeśli tak, to przejdź do (7).

- (5) Wybierz nowe $D_{\min}, D_{\max}, h$.
- (6) Przejdź do (2).
- (7) Koniec.

Punkty (3), (4), (5) może wykonywać człowiek lub program.

Poniżej zamieszczamy tekst programu rozwiązującego zadanie o kozie według podanej metody, z kontrolą wyników dokonywaną przez człowieka. Jako ćwiczenie proponujemy takie zmodyfikowanie programu, by analiza wyników była wykonywana automatycznie przez program.

```

program metodai;
var R,R2,alfa,Dmin,Dmax,d2,h,S,pom: real;
    litera: char;
begin
  clrscr;      {czyszczenie ekranu}
  repeat
    writeln('Podaj promień R (R>0:');
    readln(R);
  until R>0;
  writeln(' Promień koła R=',R:5:2);
  R2:=sqr(R);
  pom:=4*R2;
  writeln(' Połowa pola koła wynosi:', 0.5*pi*R2:13:8);
  litera:='y';
  while litera='y' do
  begin
    writeln('Podaj Dmin,Dmax,h');
    readln(Dmin,Dmax,h);
  end
end

```

```

while Dmin<=Dmax do
begin
  d2:=sqr(Dmin);
  alfa:=arctan(sqrt(pom-d2)/Dmin);
  S:=alfa*d2-R*Dmin*sin(alfa)+(pi-2*alfa)*R2;
  writeln('  D=',Dmin:10:6,'  S=',S:13:8);
  Dmin:=Dmin+h
end {while};
writeln('Czy obliczenia powtorzyc(y/n)?');
readln(litera)
end {litera='y'}
end.

```

Oto wyniki otrzymane po uruchomieniu programu dla danych początkowych Dmin=10, Dmax=20, h=0.8:

Podaj promień R (R>0):

10.0

Promień kola R=10.00

Półowa pola kola wynosi: 157.07963268

Podaj Dmin, Dmax, h

10.0 20.0 0.8

D= 10.000000	S= 122.83696986
D= 10.800000	S= 139.86949921
D= 11.600000	S= 157.36044825
D= 12.400000	S= 175.15768138
D= 13.200000	S= 193.09678421
D= 14.000000	S= 210.99767204
D= 14.800000	S= 228.65975121
D= 15.600000	S= 245.85468009
D= 16.400000	S= 262.31485878
D= 17.200000	S= 277.71356625
D= 18.000000	S= 291.62640904
D= 18.800000	S= 303.44106453
D= 19.600000	S= 312.04942239

Czy obliczenia powtorzyc(y/n)?

y

Podaj Dmin, Dmax, h

10.8 11.6 0.1

D= 10.800000	S= 139.86949921
D= 10.900000	S= 142.03381501
D= 11.000000	S= 144.20501835
D= 11.100000	S= 146.38281906

D= 11.200000 S= 148.56692449
 D= 11.300000 S= 150.75703937
 D= 11.400000 S= 152.95286583
 D= 11.500000 S= 155.15410326
 D= 11.600000 S= 157.36044824

Czy obliczenia powtorzyc (y/n)?

y

Podaj Dmin, Dmax, h

11.5 11.6 0.01

D= 11.500000 S= 155.15410326
 D= 11.510000 S= 155.37451295
 D= 11.520000 S= 155.59497341
 D= 11.530000 S= 155.81548434
 D= 11.540000 S= 156.03604543
 D= 11.550000 S= 156.25665638
 D= 11.560000 S= 156.47731688
 D= 11.570000 S= 156.69802662
 D= 11.580000 S= 156.91878530
 D= 11.590000 S= 157.13959261
 D= 11.600000 S= 157.36044824

Czy obliczenia powtorzyc (y/n)?

y

Podaj Dmin, Dmax, h

11.58 11.59 0.001

D= 11.580000 S= 156.91878530
 D= 11.581000 S= 156.94086385
 D= 11.582000 S= 156.96294288
 D= 11.583000 S= 156.98502240
 D= 11.584000 S= 157.00710241
 D= 11.585000 S= 157.02918290
 D= 11.586000 S= 157.05126387
 D= 11.587000 S= 157.07334533
 D= 11.588000 S= 157.09542727
 D= 11.589000 S= 157.11750970
 D= 11.590000 S= 157.13959261

Czy obliczenia powtorzyc (y/n)?

n

Obliczenia programu zostały w tym miejscu przerwane. Wartość zmiennej S (oznaczającej obliczone pole wypasu kozy) najbliższa połowie pola koła o promieniu $R=10$ wynosi 157.07334533 (połowa pola koła wynosi 157.07963268), a więc dokładność obliczeń jest mała. Obliczona długość

sznura kozy $D=11.587$. Przedział $[D_{\min}, D_{\max}]$ był 3-krotnie zawężany, przy czym jednocześnie był zmniejszany krok h . Program wymaga ciągłej analizy wyników przez użytkownika i wprowadzania odpowiednich danych w kolejnych krokach algorytmu.

Rozważmy teraz drugi sposób rozwiązania zadania o kozie.

Metoda 2

Chcemy znaleźć takie D , by $S(D) = \frac{\pi R^2}{2}$, czyli znaleźć rozwiązanie równania

$$(\%) \quad W(D) = \alpha D^2 - R \cdot D \cdot \sin \alpha + \left(\frac{\pi}{2} - 2\alpha\right) R^2 = 0,$$

gdzie $\alpha = \arctg \frac{\sqrt{4R^2 - D^2}}{D}$.

W celu znalezienia rozwiązania równania (%) możemy zastosować np. metodę połowienia przedziału. W metodzie tej konstruujemy ciąg kolejnych przybliżeń x , następującym algorytmem:

- (1) jeśli $W(D_{\min})$ i $W(D_{\max})$ mają takie same znaki, to brak rozwiązania i stop,
- (2) $x = (D_{\min} + D_{\max})/2$,
- (3) jeśli $W(x)$ i $W(D_{\max})$ mają różne znaki, to $D_{\min} := x$ i idź do (2),
- (4) $D_{\max} := x$ i idź do (2).

Warunkiem zakończenia obliczeń jest:

$k=-1$, jeśli rozwiązanie nie istnieje,

$k=0$, jeśli $|W(x)| \leq \epsilon$,

$k=1$, jeśli $|D_{\max} - D_{\min}| \leq \epsilon$.

program metoda2;

var pi2,P,R,R2,R4,Dmin,Dmax,WDmin,WDmax,x,Wx,eps: real;

k: -1..2;

function W(D,R,R2,R4,pi2: real): real;

var D2,alfa: real;

begin

D2:=sqr(D);

alfa:=arctan(sqrt(R4-D2)/D);

W:=alfa*D2-R*D*sin(alfa)+(pi2-2*alfa)*R2

end {W};

begin

{wczytywanie danych}

```

writeln('Podaj promień koła R');
readln(R);
writeln('Podaj eps');
readln(eps);
writeln('Podaj Dmin, Dmax');
readln(Dmin,Dmax);
  {obliczanie wielkości pomocniczych}
R2:=sqr(R);
R4:=4*R2;
pi2:=0.5*pi; {pi=stała predefiniowana}
P:=pi2*R2;
writeln(' Połowa pola koła wynosi:',P:13:8);

  {szukanie miejsca zerowego funkcji W w przedziale [Dmin,Dmax]
   metodą połowienia przedziału}

WDmin:=W(Dmin,R,R2,R4,pi2);
WDmax:=W(Dmax,R,R2,R4,pi2);
k:=2;
if WDmin*WDmax>0
  then k:=-1
  else
    repeat
      x:=0.5*(Dmin+Dmax);
      Wx:=W(x,R,R2,R4,pi2);
      if Wx*WDmax<0
        then Dmin:=x
        else
          begin
            Dmax:=x;
            WDmax:=Wx
          end;
      if abs(Wx)<=eps
        then k:=0
        else if abs(Dmax-Dmin)<=eps then k:=1
      until (k=0) or (k=1);
    {wyprowadzanie wyników}
    case k of
      -1: writeln('Brak rozwiązania w przedziale [' ,Dmin,',',Dmax,']');
      0,1: begin
        writeln(' Obliczona długość sznura wynosi:',x:10:6);
        writeln(' Obliczone pole wypasu kozy wynosi:',Wx+P:13:8)
      end
    end
  end

```

```

    end {case}
end.

```

Poniżej zamieszczono tabulogram otrzymany podczas uruchamiania programu metoda2.

Podaj promień kola R

10.00

Podaj eps

0.00001

Podaj Dmin, Dmax

10.0 20.0

Polowa pola kola wynosi: 157.07963268

Obliczona dlugosc sznura wynosi: 11.587286

Obliczone pole wypasu kozy wynosi: 157.07966062

4.7. Zadania

1. Czym różnią się podane symbole języka Pascal?

- | | | |
|----|------|-------|
| a) | a | 'a' |
| b) | b[i] | bi |
| c) | 1 | '1' |
| d) | 5 | 5.0 |
| e) | := | = |
| f) | end | {end} |
| g) | / | div |
| h) | * | . |

2. Zadeklarowano zmienne

```

var x,y,z: real;
    i,j,k: integer;

```

Które z poniższych wyrażeń są syntaktycznie poprawne, jakiego są typu, ile jest niejawnych operacji przejścia z typu integer do typu real?

```

x+y*i;
i div j+x;
i div (j+x);
i mod (j+y);
i*x+j*y;

```


$$(x+y) < (i+j);$$

$$2/x;$$

$$3/5;$$

$$i/j/x;$$

3. Zapisać w języku Pascal następujące wyrażenia logiczne:

$$7x \geq a;$$

$$y < -2.5 \vee x \geq 8.75;$$

$$-1 < i < 1 \wedge -2 \leq j \leq 2;$$

$$\frac{x^2}{a} + \frac{y^2}{b} = 1 \wedge z > -1;$$

$$\varphi \geq \sin \psi \wedge \varphi + \log_{10} \psi \neq 0;$$

$$\sim(a > b) \vee (z < 4);$$

4. Zapisać w języku Pascal następujące wyrażenia arytmetyczne:

$$\frac{a \cdot b}{c \cdot d};$$

$$\frac{a^{-b}}{c};$$

$$\frac{1+a}{a+a^2};$$

$$\frac{a + \operatorname{tg} \beta}{\sin \alpha + \cos \beta};$$

$$\frac{e^{x+y}}{\operatorname{ctg}(x-y)};$$

$$\frac{x^a - y^b}{\sqrt{|a| - x}};$$

$$\sqrt[3]{x + \sqrt{x+1}};$$

$$\phi + \frac{1}{\zeta + \frac{2}{\pi}};$$

$$\frac{\ln(a-i \cdot b)}{z+1};$$

$$a^2 - c + \frac{a}{b \cdot c + \frac{c}{d + \frac{e}{f}}};$$

$$\frac{1}{2} \ln \left| \frac{w-a}{w+a} \right| + \sqrt{a}.$$

5. Niech

const trzyspacje = ' ';

var m, n: integer;

a, b: real;

p, q: boolean;

c1, c2: char;

Które instrukcje są poprawne, a które błędne? Odpowiedź uzasadnić.

- a) $m := \text{trunc}(b) + a;$
- b) $p := m + n;$
- c) $\text{read}(c1, c2, ' ');$
- d) $c1 := \text{trzyspacje};$
- e) $c2 := 'AB';$
- f) $p := a > b;$
- g) $m := n \bmod a;$
- h) $'c1' := 'c2'$
- i) $c2 := \text{chr}('a');$
- j) $m := m - \text{ord}('0');$
- k) $\text{writeln}(a, b, m, n, q, p, q);$
- l) $\text{read}(m, m);$
- ł) $a := m / n;$
- m) $m := a \text{ div } n;$

6. Zaprogramować obliczanie następujących wielkości:

- a) $b = \frac{4a}{3}$
- b) $c = p^{0.5}$
- c) $y = 5x^3 + 3x^2 - 2x + 2$
- d) $t = \sin \alpha - \sin^2 \alpha$
- e) $w = (k^2 - 4) : (k - 2);$
- f) $f = p/2 + (6 - r)/4 - \frac{f}{2};$
- g) $x = 6 + t^5$
- h) $t = 2\pi/4$
- i) jeśli $a < b$ lub $c > d$, to $x = 4$, w przeciwnym przypadku $y = 0$.

7. Czy w następujących przykładach kolejność instrukcji jest istotna?

- a) $n := m;$
 $m := p;$
 $p := 2 * n - m;$
- b) $y := x;$
 $z := 2 * x;$
- c) $x := 3 * x - 1;$
 $S := S + x;$
- d) $z := z + x;$
 $z := z - x;$
- e) $a := a + x;$
 $x := a + x;$

8. Za pomocą jednej instrukcji przypisania zapisać:

- a) nadanie zmiennej całkowitej wartości m , jeśli zmienna miała wartość n natomiast wartości n , jeśli miała wartość m ,
- b) obliczanie kwartału w zależności od numeru miesiąca,
- c) nadawanie zmiennej logicznej p wartości $true$, jeśli $x > 2$ oraz $false$, jeśli $x \leq 2$.

9. Czy dane fragmenty programów są równoważne, tzn. czy nadają zmiennym takie same wartości?

- a) 1) `if k=1 then begin k:=k+2; S:=S+k end`
2) `if k=1 then k:=k+2; S:=S+k`
- b) 1) `if x=y then p:=1 else p:=2`
2) `if x=y then p:=1; p:=2`
- c) 1) `k:=1; for i:=1 to n do if a[i] > a[k] then k:=i`
2) `k:=1; for i:=1 to n do if a[k] <= a[i] then k:=i`
- d) 1) `for j:=1 to n do S:=S+a[j]; S:=S/n`
2) `for j:=1 to n do begin S:=S+a[j]; S:=S/n end`

10. Zbadaj poprawność semantyczną przypisań w następującym fragmencie programu:

```
var a,b,c: real;
    m,n,k: integer;
    s,p,q: boolean;
...
s:=a*c;
q:=m<n;
k:=b/c-2*m;
n:=a;
b:=m;
c:=sqr(k);
p:=q=s;
```

11. Czy poprawna jest instrukcja:

```
for i:=1 to 20 do
begin
    writeln(i);
    i:=i+1
end
```

12. Napisać program obliczający wielkości:

a) $t = 2\pi\sqrt{l/g}$

dane: l - liczba rzeczywista dodatnia,
 g - stała grawitacji ($g=981$ cm/s),

b) $F = Gm_1m_2/r^2$

dane: m_1, m_2, r - liczby rzeczywiste dodatnie,
 G - uniwersalna stała grawitacji ($G = 6.673 \cdot 10^{-8} \text{ cm}^3 / (\text{g} \cdot \text{s}^2)$).

13. Z podanych wyrażeń wybrać te, które są poprawne w języku Turbo Pascal i określić ich wartość w kodzie ASCII:

- a) `ord('a');`
- b) `ord(1);`
- c) `ord('3');`
- d) `ord('abc');`
- e) `ord(.);`
- f) `ord(' ');`

14. Jaki będzie wydruk?

- a) `writeln(ord(false):3, ord(true):3)`
- b) `writeln(ord(succ(false)):3, ord(pred(true)):3)`

15. Jak należy przygotować dane dla programu zawierającego deklarację zmiennych:

```
var i,j,k,l,m,n: integer;
```

oraz następujące instrukcje czytania:

- a) `read(n,m,i,j); read(k,l);`
- b) `read(n,m,i,j,k,l);`
- c) `readln(n,m,i,j); read(k,l);`
- d) `readln(n); readln(m); readln(i); readln(j);`
`readln(k); readln(l);`
- e) `readln (n,m,i,j,k,l);`

Który z zapisów (a)-(e) uważasz za najlepszy?

16. Niech

```
const n=10;
var in,im,p,q: integer;
    s,t: real;
```


Czy podane instrukcje są poprawne?

- a) read(n);
- b) p:=s div t;
- c) q:=n=10;
- d) im:=in/n;
- e) write(n,p,q,im)

17. Czy poprawne są następujące definicje typu wyliczeniowego:

- a) type waga=(3.75,4.5,28.2,13.7);
- b) type litera=(a,b,z,s,w);
- c) type SemestrLetni=(luty,...,czerwiec);
- d) type znak=('A','B','C');
- e) type cyfra=(0,1,2,3,4);
- f) type dni=(p,w,s,c,p,s,n);

18. Czy poprawne są następujące definicje typu okrojonego:

- a) type rok=1900..1900;
- b) type wojna=1914..1917,1939..1945;
- c) type litera='A'..'z';
- d) type wzrost=1.5..2.1;

19. Czy poprawny jest program:

```
program nic;
begin
    ;;;;
end.
```

20. Wydrukować napis:

Tytuł: 'PASCAL'.

21. Ile razy zostanie wywołana procedura wyjścia w danym fragmencie programu?

```
for licznik1:=1 to 5 do
    for licznik2:=4 to 9 do
        writeln(licznik1#licznik2)
```

22. Napisać program, który danym zmiennym rzeczywistym o wartościach x, y (x≠y) nadaje nowe wartości w następujący sposób: zmiennej o wartości mniejszej przypisuje wartość ich średniej arytmetycznej, zmiennej o wartości większej przypisuje wartość x*y.

23. Cyfry od 1 do 5 oznaczają kolory:

- 1 - szary,
- 2 - czerwony,
- 3 - zielony,
- 4 - niebieski,
- 5 - czarny.

Napisz program, który dla danej liczby całkowitej kolor ($1 \leq \text{kolor} \leq 5$), wydrukuje tekst w postaci (np. dla kolor=4)

Moim ulubionym kolorem jest niebieski.

24. Danych jest n liczb naturalnych. Dla każdej z tych liczb wydrukować napis "parzysta" lub "nieparzysta".

25. Daną cyfrę k (od 0 do 9) wydrukować słownie.

26. Dana jest cyfra rzymska od I do X. Wydrukować ją w postaci arabskiej.

27. Dana jest liczba całkowita z przedziału $[1, 12]$ oznaczająca miesiąc. Wydrukować, który to kwartał.

28. W zależności od podanego dnia i miesiąca wydrukować nazwę odpowiedniej pory roku.

29. Dana jest liczba naturalna oznaczająca rok. Napisz program sprawdzający czy jest to rok przestępny, czy nie i drukujący odpowiedni tekst w każdym przypadku (rok jest przestępny, jeśli dzieli się przez 4 i nie dzieli się przez 100 lub dzieli się przez 400).

30. Dane są liczby rzeczywiste a, b . Obliczyć

$$P = \begin{cases} a^2 & \text{jeśli } a=b, \\ \frac{1}{2} ab & \text{jeśli } a \neq b. \end{cases}$$

31. Dane są a, b, c - długości boków trójkąta. Obliczyć:

- promień okręgu wpisanego - r ,
- promień okręgu opisanego - R ,
- pole trójkąta - S ,
- kąty trójkąta α, β, γ (w stopniach).

Wzory:

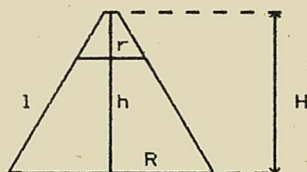
$$p = \frac{1}{2}(a+b+c),$$

$$r = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}, \quad S = pr, \quad R = \frac{abc}{4S},$$

$$\alpha = 2\arctg \frac{r}{p-a}, \quad \beta = 2\arctg \frac{r}{p-b}, \quad \gamma = 180^\circ - \alpha - \beta.$$

32. Napisać programy dla następujących zadań:

a) Dla stożka ściętego dane są wartości r , R , h .



Obliczyć:

- l - tworzącą stożka,
- M - pole powierzchni bocznej,
- V - objętość stożka,
- H - wysokość stożka.

Wzory:

$$l = \sqrt{h^2 + (R-r)^2},$$

$$M = \pi l (R+r),$$

$$V = \frac{\pi h}{3} (R^2 + r^2 + rR),$$

$$H = \frac{hR}{R-r}.$$

b) Dany jest stożek o promieniu podstawy r , wysokości h i tworzącej l . Obliczyć:

- M - pole powierzchni bocznej,
- S - pole powierzchni całkowitej,
- V - objętość stożka.

Wzory:

$$M = \pi r l, \quad S = \pi r (r+l), \quad V = \frac{1}{3} \pi r^2 h.$$

33. Dana jest liczba rzeczywista x . Obliczyć:

$$A(x) = \begin{cases} x^2+1 & \text{jeśli } x < -1, \\ x^3+7 & \text{jeśli } x = -1, \\ x^5-x^2+1 & \text{jeśli } x > -1. \end{cases}$$

34. Dla każdej z podanych n liczb całkowitych (n - dana liczba naturalna) obliczyć wielkość S określoną wzorem:

$$S = \begin{cases} x^2+3|x| & \text{jeśli } x \text{ - parzyste,} \\ \frac{x+7}{2} & \text{jeśli } x \text{ - nieparzyste i } x > 0, \\ \frac{x-7}{2} & \text{jeśli } x \text{ - nieparzyste i } x < 0. \end{cases}$$

35. Dane są liczby rzeczywiste a, b, c, d ($a < b, c > 0$). Obliczyć:

$$Z = \begin{cases} e^x + |y^3| & \text{jeśli } x^2 + y^2 \leq a, \\ \sin(x) + d^2 & \text{jeśli } a < x^2 + y^2 < b, \\ 12y^4 + \sqrt[3]{x} & \text{jeśli } b \leq x^2 + y^2, \end{cases}$$

gdzie:

$$x = \ln c + \cos(a+b), \quad y = \sqrt{|ax^3 + bx + c|}.$$

36. Dane są x, y - liczby rzeczywiste. Obliczyć:

$$f(x, y) = \begin{cases} (x^2 + y^3 x)(x+y) & \text{jeśli } x \neq y, \\ (y - x^4 y)(y^3 + x) & \text{w przeciwnym przypadku.} \end{cases}$$

37. Dane są a, b, c - liczby rzeczywiste, m - liczba całkowita, ($1 \leq m \leq 5$). Obliczyć

$$S = \begin{cases} \max(a, b, c) & \text{jeśli } m=1, \\ \min(a, b, c) & \text{jeśli } m=2, \\ \frac{a+b+c}{3} & \text{jeśli } m=3, \\ a^2 + b^2 + c^2 & \text{jeśli } m=4, \\ |a| + |b| + |c| & \text{jeśli } m=5. \end{cases}$$

38. Dana jest liczba naturalna n oraz liczba rzeczywista x . Obliczyć:

$$S = \sum_{i=1}^n x^i \quad (\text{bez użycia potęgowania}).$$

39. Dana jest liczba naturalna n . Obliczyć sumę

$$\sum_{i=1}^n \frac{1}{2^i}$$

40. Dana jest liczba naturalna n oraz liczba rzeczywista x .
Napisz program obliczający sumę:

$$\sum_{i=0}^n \frac{x^i}{i!}$$

raz od początku a raz od końca. Porównać otrzymane wyniki. Który sposób jest lepszy? Dlaczego?

41. Dana jest liczba naturalna n oraz liczba rzeczywista x .
Obliczyć

$$S = \begin{cases} n!x & \text{jeśli } x < 0, \\ \sum_{i=1}^n x^i & \text{jeśli } x \geq 0. \end{cases}$$

42. Dla danej liczby naturalnej n oraz liczby rzeczywistej x obliczyć

$$G(x, n) = \begin{cases} \sum_{i=1}^n \frac{x+i}{i!} & \text{jeśli } x \leq 0, \\ \prod_{i=1}^n \frac{x+i}{i} & \text{jeśli } x > 0. \end{cases}$$

43. Dane są: n - liczba naturalna, x - liczba rzeczywista. Obliczyć

$$F_n(x) = \frac{n}{3} x^n + \frac{n-1}{3} x^{n-1} + \dots + \frac{1}{3} x + 1.$$

44. Dla danych liczb naturalnych m, n oraz zmiennej rzeczywistej y , która przyjmuje wartości $a+i \cdot h$, gdzie a, h - dane liczby rzeczywiste, $i=0, 1, \dots, n$, obliczyć i wydrukować wartości

$$W_m(y) = 1 - \frac{1}{2} y + \frac{2}{3} y^2 - \frac{3}{4} y^3 + \dots + (-1)^m \frac{m}{m+1} y^m.$$

45. Dane są liczby rzeczywiste x , eps . Obliczyć

$$S = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

z dokładnością eps (tzn. obliczenia zakończyć, jeśli

$$|S - e| < \text{eps}, \text{ gdzie } e \approx 2.71).$$

46. Obliczyć

$$\frac{x}{2} - \frac{x^2}{3} + \frac{x^3}{4} - \frac{x^4}{5} + \dots + (-1)^{n+1} \frac{x^n}{n+1}$$

dla $x=2,5,8,11,14,17,20$ oraz dowolnej liczby naturalnej n .47. Dane są liczby rzeczywiste x, ε . Obliczyć przybliżoną wartość $\sin(x)$ ze wzoru

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + \frac{x^{i+1}}{(i+1)!} \pm \dots$$

Obliczenia należy zakończyć, gdy

$$\varepsilon * |S_i| < |t_i|,$$

gdzie

$$t_i = \frac{x^{2i+1}}{(2i+1)!}, \quad S_i = \sum_{k=0}^i (-1)^k t_k.$$

48. Dane są liczby rzeczywiste x, eps . Obliczyć przybliżoną wartość $\cos(x)$ ze wzoru

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

Obliczenia należy zakończyć, gdy

$$t_i < \text{eps}, \text{ gdzie } t_i = \frac{x^i}{i!} \quad (i=2,4,6,\dots).$$

49. Dla danej liczby rzeczywistej $x > 0$ obliczyć przybliżoną wartość całek:

$$a) \int_0^x e^{-u^2} du = 1 - \frac{x^3}{3*1!} + \frac{x^5}{5*2!} - \frac{x^7}{7*3!} + \dots$$

$$b) \frac{\int_0^x e^{-u^2} du}{x} = x^{-1} - \frac{x^2}{3*1!} + \frac{x^4}{5*2!} - \frac{x^6}{7*3!} + \dots$$

Obliczenia należy zakończyć, gdy

$$\varepsilon * |S_i| \leq |t_i|,$$

gdzie:

 t_i - ostatnio obliczony wyraz ciągu, S_i - ostatnio obliczona przybliżona wartość całki.

50. Opracować program, który liczy

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{9999} - \frac{1}{10000}$$

- a) z lewa na prawo,
- b) z prawa na lewo,
- c) oddzielnie dodatnie i ujemne z lewa na prawo,
- d) oddzielnie ujemne i dodatnie z prawa na lewo.

Wyniki porównać.

51. Dana jest liczba naturalna $n > 0$ oraz liczba rzeczywista x .
Obliczyć wartość funkcji $F_n(x)$ danej wzorem:

$$F_n(x) = \begin{cases} x + 1 & \text{jeśli } n=1, \\ 2x^2 + x + 1 & \text{jeśli } n=2, \\ 3x^3 + 2x^2 + x + 1 & \text{jeśli } n=3, \\ 4x^4 + 3x^3 + 2x^2 + x + 1 & \text{jeśli } n=4, \\ 5x^5 + 4x^4 + 3x^3 + 2x^2 + x + 1 & \text{jeśli } n=5, \\ 0 & \text{jeśli } n \geq 6. \end{cases}$$

52. Napisać procedurę, która dla danych liczb całkowitych a , b , c wyznacza $\max(a, b, c)$.

53. Jakie będą wartości zmiennych d oraz p po wykonaniu programu:

```
program fun;
var z,y,d,p: real;

function f(var x: real): real;
begin
  x:=x+10.0;
  f:=0.0
end (f);

begin
  z:=5.0;
  y:=5.0;
  d:=f(z)+z;
  p:=y+f(y);
  writeln('d=',d:5:1,'   z=',z:5:1);
  writeln('p=',p:5:1,'   y=',y:5:1)
end.
```

54. Dany jest program

```
program zabawa;
var a,b: real;

procedure xyz (var x,y,z: real);
begin
  x:=1.0;
  y:=2.0;
  z:=3.0;
end {xyz};

begin
  xyz(a,b,a);
  writeln('a=',a:5:2);
  writeln('b=',b:5:2)
end.
```

Jakie wartości zostaną wydrukowane?

55. Porównać dane cztery programy. Co zostanie wydrukowane w każdym przypadku?

1. program p1;

```
var x: integer;

procedure zmiana1;
begin
  x:=1
end {zmiana1};

begin
  x:=0;
  zmiana1;
  write(x)
end.
```

2. program p2;

```
var x: integer;

procedure zmiana2;
var x: integer;
begin
  x:=1
end {zmiana2};

begin
  x:=0;
  zmiana2;
```



```

    write(x)
end.

```

```

3. program p3;
   var x: integer;

   procedure zmiana3(var x: integer);
   begin
       x:=1;
   end {zmiana3};

   begin
       x:=0;
       zmiana3(x);
       write(x)
   end.

```

```

4. program p4;
   var x: integer;

   procedure zmiana4(x: integer);
   begin
       x:=1
   end {zmiana4};

   begin
       x:=0;
       zmiana4(x);
       write(x)
   end.

```

56. Jaka wartość zostanie wydrukowana?

```

program sprawdzian;
var myśl: integer;

procedure oszust(var hi,ha: integer);
begin
    hi:=-1;
    ha:=-hi
end;

begin
    myśl:=1;
    oszust(myśl,myśl);
    writeln(myśl)
end.

```

57. Jaka wartość zostanie wydrukowana?

```
program hiha;
var myśl: integer;

procedure kłamca(var hi: integer; ha: integer);
begin
    hi:=10*ha
end {kłamca};

begin
    myśl:=10;
    kłamca(myśl,myśl);
    writeln(myśl)
end.
```

58. Dane są liczby rzeczywiste $a>0$, $h>0$.

Funkcję f stablicować w przedziale $[-a,a]$ z krokiem h , gdzie

$$f(x) = \begin{cases} |x| & \text{jeśli } x \leq 0, \\ \ln x & \text{jeśli } x > 0. \end{cases}$$

59. Napisać procedurę, która dzieli przedział $[a,b]$ na n podprzedziałów $p_1, \dots, p_n$ jednakowej długości i sprawdza, dla których k ($k=1, \dots, n$) ustalona funkcja $f(x)$ ma różne znaki na początku i końcu podprzedziału p_k .

60. Funkcja (ciągła) $G(x)$ przyjmuje na końcach przedziału $[x_p, x_k]$ różne znaki (ma zatem co najmniej jeden punkt zerowy w tym przedziale). Metodą dzielenia przedziału na połowy zlokalizować miejsce zerowe ustalonej funkcji $G(x)$ z dokładnością ϵ .

61. Napisać procedurę wyznaczania miejsca zerowego funkcji $f(x)$ metodą Newtona:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

gdzie x_0 - dane przybliżenie początkowe.

Obliczenia należy zakończyć, gdy

$$|f(x_{i+1})| < \epsilon.$$

Wykorzystując tę procedurę wyznaczyć miejsce zerowe funkcji:

$$f(x) = \cos(x) - x * e^x.$$

Przyjąć $x_0 = 0$, $\epsilon = 10^{-2}$.

62. Rozwiązać metodą stycznych równanie

$$f(x) = x^2 - \sin(x) - 1 = 0.$$

Metoda:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \text{dla } n=0,1,\dots,$$

gdzie

x_0 - dane przybliżenie początkowe (przyjąć $x_0 = 1.4$).

Obliczenia należy zakończyć, gdy

$$|x_{n+1} - x_n| < \text{eps}.$$

63. Znaleźć miejsce zerowe funkcji $f(x)$ ze wzorów:

x_0 - dane,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{(f(x_n))^2 \cdot f''(x_n)}{2 \cdot (f'(x_n))^3}.$$

Obliczenia należy zakończyć, gdy $|x_{n+1} - x_n| < \varepsilon$.

Przyjąć:

a) $x_0 = \pi$, $f(x) = \sin(x) - \frac{1}{2}x$, $\varepsilon = 10^{-4}$,

b) $x_0 = 1$, $f(x) = \cos(x) - xe^x$, $\varepsilon = 10^{-2}$.

64. Napisać procedurę rozwiązywania równania kwadratowego.

65. Dane: $n > 2$, $r > 0$. Obliczyć

$$W = \prod_{k=0}^n \left(1 + \frac{1}{a_k}\right),$$

gdzie:

$$a_0 = a_1 = \frac{1}{2},$$

$$a_k = a_{k-1} + r \cdot a_{k-2} \quad \text{dla } k=2,3,\dots,n.$$

66. Dany jest ciąg rosnący i ograniczony zadany rekurencyjnie

$$u_0 = 0,$$

$$u_k = \frac{u_{k-1} + 1}{u_{k-1} + 2} \quad \text{dla } k=1,2,\dots$$

Znaleźć pierwszy wyraz u_n spełniający warunek

$$u_n - u_{n-1} < \varepsilon,$$

gdzie $\varepsilon > 0$.

67. Obliczyć pierwiastek sześcienny a z liczby x wykorzystując do tego celu wzór iteracyjny Newtona

$$a_{i+1} = \frac{1}{3} \left(2a_i + \frac{x}{a_i^2} \right) \quad \text{dla } i=0,1,\dots,n,$$

gdzie

a_0, ε, n - dane.

Sygnałem zakończenia obliczeń będzie albo zrealizowanie żądanej dokładności ε obliczenia wartości pierwiastka sześciennego, albo wykonanie zadanej maksymalnej liczby iteracji n .

68. Dla danych liczb rzeczywistych $a, \varepsilon > 0$ oraz liczby naturalnej k obliczać kolejne przybliżenia x_i ze wzorów:

$$x_0 = a, \quad x_i = \frac{k-1}{k} x_{i-1} + \frac{a}{x_{i-1}^{k-1}} \quad \text{dla } i=1,2,\dots$$

tak długo, aż $|x_i^k - a| \leq \varepsilon$.

Ostatnie obliczone przybliżenie wydrukować (ciąg $x_0, x_1, \dots$ jest zbieżny do $\sqrt[k]{a}$).

69. Obliczyć $\frac{\sqrt{2}}{2}$ ze wzoru iteracyjnego

$$x_1 = 1,$$

$$x_{k+1} = \left(\frac{3}{2} - x_k^2 \right) \quad \text{dla } k=1,2,\dots$$

70. Dane są:

M - liczba naturalna,

a, b - liczby rzeczywiste takie, że $1 < a < b$,

$$f_1 = b,$$

$$g_1 = a,$$

$$f_{n+1} = \frac{1}{2} (f_n + g_n), \quad g_{n+1} = \frac{2f_n g_n}{f_n + g_n}.$$

Sprawdzić dla $n=0,1,\dots,M$ prawdziwość nierówności

$$0 < f_{n+1} - g_{n+1} < \frac{(b-a)^{2^n}}{2^n}.$$

71. Napisać program testujący prawdziwość nierówności

$$\frac{1}{n+1} < \ln\left(\frac{n+1}{n}\right) < \frac{1}{n},$$

gdzie n jest dowolną liczbą naturalną.

72. Napisać w języku Pascal definicję następujących funkcji:

$$\begin{aligned} \text{a) } \sinh(x) &= (e^x - e^{-x})/2, \\ \cosh(x) &= (e^x + e^{-x})/2, \\ \tgh(x) &= \sinh(x)/\cosh(x), \end{aligned}$$

$$\begin{aligned} \text{b) } \sinh^{-1}(x) &= \ln(x + \sqrt{x^2 + 1}), \\ \cosh^{-1}(x) &= \ln(x + \sqrt{x^2 - 1}). \end{aligned}$$

73. Napisać procedurę, która zamienia wartości dwóch zmiennych rzeczywistych, np. jeśli przed wywołaniem procedury

$$x = -7.5, y = 13.4,$$

to po wywołaniu

$$x = 13.4, y = -7.5.$$

74. Niech

type nieujemne = 0..maxint;

Co oblicza następująca funkcja rekurencyjna?

```
function f(n: nieujemne): integer;
begin
  if n=0 then f:=0
  else f:=n+f(n-1)
end {f};
```

Napisać iteracyjną funkcję odpowiadającą funkcji rekurencyjnej f.

75. Obliczyć wartość funkcji

$$P_n(x) = \begin{cases} 1 & \text{jeśli } n=0, \\ x & \text{jeśli } n=1, \\ \frac{(2n-1)P_{n-1}(x) - (n-1)P_{n-2}(x)}{n} & \text{jeśli } n>1. \end{cases}$$

76. Obliczyć wartość funkcji Ackermanna ($m \geq 0, n \geq 0$)

$$\begin{aligned} \text{Ack}(0, n) &= n+1, & \text{jeśli } m=0 \text{ i } n \geq 0, \\ \text{Ack}(m, 0) &= \text{Ack}(m-1, 1), & \text{jeśli } m > 0 \text{ i } n=0, \\ \text{Ack}(m, n) &= \text{Ack}(m-1, \text{Ack}(m, n-1)) & \text{jeśli } m > 0 \text{ i } n > 0. \end{aligned}$$

77. Dane są kolejne liczby naturalne z przedziału $[n_p, n_k]$. Sprawdzić, które z tych liczb są liczbami pierwszymi, a które nie. Wydrukować odpowiedni tekst przy każdej liczbie.

78. Napisać program rozmieniający pewną kwotę pieniędzy (do 99 centów) za pomocą banknotów o wartości:

1, 5, 10 i 25 centów.

Uwaga. Zapisujemy za pomocą banknotów o największej wartości.

79. Napisać funkcję o wartości logicznej, która sprawdza, czy dane trzy punkty są współliniowe.

80. Wydrukować napis

Dzisiaj jest

(w miejscu kropek wydrukować odpowiedni dzień tygodnia) w zależności od wartości zmiennej całkowitej dz ($1 \leq dz \leq 7$) oznaczającej dzień tygodnia (1 – poniedziałek, ..., 7 – niedziela).

81. Dana jest liczba naturalna n ($n \leq 100$), określająca wiek pewnej osoby (w latach). Opracować program, który podaje ten wiek z odpowiednim napisem "rok", "lata" lub "lat", np.

1 rok

3 lata

18 lat

43 lata

Zmodyfikować program tak, aby można było go uruchomić dla dowolnej liczby zestawów danych.

82. Napisać program, który czyta n znaków i dla każdego znaku drukuje napis

'cyfra', 'mała litera', 'wielka litera' lub 'inny znak' w zależności od podanego znaku.

83. Za pomocą n różnych znaków wydrukować trójkąty następującej postaci (m – dana liczba naturalna)

a)

```

      *
     **
    ***
   ****
  } m wierszy

```

b)

```

                        *
                       **
                      ***
                     ****
                    *****
                   ******
                  *******
                 *******
                *******
               *******
              *******
             *******
            *******
           *******
          *******
         *******
        *******
       *******
      *******
     *******
    *******
   *******
  *******
 } m wierszy

```

84. Napisać program, który drukuje znak * przesuwający się z lewej strony ekranu do prawej.

85. Wydrukować formularz na wpisanie ocen z przedmiotów dla ucznia klasy I postaci

Nazwisko i imię ucznia	
1. Sprawowanie	
2. Język polski	
3. Matematyka	
4. Środowisko	
5. Muzyka	
6. Wych. plastyczne	
7. Praco - technika	
8. Wych. fizyczne	

podpis wychowawcy	

Zmodyfikować program tak, aby można było drukować również odpowiednie nazwisko i imię ucznia oraz (w miejscu kropek) jego oceny uzyskane w czasie semestru.

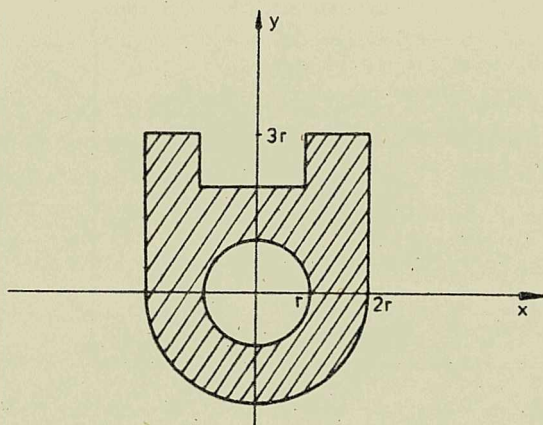
86. Wydrukować układ współrzędnych.

87. Napisać program drukujący zawiadomienie o zebraniu następującej treści:

Wrocław, 5.05.1988	
ZAWIADOMIENIE	
Uprzejmie zawiadamiamy	
ob.	JANA NOWAKA
że dnia	28 maja 1988 roku
o godz.	17.15
	w budynku D-2
w sali	301
	odbędzie się
zebranie	członków PTI
	Adam Paluch
	sekretarz PTI

Słowa wykropkowane (kropek nie trzeba drukować) mogą być zmieniane (powinny być wprowadzane jako dane).

88. Na płaszczyźnie danych jest n punktów o współrzędnych (x_i, y_i) , gdzie $i=1, \dots, n$. Wydrukować współrzędne punktów leżących wewnątrz obszaru zakreskowanego lub na jego brzegu (r - dana liczba rzeczywista dodatnia).



5. TYP TABLICOWY

Pojęcie tablic w językach programowania wywodzi się z pojęć matematycznych. W matematyce zbiór zmiennych o wartościach liczbowych $a_1, a_2, \dots, a_n$ jest traktowany jako wektor n -wymiarowy lub jako ciąg n -elementowy, natomiast w językach programowania taka struktura jest nazywana tablicą. W języku Turbo Pascal 3.0 tablicę n -elementową o wartościach rzeczywistych można zadeklarować następująco:

```
var a: array[1..n] of real;
```

Pojęcie tablicy rozszerza się następnie na regularne układy zmiennych z dwoma wskaźnikami, otrzymując w ten sposób strukturę analogiczną do macierzy. Na przykład zbiór liczb rzeczywistych

$$B = \{b_{ij} : i=1, \dots, n; j=1, \dots, m\}$$

w języku Turbo Pascal można zadeklarować jako dwuwymiarową tablicę B

```
var B: array[1..n, 1..m] of real;
```

Wszystkie elementy tablicy B są typu real.

Dostęp do poszczególnych elementów tablic uzyskuje się przez użycie indeksów, np. $a[i]$, $B[k+j, 3]$.

Wartości indeksów są obliczane w chwili odwołania do tablicy, a czas dostępu do elementu tablicy jest dla każdego elementu tablicy taki sam. Z tego względu typ tablicowy jest typem strukturalnym o dostępie bezpośrednim.

Dwuwymiarowa tablica A zadeklarowana jako

```
var A: array[1..n, 1..m] of real;
```

może być traktowana jako jednowymiarowa tablica, której każdy element A_i jest jednowymiarową tablicą, tzn.

```
var A: array[1..n] of array[1..m] of real;
```

Ten sposób deklaracji jest polecany wówczas, gdy w programie chcemy mieć dostęp do całych wierszy tablicy. Przy takiej deklaracji tablicy A do jej elementów można odwoływać się następująco:

$A[i]$ - dostęp do i -tego wiersza,

$A[i][j]$ lub $A[i, j]$ - dostęp do elementu w i -tym wierszu i w j -tej kolumnie.

Przykład 5.1

Założmy, że w pewnym programie wystąpiły następujące deklaracje:

```
const m=3; n=4;
var a: array[1..m,1..n] of integer;
    b: array[1..n] of integer;
```

Wówczas instrukcja przypisania

```
b:=a[1]
```

będzie sygnalizowana jako błędna, natomiast będzie ona dozwolona jeśli napiszemy np.

```
const m=3; n=4;
type wiersz=array[1..n] of integer;
var a: array[1..m] of wiersz;
    b: wiersz;
```

Wszystkie elementy zmiennej tablicowej są tego samego ustalonego typu. Ogólna postać deklaracji zmiennej tablicowej (tablicy) ma postać:

```
type nazwa_typu_tablicowego=array[T1,...,Tn] of T;
var nazwa_zmiennej_tablicowej: nazwa_typu_tablicowego;
```

lub

```
var nazwa_zmiennej_tablicowej: array[T1,...,Tn] of T;
```

gdzie

T1,...,Tn – nazwy lub opisy typów porządkowych (z wyjątkiem typu integer) określające typ indeksów,

T – dowolny typ określający typ elementów.

Tablice jednowymiarowe o elementach typu char i indeksach o wartościach całkowitych nazywa się tablicami znakowymi.

Przykład 5.2

Niech

```
var drzewo: array[1..7] of char;
    kwiat: array[1..9] of char;
```

Poprawnymi przypisaniami są, np.

```
drzewo:='lipa';
kwiat:='stokrotka'
```

Natomiast przypisanie

```
drzewo:='klon';
kwiat:='irys'
```

są niepoprawne, ponieważ w apostrofach jest za mało znaków (liczba znaków w apostrofach powinna być równa liczbie znaków danej tablicy).

W programach tablice można wykorzystać na przykład do zapisu odwzorowań określonych na zbiorze n -elementowym.

Przykład 5.3

Relacja ρ określona na zbiorze n -elementowym $a_1, a_2, \dots, a_n$ jest zapisana w postaci macierzy $m \times$ stopnia n (tzn. m jest macierzą kwadratową o n wierszach i n kolumnach, co zapisujemy umownie $m \times n \times n$) następująco:

$$m[i, j] = \begin{cases} 1 & \text{jeśli } a_i \rho a_j, \\ 0 & \text{w przeciwnym przypadku,} \end{cases}$$

gdzie $i, j = 1, \dots, n$.

Należy sprawdzić, czy relacja ρ jest częściowym porządkiem, tzn. czy jest

- zwrotna,
- antysymetryczna,
- przechodnia.

Przypomnijmy, że jeśli $\rho: D \rightarrow D$, to

- relacja jest zwrotna $\Leftrightarrow \forall a \in D, a \rho a$,
- relacja jest antisymetryczna $\Leftrightarrow \forall a, b \in D, a \rho b \wedge b \rho a \Rightarrow a = b$,
- relacja jest przechodnia $\Leftrightarrow \forall a, b, c \in D, (a \rho b \wedge b \rho c) \Rightarrow a \rho c$.

Przykładem relacji, która jest częściowym porządkiem może być relacja " $\leq$ " określona na zbiorze liczb rzeczywistych.

Poniżej przedstawiamy program sprawdzający, czy dana zerojedynkowa tablica MX opisuje częściowy porządek.

```
program relacja;
const nmax=5;
type tablica=array[1..nmax,1..nmax] of integer;
var MX: tablica;
    i,j,n: integer;

function czporz(A: tablica; n: integer): boolean;
{zmienną czporz zostanie nadana wartość true, jeśli relacja
określona za pomocą tablicy A jest częściowym porządkiem,
```

```

    natomiast false w przeciwnym przypadku)
var i,i1,j,k: integer;
    p: boolean;
begin
    p:=true;
        {test na zwrotność}
    k:=1;
    while p and (k<=n) do
        if A[k,k]=0 then p:=false else k:=k+1;
            {test na antysymetryczność i przechodność}
    i:=2;
    while p and (i<=n) do
        begin
            j:=1;
            i1:=i-1;
            while p and (j<=i1) do
                if A[i,j]*A[j,i1]=1
                    then p:=false
                    else
                        begin
                            if A[i,j]=0 then
                                begin
                                    k:=1;
                                    while p and (k<=n) do
                                        if A[i,k]*A[k,j]=1 then p:=false else k:=k+1
                                    end {A[i,j]=0};
                                    if A[j,i1]=0 then
                                        begin
                                            k:=1;
                                            while p and (k<=n) do
                                                if A[j,k]*A[k,i1]=1 then p:=false else k:=k+1
                                            end {A[j,i1]=0};
                                            j:=j+1
                                        end {else};
                                    i:=i+1
                                end {i};
                                czporz:=p
                            end {czporz};
                        begin
                            clrscr;
                            write('n='); readln(n);
                            writeln(' Podaj elementy macierzy:');

```



```

for i:=1 to n do
  for j:=1 to n do readln (MX[i,j]);
writeln;
writeln(' Macierz MX:');
for i:=1 to n do
begin
  for j:=1 to n do write(MX[i,j]:3);
  writeln
end {i};
writeln;
write(' Relacja określona macierzą MX');
if not czporz(MX,n) then write(' nie');
writeln(' jest częściowym porządkiem')
end.

```

Na zakończenie tego rozdziału podamy przykład ilustrujący wykorzystywanie menu w programach oraz programowanie konwersacji.

Przykład 5.4

Napiszemy program, który umożliwi wykonywanie następujących zadań:

1. czytanie wymiarów i elementów macierzy,
2. drukowanie macierzy,
3. drukowanie wybranego wiersza macierzy,
4. obliczanie iloczynu skalarnego dwóch wybranych wierszy macierzy,
5. zakończenie wykonywania programu.

Program przed wykonaniem zadań 2 + 4 powinien sprawdzać, czy została wcześniej wczytana macierz. Ponadto, w przypadku 3 i 4, powinien sprawdzać, czy podane numery wierszy są odpowiednie.

```

program macierze;
const nmax=10;
      mmax=10;
      spacje='      ';
type wiersz=array[1..mmax] of integer;
      tablica=array[1..nmax] of wiersz;
var A: tablica;
      i,j,m,n,numer,nrwiersza: integer;
      znak: char;
      dozwolone: boolean;

procedure czytaj(var X: tablica; var n,m: integer);
var i,j: integer;

```

```

begin
  clrscr;
  writeln(spacje,'Podaj wymiary macierzy:');
  write(spacje,' n= '); readln(n);
  write(spacje,' m= '); readln(m);
  writeln(spacje,'Podaj elementy macierzy (wierszami):');
  writeln;
  for i:=1 to n do
  begin
    for j:=1 to m do
    begin
      read(X[i,j]); write(' ');
    end (j);
    writeln
  end (i)
end {czytaj};

procedure drukuj(X: tablica; n,m: integer);
var i,j: integer;
begin
  clrscr;
  writeln('Elementy macierzy:');
  for i:=1 to n do
  begin
    writeln;
    for j:=1 to m do write(X[i,j]:5)
  end (i)
end {drukuj};

procedure drukuj_wiersz(X: wiersz; m: integer);
var j: integer;
begin
  for j:=1 to m do write(X[j]:4);
  writeln
end {drukuj_wiersz};

function iloczyn_sk(x,y: wiersz; n: integer): integer;
var i,pom: integer;
begin
  pom:=0;
  for i:=1 to n do
    pom:=pom+x[i]*y[i];
  iloczyn_sk:=pom
end {iloczyn_sk};

```

```

begin
  dozwolone:=false;
  repeat
    clrscr;
    gotoxy(10,5);
    writeln('Menu programu:');
    writeln(spacje,
      '1. Czytanie danych (wymiary i elementów macierzy).');
    writeln(spacje,'2. Drukowanie całej macierzy');
    writeln(spacje,'3. Drukowanie wybranego wiersza macierzy');
    write(spacje,
      '4. Obliczanie iloczynu skalarnego dwóch wybranych wierszy');
    writeln(spacje,'5. Koniec programu');
    writeln;
    write(spacje,'Wybierz dowolny numer od 1 do 5: ');
    readln(number);
    case number of
      1: begin
          dozwolone:=true;
          czytaj(A,n,m);
          end {number=1};
      5: begin
          writeln;
          writeln(spacje,spacje,' G O O D B Y E');
          halt {zakończenie wykonywania programu}
          end {number=5}
    else
      if dozwolone then
        case number of
          2: drukuj(A,n,m);
          3: begin
              write(spacje, 'Podaj numer wiersza: ');
              readln(nrwersza);
              if (0<nrwersza) and (nrwersza<=n) then
                begin
                  writeln(' Elementy ',nrwersza,'-go wiersza');
                  drukuj_wiersz(A[nrwersza],m)
                end
              else writeln('Podałeś zły numer wiersza')
            end {number=3};
          4: begin
              writeln;

```

```

write(spacje,'Podaj numery wierszy, dla których');
writeln(' chcesz obliczyć iloczyn skalarny');
write(spacje,' k= '); readln(nrwiersza);
write(spacje,' l= '); readln(i);
if (0<nrwiersza) and (nrwiersza<=n)
  and (0<i) and (i<=n) then
  begin
    writeln(spacje,'Elementy ',nrwiersza,
      '-go wiersza:');
    drukuj_wiersz(A[nrwiersza],m);
    writeln(spacje,'Elementy ',i,'-go wiersza:');
    drukuj_wiersz(A[i],m);
    writeln(' iloczyn skalarny wybranych wierszy= ',
      iloczyn_sk(A[nrwiersza],A[i],m))
  end {then}
  else writeln('Podałeś złe numery')
end {number=4}
end {case}
else writeln(spacje,'Podałeś zły numer')
end {case};
writeln;
write('Czy zakończyć program (y/n)? ');
readln(znak)
until znak='y'
end.

```

5.1. Zadania

Uwaga. Wiele zadań z tego rozdziału dotyczy ciągów, wektorów i macierzy. W każdym przypadku należy zastanowić się, czy w programie jest konieczne użycie zmiennych typu tablicowego. Wybór odpowiedniej struktury danych należy do Czytelnika.

1. Kiedy używać deklaracji (a) a kiedy deklaracji (b)? Odpowiedź uzasadnij.

- a) var x: array[1..n,1..m] of real;
- b) var y: array[1..n] of array[1..m] of real;

2. Które z poniższych deklaracji są błędne? Dlaczego?

- a) wektor: array[1..2] of real;
- b) ceny: array[1..10] of 15.0..72.5;
- c) oceny: array[5..2] of integer;
- d) test: array[boolean] of true..false;

3. Ile elementów ma tablica a? Jak prościej ją zadeklarować?

```
const n=3;
      m=3;
var a: array[-3..3,0..0,n..m] of integer;
```

4. Zadeklarować zmienne tablicowe jednowymiarowe

- a) o wskaźnikach logicznych i elementach rzeczywistych,
- b) o wskaźnikach typu kolor i elementach całkowitych, gdzie
type kolor=(pik, trefl, karo, kier);
- c) o wskaźnikach typu znakowego i elementach typu logicznego.

5. Tablice a i b zadeklarowano następująco:

```
var x: array[1..5] of array[1..5] of real;
    y: array[1..5,1..5] of real;
```

Które z podanych instrukcji są poprawne?

- a) read(x);
- b) y:=x;
- c) read(x[1]);
- d) read(y[1]);
- e) x[1]:=0.0;
- f) x[2,2]:=5.5;
- g) y[3][3]:=4;
- h) x[2][3]:=y[3][1];

6. Której z podanych instrukcji należy użyć do obliczania wartości zmiennej x? Dlaczego?

- a) x:=(a[i]+sin(a[i]))/(a[i]+2.5)
- b) begin
 ai:=a[i];
 x:=(ai+sin(ai))/(ai+2.5)
 end

7. Niech

```
var a: array[1..20] of integer;
    b: array[1..50] of integer;
```

Jakie będą wartości elementów tablicy b po wykonaniu podanych instrukcji? Czy możesz zaproponować lepszy sposób obliczenia tych wartości?

```
a) for i:=1 to 20 do
      for k:=1 to 50 do
        b[k]:=b[k]+a[i]
```

```
b) for i:=1 to 20 do
      begin
        ai:=a[i];
        for k:=1 to 50 do
          b[k]:=b[k]+ai
        end (i)
```

8. Co robi dany fragment programu?

```
if x<>0
then begin
  y:=a[1];
  for i:=2 to n do
    y:=y*x+a[i]
  end {x<>0}
else y:=a[n]
```

9. W danym n -elementowym ciągu $x_1, x_2, \dots, x_n$ zamienić parami elementy o numerach

- 1 i n ,
- 2 i $n-1$.

Wydrukować ciąg wejściowy i wyjściowy.

10. Dany jest n -elementowy ciąg a oraz liczba x . Wydrukować:

- a) liczbę elementów w ciągu a o wartości x ,
- b) numery elementów w ciągu a o wartości x .

11. Danych jest n liczb rzeczywistych. Obliczyć, ile jest liczb dodatnich i ich sumę oraz ile jest liczb niedodatnich i ich średnią arytmetyczną.

12. Na wejściu dane są liczby całkowite. Wczytywać je i dodawać tak długo, aż

- a) pojawi się liczba 0,
- b) pojawi się liczba mniejsza od 0.

13. Dany jest n -elementowy ciąg $a_1, a_2, \dots, a_n$. Znaleźć liczbę elementów ujemnych, zerowych i dodatnich ciągu.

14. Ciąg $a_1, a_2, \dots, a_n$ posegregować na liczby ujemne $u_1, u_2, \dots, u_p$ i nieujemne $v_1, v_2, \dots, v_q$ ($p+q=n$).

15. Dla dowolnego n -elementowego ciągu liczb całkowitych napisać program wyznaczający:

- sumę elementów ciągu, jeśli co najmniej jeden z wyrazów ciągu ma wartość 0,
- iloczyn elementów, w przeciwnym przypadku.

16. Dana jest liczba naturalna parzysta n oraz ciąg $a_1, a_2, \dots, a_n$. Obliczyć i wydrukować:

$$S_p = a_2 + a_4 + a_6 + \dots + a_n,$$

$$S_{np} = a_1 + a_3 + a_5 + \dots + a_{n-1}.$$

17. Znaleźć minimum ciągu n -elementowego.

18. Dany jest n -elementowy ciąg $a_1, a_2, \dots, a_n$. Obliczyć

$$\max_{1 \leq i \leq n} |a_i| * \sum_{i=1}^n a_i * \prod_{i=1}^n (a_i + 1).$$

19. Dla danego n -elementowego ciągu $a_1, a_2, \dots, a_n$ obliczyć i wydrukować

$$(\max_{1 \leq i \leq n} |a_i| - \min_{1 \leq i \leq n} a_i) * \prod_{i=1}^n a_i.$$

20. Dane: $n, a_1, a_2, \dots, a_n$. Obliczyć

$$\min \left\{ \sum_{i=1}^{n-1} \left| \frac{a_i}{a_n} \right|, \prod_{i=1}^n a_i \right\}.$$

21. Dane są liczby całkowite: n, k ($n, k > 0$), $a_1, a_2, \dots, a_n$, $b_1, b_2, \dots, b_k$. Obliczyć

$$S = \sqrt{\prod_{i=1}^n a_i} + \max_{1 \leq i \leq k} b_i.$$

22. Dla danego n -elementowego ciągu a obliczyć i wydrukować

$$\max_{1 \leq i \leq n} |a_i| * \sum_{i=1}^n a_i.$$

Ponadto znaleźć liczbę k taką, że

$$|a_k| = \max_{1 \leq i \leq n} |a_i|.$$

Jeśli jest kilka takich elementów, to wydrukować indeks ostatniego z nich.

23. Dane są wartości całkowite: $n, a_1, a_2, \dots, a_n; m, b_1, b_2, \dots, b_m$.

Obliczyć

$$\min(m, n) \sum_{i=1}^{\min(m, n)} (a_i - b_i) * \max_{1 \leq i \leq \min(m, n)} (a_i - b_i).$$

24. Znaleźć maksimum ciągu n -elementowego oraz

- wskaźnik pierwszego elementu maksymalnego,
- wskaźnik ostatniego elementu maksymalnego,
- wskaźniki wszystkich elementów maksymalnych.

25. Dane są zarobki n pracowników. Obliczyć i wydrukować średni zarobek oraz liczbę pracowników zarabiających poniżej tej średniej.

26. Dla lat 1950 do 1987 podane są średnie temperatury roczne. Podać ostatni najcieplejszy rok i jego temperaturę.

27. Dane są liczby rzeczywiste $q_1, q_2, \dots, q_n$ oznaczające wydajności zbiorów pszenicy z n pól. Obliczyć średnią wydajność dwoma sposobami

$$1) S1 = \frac{1}{n} \sum_{i=1}^n q_i,$$

$$2) S2 = \frac{1}{n-2} \left(\sum_{i=1}^n q_i - \max_{1 \leq i \leq n} q_i - \min_{1 \leq i \leq n} q_i \right).$$

Porównać otrzymane wyniki.

28. W danym n -elementowym ciągu przestawić jego wyrazy tak, by elementy o wartościach parzystych znalazły się na początku.

29. Danych jest n liczb całkowitych $a_1, a_2, \dots, a_n$. Posegregować te liczby na parzyste i nieparzyste. Podać liczbę elementów parzystych i nieparzystych.

30. Dany jest n -elementowy ciąg $a_1, a_2, \dots, a_n$ liczb całkowitych z przedziału $[0, m-1]$. Zastosować algorytm koszykowy do uporządkowania wartości elementów ciągu a w ciąg rosnący b , przy czym elementy powtarzające się wydrukować tylko raz.

Algorytm koszykowy:

- (1) $b_j = 0$ dla $j=0, 1, \dots, m-1$,
- (2) $b_{a_i} = b_{a_i} + 1$ dla $i=1, 2, \dots, n$,
- (3) drukować te wskaźniki j , dla których $b_j > 0$ ($j=0, 1, \dots, m-1$).

31. Powtarzające się elementy z zadania 30 drukować tyle razy, ile razy występują.

32. Dane są dwa wektory n -wymiarowe. Za pomocą instrukcji:

- a) **for**,
- b) **while**,
- c) **repeat**

napisać program obliczania ich iloczynu skalarnego.

33. Jakie będą wartości elementów w tablicy a po wykonaniu poniższego fragmentu programu?

```
var i,j,n: integer;
    a: array[1..10,1..10] of real;
begin
    read(n);
    for i:=1 to n do
        for j:=1 to n do
            a[i,j] := (i div j) * (j div i);
```

34. W tablicy $PLACA$ dane są zarobki n pracowników ($n \leq 50$). Obliczyć nowe zarobki pracowników według wzoru

$$NOWAPLACA_i = \begin{cases} PLACA_i + 15\% SR & \text{jeśli } PLACA_i \leq SR, \\ PLACA_i + 10\% SR & \text{jeśli } PLACA_i > SR, \end{cases}$$

gdzie

$$i=1, \dots, n, \quad SR = \frac{1}{n} \sum_{i=1}^n PLACA_i.$$

Wyniki wydrukować w czytelnej postaci.

35. Daną kwotę pieniędzy rozdzielić na n szkół proporcjonalnie do ilości makulatury zebranej przez każdą szkołę.

36. Danych jest p temperatur mierzonych o ustalonej godzinie w ciągu kolejnych p dni. Wydrukować numery dni w porządku rosnących temperatur.

37. Danych jest n liczb rzeczywistych $x_1, \dots, x_n$. Obliczyć wariancję S dwoma sposobami:

$$1) \quad S = \frac{1}{n} \sum_{i=1}^n (x_i - SA)^2,$$

$$2) \quad S = \frac{1}{n} \sum_{i=1}^n x_i^2 - (SA)^2,$$

gdzie

$$SA = \frac{1}{n} \sum_{i=1}^n x_i.$$

Porównać obydwa programy.

38. Z ciągu $a_1, a_2, \dots, a_n$ utworzyć ciąg $b_1, b_2, \dots, b_n$ przez umieszczenie na początku elementów ujemnych, potem dodatnich, a na końcu zerowych. Następnie obliczyć:

$$SREDNIA = \frac{1}{n} \sum_{i=1}^n b_i, \quad MIN = \min_{1 \leq i \leq n} b_i, \quad MAX = \max_{1 \leq i \leq n} b_i.$$

39. Dla wektora $x = (x_1, \dots, x_n)$ obliczyć:

$$\text{— normę pierwszą } \|x\|_1 = \sum_{i=1}^n |x_i|,$$

$$\text{— normę drugą } \|x\|_2 = \left(\sum_{i=1}^n x_i^2 \right)^{1/2},$$

$$\text{— normę maksimum } \|x\|_\infty = \max_{1 \leq i \leq n} |x_i|,$$

a następnie sprawdzić prawdziwość następujących nierówności:

$$\|x\|_\infty \leq \|x\|_1 \leq n \|x\|_\infty,$$

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty,$$

$$n^{-1/2} \|x\|_1 \leq \|x\|_2 \leq \|x\|_1.$$

40. Dla danej liczby n obliczyć

$$S = \sum_{i=1}^n x_i^2,$$

gdzie

$$x_i = \begin{cases} 1 & \text{jeśli } i=1, \\ (i-1)*1 & \text{jeśli } i \neq 1. \end{cases}$$

41. Dla n -elementowego ciągu liczb rzeczywistych $a_1, a_2, \dots, a_n$ obliczyć (jeśli można)

a) średnią geometryczną $SG = \sqrt[n]{a_1 * a_2 * \dots * a_n}$,

b) średnią harmoniczną $SH = \frac{n}{\frac{1}{a_1} + \dots + \frac{1}{a_n}}$.

42. Zapisać w języku Turbo Pascal

$$z = \sqrt[n]{x_1^k * x_2^k * \dots * x_m^k}.$$

43. Pewna osoba ważyła się co miesiąc od stycznia do grudnia. Podać, w którym miesiącu nastąpił największy:

- przyrost wagi,
- spadek wagi.

44. Dany jest n -elementowy ciąg $a_1, a_2, \dots, a_n$. Utworzyć ciąg $y_1, y_2, \dots, y_{n-1}$, w którym

$$y_j = \begin{cases} \frac{a_{j+1} - a_j}{j} & \text{jeśli } a_{j+1} > a_j, \\ j+5 & \text{jeśli } a_{j+1} = a_j, \\ \frac{a_j - a_{j+1}}{j+1} & \text{jeśli } a_{j+1} < a_j. \end{cases}$$

45. Dana jest liczba naturalna n oraz liczby rzeczywiste:

$$a_1, a_2, \dots, a_n, \quad b_1, b_2, \dots, b_n.$$

Utworzyć ciąg $c_1, c_2, \dots, c_n$, którego elementy są określone wzorem:

$$c_i = a_i + b_i + \frac{i}{3} \quad \text{dla } i=1, \dots, n.$$

Dla każdego ciągu a, b, c obliczyć:

- średnią arytmetyczną,
- iloczyn elementów,
- element minimalny w ciągu.

46. Utworzyć ciąg $u_1, \dots, u_m$ według wzoru

$$u_i = i! \cdot x^i \quad \text{dla } i=1, \dots, m,$$

gdzie

m, x - liczby dane.

Obliczyć sumę elementów ciągu u .

47. Dana jest liczba naturalna n oraz liczba rzeczywista x .
Napisać program, który oblicza wartość

$$S = \sum_{k=1}^n y_k,$$

gdzie

$$y_k = \frac{(-1)^k x^k}{k!} \quad \text{dla } k=1, \dots, n.$$

48. Utworzyć ciąg $d_1, \dots, d_k$ według wzoru

$$d_i = \frac{i+6}{i!} \quad \text{dla } i=1, \dots, k$$

oraz obliczyć sumy częściowe S_i

$$S_i = \sum_{j=1}^i d_j \quad \text{dla } i=1, \dots, k.$$

49. Dane: $n, x_1, \dots, x_n$ ($x_i > -1, i=1, \dots, n$). Utworzyć ciąg $y_1, \dots, y_n$,
którego elementy są określone wzorem

$$y_i = \frac{e^{x_i}}{\sqrt{1+x_i}} \quad \text{dla } i=1, \dots, n.$$

50. Dana jest liczba naturalna n oraz liczba rzeczywista x ($x \neq 0$). Utworzyć ciąg u , w którym

$$u_1 = x,$$

$$u_k = \ln\left(\frac{e^{u_{k-1}} - 1}{u_{k-1}}\right) \quad \text{dla } k=2, 3, \dots, n$$

oraz obliczyć iloczyn elementów ciągu u .

51. Objętość V_k k -wymiarowej kuli o promieniu r jest obliczana
według zależności

$$V_0 = 1,$$

$$V_1 = 2r,$$

$$v_k = \frac{2\pi r^2}{k} v_{k-2} \quad \text{dla } k=2,3,\dots$$

Obliczyć v_n (r, n - dane).

52. Dana jest liczba naturalna n . Obliczyć a_n ze wzorów:

$$a_0 = a_1 = 0,$$

$$a_k = k + a_{k-1} + a_{k-2} \quad \text{dla } k \geq 2.$$

53. Niech

$$a_0 = 1,$$

$$a_k = a_0 a_{k-1} + a_1 a_{k-2} + \dots + a_{k-2} a_1 + a_{k-1} a_0, \quad \text{dla } k > 0.$$

Dla danej liczby naturalnej n obliczyć a_n .

54. Dane: $n, a_1, \dots, a_n$. Utworzyć ciąg $y_1, \dots, y_n$ oraz wyznaczyć liczbę zmian znaków w ciągu $y_1, \dots, y_n$, w którym

$$y_1 = a_1^2,$$

$$y_k = \frac{(k-1)(n-k)}{(n-k+1)k} a_k^2 - a_{k-1} a_{k+1} \quad \text{dla } k=2, \dots, n-1,$$

$$y_n = a_n^2.$$

Obliczając liczbę zmian znaków nie uwzględniać wyrazów ciągu y o wartości równej 0.

55. Napisać procedurę, która dla danych

$$n, a_1, \dots, a_n,$$

$$m, b_1, \dots, b_m$$

utworzy ciąg $c_1, \dots, c_k, c_{k+1}, \dots, c_t$, w którym

$$k = \min(n, m),$$

$$t = \max(n, m),$$

$$c_i = \begin{cases} a_i + b_i & \text{jeśli } 1 \leq i \leq k, \\ a_i & \text{jeśli } k < i \leq t \text{ oraz } t=n, \\ b_i & \text{jeśli } k < i \leq t \text{ oraz } t=m, \end{cases}$$

dla $i=1, 2, \dots, t$.

56. Dany jest n -elementowy ciąg liczbowy a_i ($i=1, \dots, n$). Obliczyć wartości P oraz Q ze wzorów:

$$P = \prod_{i=1}^n \prod_{\substack{j=1 \\ j \neq i}}^n (a_i - a_j),$$

$$Q = \prod_{i=1}^n \sum_{j=1}^n (a_i - a_j).$$

57. Dane: $n, a_1, \dots, a_n$. Obliczyć

$$\prod_{j=1}^n \frac{\sum_{i=1}^n (a_i - a_j)}{j!}.$$

58. Dane: $n, x_1, \dots, x_n, y_1, \dots, y_n$. Obliczyć

$$\sum_{i=1}^n \frac{(x_i - y_i)^i}{\prod_{j=1}^n x_j}.$$

59. Dane: $n, a_1, \dots, a_n$. Obliczyć wartości

$$w(k) = \prod_{i=1}^k \prod_{\substack{j=1 \\ j \neq i}}^k (a_i - a_j) \quad \text{dla } k=2, \dots, n.$$

Wyniki wydrukować w postaci

$$k=2 \quad w(2)=\dots\dots\dots$$

$$k=3 \quad w(3)=\dots\dots\dots$$

...

60. Dane: $n, a_1, \dots, a_n$. Obliczyć średnie bieżące

$$S_i = \frac{a_1 + a_2 + \dots + a_i}{i} \quad \text{dla } i=1, 2, \dots, n$$

oraz

$$S = \left(\sum_{i=1}^n S_i \right) / n.$$

Wyniki wydrukować w postaci

i	$S(i)$
1	
2	
...	
n	

średnia średnich $S = \dots\dots\dots$

61. Dla danej liczby n obliczyć liczby Fibonacciego $u_1, \dots, u_n$ ze wzorów:

$$u_1 = 1, \quad u_2 = 2,$$

$$u_k = u_{k-1} + u_{k-2} \quad \text{dla } k=3, \dots, n$$

oraz obliczyć

$$v_k = \frac{u_k}{u_{k-1}} \quad \text{dla } k=3, \dots, n.$$

Wyniki wydrukować w postaci

k	u	v
3	3	1.50000000
4	5	1.66666667
...		

(liczbę $v = (1 + \sqrt{5})/2 = 1.6180339887 \dots$ nazywa się złotym stosunkiem).

62. Dla danego $n+1$ -elementowego ciągu $x_0, x_1, \dots, x_n$ liczb parami różnych oraz danej liczby x obliczyć sumy

$$S_m = \sum_{k=0}^n ((x - x_k)^m * \prod_{\substack{j=0 \\ j \neq k}}^n \frac{x - x_j}{x_k - x_j}) \quad \text{dla } m=1, 2, \dots, n.$$

63. Napisać podprogram, który dla ciągu liczb rzeczywistych $x_p, \dots, x_k$ wyznacza wartość zmiennej logicznej S określonej następująco:

$$S = \begin{cases} \text{true} & \text{jeśli ciąg } x_p, \dots, x_k \text{ jest uporządkowany rosnąco,} \\ \text{false} & \text{w przeciwnym przypadku.} \end{cases}$$

Dla danego ciągu $a_1, \dots, a_n$ sprawdzić, czy uporządkowane są rosnąco kolejne pięcioelementowe podciągi

$$a_1, \dots, a_5;$$

$$a_6, \dots, a_{10} \quad \text{itd.}$$

oraz czy jest uporządkowany rosnąco cały ciąg $a_1, \dots, a_n$.

64. Napisać program wybierający z danego ciągu liczb całkowitych wszystkie (co najmniej trzyelementowe) sekwencje stanowiące ciąg arytmetyczny, np. w ciągu

$$5, -7, -3, -1, 1, 2, 4, 6, 8, 3, -2, 4, -8, 16, 40;$$

są następujące ciągi arytmetyczne:

$$-3, -1, 1;$$

2,4,6,8;
8,3,-2;
-8,16,40.

65. Napisz program wybierający z danego ciągu liczb całkowitych wszystkie (co najmniej trzejelementowe) sekwencje stanowiące ciąg geometryczny, np. w ciągu

5,-7,-3,-1,1,2,4,6,8,3,-2,4,-8,16,40;

są następujące ciągi geometryczne

1,2,4;
-2,4,-8,16.

66. W danym n -elementowym ciągu liczb całkowitych ($n > 0$) znaleźć element występujący w nim najczęściej oraz liczbę jego wystąpień. Jeżeli dwa różne elementy występują w ciągu tyle samo razy, to można wybrać którykolwiek z nich.

67. Dane są :

- ciąg liczb całkowitych $a_1, a_2, \dots, a_n$ ($n > 0$),
- ciąg wartości logicznych $b_1, b_2, \dots, b_n$.

Wiadomo, że w ciągu a jest k ($0 \leq k \leq n$) liczb ujemnych, w ciągu b jest k wartości true. Napisz procedurę, która uporządkuje ciąg a tak, aby liczby ujemne w ciągu a znalazły się na tych samych pozycjach, na których w ciągu b znajdują się wartości true.

68. W danym n -elementowym ciągu liczb naturalnych znaleźć najmniejszą liczbę podzielną przez 3.

69. Danych jest n ciągów m -elementowych zerojedynekowych oraz liczba k ($1 \leq k \leq m$). Obliczyć ile ciągów

- a) nie zawiera jedynki na k -tym miejscu,
- b) ma same jedynki,
- c) ma same zera.

70. Dany jest n -elementowy ciąg liczb całkowitych $a_1, \dots, a_n$. Sprawdzić, czy ten ciąg jest złożeniem ciągu niemalejącego i ciągu nierosnącego, tzn. czy jest spełniony jeden z trzech warunków:

- 1) ciąg jest niemalejący,
- 2) ciąg jest nierosnący,
- 3) istnieje t ($1 < t < n$) takie, że

$$a_i \leq a_{i+1} \text{ dla } 1 \leq i < t \quad \text{oraz} \quad a_i \geq a_{i+1} \text{ dla } t \leq i < n.$$

71. Dany jest n -elementowy ciąg uporządkowany rosnąco. Zmienić porządek ciągu na przeciwny.

72. Dane są dwa ciągi uporządkowane rosnąco. Połączyć je w jeden ciąg rosnący.

73. Dla danych n -wymiarowych wektorów x oraz y sprawdzić, czy

$$\sum_{i=1}^n x_i^2 + \sum_{i=1}^n y_i^2 \leq \sum_{i=1}^n x_i y_i.$$

74. Dla danych wektorów n -wymiarowych x oraz y sprawdzić, że

$$\max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left(\frac{x_i}{y_j} \right) = \frac{\max_{1 \leq i \leq n} x_i}{\min_{1 \leq j \leq n} y_j},$$

przy czym $y_j \neq 0$ dla $j=1, \dots, n$.

75. Dla danych n -wymiarowych wektorów a , b oraz liczby rzeczywistej k znaleźć n -wymiarowy wektor c o postaci

$$c = b + k \cdot a,$$

ortogonalny do wektora a .

76. Dana jest liczba naturalna n i dwa ciągi n -elementowe d oraz g o elementach całkowitych, reprezentujące ograniczenia dolne i górne tablicy n -wymiarowej o elementach całkowitych, np. dane:

$$n=5,$$

$$d=(2,5,3,-7,4),$$

$$g=(7,5,6,0,6)$$

reprezentują tablicę zadeklarowaną

```
var a: array[2..7,5..5,3..6,-7..0,4..6] of integer;
```

Napisać program sprawdzający, czy dane dwa ciągi d , g reprezentują tablicę n -wymiarową (tzn. czy $d_i \leq g_i$ dla $i=1, \dots, n$). Jeśli tak, to podać liczbę elementów tablicy a .

77. Na podstawie wartości funkcji c obliczyć, ile było w tekście liter, a ile znaków operacji. Funkcja c jest określona następująco:

dla danego ciągu $s_1 s_2 \dots s_n$ liter i znaków operacji

$$c(0) = 0,$$

$$c(i) = \begin{cases} c(i-1)+1 & \text{jeśli } s_i \text{ jest literą,} \\ c(i-1)-1 & \text{jeśli } s_i \text{ jest znakiem operacji.} \end{cases}$$

Na przykład, dla ciągu znaków

$$X+Y-ZAL*A*B/C$$

funkcja c przyjmuje wartości

$$c(0)=0, \quad c(1)=1, \quad c(2)=0, \quad c(3)=1, \quad c(4)=0, \quad c(5)=1, \quad c(6)=2, \\ c(7)=3, \quad c(8)=2, \quad c(9)=3, \quad c(10)=2, \quad c(11)=3, \quad c(12)=2, \quad c(13)=3.$$

Uwaga. Danymi dla programu mają być wartości funkcji c (a nie ciąg znaków $s_1 \dots s_n$).

78. Dla danego n -elementowego ciągu x

a) uporządkować elementy ciągu rosnąco metodą przestawiania elementów sąsiednich nie stojących we właściwym porządku; sortowanie zakończyć, jeżeli w danym kroku algorytmu nie będzie ani jednego przestawienia,

b) zaprogramować inne metody porządkowania ciągu w ciąg rosnący. Dla każdej metody wydrukować liczbę porównań i liczbę przestawień elementów ciągu.

79. Dany jest ciąg n trójek liczb rzeczywistych

$$a_1, b_1, c_1$$

$$a_2, b_2, c_2$$

$$\dots\dots\dots$$

$$a_n, b_n, c_n.$$

Wydrukować wartości wskaźników i oraz a_i, b_i, c_i , dla których

$$b_i^2 - 4a_i c_i > 0.$$

Wydrukować liczbę trójek spełniających powyższy warunek.

80. W danym uporządkowanym niemalejąco ciągu liczb całkowitych znaleźć wartość występującą najczęściej oraz liczbę jej wystąpień, np. w danym ciągu

$$1, 1, 3, 3, 3, 5, 7, 9, 9, 9, 9, 20$$

liczba 9 występuje najczęściej (4 razy).

81. Dla danych trójek (nr, il, c) , gdzie

nr - numer towaru ($1 \leq nr \leq 500$),

il - ilość towaru w jednostkach umownych,

c - cena jednostki towaru w złotych,

dokonać następującego spisu towarów w sklepie:

a) wydrukować zestawienie:

$$nr \quad il \quad c \quad il * c$$

uporządkowane według numerów towarów,

b) wydrukować ogólną wartość towarów w sklepie.

Uwagi:

- sygnałem końca danych jest 0,
- mogą występować trójki z tym samym numerem, np.
(5,10,160);
(5,12,220).

82. Dane są dwa ciągi u oraz w . Sprawdzić, czy u jest podciągiem w , złożonym z kolejnych elementów ciągu w , np.

$w = [2, 7, 4, -3, 5, 1, 8],$

$u = [4, -3, 5]$ u jest podciągiem w ,

$u = [2, 4, 5]$ u nie jest podciągiem w ,

$u = [1, 8, 2]$ u nie jest podciągiem w .

83. Napisać procedury obliczania sumy, różnicy i iloczynu dla dwóch ciągów zerojedynkowych reprezentujących binarnie liczby całkowite nieujemne.

84. Danych jest n serii pomiarów. Każda z serii może zawierać od 2 do 100 pomiarów. Napisać program, który:

- a) dla każdej serii wyznacza średnią arytmetyczną pomiarów,
- b) drukuje wykres, w którym
oś odciętych = numer serii,
oś rzędnych = średnia w serii,
- c) drukuje numery serii w porządku malejących średnich.

85. Obliczyć pole wielokąta o wierzchołkach:

$P_1(x_1, y_1), P_2(x_2, y_2), \dots, P_n(x_n, y_n)$

ze wzoru:

$$S = \frac{1}{2}[(x_1 - x_2)(y_1 + y_2) + (x_2 - x_3)(y_2 + y_3) + \dots + (x_n - x_1)(y_n + y_1)].$$

Uwaga. Jeśli wierzchołki ponumerowane są zgodnie z ruchem wskazówek zegara, to $S < 0$, natomiast w przeciwnym przypadku $S > 0$.

86. Dana jest macierz $A-n \times m$. Obliczyć liczbę zer w każdym wierszu.

87. Dane są liczby naturalne n, m ($n, m \leq 10$) oraz macierz $A-n \times m$. Obliczyć:

$$S_i = \sum_{j=1}^m a_{ij} \quad \text{dla } i=1, \dots, n.$$

Wydrukować macierz A oraz obliczone sumy S_i ($i=1, \dots, n$) w postaci np.:

MACIERZ A	Suma
$a_{11} \ a_{12} \ \dots \ a_{1m}$	S_1
.....	
$a_{n1} \ a_{n2} \ \dots \ a_{nm}$	S_n

88. Dane są liczby naturalne $n \leq 10$, $m \leq 10$ oraz macierz B $n \times m$.
Obliczyć:

$$M_j = \min_{1 \leq i \leq n} b_{ij} \quad \text{dla } j=1, \dots, m.$$

Wydrukować macierz B oraz obliczone wartości minimum w postaci np.

	B
	$b_{11} \ b_{12} \ \dots \ b_{1m}$
	
	$b_{n1} \ b_{n2} \ \dots \ b_{nm}$
min	$M_1 \ M_2 \ \dots \ M_m$

89. Dane są liczby naturalne $n, m (\leq 10)$ oraz macierz A $m \times n$.
Utworzyć wektor b o wartościach

$$b_i = \max_{1 \leq j \leq n} |a_{ij}| \quad \text{dla } i=1, \dots, m.$$

Wydrukować macierz A oraz wektor b w czytelnej postaci.

90. Dana jest macierz A $m \times n$. Wyznaczyć wartość oraz indeksy elementu o maksymalnej wartości bezwzględnej. Jeśli takich elementów jest kilka, to wyznaczyć wartość pierwszego napotkanego.

91. Dana jest macierz prostokątna A $n \times m$. Znaleźć

$$W = \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} |a_{ij}|.$$

Podać indeksy wszystkich elementów macierzy mających wartość W .

92. Dana jest macierz A $n \times m$. Obliczyć:

$$a) \sum_{i=1}^n \max_{1 \leq j \leq m} a_{ij},$$

$$b) \sum_{j=1}^m \min_{1 \leq i \leq n} a_{ij},$$

$$c) \min_{1 \leq j \leq m} \sum_{i=1}^n |a_{ij}|.$$

93. Dana jest macierz $A-n*m$. Znaleźć dwa elementy o równych wartościach. Wydrukować tę wartość i indeksy tych elementów.

94. Dla danej macierzy $A-m*n$ obliczyć i wydrukować:

$$SR_i = \frac{1}{n} \sum_{j=1}^n a_{ij}, \quad D_i = \sqrt{\frac{1}{n} \sum_{j=1}^n a_{ij}^2 - SR_i^2} \quad \text{dla } i=1,2,\dots,m.$$

95. Dla danej macierzy $A-m*n$ obliczyć i wydrukować:

$$SR_i = \frac{1}{n-1} \sum_{j=1}^n a_{ij} \quad \text{dla } i=1,\dots,m;$$

$$COV_{ij} = \frac{1}{n} \sum_{k=1}^n a_{ik} a_{jk} - SR_i SR_j \quad \text{dla } i,j=1,\dots,m.$$

96. Na pewnym kierunku studiów prowadzi się 1w wykładów i studiuj tam 1s studentów. Mając dane stopnie, obliczyć średni stopień z i-tego wykładu oraz średni stopień ze wszystkich wykładów dla każdego studenta. Policzyc również ogólny średni stopień wszystkich studentów ze wszystkich wykładów.

97. Dana jest macierz $A-n*m$. Obliczyć i wydrukować

$$S_j = \sin\left(\sum_{i=1}^n a_{ij}\right) \quad \text{dla } j=1,\dots,m.$$

98. Dana jest macierz $A-m*m$. Obliczyć:

- maksimum w wierszach nieparzystych,
- sumę elementów w wierszach parzystych,
- minimum w kolumnach nieparzystych,
- iloczyn w kolumnach parzystych.

99. Napisać procedurę obliczania maksimum w ciągu. Dla danej macierzy $B-n*m$ wykorzystać ją do obliczenia

$$a) M_i = \max_{1 \leq j \leq m} b_{ij} \quad \text{dla } i=1,\dots,n,$$

$$b) MM = \max_{1 \leq i \leq n} M_i,$$

$$c) S = \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} b_{ij},$$

d) porównać MM i S.

100. Napisać procedurę, która sprawdza, czy w danej macierzy prostokątnej $A-m*n$ istnieje element a_{ks} taki, że

$$a_{ks} = \min_{1 \leq i \leq m} a_{is} = \max_{1 \leq j \leq n} a_{kj}.$$

Jeśli taki element istnieje, to podać wartości k , s , a_{ks} .

101. Napisać funkcję, która oblicza średnią geometryczną ciągu n -elementowego. Wykorzystać ją do obliczenia średniej geometrycznej każdego wiersza macierzy $A-m*n$.

102. Napisać procedurę, która dla danego ciągu n -elementowego wyznacza liczbę elementów ujemnych, równych zero oraz dodatnich. Wykorzystać ją do obliczenia liczby elementów ujemnych, równych zero oraz dodatnich w każdej kolumnie macierzy $A-m*n$.

103. Napisać procedurę obliczania średnich bieżących $b_1, \dots, b_n$ ciągu $a_1, \dots, a_n$ ze wzorów:

$$b_1 = \frac{a_1 + a_2}{2},$$

$$b_i = \frac{a_{i-1} + a_i + a_{i+1}}{3} \quad \text{dla } i=2, 3, \dots, n-1,$$

$$b_n = \frac{a_{n-1} + a_n}{2}.$$

Obliczyć średnie bieżące w każdym wierszu macierzy $X-m*n$.

104. Wśród n osób przeprowadzono następujące głosowanie: każda osoba oddaje jeden głos na dowolną spośród n osób. Wydrukować, ile głosów otrzymała każda z osób oraz która osoba otrzymała maksymalną liczbę głosów.

105. Dane są dwie macierze $X-l*k$ oraz $Y-l*k$. Dla każdej z tych macierzy X i Y utworzyć wektory S_x , S_y , których elementami są średnie arytmetyczne kolejnych wierszy odpowiednio macierzy X i Y . Obliczyć iloczyn skalarny S_x i S_y .

106. Uporządkować każdy wiersz macierzy $A-m*n$ w ciąg malejący.

107. Uporządkować każdą kolumnę macierzy $A-m*n$ w ciąg rosnący.

108. Dana jest macierz kwadratowa $A-n*n$. Wyznaczyć iloczyn skalarny elementów głównej przekątnej z każdym wierszem tej macierzy.

109. Danych jest m wektorów n -wymiarowych

$$x_1 = (x_{11}, x_{12}, \dots, x_{1n}),$$

$$x_2 = (x_{21}, x_{22}, \dots, x_{2n}),$$

...

$$x_m = (x_{m1}, x_{m2}, \dots, x_{mn}).$$

Sprawdzić, które pary wektorów są ortogonalne.

Uwaga. Wektory X i Y są ortogonalne, jeśli ich iloczyn skalarny równa się 0.

110. Dla danej macierzy $A-m \times n$ wydrukować napis

'zero w każdym wierszu',

jeśli w każdym wierszu macierzy co najmniej jeden element jest równy 0.

111. Dla danej macierzy $A-m \times n$ wydrukować napis

'dokładnie jedno zero w każdym wierszu',

jeśli w każdym wierszu macierzy dokładnie jeden element jest równy 0.

112. Dana jest macierz $A-n \times n$. Sprawdzić, czy jest to macierz symetryczna.

113. Dana jest macierz $A-n \times m$. Zamienić miejscami elementy kolumn o numerach k oraz t ($1 \leq k \leq m$, $1 \leq t \leq m$, $k \neq t$).

114. Dane są dwie macierze prostokątne $A-n \times m$ i $B-n \times m$ oraz liczba rzeczywista h . Utworzyć macierze $C-n \times m$ i $D-n \times m$, gdzie

$$C = A - h \cdot B, \quad D = 2A + 3B.$$

Wydrukować macierze A , B , C , D .

115. Dane są macierze $A-r \times s$, $C-m \times r$ oraz liczba całkowita k . Obliczyć iloczyn skalarny k -tej kolumny macierzy A z każdym wierszem macierzy C .

116. Dane są macierze $A-r \times s$, $B-m \times s$. Obliczyć iloczyn skalarny każdego wiersza macierzy A z każdym wierszem macierzy B .

117. Dane są dwie macierze prostokątne $A-m \times n$, $B-n \times l$. Obliczyć elementy macierzy $C-m \times l$, $C=A \cdot B$, gdzie

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}, \quad \text{dla } i=1, \dots, m; j=1, \dots, l.$$

118. Zastąpić zerami te elementy macierzy $A-n \times n$, które nie leżą na wskazanych przekątnych:

$$A = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

119. Dana jest macierz $A-r \times s$. Wyzerować elementy nieparzystych wierszy i parzystych kolumn.

120. Dana jest macierz $A-n \times m$, n -elementowy ciąg $x_1, \dots, x_n$ oraz liczba całkowita k ($1 \leq k \leq n$).

(1) Obliczyć sumę SA elementów macierzy A oraz sumę SX elementów ciągu X .

(2) Zastąpić k -tą kolumnę i k -ty wiersz w macierzy A ciągiem x .

(3) Obliczyć sumę SAX elementów nowo utworzonej macierzy A .

(4) Za pomocą wielkości SA , SAX , SX i x_k obliczyć sumę tych elementów wejściowej macierzy, które zostały zastąpione ciągiem x .

121. Dla danej macierzy $A-n \times n$ oraz liczby naturalnej k ($k=1,2,3$) obliczyć:

- minimum z elementów nad główną przekątną, gdy $k=1$.
- średnią arytmetyczną elementów pod główną przekątną, gdy $k=2$.
- iloczyn elementów leżących na głównej przekątnej, gdy $k=3$.

122. Dana jest macierz kwadratowa $A-n \times n$. Dwie główne przekątne dzielą macierz A na 4 części

$$A = \begin{bmatrix} & 2 & \\ 1 & & 3 \\ & 4 & \end{bmatrix}$$

Obliczyć:

- minimum w części 1,
- średnią arytmetyczną w części 2,
- liczbę zer w części 3,
- iloczyn elementów w części 4.

123. Napisać procedurę usuwającą z danej macierzy wiersze i kolumny o wszystkich elementach zerowych. Pozostałe kolumny przesunąć w lewy górny róg, np.

$$\begin{bmatrix} 1 & 0 & 2 & 0 & 3 \\ 4 & 0 & 5 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 8 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

124. Z macierzy $A-m*n$ usunąć k -ty wiersz ($1 \leq k \leq m$).

125. Dana jest macierz $A-n*n$ o elementach całkowitych. Usunąć z niej wiersze i kolumny, które zawierają dany element x , np. niech $x=7$,

$$A = \begin{bmatrix} 2 & 5 & 7 & 0 & 3 \\ 4 & 2 & 3 & 1 & 9 \\ 8 & 2 & 6 & 2 & 1 \\ -5 & 4 & -3 & 0 & 2 \\ 6 & 1 & 3 & 2 & 4 \end{bmatrix}.$$

A po usunięciu pierwszego wiersza i trzeciej kolumny

$$\begin{bmatrix} 4 & 2 & 1 & 9 \\ 8 & 2 & 2 & 1 \\ -5 & 4 & 0 & 2 \\ 6 & 1 & 2 & 4 \end{bmatrix}.$$

126. Sprawdzić, czy w macierzy $A-n*m$ są dwie kolumny takie same lub dwa takie same wiersze.

127. W macierzy prostokątnej znaleźć wszystkie kolumny jednakowe.

128. Dana jest macierz $A-n*m$. Utworzyć macierz A^T .

129. Dana jest liczba naturalna n ($n \leq 7$). Utworzyć i wydrukować macierz $A-n*n$ o elementach

$$a_{ij} = \begin{cases} \frac{i+j}{4} & \text{jeśli } i < j, \\ \sqrt{i*j} & \text{jeśli } i = j, \\ (i-j)^3 & \text{jeśli } i > j, \end{cases}$$

gdzie $i, j = 1, \dots, n$.

130. Utworzyć macierz $A-n*n$ w sposób wynikający z przykładu dla $n=3$

$$A = \begin{bmatrix} 1 & 1+2 & 1+2+3 \\ 2 & 2+3 & 2+3+4 \\ 3 & 3+4 & 3+4+5 \end{bmatrix}.$$

131. Dane: n, m - liczby naturalne. Utworzyć macierz $A-n*m$ o

elementach

$$a_{ij} = i*j \quad \text{dla } i=1, \dots, n; \quad j=1, \dots, m.$$

132. Dana jest macierz $A-n*m$. Utworzyć macierz $B-m*m$ postaci

$$\begin{bmatrix} 1 & \min A & 1 \\ \max A & & \max A \\ 1 & \min A & 1 \end{bmatrix}$$

gdzie $\min A$, $\max A$ oznaczają odpowiednio minimum i maksimum macierzy A .

133. Dana jest macierz $A-n*m$. Utworzyć macierz $B-n*n$ postaci:

$$B = \begin{bmatrix} m_1 & p_1 & m_1 & p_1 & \cdot & \cdot & \cdot & \cdot \\ m_2 & p_2 & m_2 & p_2 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ m_n & p_n & m_n & p_n & \cdot & \cdot & \cdot & \cdot \end{bmatrix},$$

gdzie m_i - minimum i -tego wiersza macierzy A ,

p_i - maksimum i -tego wiersza macierzy A .

134. Dane: n , m . Utworzyć i wydrukować macierz $C-n*m$ o postaci:

$$\begin{bmatrix} 1+1 & 1+2 & \dots & 1+m \\ 2+1 & 2+2 & \dots & 2+m \\ \dots & \dots & \dots & \dots \\ n+1 & n+2 & \dots & n+m \end{bmatrix}.$$

135. Dane m , n . Utworzyć macierz $A-m*n$ o elementach

$$a_{ij} = 2^i 3^j \quad \text{dla } i=1, \dots, m; \quad j=1, \dots, n.$$

136. Dla danej macierzy $A-m*n$ wyznaczyć macierz, która zawiera tylko elementy dodatnie macierzy A (pozostałe elementy wyzerować) oraz macierz, która zawiera tylko elementy ujemne z A (pozostałe wyzerować). Wszystkie macierze wydrukować.

137. Dana jest macierz $A-n*m$. Utworzyć macierze $B-n*m$ i $C-n*m$ takie, że:

a) macierz B zawiera elementy o wartościach parzystych z macierzy A (pozostałe elementy mają wartość 0),

b) macierz C zawiera elementy o wartościach nieparzystych z A (pozostałe elementy mają wartość 0).

Sprawdzić, czy

$$\sum_{i=1}^n \sum_{j=1}^m a_{ij} = \sum_{i=1}^n \sum_{j=1}^m b_{ij} + \sum_{i=1}^n \sum_{j=1}^m c_{ij}.$$

138. Utworzyć macierz A - $n \times n$ o postaci

$$A = \begin{bmatrix} 1 & 2 & 3 & \dots & n \\ 2 & 3 & \dots & n & 1 \\ 3 & \dots & n & 1 & 2 \\ \vdots & & & & \vdots \\ n & 1 & 2 & \dots & n-1 \end{bmatrix}.$$

139. Dane są macierze A - $n \times n$, B - $n \times n$. Utworzyć macierz C - $n \times n$ o elementach

$$c_{ik} = \min_{1 \leq j \leq n} (a_{ij} + b_{jk}) \quad \text{dla } i=1, \dots, n; k=1, \dots, n.$$

140. Napisać procedury, które realizują następujące zadania:

- transponowanie macierzy R - $m \times n$,
- dodawanie lub odejmowanie dwóch macierzy prostokątnych A - $m \times n$ oraz B - $m \times n$,
- mnożenie macierzy prostokątnych X - $m \times k$, Y - $k \times n$.

Wykorzystując te procedury dla danych macierzy A , B , C stopnia n , wyznaczyć wartości elementów macierzy M , V , W , przy czym

$$M = A * B + C^T, \quad V = (M * C)^T + B, \quad W = M - A^T,$$

A^T oznacza macierz transponowaną do A .

141. Danymi dla programu są: liczba naturalna n oraz elementy macierzy A - $n \times n$ podane w następującej kolejności

$$a_{11}, a_{21}, a_{12}, a_{31}, a_{22}, a_{13}, \dots, a_{n,n-1}, a_{n-1,n}, a_{nn}.$$

Napisać program, który tworzy macierz A .

142. Sprawdzić, czy

$$A * A^T = A^T * A = E,$$

gdzie

A - dowolna macierz $n \times n$,

E - macierz jednostkowa $n \times n$, tzn.

$$E = \begin{bmatrix} 1 & & \\ & 1 & \\ & & \ddots \\ & & & 1 \end{bmatrix}.$$

143. Dane są dwie macierze diagonalne stopnia n (tzn. tylko główne przekątne mają elementy niezerowe). Obliczyć ich iloczyn. Jak najlepiej jest pamiętać macierze diagonalne?

144. Dana jest macierz dwuwstęgowa A $n \times n$ (n - duże), tzn. wszystkie elementy poza wyróżnionymi przekątnymi mają wartość 0.

$$A = \begin{bmatrix} * & * & * & & & & & & \\ * & * & * & * & & & & & \\ * & * & * & * & * & & & 0 & \\ * & * & * & * & * & * & & & \\ & * & * & * & * & * & * & & \\ & & * & * & * & * & * & * & \\ & & & * & * & * & * & * & \\ 0 & & & & * & * & * & * & * \\ & & & & & * & * & * & * \\ & & & & & & * & * & * \end{bmatrix}.$$

Zapamiętać elementy z pięciu wyróżnionych przekątnych macierzy A w tablicy $b[-2..2, 1..n]$ następująco:

$$\begin{aligned} b_{-2,i} &= a_{i,i-2} & \text{dla } i=3,4,\dots,n, \\ b_{-1,i} &= a_{i,i-1} & \text{dla } i=2,3,\dots,n, \\ b_{0,i} &= a_{ii} & \text{dla } i=1,2,\dots,n, \\ b_{1,i} &= a_{i,i+1} & \text{dla } i=1,2,\dots,n-1, \\ b_{2,i} &= a_{i,i+2} & \text{dla } i=1,2,\dots,n-2, \\ b_{-1,1} &= b_{-2,1} = b_{-2,2} = b_{1,n} = b_{2,n-1} = b_{2,n} = 0. \end{aligned}$$

145. Macierz s -wstęgową A $n \times n$ postaci

$$A = \begin{bmatrix} * & * & * & * & & & & & \\ * & * & * & * & * & & & & \\ * & * & * & * & * & * & & 0 & \\ * & * & * & * & * & * & * & & \\ * & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & & * & * & * & * & * & * & \\ 0 & & & * & * & * & * & * & \\ & & & & * & * & * & * & \\ & & & & & * & * & * & \\ & & & & & & * & * & \\ & & & & & & & * & * \end{bmatrix} \left. \vphantom{\begin{bmatrix} * & * & * & * & * & * & * & * & \\ * & * & * & * & * & * & * & * & \\ * & * & * & * & * & * & * & * & \\ * & * & * & * & * & * & * & * & \\ * & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & & * & * & * & * & * & * & \\ 0 & & & * & * & * & * & * & \\ & & & & * & * & * & * & \\ & & & & & * & * & * & \\ & & & & & & * & * & \\ & & & & & & & * & * \end{bmatrix}} \right\} s \text{ przekątnych}$$

gdzie

$$s < n-1, \quad a_{ij} = 0 \text{ dla } |i-j| > s,$$

zapamiętać w tablicy jednowymiarowej C :

$$C = [a_{11}, \dots, a_{1,s+1}, \dots, a_{n,n-s}, \dots, a_{nn}].$$

Ile elementów zawiera tablica C ?

146. Elementy macierzy s -wstęgowych A $n \times n$, B $n \times n$ zapamiętane są w tablicach jednowymiarowych AW , BW . Obliczyć $CW = AW + BW$. Tablice AW , BW , CW wydrukować w postaci macierzy wymiaru $n \times n$ o s wstęgach.

147. Obliczyć wyznacznik macierzy A - $n \times n$ postaci:

$$A = \begin{bmatrix} * & * & & & 0 \\ * & * & * & & \\ & * & * & * & \\ 0 & * & * & * & * \\ & & * & * & * \end{bmatrix}.$$

148. Dana jest macierz X - $n \times m$. Obliczyć i wydrukować

- normę wierszową: $SA = \max_{1 \leq i \leq n} \left(\sum_{j=1}^m x_{ij}^2 \right)^{1/2},$

- normę kolumnową: $SB = \max_{1 \leq j \leq m} \left(\sum_{i=1}^n x_{ij}^2 \right)^{1/2},$

- normę Euklidesową: $SE = \left(\sum_{i=1}^n \sum_{j=1}^m x_{ij}^2 \right)^{1/2}.$

149. Dany jest n -elementowy wektor a , którego elementami są wartości głównej przekątnej macierzy diagonalnej A . Wydrukować macierz A oraz obliczyć:

a) normę Euklidesową,

b) normę drugą.

(zobacz zadanie 148).

150. Dla danej macierzy A - $m \times n$ i n -elementowego wektora x obliczyć normę wektora y ze wzoru

$$\|y\| = \sqrt{\sum_{i=1}^n y_i^2},$$

gdzie

$$y_i = \sum_{k=1}^n a_{ik} x_k \quad \text{dla } i=1, \dots, m.$$

151. Daną macierz B - $n \times m$ zapamiętać w tablicy jednowymiarowej C . Wydrukować B , C .

152. Macierz A - $n \times n$ ma postać

$$A = \begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ . & . & . & . & . \\ . & . & . & . & 0 \\ a_{n1} & a_{n2} & . & . & a_{nn} \end{bmatrix}.$$

Zapamiętać elementy z głównej przekątnej i spod niej w tablicy jednowymiarowej C

$$C = [a_{11}, a_{21}, a_{22}, \dots, a_{n1}, a_{n2}, \dots, a_{nn}]$$

153. Napisać procedurę, która rozwiązuje układ równań stopnia n postaci:

$$a_{11}x_1 = b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

.....

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n$$

Wskazówka. Z pierwszego równania obliczyć x_1 , wstawić do drugiego równania, obliczyć x_2 itd.

154. Dane są macierze trójkątne $A-n \times n$, $B-n \times n$ postaci

$$A = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ 0 & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & b_{nn} \end{bmatrix}$$

Zapamiętać elementy tych macierzy w macierzy $AB-n \times (n+1)$ postaci

$$AB = \begin{bmatrix} a_{11} & b_{11} & b_{12} & \dots & b_{1n} \\ a_{21} & a_{22} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_{nn} \end{bmatrix}$$

155. Pomnożyć macierz trójkątną górną $A-n \times n$ przez trójkątną dolną $B-n \times n$.

156. Dane są dwie macierze trójkątne górne A, B wymiaru $n \times n$

$$A = \begin{bmatrix} * & \dots & * \\ 0 & \ddots & \vdots \\ * \end{bmatrix}, \quad B = \begin{bmatrix} * & \dots & * \\ 0 & \ddots & \vdots \\ * \end{bmatrix}$$

Napisać program, który wczytuje elementy macierzy A i B do tablicy wymiaru $(n+1) \times n$ oraz oblicza iloczyny $A \cdot B$, $B \cdot A$, które są także macierzami trójkątnymi górnymi i zapamiętuje te iloczyny w dwóch wektorach $(n^2 + \frac{n}{2})$ -elementowych.

157. Dana jest macierz A stopnia n o elementach:

$$a_{ij} = \begin{cases} 1 & \text{dla } i \neq j, \\ d & \text{dla } i=j, \end{cases}$$

d - dowolna liczba rzeczywista różna od zera.

1. Obliczyć metodą eliminacji Gaussa z częściowym wyborem elementu głównego macierz X odwrotną do A , tzn.

$A \cdot X = E$, gdzie E oznacza macierz jednostkową.

Jeśli przez x_{ij} oznaczymy elementy macierzy odwrotnej do A , to

$$x_{ii} = \frac{d+n-2}{d(d+n-2)-n+1} \quad \text{dla } i=1, \dots, n,$$

$$x_{ij} = -\frac{1}{d(d+n-2)-n+1} \quad \text{dla } i, j=1, \dots, n; i \neq j.$$

2. Obliczyć i wydrukować:

$$\max |x_{ij} - a_{ij}^{-1}|,$$

$$\|X\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |x_{ij}|.$$

158. Dla danego grafu jest określona zerojedynkowa, symetryczna macierz A - $n \times n$ o elementach:

$$a_{ij} = \begin{cases} 1 & \text{jeśli węzeł } i \text{ jest połączony krawędzią z węzłem } j, \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

Korzystając z A utworzyć B - $n \times n$ o elementach:

$$b_{ij} = \begin{cases} \text{true} & \text{jeśli istnieje droga z węzła } i \text{ do } j, \\ \text{false} & \text{w przeciwnym przypadku,} \end{cases}$$

według następującego algorytmu:

$$B^0 = A,$$

$$b_{ij}^k = b_{ij}^{k-1} \vee (b_{ik}^{k-1} \wedge b_{kj}^{k-1}) \quad \text{dla } k=1, \dots, n; i, j=1, \dots, n.$$

B^n jest macierzą szukaną.

159. Relacja R określona na zbiorze $a_1, \dots, a_n$ jest zapisana w tablicy M - $n \times n$, gdzie

$$m_{ij} = \begin{cases} 1 & \text{jeśli } a_i R a_j, \\ 0 & \text{w przeciwnym przypadku,} \end{cases}$$

dla $i, j=1, \dots, n$.

Sprawdzić, czy relacja R jest

- a) zwrotna,
- b) symetryczna,
- c) przechodnia.

160. Działanie $\circ$ określone na zbiorze $a_1, \dots, a_n$ jest zapisane w tablicy $M-n \times n$ następująco

$$m_{ij}=k \Leftrightarrow a_i \circ a_j = a_k \quad \text{dla } i, j, k=1, \dots, n.$$

Sprawdzić, czy działanie ma element neutralny.

Uwaga. $e \in \{a_1, \dots, a_n\}$ jest elementem neutralnym wtedy i tylko wtedy,

$$\text{gdy } \bigwedge_{1 \leq i \leq n} a_i \circ e = e \circ a_i = a_i.$$

161. Napisać procedurę tworzenia tablicy $W-n \times n$ o elementach

$$w_{ij} = \begin{cases} i+f(R*j) & \text{dla } i \geq j, \\ 2*(i+j)*g(j+R*i) & \text{dla } i < j, \end{cases}$$

gdzie

$i, j=1, \dots, n$,

n - dana liczba naturalna,

R - dana liczba rzeczywista,

f, g - dane funkcje.

Wykorzystać tę procedurę do utworzenia tablicy:

- $A-n \times n$, gdzie $n=7$, R - dowolne,

$$f(x) = x^2 + e^x,$$

$$g(x) = \begin{cases} 2x + |\sin(x)| & \text{dla } x < 0, \\ 1 & \text{dla } x = 0, \\ \frac{\cos(x)}{x+1} & \text{dla } x > 0. \end{cases}$$

- $B-n \times n$, gdzie $n=7$, R - dowolne,

$$f(x) = \sin(x),$$

$$g(x) = \cos(x).$$

162. Uporządkować ciąg wierszy macierzy prostokątnej A według malejących wartości elementów kolumny o numerze k (należy tak przestawić wiersze macierzy A , by k -ta kolumna była uporządkowana malejąco).

163. W macierzy $A-n \times m$ policzyć liczbę zmian znaków w każdym wierszu, np.

$$A = \begin{bmatrix} 1 & -1 & 2 & 0 & 3 \\ -4 & -2 & 7 & -1 & 5 \\ 3 & 3 & 2 & 5 & 7 \end{bmatrix},$$

nr wiersza liczba zmian znaku

1	2
2	3
3	0

164. Dane są macierze $A-n*n$, $B-n*n$ zawierające elementy 0 i 1 (oznaczające fałsz i prawdę). Utworzyć macierz C , która jest sumą logiczną macierzy A i B , oraz macierz D taką, że

$$d_{ij} = \bigcup_{k=1}^n (a_{ik} \wedge b_{kj}) \quad \text{dla } i, j=1, \dots, n,$$

gdzie $\cup$ - alternatywa, $\wedge$ - koniunkcja.

165. Napisać procedurę ZAMIANA, która dla danych dwóch wektorów T oraz $TMOD$ w wektorze T zastępuje zerami te elementy, które występują w $TMOD$. Następnie dla danych dwóch macierzy $A-n*m$, $B-n*k$ wykorzystując procedurę ZAMIANA w i -tym ($i=1, \dots, n$) wierszu macierzy A zastąpić zerami te elementy, które występują w i -tym wierszu macierzy B .

166. Magicznym kwadratem stopnia n nazywamy tablicę wymiaru $n*n$, zawierającą liczby od 1 do n^2 zapamiętane w taki sposób, że sumy liczb w każdym wierszu, w każdej kolumnie i na każdej z dwóch głównych przekątnych są sobie równe. Napisać program, który sprawdza, czy dana tablica $n*n$ jest magicznym kwadratem.

167. Zapelnąć macierz $T-n*n$ liczbami od 1 do n^2 idąc po spirali od lewego górnego rogu według wskazówek zegara, np. dla $n=6$ należy utworzyć macierz

$$T = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 20 & 21 & 22 & 23 & 24 & 7 \\ 19 & 32 & 33 & 34 & 25 & 8 \\ 18 & 31 & 36 & 35 & 26 & 9 \\ 17 & 30 & 29 & 28 & 27 & 10 \\ 16 & 15 & 14 & 13 & 12 & 11 \end{bmatrix}.$$

168. W pewnym zakładzie jest zatrudnionych n pracowników, którzy mają nienormowany czas pracy. Dla każdego pracownika znana jest liczba godzin przepracowanych przez niego w kolejnych dniach tygodnia.

Wykorzystując typ wyliczeniowy dnitygodnia, obliczyć liczbę godzin przepracowanych między datą początkową a datą końcową

- przez każdego pracownika,
- przez n pracowników.

169. Dla dowolnej sieci określa się macierz incydencji $A-n*n$ (n - liczba węzłów sieci) o elementach 0 lub 1 następująco:

$$a_{ij} = \begin{cases} 1 & \text{jeśli w sieci jest strzałka od węzła } i \text{ do węzła } j, \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

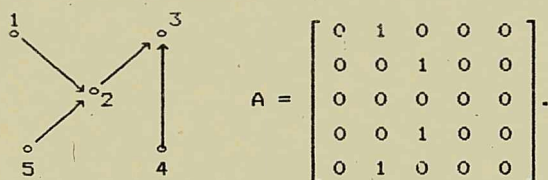
dla $i, j=1, \dots, n$.

Utworzyć macierz połączeń $M-n*n$ określoną następująco:

$$m_{ij} = \begin{cases} 1 & \text{jeśli w sieci można dojść od węzła } i \text{ do } j, \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

dla $i, j=1, \dots, n$.

Na przykład



Dla powyższej sieci macierz M ma postać:

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

170. Dana jest liczba naturalna n oraz liczby rzeczywiste:

x ,

$x_1, x_2, \dots, x_n$ ($x_i \neq x_j$ dla $i, j=1, \dots, n, i \neq j$),

$y_1, y_2, \dots, y_n$, gdzie $y_i = f(x_i)$ dla $i=1, \dots, n$.

Obliczyć wartość wielomianu interpolacyjnego Lagrange'a w zadanym punkcie x według wzoru

$$L(x) = \sum_{i=1}^n (y_i * \prod_{\substack{j=1 \\ j \neq i}}^n \frac{x-x_j}{x_i-x_j}).$$

171. Obliczyć wartość zespoloną wielomianu

$$w(x) = a_0 x^n + a_1 x^{n-1} + \dots + a_n, \quad \text{gdzie } x = R + i \cdot I$$

według schematu

$$w_0 = a_0, \quad R_0 = a_0, \quad I_0 = 0,$$

$$w_k = (R_{k-1} R - I_{k-1} I + a_k) + i(I_{k-1} R + R_{k-1} I),$$

gdzie

$$R_{k-1} - \text{część rzeczywista } w_{k-1},$$

$$I_{k-1} - \text{część urojona } w_{k-1}$$

dla $k=1, 2, \dots, n$.

172. Obliczyć ze schematu Hornera wartość wielomianu

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0.$$

Schemat Hornera: wartość wielomianu

$$w_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

można obliczyć ze wzorów:

$$w_0 = a_n,$$

$$w_i = w_{i-1} x + a_{n-i} \quad \text{dla } i=1, \dots, n.$$

173. Dane są liczby naturalne m, n ($m, n \leq 20$) oraz współczynniki wielomianów:

$$p(x) = a_0 x^m + a_1 x^{m-1} + \dots + a_m,$$

$$q(x) = b_0 x^n + b_1 x^{n-1} + \dots + b_n.$$

Obliczyć współczynniki wielomianów:

$$v(x) = p(x) + q(x),$$

$$w(x) = p(x)q(x) = c_0 x^r + c_1 x^{r-1} + \dots + c_r,$$

gdzie

$$r = m + n,$$

$$c_k = \sum_{j=t}^l a_j b_{k-j} \quad \text{dla } k=1, \dots, r; \quad l = \min\{m, k\}, \quad t = \max\{0, k-n\}.$$

174. Dla wielomianu

$$w(x) = a_0 x^n + a_1 x^{n-1} + \dots + a_n,$$

gdzie a_i są liczbami rzeczywistymi ($i=0, \dots, n$), $a_0 \neq 0$, oszacować przedział $(0, 6)$, w którym znajdują się jego wszystkie pierwiastki rzeczywiste dodatnie, przy czym

$$G = 1 + \sqrt[m]{\frac{B}{a_0}}, \quad B = \left| \min_{\substack{1 \leq i \leq n \\ a_i < 0}} a_i \right|,$$

m - wskaźnik pierwszego ujemnego współczynnika.

Wartość współczynnika a_0 powinna być dodatnia (jeśli $a_0 < 0$, to należy pomnożyć $w(x)$ przez -1).

Jeśli $a_i \geq 0$ dla $i=0,1,\dots,n$, to wydrukować napis:

NIE MA PIERWIASTKÓW DODATNICH.

175. Dla danych m punktów $x_1, x_2, \dots, x_m$ obliczyć i wydrukować wartość wielomianu

$$w(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

(skorzystać ze schematu Hornera). Dla ilu punktów $w(x) < 0$?

Wyniki wydrukować w czytelnej postaci.

176. Napisać funkcję, która dla danych liczb n, x oblicza wartość wielomianu

$$w(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0.$$

Wykorzystać tę funkcję do obliczenia wartości wyrażenia

$$\delta = \sum_{i=1}^M \frac{A(r+s_i)}{B(r-s_i)},$$

gdzie

$$A(x) = A_p x^p + \dots + A_1 x + A_0,$$

$$B(x) = B_p x^p + \dots + B_1 x + B_0,$$

M, p - dane liczby naturalne,

A_i, B_i ($i=0,\dots,p$), $r, s_1, \dots, s_M$ - dane liczby rzeczywiste.

Jeśli $B(r-s_i)=0$ dla pewnego i , to należy program przerwać i sygnalizować błąd.

177. Na płaszczyźnie jest danych n punktów o współrzędnych (x_i, y_i) , $i=1,\dots,n$. Wydrukować współrzędne tych punktów, które leżą wewnątrz koła o danym promieniu R i środku $(0,0)$, tzn. spełniają warunek

$$x_i^2 + y_i^2 < R^2.$$

178. Dane są współrzędne n punktów leżących na płaszczyźnie. Wyznaczyć punkty leżące

a) wewnątrz kwadratu wyznaczonego przez proste:

$$y = 0, \quad x = 0, \quad y = 2, \quad x = 2.$$

b) wewnątrz trójkąta wyznaczonego przez proste:

$$y = 0, \quad y = \frac{1}{3}x, \quad y = -x + 4.$$

179. Napisać procedury sprawdzające, czy dany punkt (x, y) leży:

(1) na okręgu o środku $(0, 0)$ i promieniu R ,

(2) wewnątrz trójkąta ABC wyznaczonego przez proste

$$y = 0, \quad y = 2x, \quad y = -2x + a,$$

(a - dana liczba rzeczywista, $|a| > 1$).

W programie, dla danych n punktów, wydrukować współrzędne tych punktów, które spełniają:

a) warunek (1),

b) warunek (2),

c) warunki (1) i (2).

180. Dane a, b, c - liczby rzeczywiste, n - liczba naturalna oraz współrzędne (x_i, y_i) dla $i=1, \dots, n$.

Wydrukować numery oraz współrzędne punktów leżących na prostej

$$ax + by + c = 0.$$

181. Dana jest prosta $y = ax + b$ oraz m punktów na płaszczyźnie.

Wydrukować współrzędne oraz liczbę punktów leżących ponad tą prostą.

182. Na płaszczyźnie danych jest n punktów o współrzędnych

$$(x_i, y_i) \text{ dla } i=1, \dots, n.$$

Znaleźć parę punktów położonych najbliżej siebie oraz podać odległość między nimi.

Uwaga. Odległość dwóch punktów na płaszczyźnie jest określona wzorem

$$S_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

183. Na płaszczyźnie danych jest n punktów o współrzędnych

$$(x_i, y_i), \text{ dla } i=1, \dots, n.$$

Wydrukować współrzędne punktów położonych:

a) najwyżej,

b) najniżej,

c) najbardziej na lewo,

d) najbardziej na prawo,

e) na osi OX,

f) na osi OY.

Wyniki wydrukować w czytelnej postaci.

184. Na płaszczyźnie danych jest n punktów o współrzędnych (x_i, y_i) , dla $i=1, \dots, n$.

Podać liczbę punktów w każdej ćwiartce i wydrukować ich współrzędne, np. dla $n=6$ i punktów:

$(2, 1)$, $(-3, -8)$, $(-4, -7)$, $(5, -8)$, $(1, 1)$, $(-2, -2)$

wydruk powinien mieć postać:

CWIARTKA I

Liczba punktów 2

Współrzędne punktów:

$(2, 1)$

$(1, 1)$

CWIARTKA II

Liczba punktów 0

CWIARTKA III

Liczba punktów 3

Współrzędne punktów:

$(-3, -8)$

$(-4, -7)$

$(-2, -2)$

CWIARTKA IV

Liczba punktów 1

Współrzędne punktów:

$(5, -8)$

185. Dany jest punkt (x, y) , liczba całkowita n oraz liczby rzeczywiste a_i , b_i , c_i dla $i=1, \dots, n$.
Spośród n prostych

$$a_i x + b_i y + c_i = 0 \quad (i=1, \dots, n),$$

wyznaczyć te, na których leży punkt (x, y) . Wydrukować odpowiednie wartości i , a_i , b_i , c_i .

186. Na płaszczyźnie danych jest n punktów. Wydrukować współrzędne tych trójek punktów, które są współliniowe.

187. Na płaszczyźnie danych jest n punktów. Wydrukować współrzędne par punktów, które wyznaczają najdłuższy odcinek.

188. Dane są dwie proste o równaniach:

$$a_i x + b_i y + c_i = 0, \quad \text{dla } i=1, 2.$$

Znaleźć punkt przecięcia (x_0, y_0) tych prostych.

Uwzględnić trzy przypadki:

1. proste są równoległe, punktu przecięcia nie ma
(jeśli $W = 0$, $W_x \neq 0$, $W_y \neq 0$),
2. proste pokrywają się, punktów przecięcia jest nieskończenie wiele (cała prosta)
(jeśli $W = 0$, $W_x = 0$, $W_y = 0$),
3. proste przecinają się,

$$x_0 = \frac{b_1 c_2 - b_2 c_1}{W}, \quad y_0 = \frac{c_1 a_2 - c_2 a_1}{W}, \quad \text{jeśli } W \neq 0,$$

gdzie

$$W = a_1 b_2 - a_2 b_1, \quad W_x = b_1 c_2 - b_2 c_1, \quad W_y = c_1 a_2 - c_2 a_1.$$

6. TYP ŁAŃCUCHOWY

W wielu zadaniach operuje się na danych, które są ciągami znaków, a ich długość w czasie wykonywania programu może ulegać zmianie, np. nazwisko, stan cywilny, adres itp. Aby umożliwić posługiwanie się takimi danymi, w języku Turbo Pascal wersji 3.0 wprowadzono typ danych łańcuchowych `string`. W języku standardowym Pascal typ ten nie występuje. Dane typu `string` nazywamy zmiennymi łańcuchowymi lub łańcuchami. Zmienną łańcuchową `x` deklarujemy

```
type nazwa_typu_łańcuchowego=string[max_długość];  
var nazwa_zmiennej_łańcuchowej: nazwa_typu_łańcuchowego;
```

lub

```
var nazwa_zmiennej_łańcuchowej: string[max_długość];
```

gdzie `max_długość` jest liczbą naturalną lub stałą o wartości liczbowej z przedziału `[0,255]`.

Wartością tak zadeklarowanej zmiennej łańcuchowej może być dowolny ciąg znaków. Liczba znaków łańcucha (czyli długość łańcucha) może się zmieniać, ale nie może przekroczyć ograniczenia górnego `max_długość`.

Każda zmienna łańcuchowa zadeklarowana w powyższy sposób zajmuje $1+max_długość$ bajtów w pamięci, ponieważ w jednym (pierwszym) bajcie jest pamiętana aktualna długość łańcucha (stąd ograniczenie, że $max_długość \leq 255$).

Dana łańcuchowa (łańcuch) jest ciągiem składającym się ze znaków zawartych między parą apostrofów, znaków sterujących oraz znaków zapisanych dziesiętnie lub szesnastkowo, np.

```
'J'    {jeden znak J}  
^J     {jeden znak sterujący oznaczający nową linię}  
#74    {jeden znak o kodzie dziesiętnym 74, czyli J}  
#$4A   {jeden znak o kodzie szesnastkowym 4A, czyli J}.
```

Przykład 6.1

- Załóżmy, że zadeklarowano zmienne:

```
var długi_łańcuch: string[250];  
    nazwisko: string[20];
```


Jeżeli zmiennej nazwisko chcemy nadać wartość O'Neill, to możemy napisać

```
nazwisko:='O'Neill'
```

pamiętając, że apostrof wewnątrz łańcucha zapisujemy zawsze jako podwójny apostrof.

Zapis

```
nazwisko:=''
```

oznacza, że nazwisko jest łańcuchem pustym o długości 0 czyli, że nie zawiera żadnych znaków.

Wartością zmiennej długi_łańcuch może być ciąg zawierający maksymalnie 250 znaków. Gdyby wystąpiło więcej niż 250 znaków, wówczas znaki od 251 do końca zostaną pominięte.

Przykład 6.2

W wyniku wykonania programu

```
var l1,l2: string[4];
begin
  clrscr;
  l1:=^J; l2:=#73;
  writeln(l2,' TAK',l1,l2,' NIE')
end.
```

na ekranie pojawi się napis

I TAK

I NIE

6.1. Operacje na łańcuchach

Łańcuchy w języku Turbo Pascal wersji 3.0 można łączyć, porównywać, przepisywać w całości lub tylko pewną część, skracać przez usunięcie określonych elementów itp. Dla zmiennych łańcuchowych jest określony operator konkatenacji +. Zapis

```
napis1 + napis2
```

gdzie napis1, napis2 są łańcuchami, oznacza utworzenie łańcucha o elementach z napis1 i następujących po nich elementach z napis2, np.

```
napis1:='INFORMATYK';
```

```
napis2:='A';
```

```
wynik1:=napis1+napis2;           {wynik1='INFORMATYKA'}
wynik2:=napis2+'NN'+'+'+napis2;  {wynik2='ANNA'}
```

Łańcuchy można porównywać za pomocą operatorów relacji:

>, >=, =, < >, <=, <.

A oto przykłady, które ilustrują sposób, w jaki są określone poszczególne relacje:

Wyrażenie	Wynik
'AA'='AA'	true
'A'='AA'	false
'AB'<>'AA'	true
'morze'>'Morze'	true {bo ord('m') > ord('M')}
'KOT'<'KOTKA'	true {bo 'KOT' ma mniej znaków niż 'KOTKA'}
'123'>'1000'	true {bo ord('2')>ord('0')}

Poniżej podamy funkcje i procedury określone dla łańcuchów. Pełniejszy opis tych procedur i funkcji można znaleźć w [2].

Funkcje

concat(ł1,ł2,...,łk) - wynikiem jest łańcuch ł1+ł2+...+łk, np.
 ł1:='A'; ł2:='B';
 ł:=concat(ł1,ł2,ł2,ł1) {ł='ABBA'}

copy(ł,n,k) - wartością funkcji jest łańcuch o k elementach pobranych z łańcucha ł począwszy od miejsca n (jeśli jest to możliwe), np.

```
ł1:=copy('ASTER',2,4);           {ł1='STER'}
ł1:=copy('DOM',5,1);             {ł1='' }
ł1:=copy('PARA',3,3)             {ł1='RA' }
```

length(ł) - wartością funkcji jest liczba znaków łańcucha ł, np.
 n:=length('12'); {n=2}
 n:=length(''); {n=1}
 n:=length(''); {n=0}
 n:=length(' ') {n=1}

pos(ł1,ł2) - wynikiem jest liczba zero lub najmniejszy numer pozycji w łańcuchu ł2, od której zaczyna się podciąg identyczny z łańcuchem ł1, np.
 n:=pos('AB','ABRAKADABRA'); {n=1}
 n:=pos('W','MAGIA'); {n=0}
 n:=pos('RABA','KRAB') {n=0}

Procedury

delete(ł,n,k) - usunięcie k znaków z łańcucha ł począwszy od znaku znajdującego się na pozycji n, np.

```

ł:='KOWALSKI';
delete(ł,9,2);           {ł='KOWALSKI'}
delete(ł,6,3);           {ł='KOWAL'}
delete(ł,1,1)            {ł='OWAL'}

```

insert(ł1,ł2,n) - wstawienie do łańcucha ł2 przed znak znajdujący się na pozycji n znaków z łańcucha ł1, przy czym:

- a) n musi należeć do przedziału [1,255],
- b) jeśli $n > \text{length}(\lambda_2)$, to $n := \text{length}(\lambda_2) + 1$,
- c) jeśli wartość wyrażenia $\text{length}(\lambda_1) + \text{length}(\lambda_2)$ jest większa niż zadeklarowana maksymalna długość λ2, to przypisuje się znaki od lewej do prawej odrzucając znaki nadmiarowe.

Przykład 6.3

```

var ł: string[5];
...
ł:='A';
insert('R',ł,4);          {ł='AR'}
insert('KA',ł,3);         {ł='ARKA'}
insert('B',ł,1);          {ł='BARKA'}
insert('TOR',ł,1)         {ł='TORBA'}

```

str(wyr,ł1) lub **str(wyr:m,ł1)** lub **str(wyr:m:n,ł1)** -

przekształcenie danej reprezentowanej przez wyrażenie arytmetyczne wyr w ciąg znakowy (mający postać liczby o wartości tej danej), który zostanie przekształcony w daną łańcuchową i przypisany zmiennej ł1. W powyższych wywołaniach procedury

- m - wyrażenie typu integer, określa długość ciągu znakowego,
- n - wyrażenie typu integer, określa liczbę cyfr ułamkowych w ciągu znakowym.

Przykład 6.4

```

var x: string[4];
y: string[10];

```

Wywołanie	Ciąg	Wynik
str(17,x)	17	{x='17'}
str(17:3,x)	b17	{x=' 17'}
str(17:5,x)	bbb17	{x=' 17'}
str(17.98:6:2,y)	b17.98	{y='17.98'}
str(17.98:10:2,y)	bbbbbb17.98	{y=' 17.98'}

Uwaga. Symbol *b* oznacza spację.

val(l,x,k) - znaki łańcucha *l* są traktowane jako zapis liczby, która zostaje przypisana zmiennej *x* typu integer lub real. W łańcuchu *l* nie mogą występować spacje. Ponadto, jeśli *x* jest typu integer, to liczba reprezentowana przez łańcuch *l* też musi być typu integer. Jeśli opisane przypisanie jest wykonalne, to *k=0*, w przeciwnym przypadku *k* ma wartość numeru znaku w łańcuchu *l*, który uniemożliwił wykonanie konwersji (wówczas *x* się nie zmienia).

Przykład 6.5

```
var x,k: integer;
...
val('17',x,k);           {x=17,k=0}
val('19.75',x,k);        {x=17,k=3}
val(' 20',x,k);          {x=17,k=1}
val('21 ',x,k);          {x=17,k=3}
val('25',x,k);           {x=25,k=0}
```

Podamy teraz przykład programu, w którym wykorzystano zmienne łańcuchowe.

Przykład 6.6

Danych jest *n* nazwisk. Podać liczbę liter w każdym nazwisku. Wydrukować nazwisko najdłuższe (jeśli jest takich kilka, to pierwsze wczytane).

```
program names;
type napis=string[15];
var name,maxname: napis;
    lengthname,maxlength,i,n: integer;
    szablon: string[25];
```



```

procedure druk(x: napis; var d: integer);
begin
  d:=length(x);
  writeln('      Liczba liter: ',d:5);
  writeln
end {druk};

begin
  clrscr;
  write('Podaj liczbę nazwisk: ');
  readln(n);
  writeln;
  writeln('Podaj   1 nazwisko');
  readln(maxname);
  druk(maxname,maxlength);
  for i:=2 to n do
  begin
    writeln('Podaj ',i:3,' nazwisko');
    readln(name);
    druk(name,lengthname);
    if lengthname > maxlength
    then
    begin
      maxlength:=lengthname;
      maxname:=name
    end {if}
  end {i};

  {drukowanie wyników}
  szablon:='*';
  for i:=1 to 10 do
    szablon:=szablon+'-*';
  writeln;
  writeln;
  writeln(szablon,'WYNIKI',szablon);
  writeln;
  writeln('      Najdłuższe nazwisko: ',maxname);
  writeln('      Liczba liter: ',maxlength:5)
end.

```

6.2. Zadania

1. Danych jest n nazwisk. Uporządkować je alfabetycznie.

2. W tablicy MIASTO są zapamiętane nazwy n miast natomiast w tablicy KOD są zapamiętane kody głównych urzędów pocztowych tych miast, np.

```

MIASTO[1]='PRUDNIK',      KOD[1]='42-800',
MIASTO[2]='WARSZAWA'     KOD[2]='00-001'.

```

Utworzyć tablicę KM zawierającą nazwy miast z odpowiednimi kodami, np.

```

KM[1]='42-800 PRUDNIK',
KM[2]='00-001 WARSZAWA'.

```

3. Sprawdzić, czy łańcuch ł1 jest zawarty w łańcuchu ł2 , np.

```

ł2='LOKOMOTYWA',
ł1='LOK'           ł1 jest zawarty w ł2,
ł1='MOTYKA'       ł1 nie jest zawarty w ł2,
ł1='WAL'          ł1 nie jest zawarty w ł2.

```

4. Podać, ile razy łańcuch ł1 jest zawarty w łańcuchu ł2 .

5. Danych jest m łańcuchów. Podać:

a) długość każdego łańcucha,

b) najdłuższy łańcuch.

c) wydrukować wszystkie łańcuchy od najkrótszego do najdłuższego, każdy w nowej linii tak, aby ostatnie znaki w łańcuchach były w tej samej kolumnie (jeśli jest kilka łańcuchów tej samej długości, to drukować je w porządku alfabetycznym), np.

```

CHEMIA
FIZYKA
BIOLOGIA
ZOOLOGIA
MATEMATYKA
ELEKTRONIKA
INFORMATYKA

```

6. W pewnym wykazie osób nazwiska kobiet kończą się literą a, nazwiska mężczyzn kończą się literą i. Napisać procedurę, która przed dane nazwisko (zapamiętane jako łańcuch) dopisuje odpowiednio Pan lub Pani.

7. Napisać procedurę, która dla danego łańcucha zawierającego nazwę państwa i jego stolicy, np.

'POLSKA-WARSZAWA'

tworzy dwa łańcuchy, z których jeden zawiera tylko nazwę państwa, a drugi tylko nazwę stolicy, np.

'POLSKA', 'WARSZAWA'.

8. Danych jest n nazw miast, napisanych małą literą. Zastąpić pierwsze litery (małe) odpowiednimi wielkimi literami.

9. W tablicy NAZWISKA jest zapamiętanych n nazwisk zapisanych małymi i wielkimi literami. Wszystkie małe litery występujące w tych nazwiskach zastąpić odpowiednimi wielkimi literami. Uporządkować te nazwiska według następującej zasady: na początku umieszczamy (w porządku alfabetycznym) te nazwiska, których na daną literę jest najwięcej, a na końcu te, których najmniej. Na przykład nazwiska

KOS, MUCHA, KRET, BARAN, MAMUT, DZIK

naależy uporządkować w następujący sposób:

KOS, KRET, MAMUT, MUCHA, BARAN, DZIK.

10. Wykorzystując typ łańcuchowy zaprogramować grę w inteligencję dla dwóch graczy.

Opis gry: Jeden z graczy podaje pierwszą literę rzeczownika w liczbie pojedynczej. Następnie gracze na przemian dodają po jednej literze tak, aby powstał jakiś rzeczownik w mianowniku liczby pojedynczej (bez zdrobnień). Żaden z graczy nie wie, jaki wyraz ma na myśli drugi gracz. Który gracz poda ostatnią literę, ten przegrywa (w grze chodzi o tworzenie jak najdłuższych rzeczowników), np.

Gracz 1	Gracz 2	Tworzony wyraz
k		k
	o	ko
z		koz
	i	kozi
o		kozio
	ł	koziół

Wygrał Gracz 1.

Po zakończonej grze, program powinien podać:

- utworzony przez graczy rzeczownik i jego długość,
- numer (lub inicjały) gracza, który wygrał,
- komunikat zawierający gratulacje dla zwycięzcy i pocieszenie dla pokonanego (program powinien drukować różne komunikaty w zależności, np. od długości utworzonego wyrazu),
- komunikat z zapytaniem, czy gracze grają dalej.

W grze mogą być np. sumowane punkty przydzielane zwycięzcy (w liczbie równej długości utworzonego wyrazu) podczas kilku partii.

11. Dana jest rodzina m wyrazów pokrewnych. Znaleźć ich rdzeń (tzn. powtarzającą się część wyrazu), np. dla rodziny wyrazów:

dom, domowy, domownik, domator, domek;

rdzeniem jest dom.

12. Dany jest tekst zakończony kropką. Wydrukować wyrazy, w których litery występują w porządku rosnącym, np. dla danego tekstu:

Ach, co to za kot.

należy wydrukować wyrazy

Ach

co

kot

13. Napisać procedurę, która sprawdza, czy dany wyraz jest palindromem, tzn. czy jest to taki sam wyraz czytany od początku i od końca, np. palindromami są wyrazy:

KOK, OKO, SOS, ANNA, KAJAK, KASAK, RADAR.

14. Przeczytać n znaków i wydrukować napis

JEST WIELKA LITERA

jeśli w danym ciągu znaków występuje przynajmniej jedna wielka litera.

15. Z danego tekstu zakończonego wykrzyknikiem usunąć wybrany znak.

16. Dana jest pewna litera oraz tekst zakończony kropką. Obliczyć częstość występowania tej litery w tym tekście.

17. Napisać program czytający dany tekst zakończony kropką i redukujący ciągi kolejnych spacji do jednej spacji.

18. Dany jest tekst zakończony kropką. Pomiędzy każde dwa znaki nie będące spacją wstawić jedną spację, a występujące ciągi spacji podwoić.

19. Dany jest tekst zakończony kropką. Policzyć, ile razy występuje w nim dwugłoska cz.

20. Dany jest ciąg zdań. Wydrukować każde zdanie od nowej linii.

21. Napisać program, który czyta tekst i drukuje liczbę słów długości 1,2,...,d, gdzie d jest maksymalną długością słowa w tym tekście.

22. Dany jest tekst (składający się z wyrazów oddzielonych spacjami) zakończony kropką oraz liczba naturalna k. Wydrukować wszystkie k-literowe wyrazy tekstu.

23. Napisać program, który czyta tekst tak długo, aż napotka wyraz zawierający cztery różne litery.

24. Napisać program, który czyta ciąg znaków zakończonych kropką i drukuje liczbę liter, przecinków, spacji i cyfr w tym tekście, np. dla danego tekstu

Dodaj $x+2$, gdzie $x=5$.

należy wydrukować:

W tekście wystąpiło:

- 12 liter
- 1 przecinek
- 3 spacje
- 2 cyfry

25. Napisać program, który czyta ciąg znaków zakończony kropką i drukuje powtarzające się znaki, występujące obok siebie, np. dla danych

Inna panna, Abba, week.

należy wydrukować

nn, nn, bb, ee

26. Napisać program, który:

a) czyta n liczb naturalnych z przedziału [33,126] będących kodami znaków w kodzie ASCII,

b) dla każdej liczby wypisuje tę liczbę, znak, którego kodem jest ta liczba oraz odpowiedni napis "litera" lub "cyfra" lub "inny znak".

27. Dany jest tekst zakończony kropką. Znaleźć literę występującą w nim najczęściej.

28. Danych jest n wyrazów uporządkowanych alfabetycznie. Napisać program, który porządkuje te wyrazy alfabetycznie względem liter branych od końca wyrazu, np.

DANE	WYNIK
adres	instrukcja
bit	cyfra
cyfra	etykieta
etykieta	kod
instrukcja	adres
kod	bit

29. Dany tekst (do 40 znaków) wydrukować na środku wiersza zawierającego m znaków, gdzie m jest stałą zależną od wykorzystywanego urządzenia zewnętrznego (drukarki, monitora).

7. TYP REKORDOWY

Do opisu obiektów złożonych, których składowe są tego samego typu służą tablice. Założmy jednak, że chcemy opisać obiekt złożony, np. studenta, podając jako elementy opisu:

- nazwisko studenta - łańcuch,
- imię studenta - łańcuch,
- datę urodzenia - trzy liczby całkowite:
 dzień - z przedziału [1,31],
 miesiąc - z przedziału [1,12],
 rok - z przedziału [1900,1980],
- wydział - łańcuch,
- średnią ocen - liczba rzeczywista,
- rok studiów - liczba całkowita z przedziału [1,5],
- liczba wypożyczonych książek z biblioteki - liczba całkowita.

Do opisu takiego obiektu tablica nie byłaby strukturą odpowiednią. W takich przypadkach wykorzystujemy strukturę zwaną rekordem. Daną o nazwie *r* typu rekordowego, czyli rekord deklarujemy następująco:

```
type nazwa_typu_rekordowego=record
                                lista_pól
                                end;
var r: nazwa_typu_rekordowego;
```

lub

```
var r: record
        lista_pól
        end;
```

gdzie *lista_pól* ma postać:

```
p11, p12, ..., p1m1 : T1;
p21, p22, ..., p2m2 : T2;
...
pn1, pn2, ..., pnmn : Tn
```

przy czym $T_1, T_2, \dots, T_n$ są nazwami lub opisami typów.

Do poszczególnych pól rekordu odwołujemy się, pisząc np.

$r.p_{11}, r.p_{12}, \dots, r.p_{nm_n}$.

Przykład 7.1

Prostym przykładem typu rekordowego jest data, która może być zdefiniowana jako rekord zawierający trzy pola: dzień, miesiąc i rok.

```
type data=record
    dzień: 1..31;
    miesiąc: 1..12;
    rok: integer
end;
```

■

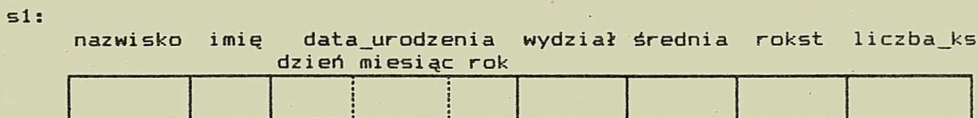
Przykład 7.2

Zdefiniujemy typ rekordowy dla opisu studenta oraz zmienną rekordową s1 typu student

```
type student=record
    nazwisko, imię: string[10];
    data_urodzenia: record
        dzień: 1..31;
        miesiąc: 1..12;
        rok: integer
    end;
    wydział: string[7];
    średnia: real;      {średnia ocen}
    rokst: 1..5;        {rok studiów}
    liczba_ks: integer  {liczba wypożyczonych książek}
end;

var s1: student;
```

Na rysunku 7.1 rekord s1 jest przedstawiony graficznie.



Rys. 7.1. Graficzne przedstawienie zmiennej rekordowej s1

Do poszczególnych pól rekordu s1 odwołujemy się, pisząc np.

```
read(s1.nazwisko);
s1.data_urodzenia.dzień:=13
```

■

Mając zdefiniowany typ rekordowy można zdefiniować na przykład zmienną typu tablicowego o elementach typu rekordowego.

Przykład 7.3

Zadeklarujemy zmienną tablicową *s* następująco:

```
var s: array[1..5] of student;
```

gdzie *student* jest typem zdefiniowanym w poprzednim przykładzie.

Obraz graficzny *s* przedstawia rysunek 7.2.

	s:	nazwisko	imię	data_urodzenia dzień miesiąc rok	wydział	średnia	rokst	liczba_ks
s[1]								
s[2]								
s[3]								
s[4]								
s[5]								

Rys. 7.2. Graficzne przedstawienie zmiennej tablicowej *s*

Do poszczególnych pól zmiennej *s* można odwoływać się następująco

```
read(s[i].data_urodzenia.miesiąc);
write(s[i].nazwisko);
s[4].rokst:=3;
s[i+j].wydział:='PPT';
```

7.1. Instrukcja wiążąca

Operując na zmiennych rekordowych często stosujemy instrukcję wiążącą *with*. Dzięki tej instrukcji programy są czytelniejsze i krótsze w zapisie, np. zamiast pisać

```
for i:=1 to 5 do
  read(s[i].nazwisko, s[i].imię, s[i].data_urodzenia.dzień,
    s[i].data_urodzenia.miesiąc, s[i].data_urodzenia.rok,
    s[i].wydział, s[i].średnia, s[i].rokst, s[i].liczba_ks);
```

można zapis znacznie skrócić, pisząc

```

for i:=1 to 5 do
  with s[i],data_urodzenia do
    read(nazwisko,imię,dzień,miesiąc,rok,wydział,średnia,rokstud,
      liczba_ks);

```

Z zamieszczonych dalej przykładów widać, że nie należy stosować tych samych nazw dla różnych zmiennych, ponieważ jeśli nawet nie ma błędów syntaktycznych, a więc składnia jest poprawna, powoduje to znaczne ograniczenia i utrudnienia w swobodnym operowaniu tymi zmiennymi.

Przykład 7.4

Dany jest fragment programu

```

var a: array[1..7] of record
      b,c: integer
    end;
    i: integer;
begin
  i:=6;
  a[i].b:=5;
  with a[i] do
    begin
      c:=17;
      i:=i+1;
      b:=4;
    end;
  ...

```

te dwa przypisania odnoszą się do zmiennej a[6]

W wyniku wykonania tego fragmentu programu zmienna a[6] otrzyma wartości a[6].b=4, a[6].c=17.

Poniżej przedstawiono graficznie zmienną tablicową a.

a:	b	c
1		
2		
3		
4		
5		
→ 6	4	17
7		

Przykład 7.5

Dany jest fragment programu:

```
var k: integer;
    r: record
        k: char;
        b: boolean
    end;
begin
    ...
    with r do read(k);
    k:=5;
    ...
```

W programie tym zadeklarowano zmienną całkowitą *k* oraz zmienną rekordową *r* o polu *k* typu znakowego. Użycie tej samej nazwy *k* w tym przypadku nie jest błędem, ponieważ pisząc w programie instrukcję

```
k:=5
```

odwołujemy się do zmiennej całkowitej, natomiast wewnątrz instrukcji *with* zmienna całkowita *k* jest niedostępna. Instrukcja

```
with r do read(k)
```

jest równoważna instrukcji

```
read(r.k)
```

i powoduje wczytanie jednego znaku.

Przykład 7.6

W programie zdefiniowano typ *termin* oraz zadeklarowano dwie zmienne rekordowe *odczyt* i *osoba* w następujący sposób:

```
type termin=record
    dz: 1..31;
    m: 1..12;
    r: integer
end (termin);

var odczyt: record
    tytuł: string[20];
    data: termin (data odczytu);
    sala: integer
end (odczyt);
```

```

osoba: record
    nazw: string[10];
    data: termin    {data urodzenia};
    tytuł: (mgr,dr,doc,prof)
end {osoba};

```

Przykładowe instrukcje dla tych zmiennych:

```

(1) odczyt.data.dz:=1;
(2) osoba.data.dz:=13;
(3) with odczyt,osoba do
    tytuł:=dr;
(4) with odczyt,osoba,data do
begin
    sala:=103;
    odczyt.tytuł:='PASCAL C';
    tytuł:=doc;
    dz:=13; m:=1; r:=1940
    {13.1.1940 jest to data urodzenia wykładowcy a nie
    termin odczytu }
end {with};

```

W instrukcji (1) i (2) wprost wskazujemy zmienne, do których się odwołujemy. W instrukcji (3) pisząc

```
tytuł:=dr
```

odwołujemy się do pola tytuł zmiennej rekordowej osoba, ponieważ ta zmienna jest zapisana jako ostatnia na liście instrukcji with. W instrukcji (4) podano kilka przykładów prawidłowych odwołań do pól zmiennych odczyt i osoba.

■

7.2. Rekordy z wariantami

Dotychczas rc. ważne rekordy miały stałą liczbę pól ustalonego typu. W języku Turbo Pascal 3.0, podobnie jak w Pascalu standardowym, mamy możliwość definiowania rekordów z wariantami, które umożliwiają interpretację zmiennych rekordowych różnymi sposobami.

Przykład 7.7

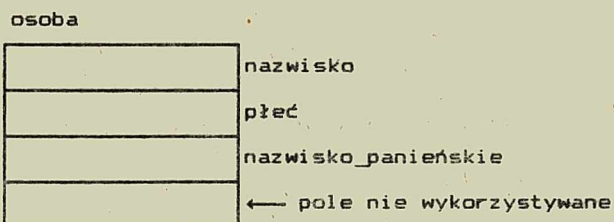
Załóżmy, że interesują nas następujące informacje o danych osobach: nazwisko i płeć oraz nazwisko panińskie, jeśli dana osoba jest kobietą, stosunek do służby wojskowej oraz cechy charakterystyczne w przypadku mężczyzny, jeśli natomiast dana osoba jest dzieckiem, to nie interesują nas żadne dodatkowe informacje.

W celu podania opisu uwzględniającego wymienione żądania możemy zadeklarować zmienną rekordową

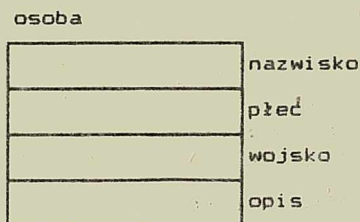
```
var osoba: record
    nazwisko: string[10];
    case płeć: (kobieta, mężczyzna, dziecko) of
        kobieta: (nazwisko_panińskie: string[10]);
        mężczyzna: (wojsko: boolean;
                    opis: (broda, wąsy, okulary, nic));
        dziecko: ();
    end;
```

W pamięci komputera jest rezerwowane miejsce na rekord zawierający maksymalną liczbę pól. Dla każdego przypadku pola z części zmiennej mają inną interpretację, np.

wariant kobieta:



wariant mężczyzna:



wariant dziecko:



Przykład 7.8

Dane są informacje o n osobach:

- nazwisko i imię,
- data urodzenia,
- liczba osób na utrzymaniu,
- stan cywilny (żonaty, mężatka, wdow, rozw, wolny),
- liczba rozwodów.

W zależności od stanu cywilnego są podane dodatkowe informacje:

- żonaty lub mężatka - data ślubu,
- wdow - data ślubu i śmierci współmałżonka,
- rozw - data ostatniego ślubu i rozwodu,
- wolny - liczba dzieci oraz informacja, czy dana osoba ma mieszkanie czy nie.

Napisać program drukujący imię, nazwisko, datę urodzenia oraz liczbę rozwodów osoby, która rozwodziła się najczęściej (jeśli jest takich kilka, to program powinien drukować informację o pierwszej z nich).

```
program rozwody;
const nmax=10;
type data=record
    dzień: 1..31;
    miesiąc: 1..12;
    rok: integer
end {data};

osoba=record
    nazwisko, imię: string[10];
    data_ur: data;
    liczba_osób: integer;
    liczba_rozw: integer;
```

```

        case stan: (żonaty,mężatka,wdow,rozw,wolny) of
            żonaty,mężatka: (data_śl: data);
            wdow: (ślub,śmierć: data);
            rozw: (ożenek,rozwód: data);
            wolny: (ma_mieszk: boolean;
                    liczba_dzieci: integer)
        end (osoba);
var wsk,nr,max,i,j,n,pom: integer;
    WYKAZ: array[1..nmax] of osoba;
    ROZWIEDZENI: array[1..nmax] of integer;
    StanCywilny: 1..5;

begin
    clrscr;
    write(' Podaj liczbę osób w wykazie: '); readln(n);
    j:=0;                (liczba osób rozwiedzionych)
    for i:=1 to n do
        with WYKAZ[i],data_ur do
            begin
                writeln('          Podaj dane ',i:3,'-ej osoby: ');
                write('          nazwisko (do 10 znaków): '); readln(nazwisko);
                write('          imię (do 10 znaków): '); readln(imię);
                write('          datę urodzenia postaci: dzień,miesiąc,rok: ');
                readln(dzień,miesiąc,rok);
                write('          liczbę osób na utrzymaniu: ');
                readln(liczba_osób);
                write('          Podaj liczbę rozwodów: '); readln(liczba_rozw);
                if liczba_rozw > 0
                then
                    begin
                        j:=j+1;
                        ROZWIEDZENI[j]:=i
                    end (if);
                repeat
                    writeln('          Podaj stan cywilny:');
                    writeln('          1-żonaty');
                    writeln('          2-mężatka ');
                    writeln('          3-wdowiec ');
                    writeln('          4-rozwiedziony ');
                    writeln('          5-wolny');
                    readln(StanCywilny)
                until (0<StanCywilny) and (StanCywilny<6);
            end
        end
    end

```

```

case StanCywilny of
  1,2: with data_sl do
    begin
      write('Podaj datę ślubu: ');
      readln(dzień,miesiąc,rok)
    end;
  3: begin
    with ślub do
      begin
        write('Podaj datę ślubu: ');
        readln(dzień,miesiąc,rok)
      end;
    with śmierć do
      begin
        write('Podaj datę śmierci: ');
        readln(dzień,miesiąc,rok)
      end
    end;
  4: begin
    with ożenek do
      begin
        write('Podaj datę ślubu: ');
        readln(dzień,miesiąc,rok)
      end;
    with rozwód do
      begin
        writeln('Podaj datę rozwodu: ');
        readln(dzień,miesiąc,rok)
      end
    end;
  5: begin
    write('Podaj 1-jeśli ma mieszkanie',
          ' 0-jeśli nie ma mieszkania: ');
    readln(pom); {zmienna pomocnicza}
    if pom=1
      then ma_mieszk:=true
      else ma_mieszk:=false;
    write('Podaj liczbę dzieci: ');
    readln(liczba_dzieci)
  end
end {case}
end {with};

```



```

{szukanie osoby, która rozwodziła się najczęściej}
if j=0
then writeln('Brak osób z rozwodami')
else begin
    writeln;
    write('  Informacje o osobie, która rozwodziła się',
          '  najczęściej:');
    nr:=ROZWIEDZENI[i];
    max:=WYKAZ[nr].liczba_rozw;
    for i:=2 to j do
    begin
        wsk:=ROZWIEDZENI[i];
        pom:=WYKAZ[wsk].liczba_rozw;
        if pom > max
        then
            begin
                nr:=wsk;
                max:=pom
            end {if}
        end {i};
    with WYKAZ[nr],data_ur do
        writeln(nazwisko,imie:11,' ur: ',dzień:2,'.',
                miesiąc:2,'.',rok:4,chr(10),chr(13),
                '  liczba rozwodów=',liczba_rozw:2,
                '  liczba osób na utrzymaniu=',
                liczba_osób:2)
    end {j>0}
end.

```

Jako ćwiczenie dla Czytelnika proponujemy napisanie programu, który:

- podaje informacje o wszystkich osobach, które rozwodziły się najczęściej,
- dla osób owdowiałych oblicza, ile dni pozostawały w związku małżeńskim,
- drukuje informacje o osobie, która ostatnia wzięła ślub,
- drukuje informacje o najstarszym bezdzietnym kawalerze z mieszkaniem,

7.3. Zadania

1. Zdefiniować typ rekordowy POJAZD o polach:

- rodzaj samochodu,
- marka,
- rok produkcji,
- numer rejestracyjny.

2. Za pomocą rekordu opisać towar, podając jego nazwę, cenę, wagę, rok produkcji, termin ważności, producenta itp.

3. W programie zadeklarowano zmienne:

```
var i: integer;
    opis: array[1..10] of record
        i: char;
        jestcyfra: boolean
    end;
```

Jaki będzie efekt wykonania poniższej instrukcji?

```
for i:=1 to 10 do
begin
    with opis[i] do
    begin
        write('Podaj znak '); readln(i);
        if ('0' <= i) and (i <= '9')
            then jestcyfra:=true
            else jestcyfra:=false;
        writeln(i)
    end {with};
    writeln(i)
end {i}
```

4. Jakie będą wartości w tablicy a po wykonaniu podanego fragmentu programu?

```
var a: array[1..5] of record
    b,c: integer
end;

i: integer;
```

```

begin
  for i:=1 to 5 do
    begin
      a[i].b:=0;
      a[i].c:=1
    end {i};
    i:=4;
    with a[i] do
      begin
        b:=3;
        i:=i+1;
        c:=19;
        b:=9
      end {with};
    ...

```

5. Dane są informacje o n osobach. Wydrukować imię, nazwisko, zawód oraz stan cywilny osób urodzonych w lipcu 1950 roku.

6. Dane są następujące informacje o n studentach: nazwisko, imię, miejsce zamieszkania, data urodzenia, wydział, średnia ocen, liczba wypożyczonych książek z biblioteki. Napisać program, który

- oblicza, ilu studentów na wydziale elektroniki ma średnią 4.0, a ilu średnią 5.0,
- drukuję nazwiska i imiona studentów zamieszkałych we Wrocławiu,
- drukuję nazwiska, imiona i wydział studentów, którzy wypożyczyli więcej niż 5 książek.

7. Dane są informacje o n studentach. Sprawdzić, czy student o nazwisku nazw₁ ma taką samą średnią ocen jak student o nazwisku nazw₂. Jeśli nie, to podać, który z nich ma wyższą średnią.

8. Dane są informacje o n osobach: nazwisko, imię, data urodzenia, płeć, stan cywilny oraz liczba dzieci. Policzyc, ile jest bezdzietnych kobiet stanu wolnego. Wydrukować ich imię, nazwisko, datę urodzenia.

9. Dane są informacje o n osobach: nazwisko, waga, wzrost, kolor oczu i włosów oraz płeć danej osoby. Napisać program, który:

- wczytuje dane do tablicy,
- drukuję liczbę kobiet o wadze ponad 80kg,
- drukuję nazwisko i wzrost niebieskookich blondynek.

10. Napisać program, który:

- wczytuje i drukuje w postaci tabeli imiona i nazwiska n sportowców oraz wyniki m konkurencji sportowych,
- wyprowadza imię i nazwisko oraz średni wynik najlepszego sportowca. Za kryterium przyjąć sumę wyników z m konkurencji.

11. Zdefiniować zmienną o nazwie FIGURA jako rekord z wariantami zawierający następujące informacje:

- promień koła, jeśli opisuje koło,
- długości dwóch boków, w przypadku prostokąta,
- długości trzech boków, jeśli opisuje trójkąt.

Napisać program, który:

- a) dla danej figury (koło, prostokąt, trójkąt) oblicza jej pole,
- b) dla danych dwóch figur sprawdza, czy te figury są kołem, kwadratem lub trójkątem równobocznym. Jeśli tak, to sprawdza, czy jedna z figur zawiera się w drugiej.

12. Typ osoba ma zawierać pola:

- nazwisko, imię, data urodzenia, płeć, liczba rozwodów,
 - stan cywilny: żonaty, mężatka, wdow, rozw, panna, kawaler
- oraz jeśli stan cywilny jest:
- żonaty lub mężatka, to datę ślubu,
 - wdow, to datę ślubu i datę śmierci współmałżonka,
 - rozw, to datę ostatniego ślubu i datę ostatniego rozwodu,
 - panna lub kawaler, to czy ma mieszkanie, czy nie, oraz liczbę dzieci.

Napisać program, który dla danych n osób drukuje:

- a) nazwisko, imię, datę urodzenia, płeć i stan cywilny osób, które wzięły ślub w maju 1985 roku.
- b) nazwisko i imię bezdzietnych kawalerów z mieszkaniem,
- c) nazwisko i imię mężatki o najdłuższym stażu małżeńskim,
- d) nazwisko i imię wdowy, która owdowiała ostatnia,
- e) informacje o osobie, która rozwiodła się najwięcej razy.

13. Dany jest wykaz dzieci z przedszkola (nazwisko, imię, średni zarobek matki, średni zarobek ojca, liczba osób na utrzymaniu rodziców).

Napisać program, który:

- a) wczytuje dane;
- b) oblicza opłatę za przedszkole według zasady:
 - (1) osoby samotne płacą ustaloną kwotę złotych,
 - (2) małżeństwa płacą 20% średniej wypadającej na osobę w rodzinie;

c) drukuje alfabetycznie nazwiska i imiona dzieci z przedszkola podając przy każdym dziecku wysokość opłaty.

14. Daną liczbę rzeczywistą można zapamiętać w postaci rekordu:

```
type lrzecz=record
    całk, ułam: 0..maxint
end;
```

Dla danego ciągu $z_1, z_2, \dots, z_n$ liczb rzeczywistych, z których każda

- a) zapamiętana jest w postaci powyższego rekordu,
- b) część ułamkową ma nie mniejszą niż 0.1,

napisz program, znajdujący liczbę największą według metody:

- (1) znajdź maksymalną część całkowitą,
- (2) jeśli kilka liczb ma taką samą część całkowitą, to wybierz spośród nich tę, która ma maksymalną część ułamkową.

15. Dane są dwie tablice STUDENT i PRACOWNIK. Elementami tablicy STUDENT są rekordy opisujące studentów (nazwisko, imię, średnia ocen), elementami tablicy PRACOWNIK – rekordy opisujące pracowników (nazwisko, imię, wynagrodzenie, premia). Obie tablice są uporządkowane alfabetycznie według nazwisk i imion. Zarówno w tablicy STUDENT jak i w PRACOWNIK nie występują dwie osoby o takim samym nazwisku i imieniu. Napisz procedurę przydzielającą 10% premię od wynagrodzenia dla każdego pracownika, który jest jednocześnie studentem i ma średnią wyższą niż 3.0.

16. Dane są:

- a) aktualny dzień, miesiąc i rok,
- b) dzień, miesiąc i rok urodzenia danych n osób.

Obliczyć wiek tych osób z dokładnością do dnia. Wydrukować ich nazwiska od najmłodszego do najstarszego. Uwzględnić w programie, że luty może mieć 28 lub 29 dni.

17. Dane są informacje o n osobach: nazwisko, numer telefonu, miejsce zamieszkania, data urodzenia, płeć. Napisz program sortujący informacje o tych osobach według wybranego pola.

18. Dane są informacje o n pracownikach: nazwisko i imię, stawka godzinowa w złotych oraz czas pracy w poszczególne dni tygodnia. Napisz program, który dla każdego pracownika oblicza całkowity czas

pracy w tygodniu oraz zapłatę według zasady: 40 godzin pracy w tygodniu jest płatnych według podanej stawki, wszystkie godziny ponad 40 są o 50% droższe.

19. Elementami tablicy PRACOWNIK są rekordy opisujące górników zatrudnionych w pewnej kopalni. Wydrukować dane o tych górnikach, którzy są braćmi lub stanowią rodzinę (ojciec i synowie).

20. Elementami tablicy LISTA są rekordy opisujące członków pewnego towarzystwa (nazwisko, imię, data urodzenia, adres, miejsce pracy, itp.). Wczytać dane do tablicy SKŁADKA następująco:

$$SKŁADKA_i = \begin{cases} 1 & \text{jeśli } i\text{-ty członek towarzystwa zapłacił składkę,} \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

Na podstawie tablic LISTA i SKŁADKA wydrukować informacje o osobach, które nie zapłaciły składki i skreślić te osoby z listy członków, tzn. z tablicy LISTA usunąć informacje o tych osobach.

8. TYP ZBIOROWY

W wielu przypadkach nie interesuje nas porządek występujących wartości (co jest istotne np. w przypadku tablic), ale jedynie wartości elementów. Mówimy wówczas o zbiorach.

Zmienną typu zbiorowego w języku Turbo Pascal 3.0 deklarujemy następująco:

```
type nazwa_typu_zbiorowego=set of T;  
var nazwa_zmiennej: nazwa_typu_zbiorowego;
```

lub

```
var nazwa_zmiennej: set of T;
```

gdzie T jest nazwą lub opisem typu porządkowego (T jest nazywany typem bazowym typu zbiorowego). Typ T nie może zawierać więcej niż 256 elementów. Ponadto, dla $x \in T$, $\text{ord}(x)$ musi należeć do przedziału $[0, 255]$.

Typ zbiorowy jest zbiorem potęgowym zbioru bazowego.

Przykład 8.1

Rozważmy typ MałyZbiór, którego zbiorem bazowym jest zbiór $\{0, 1, 2, 3\}$

```
type MałyZbiór=set of 0..3;
```

oraz zmienną x, zadeklarowaną następująco:

```
var x: MałyZbiór;
```

Typ MałyZbiór jest odpowiednikiem zbioru potęgowego, czyli zbiorem wszystkich podzbiorów zbioru liczb 0,1,2,3. Możliwymi wartościami zmiennej x są:

[] – zbiór pusty,

[0],[1],[2],[3] – podzbiory jednoelementowe zbioru bazowego
{0,1,2,3},

[0,1],[0,2],[0,3],[1,2],[1,3],[2,3] – podzbiory dwuelementowe
zbioru bazowego,

[0,1,2],[0,1,3],[0,2,3],[1,2,3] – podzbiory trzejelementowe zbioru
bazowego,

[0,1,2,3] – cały zbiór bazowy.

Typ MałyZbiór z omawianego przykładu ma 16 (czyli 2^4) możliwych wartości (podzbiorów). Ogólnie, jeśli typ bazowy ma n wartości, to typ zbiorowy ma 2^n możliwych wartości.

Zmiennym typu zbiorowego możemy nadawać pewne wartości początkowe, zmieniać te wartości oraz je porównywać. Na elementach typu zbiorowego są określone następujące operacje:

<u>operator</u>	<u>operacja</u>
+	suma zbiorów,
-	różnica zbiorów,
*	część wspólna zbiorów,
in	należenie do zbioru,
=	równość zbiorów,
<>	nierówność zbiorów,
<,<=,>=,>	zawieranie zbiorów.

Za pomocą zmiennych typu zbiorowego oraz operacji na nich określonych możemy zapisywać w TP 3.0 obliczenia teoriomnogościowe.

Przykład 8.2

Założmy, że zadeklarowano zmienne x oraz y

```
var x,y: set of 0..9;
```

Dla tych zmiennych można napisać następujące instrukcje:

$x := [];$	{ $x = \{ \}$ }
$x := x + [0];$	{ $x = \{0\}$ }
$y := x + [2];$	{ $y = \{0, 2\}$ }
if not(3 in x) then $x := x + [3];$	{ $x = \{0, 3\}$ }
$y := y * x;$	{ $y = \{0\}$ }
$y := y - [0];$	{ $y = \{ \}$ }
$x := [0..3, 6, 8..9];$	{ $x = \{0, 1, 2, 3, 6, 8, 9\}$ }
if $x < y$ then $x := y;$	{ $x = \{0, 1, 2, 3, 6, 8, 9\}$ }

Przykład 8.3

Założmy, że mamy pojemnik zawierający n piłek ($n \geq 5$) i chcemy wyciągnąć k piłek spośród nich. Szczególnie interesuje nas sposób generowania rozwiązań. Jeśli założymy, że piłki są oznaczone numerami od 1 do n , to zadanie sprowadza się do wygenerowania jednej k -elementowej permutacji ze zbioru n -elementowego.

Napišemy program generujący wszystkie k -elementowe permutacje ze zbioru n -elementowego ($n \geq k$), wykorzystując następujący algorytm:

Założmy, że mamy już wybranych 1 piłek. Jeśli

(1) $l < k$, to musimy wyciągnąć następną piłkę,

(2) $l = k$, to koniec wyciągania.

Przedstawiony algorytm jest rekurencyjny i dlatego procedura WYBIERZ w programie permutacje jest rekurencyjna.

```

program permutacje;
const nmax=20;
type licznik=0..nmax;
      zabawka=1..nmax;
      pojemnik=set of zabawka;
      tablica=array[1..nmax] of integer;
var permutacja: tablica;
      n,k: licznik;

procedure WYBIERZ(WOREK: pojemnik; n,l,r: licznik; ZBIOR: tablica);
  {procedura WYBIERZ zapisuje do tablicy ZBIOR numery piłek
   wyciąganych z n-elementowego zbioru WOREK, z którego trzeba
   wyciągnąć 1 piłek, przy czym r piłek już wyciągnięto}
var piłka: zabawka;
begin
  if r=1
  then
    {koniec wyciągania, drukowanie numerów wyciągniętych piłek}
    begin
      for piłka:=1 to l do
        write(ZBIOR[piłka]:5);
      writeln
    end {r=1}
  else
    {wyciąganie 1-r piłek}
    for piłka:=1 to n do
      if piłka in WOREK
      then
        begin
          ZBIOR[r+1]:=piłka;
          WYBIERZ(WOREK-[piłka],n,l,r+1,ZBIOR)
        end
      end
    end {WYBIERZ};
begin
  clrscr;
  writeln('Program generuje wszystkie k-elementowe permutacje ');
  writeln('ze zbioru n-elementowego');

```

```

write('Podaj liczbę elementów zbioru n=');
readln(n);
write('Podaj k=');
readln(k);
writeln;
if n>=k then
    begin
        writeln('          WYNIKI');
        writeln;
        WYBIERZ([1..n],n,k,0,permutacja)
        end {n>=k}
    else writeln('ZŁE DANE')
end.

```

Oto otrzymane wyniki dla $n=3$ i $k=3$:

Program generuje wszystkie k -elementowe permutacje
ze zbioru n -elementowego
Podaj liczbę elementów zbioru $n=3$
Podaj $k=3$

```

      WYNIKI
1      2      3
1      3      2
2      1      2
2      3      1
3      1      2
3      2      1

```

Przykład 8.4

Danych jest n znaków. Dla każdego znaku wydrukować jeden z napisów:

- litera
- cyfra
- inny znak

w zależności od tego, do której grupy znak należy.

```

program znaki;
var znak: char;
    i,n: integer;
begin
    clrscr; gotoxy(5,5);
    write(' Podaj n: '); readln(n);
    for i:=1 to n do
        begin
            gotoxy(5,5+i);

```

```

write(' Podaj znak: ');
read(znak);
if znak in ['A'..'Z','a'..'z']
  then write(' - LITERA')
  else
    if znak in ['0'..'9']
      then write(' - CYFRA')
      else write(' - INNY ZNAK')
    end if
  end if
end if
end.

```

Przykład 8.5

W grupie laboratoryjnej n studentów za pomocą 7-elementowych ciągów zerojedynkowych określa dni, w które chcieliby mieć laboratorium z informatyki, np. dane

1, 0, 0, 1, 0, 0, 0,

oznacza, że student chciałby mieć laboratorium w poniedziałki i w czwartki. Wydrukować dni odpowiadające wszystkim studentom oraz dni nie odpowiadające żadnemu studentowi.

```

program laborka;
const nmax=10;
type tydzień=(poniedz,wtorek,środa,czwartek,piątek,sobota,niedz);
  OdpowiednieDni=set of tydzień;
var dni: array[1..nmax] of OdpowiednieDni;
  {dni[i] - dni odpowiadające i-temu studentowi}
  DobreDni,ZłeDni: OdpowiednieDni;
  {DobreDni - dni odpowiadające wszystkim studentom}
  {ZłeDni - dni nie odpowiadające żadnemu studentowi}
NieTak: 0..1;
dzień: tydzień;
i,n: integer;

procedure Pisz(dz: tydzień);
begin
  case dz of
    poniedz: write(' poniedziałek ');
    wtorek : write(' wtorek ');
    środa : write(' środa ');
    czwartek: write(' czwartek ');
    piątek: write(' piątek ');

```

```

sobota: write(' sobota ');
niedz: write(' niedziela ')
end {case}
end {Pisz};

procedure Drukuj(zbiór: OdpowiednieDni);
var dzień: tydzień;
begin
  for dzień:=poniedz to niedz do
    if dzień in zbiór then
      begin
        Pisz(dzienie);
        writeln
      end {if}
    end {Drukuj};
  begin
    {czytanie danych}
    clrscr;
    write('Podaj liczbę studentów: ');
    readln(n);
    for i:=1 to n do
      begin
        writeln(' Podaj za pomocą 0 i 1 (0-nie, 1-tak)');
        write(' czy w dany dzień chcesz mieć');
        writeln(' laboratorium z informatyki:');
        dni[i]:=0;
        for dzień:=poniedz to niedz do
          begin
            Pisz(dzienie);
            readln(NieTak);
            case NieTak of
              0: ;
              1: dni[i]:=dni[i]+[dzień]
            else
              begin
                writeln;
                writeln(' ZLE PODAJESZ DANE!');
                halt
              end {else}
            end {case}
          end {dzień}
        end {i};
      end
    end
  end
end {i};

```



```

      {obliczenia}
DobreDni:=[poniedz..niedzl];
ZłeDni:=DobreDni;
for i:=1 to n do
begin
  DobreDni:=DobreDni*dni[i];
  ZłeDni:=ZłeDni-dni[i]
end {i};
      {wyprowadzanie wyników}
writeln;
if DobreDni<>[]
then begin
  writeln('Dni odpowiadające wszystkim studentom:');
  Drukuj(DobreDni)
end {DobreDni<>[]}
else writeln('Brak dni odpowiadających wszystkim studentom');
writeln;
if ZłeDni<>[]
then begin
  writeln('Nie ma chętnych na dzień:');
  Drukuj(ZłeDni)
end
else writeln('Na każdy dzień tygodnia są chętni')
end.

```

Przykład 8.6

W danym tekście zakończonym kropką, policzyć wystąpienia par małych liter, np. w tekście

Tata lata.

są następujące pary małych liter

at - 2,
ta - 2,
la - 1.

Oto program realizujący powyższe zadanie:

```

program liczpary;
type litera='a'..'z';
var LiczbaPar: array[litera,litera] of integer;
    tekst: string[20];
    i1,i2: litera;

```

```

    z1,z2: char;
    k: integer;

begin
    for l1:='a' to 'z' do
        for l2:='a' to 'z' do
            LiczbaPar[l1,l2]:=0;
        clrscr;
        writeln(' Podaj tekst (do 20 znaków):');
        readln(tekst);
        z1:=' ';
        for k:=1 to length(tekst) do
            begin
                z2:=tekst[k];
                if [z1,z2]<=['a'..'z'] then
                    LiczbaPar[z1,z2]:=LiczbaPar[z1,z2]+1;
                z1:=z2
            end {k};
        writeln;writeln;
        for l1:='a' to 'z' do
            for l2:='a' to 'z' do
                begin
                    k:=LiczbaPar[l1,l2];
                    if k<>0 then writeln(l1,l2,'-liczba wystąpień:',k:3)
                end {l2}
            end {l1};
        end.

```

8.1. Zadania

1. Dane są zbiory A, B liczb całkowitych od 4 do 10. Obliczyć

$$C = A \cup B.$$

Wydrukować elementy zbiorów A, B, C.

2. Dane są podzbiory E, F, G wielkich liter. Utworzyć zbiór

$$S = E \cup F \cup G.$$

Wydrukować elementy zbiorów E, F, G, S.

3. Dane są podzbiory A, B, C małych liter. Utworzyć zbiór

$$D = A \cap B \cap C.$$

Wydrukować elementy zbiorów A, B, C, D.

4. Dla danych dwóch zbiorów A, B liczb całkowitych z przedziału [1,20] utworzyć zbiory C, D, E, F, gdzie

$$C = A \cup B,$$

$$D = A \cap B,$$

$$E = A - B,$$

$$F = B - A.$$

Wydrukować elementy zbiorów A, B, C, D, E, F. Sprawdzić, czy

$$A \subseteq C,$$

$$D \subseteq C,$$

$$C = E \cup F \cup D,$$

$$E \cap F = \emptyset.$$

5. Znaleźć wielkie litery w danym tekście zakończonym średnikiem. Podać liczbę wielkich liter w tym tekście.

6. Dany jest tekst zakończony znakiem \$. Obliczyć częstość wystąpienia poszczególnych wielkich liter w tym tekście.

7. Dany jest tekst zakończony kropką. Wydrukować liczbę głosek występujących w tym tekście.

8. Dane są dwa zbiory A, B liczb całkowitych z przedziału [1,n] oraz funkcja $f(x)$ określona dla x należących do zbioru A. Napisać program, tworzący zbiór Z, którego elementy x spełniają relację:

$$(x \in A) \wedge (f(x) \in B).$$

Na przykład:

$$n = 10, \quad A = [2..8], \quad B = [5..10], \quad f(x) = 2x,$$

wówczas

$$Z = \{3, 4, 5\}.$$

9. Znaleźć liczby pierwsze mniejsze niż dana liczba MAXN metodą sita Eratostenesa (zob. [1] s. 155).

10. W programie zadeklarowano tablicę rekordów zawierających informacje, o miesiącach, w jakich osoba o danym nazwisku chciałaby wziąć urlop

```

type mies=(i,ii,iii,iv,v,vi,vii,viii,ix,x,xi,xii);
var osoba: record
    nazw: string[15];
    urlop: set of mies;
end;
prac: array[1..50] of osoba;

```

Napisać program, który drukuje zestawienie postaci:

MIESIĄC	NAZWISKA
styczeń	Królikowski
	Stępień
	Kowal
luty	Paluch

itd.

11. Zmodyfikować program laborka z przykładu 8.5 tak, aby:

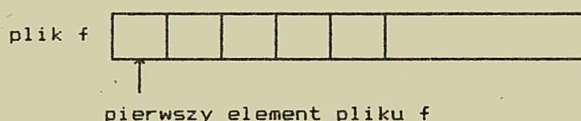
a) można było jako dane podawać nazwy dni,

b) program drukował procentowe zestawienie studentów chętnych na laboratorium w poszczególne dni tygodnia.

9. TYP PLIKOWY

Plik jest strukturą danych przypominającą taśmę magnetofonową. Elementy pliku tworzą ciąg. Dostęp do elementów jest sekwencyjny, tzn. żeby odczytać lub zapisać coś do pliku trzeba plik "przewinąć" w odpowiednie miejsce. Wszystkie elementy pliku są tego samego typu, a ich liczba jest teoretycznie nieograniczona.

W danej chwili program ma dostęp tylko do jednego elementu pliku. Graficznie plik można przedstawić następująco:



Zmienną typu plikowego deklarujemy następująco:

```
type nazwa_typu_plikowego=file of T;  
var nazwa_zmiennej: nazwa_typu_plikowego;
```

lub

```
var nazwa_zmiennej: file of T;
```

gdzie T jest nazwą lub opisem dowolnego typu z wyjątkiem plikowego.

Ze zmienną plikową wiąże się pojęcie długości pliku, tzn. liczby elementów w danej chwili wykonywania programu. Plik o długości zero nazywamy plikiem pustym.

Wszystkie operacje na plikach są wykonywane za pomocą wywołań funkcji i procedur. Należy pamiętać, że przed wykonaniem jakiegokolwiek operacji na pliku należy plik związać ze zbiorem (danych lub wyników) procedurą assign, np. jeśli plik f zadeklarowany

```
var f: file of integer;
```

chcemy związać ze zbiorem danych DANEF, to musimy napisać

```
assign(f,'DANEF')
```

Podamy teraz procedury standardowe dotyczące operacji na plikach (Turbo Pascal wersja 3.0):

`assign(f,'ZBIOR')`

- skojarzenie pliku `f` ze zbiorem o nazwie `ZBIOR`; jest wymagane, aby przed wywołaniem procedury `assign` plik `f` nie był otwarty;

`reset(f)`

- otwarcie pliku `f` (wcześniej musi być wywołana procedura `assign(f,'ZBIOR')`); dostępny staje się pierwszy element pliku;

`rewrite(f)`

- otwarcie pliku `f`; jeśli wcześniej nie był pusty, to jego zawartość zostanie skasowana. Po wykonaniu `rewrite(f)`, plik `f` jest pusty. Przed wywołaniem `rewrite(f)` należy wywołać procedurę `assign(f,'ZBIOR')`;

`read(f,x)`

- odczytanie z pliku `f` jednego elementu i przypisanie jego wartości zmiennej `x`; zmienna `x` musi być tego samego typu co elementy pliku `f`;

`write(f,x)`

- zapisanie do pliku `f` wartości zmiennej `x`; zmienna `x` musi być tego samego typu co elementy pliku `f`;

`seek(f,n)`

- ustawienie pliku `f` w takiej pozycji, że jest dostępny `n`-ty element pliku `f`, gdzie `n` jest wyrażeniem o wartości typu integer;

`close(f)`

- zamknięcie pliku `f`, jeśli był wcześniej otwarty (za pomocą `reset(f)` lub `rewrite(f)`), w przeciwnym przypadku stan pliku `f` nie zmienia się;

`rename(f,'NOWY')`

- zmiana dotychczasowej nazwy zbioru skojarzonego z plikiem `f` na nową nazwę `NOWY`, przy czym nowa nazwa nie może być nazwą żadnego zbioru już istniejącego. W chwili wywołania procedury `rename(f)` plik `f` nie może być otwarty;

`erase(f)`

- skasowanie zbioru danych skojarzonego z plikiem `f`. Zaleca się, by przed wywołaniem procedury `erase(f)` plik `f` nie był otwarty.

Dla zmiennych plikowych są określone następujące funkcje standardowe:

`eof(f)`

- (ang. end of file) przyjmująca wartości logiczne `true` i `false`:

$$\text{eof}(f) = \begin{cases} \text{true} & \text{jeśli koniec pliku } f, \\ \text{false} & \text{w przeciwnym przypadku.} \end{cases}$$

`filesize(f)`

- wynikiem funkcji jest liczba całkowita, określająca liczbę elementów pliku `f` (tzn. długość `f`); dla pliku pustego `f`
`filesize(f)=0`;

`filepos(f)`

- rezultatem funkcji jest liczba całkowita określająca numer aktualnie dostępnego elementu w pliku `f`.

Szczegółowe omówienie procedur i funkcji standardowych związanych z plikami w języku Turbo Pascal 3.0 znajduje się w [2].

Przykład 9.1

Zapisać `n` liczb całkowitych do pliku `f`, któremu odpowiada zbiór o nazwie 'DANE'.

```
program piszdane;
var f: file of integer;
    i,n,x: integer;
begin
  assign(f,'DANE');
  rewrite(f);
  clrscr;
  write('Podaj n: ');
  readln(n);
  for i:=1 to n do
  begin
    write('Podaj x: ');
    readln(x);
    write(f,x)
  end {i};
  close(f)
end.
```

Przykład 9.2

Dany jest plik o elementach całkowitych (plik ten można utworzyć wykorzystując program o nazwie pisdane z poprzedniego przykładu). Policz, ile razy występuje w tym pliku dana liczba całkowita x.

```

program zlicz;
var f: file of integer;
    x,licz,z: integer;
begin
  clrscr;
  assign(f,'DANE'); reset(f);
  if eof(f) then writeln('PLIK PUSTY')
  else
    begin
      licz:=0;
      write('Podaj x: '); readln(x);
      writeln('Elementy pliku:');
      repeat
        read(f,z); writeln('      z=',z:3);
        if x=z then licz:=licz+1
      until eof(f);
      write('liczba x=',x:3,' wystąpiła ',licz:5);
      if licz=1 then writeln(' raz') else writeln(' razy')
    end {else};
  close(f)
end {zlicz}.

```

9.1. Pliki tekstowe

Pliki tekstowe o standardowej nazwie text w języku Turbo Pascal 3.0 definiujemy następująco:

```

type nazwa_typu=text;
var nazwa_zmiennej: nazwa_typu;

lub

var nazwa_zmiennej: text;

```

Dowolny plik tekstowy składa się z różnej długości wierszy zawierających znaki. Każdy wiersz jest zakończony znakiem końca wiersza

CR/LF (carriage return/line feed). Standardowo definiuje się:

```
type text=file of char;
var input,output: text;
```

przy czym

- plik input jest standardowym plikiem wejścia, może być czytany tylko raz, nie można do niego zapisywać;
- plik output jest standardowym plikiem wyjścia, może być tylko generowany, nie można z niego czytać.

Dowolny plik tekstowy można skojarzyć ze zbiorem danych albo z urządzeniem zewnętrznym, takim jak: drukarka, konsola, terminal lub klawiatura.

Wcześniej opisane standardowe procedury i funkcje dotyczące operacji na plikach można wykorzystywać również i dla plików tekstowych. Podamy teraz zasadnicze różnice w wywołaniach procedur i funkcji, w przypadku, gdy występujący w nich parametr *f* jest nazwą pliku tekstowego:

- | | |
|--|--|
| assign(<i>f</i> , 'ZBIOR') | - ZBIOR jest nazwą zbioru lub urządzenia zewnętrznego (CON: - konsola, TRM: - terminal, KBD: - klawiatura, LST: - drukarka); |
| reset(<i>f</i>) | - <i>f</i> nie może być nazwą input ani output; jeśli plik <i>f</i> został skojarzony z urządzeniem zewnętrznym, to jest już otwarty i wywołanie tej procedury nie powoduje żadnego skutku; |
| rewrite(<i>f</i>) | - uwagi jak dla reset(<i>f</i>); |
| close(<i>f</i>) | - <i>f</i> nie może być nazwą pliku standardowego; |
| read(<i>f</i> , <i>x</i> ₁ , ..., <i>x</i> _{<i>n</i>}) | - przypisanie kolejnych wartości elementów z pliku tekstowego <i>f</i> zmiennym <i>x</i> ₁ , ..., <i>x</i> _{<i>n</i>} typu integer, real, char, lub string. Brak parametru <i>f</i> oznacza wywołanie dla pliku input; |
| readln(<i>f</i>) | - wprowadzenie jednego wiersza z pliku <i>f</i> (znaki z tego wiersza są ignorowane). Brak parametru <i>f</i> oznacza wywołanie dla pliku input; |
| readln(<i>f</i> , <i>x</i> ₁ , ..., <i>x</i> _{<i>n</i>}) | - czytanie jednego wiersza danych z pliku tekstowego <i>f</i> i przypisanie kolejnych wartości zmiennym <i>x</i> ₁ , ..., <i>x</i> _{<i>n</i>} . Brak parametru <i>f</i> oznacza wywołanie dla pliku input; |

- `write(f,x1,...,xn)` - dopisanie wartości zmiennych `x1,...,xn` (typu `integer`, `real`, `char`, `string` lub `boolean`) do pliku tekstowego `f`. Brak parametru `f` oznacza wywołanie dla pliku `output`;
- `writeln(f)` - wyprowadzenie jednego wiersza do pliku `f`. Brak parametru `f` oznacza wywołanie dla pliku `output`;
- `writeln(f,x1,...,xn)` - dopisanie jednego wiersza zawierającego wartości `x1,...,xn` do pliku `f`. Brak parametru `f` oznacza wywołanie dla pliku `output`;
- `eof(f)` - brak parametru `f` oznacza wywołanie dla pliku `input`;
- `eoln(f)` - funkcja boolowska, `eoln(f) = true`, jeśli aktualnie dostępny element pliku `f` jest znakiem końca wiersza lub gdy `eof(f) = true`. Brak parametru `f` oznacza wywołanie dla pliku `input`;
- `seekeof(f)` - funkcja boolowska zbliżona do `eof(f)`; po pominięciu najbliższych znaków spacji, tabulacji, CR i LF `seekeof(f) = eof(f)`. Brak parametru `f` oznacza wywołanie dla pliku `input`;
- `seekeoln(f)` - funkcja boolowska zbliżona do `eoln(f)`; po pominięciu najbliższych znaków spacji i tabulacji `seekeoln(f) = eoln(f)`. Brak parametru `f` oznacza wywołanie dla pliku `input`.

9.2. Pliki bez typu elementów

W języku Turbo Pascal 3.0 mogą występować pliki bez typu elementów. Są one deklarowane następująco:

```
type nazwa_typu=file;
var nazwa_zmiennej: nazwa_typu;
```

lub

```
var nazwa_zmiennej: file;
```

W tak zadeklarowanych plikach typ elementów jest bez znaczenia. Elementami pliku są bloki po 128 bajtów pamięci dyskowej. Pliki te

są wykorzystywane do niebuforowanych operacji wejścia/wyjścia wykonywanych między zmiennymi programu a zewnętrzną pamięcią dyskową.

Operacje assign, reset, rewrite, close, rename, erase, seek i eof mają taką samą interpretację jak dla plików z ustalonym typem elementów. Zamiast operacji read i write wykonuje się natomiast operacje blockread i blockwrite. Wywołania tych procedur mają postać:

```
blockread(f,x,n,licz);
blockread(f,x,n);
blockwrite(f,x,n,licz);
blockwrite(f,x,n);
```

gdzie

- f - nazwa zmiennej plikowej reprezentującej plik bez typu elementów,
- x - nazwa dowolnej zmiennej programu,
- n - wyrażenie o wartości typu integer,
- k - nazwa zmiennej typu integer.

Wynikiem wywołania procedury blockread jest wprowadzenie n bloków (po 128 bajtów każdy) z pliku f do obszaru pamięci operacyjnej zajmowanego przez zmienną x. Jeśli w wywołaniu występuje parametr k, to zostaje mu przypisana liczba wprowadzonych bloków. Znaczenie parametrów w wywołaniu procedury blockwrite jest takie jak w blockread, z tym, że dotyczy wyprowadzania z pamięci operacyjnej do pliku f (pamięci dyskowej).

9.3. Zadania

1. Dana jest liczba rzeczywista $\epsilon > 0$. Elementy ciągu $x_1, x_2, \dots$ są określone wzorem

$$x_i = \frac{i-0.1}{i^3 + |\operatorname{tg} 2i|} \quad \text{dla } i=1, 2, \dots$$

Zapisać do pliku f elementy x_i dla $i=1, \dots, k-1$, gdzie k jest najmniejszą liczbą naturalną, dla której $|x_k| < \epsilon$.

2. Dany jest plik f zawierający liczby całkowite. Utworzyć:

- a) plik h zawierający liczby niezerowe z pliku f,
- b) plik s zawierający sumy częściowe obliczane dla elementów pliku h, tzn.

$$s_i = \sum_{j=1}^i h_j,$$

gdzie h_j - j -ty element w pliku h ,

s_i - i -ty element w pliku s .

3. W pliku f są zapamiętane dowolne liczby całkowite. Do pliku g zapisać liczby parzyste występujące w pliku f , a do pliku h - liczby nieparzyste z pliku f . Zachować porządek elementów z pliku f .

4. Dane są dwa pliki f , g zawierające niemalejące ciągi liczb całkowitych. Z plików tych utworzyć plik h uporządkowany niemalejąco.

5. Napisać procedurę, która kopiuje zawartość pliku f do pliku h i podaje liczbę przekopiowanych elementów.

6. Napisać program kasujący zbiór dyskowy o podanej nazwie.

7. Dany jest plik f zawierający liczby całkowite. Wydrukować liczby, które wystąpiły w tym pliku tylko raz.

8. W danym pliku f elementy o wartościach całkowitych są zapamiętane w następującym porządku: 10 liczb dodatnich, 10 liczb ujemnych, 10 liczb dodatnich, itd. Z pliku f utworzyć plik g tak, aby było w nim:

- a) 5 liczb dodatnich, 5 liczb ujemnych, 5 liczb dodatnich, itd.
- b) 20 liczb dodatnich, 20 liczb ujemnych, 20 dodatnich, itd.

9. Elementami pliku POJAZDY są rekordy zawierające następujące informacje o samochodach:

- marka,
- rok produkcji,
- liczba wypadków,
- numer rejestracyjny.

Napisać program, który:

a) drukuje numery rejestracyjne, liczby wypadków oraz marki samochodów, które miały więcej niż pięć wypadków,

b) przepisuje informacje z pliku POJAZDY do pliku NOWY, pomijając rekordy dotyczące samochodów, które miały więcej niż k (np. $k=5$) wypadków,

c) drukuje informacje (z pliku NOWY) o samochodach wyprodukowanych przed 1980 rokiem.

10. Dany jest plik DATY, którego elementami są rekordy typu data, gdzie

```
type data=record
    dzień: 1..31;
    miesiąc: 1..12;
    rok: integer
end {data}
```

Napisać program, który znajduje:

- a) najwcześniejszy rok zapisany w pliku DATY,
- b) wszystkie daty letnie,
- c) rok, z którego zapamiętanych było najwięcej dat.

11. Elementami pliku OSOBY są rekordy zawierające nazwiska i adresy osób stojących w kolejce oraz nazwę i ilość towaru, jaki dana osoba chce kupić. Elementami pliku TOWAR są rekordy zawierające nazwę i ilości towarów, jakie są aktualnie w sprzedaży. Wydrukować:

- a) nazwiska i adresy kolejnych osób dokonujących zakupu oraz nazwy i ilości zakupionych towarów,
- b) nazwiska osób, które nadal stoją w kolejce, nazwę towaru, na który dana osoba czeka i jego ilość,
- c) nazwy i ilości towarów, które są nadal w sprzedaży.

12. Listę uczniów pewnej szkoły zawierającą następujące informacje: nazwisko i imię ucznia oraz klasa, do której uczęszcza (np. 2a), zapamiętano w pliku WYKAZ. Napisać program, który:

- a) sprawdza, czy w szkole są uczniowie o takim samym nazwisku; jeśli tak, to sprawdza, czy uczęszczają do dowolnych równoległych klas,
- b) podaje klasy, w których liczba uczniów przekracza 30,
- c) przepisuje do pliku OSTATNI_ROK informacje o uczniach z ostatniej klasy, umieszczając najpierw informacje o uczniach klasy a, potem b itd.
- d) drukuje listę uczniów wybranej klasy według określonego wzoru,
np. nazwisko imię
 nazwisko i.
 i.nazwisko

13. Standardowy plik input zawiera nazwiska osób poprzedzone odpowiednim tytułem naukowym. Każdy tytuł wraz z nazwiskiem stanowi w tym pliku oddzielny wiersz. Tytuł od nazwiska oddzielony jest jedną spacją lub kropką. Korzystając z tego pliku wydrukować nazwiska profesorów.

14. Elementami pliku PUNKTY są rekordy złożone z dwóch pól typu integer reprezentujące współrzędne punktu na płaszczyźnie. Napisać program, który czyta dane z pliku PUNKTY i zapisuje w pliku DODATNIE te przeczytane rekordy, które zawierają współrzędne punktów należących do pierwszej ćwiartki układu współrzędnych.

15. Utworzyć plik, którego elementami są rekordy zawierające informacje o mieszkaniach do wynajęcia. Napisać program drukujący adresy mieszkań jednopokojowych z kuchnią, łazienką, osobnym wejściem i w określonej cenie x złotych za miesiąc.

16. W pliku SPIS są zapamiętane numery telefonów, nazwiska, imiona i adresy osób. Napisać program, który na podstawie nazwiska, imienia i adresu podaje numer telefonu. Uwzględnić w programie możliwość dopisywania lub skreślania osób z pliku.

17. Plik PRACOWNICY zawiera następujące informacje o pracownikach zatrudnionych w danym zakładzie: nazwisko, imię, zarobek oraz opłaty jakie mają być potrącone z pensji (RTV, światło, mieszkanie, pożyczka) z uwzględnieniem, czy jest to opłata miesięczna, kwartalna czy roczna. Napisać program, który:

- a) zakłada plik PRACOWNICY,
- b) raz w miesiącu aktualizuje plik PRACOWNICY,
- c) drukuje nazwisko, imię pracownika oraz kwotę przeznaczoną do wypłaty.

18. W pliku standardowym input są zapamiętane nazwiska osób oddzielone jedną spacją od imienia. Każde nazwisko zapamiętane jest w jednym wierszu. Napisać program, który drukuje wszystkie nazwiska i imiona zaszyfrowane w następujący sposób (podstawienie Cezara):

zamiast A wydrukuj C,
zamiast B wydrukuj D, itd.

19. Elementami pliku STARYPLIK są rekordy zawierające numer porządkowy, markę i kolor samochodu oraz rok produkcji, np.

2	FIAT	czerwony	86
3	SYRENA	biały	83
4	KADA	biały	85
6	TOYOTA	metallic	87
7	TRABANT	beżowy	85

Elementami pliku TRANS są rekordy zawierające numer porządkowy, nazwę jednej z trzech operacji: ZAMIEN, USUN, WSTAW i ewentualnie markę i kolor samochodu oraz rok produkcji, np.

3 ZAMIEN OPEL zielony 87

5 WSTAW MERCEDES czerwony 86

7 USUN

Rekordy pliku TRANS opisują transakcje dotyczące samochodów z pliku STARYPLIK, np. samochód o numerze porządkowym 3 zamień na zielony Opel z 1987 roku, z numerem porządkowym 5 dopisz czerwony Mercedes z 1986 roku, usuń z pliku element o numerze porządkowym 7.

Napisz program, który:

- a) na podstawie pliku STARYPLIK i TRANS tworzy plik o nazwie NOWYPLIK, zawierający aktualne informacje o samochodach,
- b) podaje liczbę rekordów plików STARYPLIK i NOWYPLIK,
- c) drukuje liczbę operacji ZAMIEN, WSTAW i USUN wykonanych podczas uruchomienia programu.

20. Dane są dwa pliki: LICZBY i EDIT. Elementami pliku LICZBY są liczby całkowite, elementami pliku EDIT są komendy postaci:

- Z m k - zastąpienie wartości m-tego elementu w pliku LICZBY liczbą całkowitą k,
- W m k - wstawienie po m-tym elemencie w pliku LICZBY liczby k,
- U m - usunięcie m-tego elementu z pliku LICZBY.

Komendy w pliku EDIT są ustawione rosnąco według numerów elementów, których dotyczą.

Napisz program, który kopiuje plik LICZBY do pliku WYNIKI uwzględniając komendy z pliku EDIT.

21. Elementami pliku ORBIS są rekordy zawierające informacje o wczasach zagranicznych (kraj, miejscowość, termin, cena, liczba miejsc itp.). Elementami pliku KOLEJKA są informacje o osobach zgłaszających chęć wyjazdu na wczasy. Napisz program, który

- a) drukuje informacje o osobach, które wykupiły wczasy,
- b) aktualizuje plik ORBIS po każdej sprzedaży wczasów,
- c) usuwa z pliku KOLEJKA informacje o osobach, które wykupiły wczasy.

10. TYP WSKAŹNIKOWY

W poprzednich rozdziałach zostały omówione typy statyczne. Miejsce w pamięci dla zmiennych typu statycznego jest rezerwowane w czasie interpretacji deklaracji zmiennych. Takie zmienne znajdują się w pamięci komputera dopóki jest realizowana część programu, w której były zadeklarowane. Natomiast dla zmiennych dynamicznych miejsce w pamięci nie jest rezerwowane w czasie kompilacji, tylko jest przydzielane w momencie napotkania odpowiedniego żądania (podczas wykonywania programu). Do dynamicznego zarządzania pamięcią operacyjną w języku Turbo Pascal 3.0 można wykorzystać zmienne typu wskaźnikowego.

Do typu wskaźnikowego należy zbiór pewnych wartości adresów komputera oraz stała standardowa nil. Elementy typu wskaźnikowego nazywamy wskaźnikami. Wartość nil nazywamy wskaźnikiem pustym, który nie wskazuje na żadną zmienną. Typ wskaźnikowy definiujemy

```
type nazwa_typu_wskaźnikowego = ^T;
```

gdzie T jest nazwą typu (nie może być opisem typu).

Deklarację taką należy czytać następująco: "Typ *nazwa_typu_wskaźnikowego* składa się ze zbioru wartości wskazujących na zmienne typu T".

Zmienną wskaźnikową deklarujemy:

```
var nazwa_zmiennej: nazwa_typu_wskaźnikowego;
```

lub

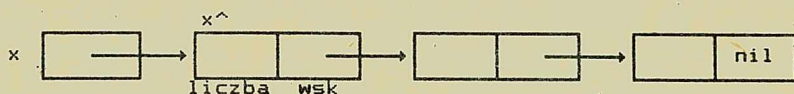
```
var nazwa_zmiennej: ^T;
```

Przykład 10.1

Prostym przykładem dynamicznej struktury danych jest lista jednokierunkowa definiowana w następujący sposób:

```
type lista = ^element;  
  element record  
    liczba: integer;  
    wsk : lista  
  end;  
var x: lista;
```


Graficznie listę jednokierunkową można przedstawić następująco:



Rys. 10.1. Graficzne przedstawienie listy jednokierunkowej

Zapis $x^{\wedge}$ oznacza zmienną, na którą wskazuje zmienna x . W tym przypadku zmienna $x^{\wedge}$ jest typu element a więc jest rekordem składającym się z dwóch pól: liczba i wsk.

Zbiór wartości typu wskaźnikowego w czasie wykonywania programu może się zmieniać. Na początku do tego zbioru należy tylko wskaźnik pusty nil. Inne elementy typu wskaźnikowego oraz zmienne wskazywane można tworzyć wywołując standardową procedurę

new

Jeżeli

```
type lista = ^T;
var q: lista;
```

to new(q) spowoduje:

- utworzenie zmiennej typu T w obszarze pamięci zwanym *stertą*; jej wartości są nieokreślone,
- utworzenie wskaźnika typu lista wskazującego na utworzoną zmienną typu T ,
- przypisanie zmiennej q wartości wskaźnika.

Poszczególne węzły listy z przykładu 10.1 można tworzyć dynamicznie przez użycie procedury standardowej new, której argumentem musi być zmienna wskaźnikowa. Każdorazowe wywołanie procedury new(x) spowoduje:

- utworzenie zmiennej typu element,
- utworzenie wskaźnika typu lista wskazującego na utworzoną zmienną,
- nadanie zmiennej x wartości wskaźnika.

Do zwalniania miejsca w pamięci wskazywanego przez q służy procedura standardowa

dispose(q)

Wywołanie procedury dispose(q) powoduje usunięcie ze zbioru wskaźników wartości zmiennej q . Wartości wszystkich zmiennych, które

były równe q , stają się wówczas nieokreślone. Obszar zajmowany przez usuwaną zmienną może być ponownie przydzielony w trakcie tworzenia innych zmiennych.

Oprócz omówionych procedur `new` i `dispose`, w języku Turbo Pascal wersji 3.0 do dynamicznego zarządzania pamięcią operacyjną, można wykorzystywać następujące procedury standardowe:

- | | |
|---------------------------|---|
| <code>mark(q)</code> | - przypisanie zmiennej wskaźnikowej q adresu szczytu sterty, |
| <code>release(q)</code> | - usunięcie ze sterty zmiennej wskazywanej przez q i wszystkich zmiennych umieszczonych po niej na sterzie, |
| <code>getmem(q,k)</code> | - zarezerwowanie na sterzie k bajtów pamięci a następnie przypisanie zmiennej q wskaźnika na początek tego obszaru, |
| <code>freemem(q,k)</code> | - zwrócenie do sterty obszaru zajmującego k bajtów pamięci o początku wskazywanym przez zmienną wskaźnikową q , |
| <code>maxavail</code> | - funkcja całkowita, której wynikiem jest rozmiar największego spójnego obszaru pamięci operacyjnej dostępnego na sterzie, wyrażony liczbą paragrafów po 16 bajtów. Jeśli <code>maxavail < 0</code> , to do otrzymanego wyniku należy dodać liczbę 65 536. |

Jeżeli w programie do utworzenia zmiennej wskazywanej wykorzystujemy procedurę `new`, to miejsce zajęte przez tę zmienną należy zwolnić procedurą `dispose`. Analogicznie należy w programie wykorzystywać parę procedur `mark` i `release`. Różnicę w wykonaniu `dispose(q)` i `release(q)` można przedstawić następująco:

sterta	dispose(q)	release(q)
x	x	x
$q^{\wedge}$	?	?
y	y	?

Przykład 10.2

Zadeklarujmy zmienne x oraz p następująco:

```

type lista = ^element;
      element = record
          var: integer;
```

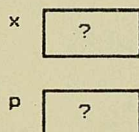
```

wsk: ^lista
end;
var x,p: lista;

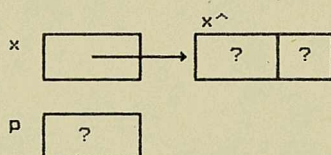
```

Na rys. 10.2 przedstawiono graficznie efekty wykonania deklaracji oraz kilku przykładowych instrukcji związanych z wyżej zadeklarowanymi zmiennymi wskaźnikowymi x, p oraz ze zmiennymi wskazywanymi. Znak zapytania ? oznacza wartość nieokreśloną. Po rysunku 10.2 znajduje się jego omówienie.

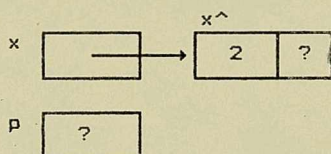
(a) var x,p: lista;



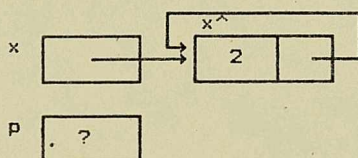
(b) new(x);



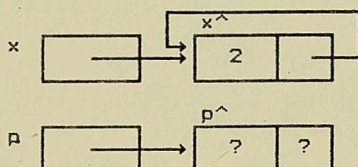
(c) x^.war:=2;



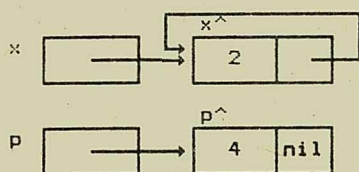
(d) x^.wsk:=x;



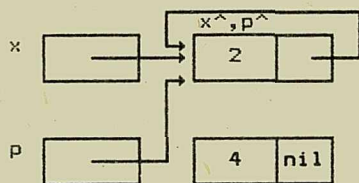
(e) new(p);



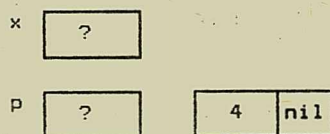
(f) $p^{\wedge}.war:=4$; $p^{\wedge}.wsk:=nil$;



(g) $p:=x$;



(h) $dispose(x)$;



Rys. 10.2. Operacje na zmiennych wskaźnikowych i wskazywanych

Omówimy pokrótce rysunek 10.2. Na początku wartości x oraz p są nieokreślone (rys. 10.2a). Element tworzony za pomocą procedury $new(x)$ oznaczamy $x^$ (rys. 10.2b). Ponieważ $x^$ jest rekordem, więc do jego pól odwołujemy się pisząc

$x^{\wedge}.war:=2$; (rys. 10.2c),

$x^{\wedge}.wsk:=x$; (rys. 10.2d).

Następnie tworzymy element $p^$ (rys. 10.2e) i nadajemy odpowiednie wartości (rys. 10.2f). Wykonując instrukcję

$p:=x$;

tracimy dostęp do poprzedniej zmiennej $p^$, przy czym miejsce w pamięci nie zostaje zwolnione. Na zmienną $x^$ wskazują teraz dwa wskaźniki: x oraz p (rys. 10.2g). Dlatego też po wykonaniu

$dispose(x)$;

zwalnia się miejsce w pamięci po zmiennej $x^$ (i jednocześnie po zmiennej $p^$) i zarówno x jak i p mają wartości nieokreślone (rys. 10.2h).

Przykład 10.3

Danych jest n liczb całkowitych. Wstawić je do listy jednokierunkowej. Wydrukować elementy listy. Wstawić element o wartości t po elemencie o wartości z (jeśli wartość z nie występuje, to element o wartości t wstawić na koniec listy).

```

program oplist;
type lista = ^węzeł;
    węzeł = record
        war: integer;
        wsk: lista
    end;
var x, p, q: lista;
    z, i, n, t: integer;

procedure druk(x: lista; n: integer);
var p: lista;
    i: integer;
begin
    if x = nil
    then writeln(' LISTA PUSTA')
    else begin
        p := x;
        for i := 1 to n do
            begin
                write(p^.war:5);
                p := p^.wsk
            end {i}
        end {else}
    end {druk};
begin
    write('Podaj n: ');
    readln(n);
        {tworzenie listy}
    if n > 0
    then
        begin
            new(x);
            writeln('Podawać elementy listy od końca');
            readln(t);
            x^.war := t;
            x^.wsk := nil;

```

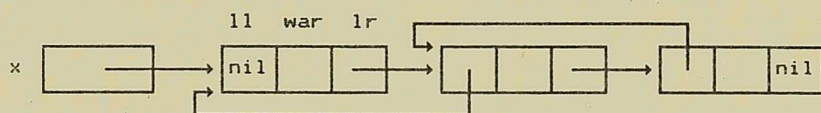
```

    for i:=2 to n do
    begin
        p:=x;
        new(x);
        x^.wsk:=p;
        readln(t);
        x^.war:=t
    end {i}
end {n>0}
else x:=nil;
    {drukowanie listy}
writeln('Elementy listy:');
druk(x,n);
    {wstawienie elementu t po elemencie z}
writeln;
write('Podaj wartość, którą chcesz wstawić: ');
readln(t);
if x=nil
then
begin
    new(x);
    x^.war:=t;
    x^.wsk:=nil
end {x=nil}
else
begin
    write('Podaj wartość, po której wstawić: ');
    readln(z);
    p:=x;
    while (p^.war<>z) and (p^.wsk<>nil) do p:=p^.wsk;
    new(q);
    q^.wsk:=p^.wsk;
    p^.wsk:=q;
    q^.war:=t
end {x<>nil};
    {drukowanie listy po wstawieniu jednego elementu}
writeln('Elementy listy po wstawieniu:');
druk(x,n+1)
end.

```

W podanych dotychczas przykładach każdy element listy ma jeden wskaźnik wyznaczający element następny. Zdefiniujemy teraz listę, w

której każdy element powiązany jest z elementem poprzednim i następnym. Jest to tzw. lista dwukierunkowa. Graficznie można ją przedstawić następująco:



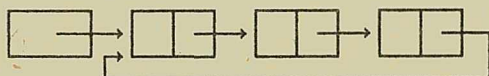
Rys. 10.3. Graficzne przedstawienie listy dwukierunkowej

Elementami powyższej listy dwukierunkowej są rekordy o trzech polach: war (wartość), ll (wskaźnik do poprzedniego elementu) oraz lr (wskaźnik do następnego elementu). W celu utworzenia takiej listy należy zdefiniować:

```
type lista = ^elem;
  elem = record
    war: integer;
    ll, lr: lista;
  end;
var x: lista;
```

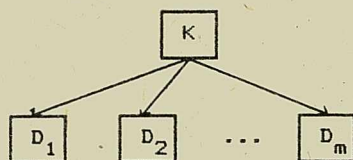
Listy mogą być wykorzystywane do realizacji struktur o specjalnym przeznaczeniu. Jeżeli na przykład elementy wstawiane są jedynie na koniec listy, a usuwane są zgodnie z zasadą: ostatni wstawiony - pierwszy usuwany, to taką strukturę nazywamy stosem.

Innym zastosowaniem listy jest kolejka, tj. lista, do której elementy są wstawiane z jednego końca, a usuwane z drugiego (pierwszy wstawiany - pierwszy usuwany). Mówiąc o listach należy również wspomnieć o cyklicznych listach jednokierunkowych, w których ostatni element zawiera wskaźnik na pierwszy element. Graficznie cykliczną listę jednokierunkową można przedstawić następująco:

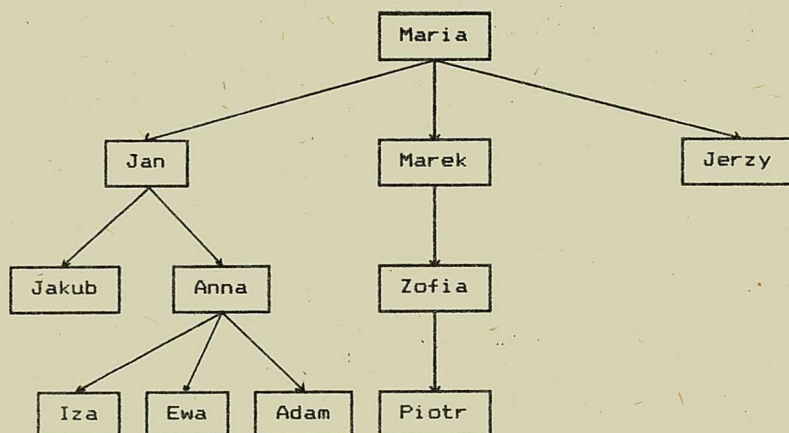


Omówmy jeszcze jedną strukturę danych a mianowicie drzewa, których szczególnym przypadkiem są drzewa binarne.

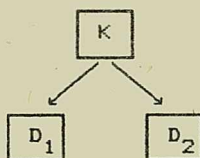
Drzewo jest to zbiór skończony D elementów (węzłów). Jeśli zbiór D nie jest pusty, to składa się z wyróżnionego węzła K zwanego korzeniem i rozłącznych podzbiorów $D_1, D_2, \dots, D_m$, które są drzewami. Podzbiory $D_1, D_2, \dots, D_m$ nazywane są poddrzewami drzewa D . Podana definicja ma charakter rekurencyjny. Graficznie drzewo można przedstawić w następujący sposób:



Typowym przykładem drzewa jest drzewo genealogiczne, np.



Jeżeli każdy węzeł drzewa ma co najwyżej dwa poddrzewa, to mówimy, że jest to drzewo binarne. Niepuste drzewo binarne można graficznie przedstawić następująco:



Rys. 10.4. Graficzne przedstawienie drzewa binarnego

gdzie

K - korzeń drzewa,

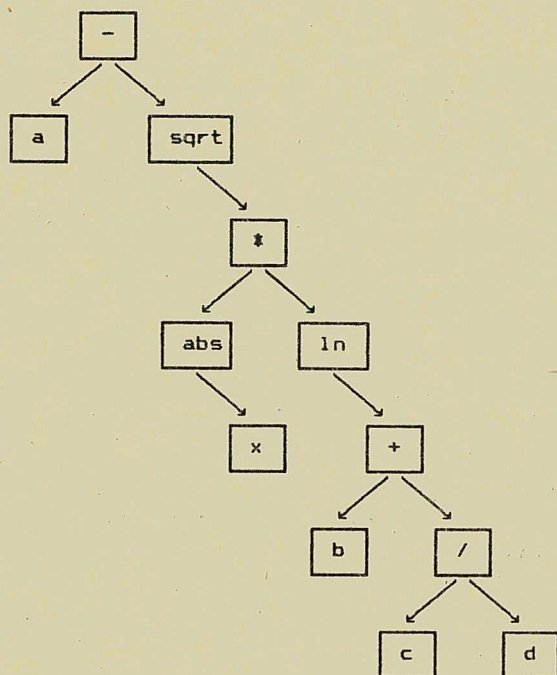
D_1 - lewe poddrzewo,

D_2 - prawe poddrzewo.

Dowolne wyrażenie arytmetyczne można zapisać w postaci drzewa binarnego, np. dla wyrażenia arytmetycznego

$$a - \sqrt{\text{abs}(x) * \ln(b + \frac{c}{d})}$$

drzewo binarne ma postać



Wiele zadań wiąże się z przeszukiwaniem drzew, które musi odbywać się w ustalonym porządku. Wyróżnia się trzy podstawowe porządki:

- (1) poprzeczny: D_1 , K , D_2 ,
- (2) wzdłużny: K , D_1 , D_2 ,
- (3) wsteczny: D_1 , D_2 , K .

Powyższy zapis należy rozumieć następująco (np. dla porządku poprzecznego): w porządku poprzecznym przeszukujemy kolejno:

- lewe poddrzewo,
- korzeń,
- prawe poddrzewo.

Przykład 10.4

Utworzyć drzewo binarne zawierające liczby całkowite według następującej metody: pierwszą liczbę wstawić do korzenia, liczby mniejsze wstawiać do lewego poddrzewa, liczby większe do prawego, a liczby występujące już w drzewie nie wstawiać.

Wydrukować węzły drzewa binarnego

- a) w porządku poprzecznym,
- b) w porządku wzdłużnym,
- c) w porządku wstecznym.

```

program drzewa;
type drzewo = ^liść;
    liść = record
        wartość: integer;
        lewe,prawe: drzewo
    end (liść);
var d: drzewo;
    i,n,w: integer;

procedure budujdrzewo(var p: drzewo; k: integer);
begin
    if p=nil
    then begin
        new(p);
        with p^ do
            begin
                wartość:=k; lewe:=nil; prawe:=nil
            end (with)
        end (p=nil)
    else
        if k<p^.wartość
        then budujdrzewo(p^.lewe,k)
        else if k>p^.wartość then budujdrzewo(p^.prawe,k)
        end (budujdrzewo);
    end (budujdrzewo);

procedure porzpoprzeczny(p: drzewo);
begin
    if p<>nil then
        begin
            write(p^.wartość:5);
            porzpoprzeczny(p^.lewe);
            porzpoprzeczny(p^.prawe)
        end
    end (porzpoprzeczny);

procedure porzwzdłużny(p: drzewo);
begin
    if p<>nil then
        begin
            porzwzdłużny(p^.lewe);
            write(p^.wartość:5);
            porzwzdłużny(p^.prawe)
        end
    end (porzwzdłużny);
end (porzwzdłużny);

```

```

procedure porzwsteczny(p: drzewo);
begin
  if p<>nil then
    begin
      porzwsteczny(p^.lewe);
      porzwsteczny(p^.prawe);
      write(p^.wartosc:5)
    end
  end {porzwsteczny};
begin
  {tworzenie drzewa}

  d:=nil;
  write('Podaj n: ');
  readln(n);
  writeln('Podaj elementy drzewa:');
  for i:=1 to n do
    begin
      readln(w);
      budujdrzewo(d,w)
    end {i};
    {wyprowadzanie wyników}
    writeln;
    writeln('Elementy drzewa w porządku poprzecznym:');
    porzpoprzeczny(d);
    writeln;
    writeln('Elementy drzewa w porządku wzdłużnym:');
    porzwzdłużny(d);
    writeln;
    writeln('Elementy drzewa w porządku wstecznym:');
    porzwsteczny(d)
  end.

```

Dla $n=8$ i danych liczb: 4,8,3,5,2,1,7,6 otrzymano następujące wyniki:

Elementy drzewa w porządku poprzecznym:

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

Elementy drzewa w porządku wzdłużnym:

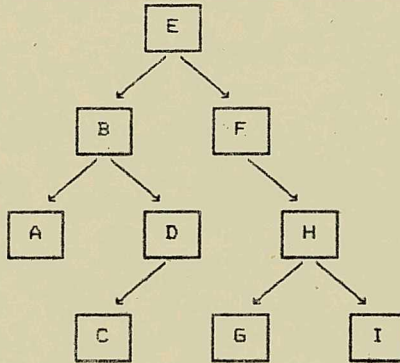
4	3	2	1	8	5	7	6
---	---	---	---	---	---	---	---

Elementy drzewa w porządku wstecznym:

1	2	3	6	7	5	8	4
---	---	---	---	---	---	---	---

Przykład 10.5

Dane drzewo binarne



można zapisać, np.

(a) w postaci nawiasowej:

E(B(A(,),D(C,)),F(,H(G,I)))

(b) w postaci wcięć:

E

B

A

D

C

F

H

G

I

Napisz program, który wczytuje drzewo binarne postaci (a) i drukuje w postaci (b).

```

program bin;
var drzewo: string[30];
    znak: char;
    liczba_spacji,i,j: integer;
begin
  clrscr;
  gotoxy(5,5);
  writeln('Podaj drzewo binarne w postaci nawiasowej:');
  readln(drzewo);
  liczba_spacji:=0;
  writeln;
  writeln('          WYNIKI');

```



```

for i:=1 to length(drzewo) do
begin
  znak:=drzewo[i];
  case znak of
    'A'..'Z','a'..'z': begin
      for j:=1 to liczba_spacji do
        write(' ');
        writeln(znak)
      end {litera};
    ' ': liczba_spacji:=liczba_spacji+1;
    ')': liczba_spacji:=liczba_spacji-1
  end {case}
end {i}
end.

```

10.1. Zadania

1. Napisać procedury dla liniowej listy jednokierunkowej realizujące następujące zadania:

- dołączanie jednego elementu na końcu listy,
- wstawianie jednego elementu na początek listy,
- wstawianie jednego elementu x po elemencie z ,
- usuwanie pierwszego elementu z listy,
- usuwanie ostatniego elementu z listy,
- usuwanie n -tego elementu listy,
- usuwanie elementu listy o wartości x ,
- usuwanie z listy elementu znajdującego się przed elementem x .

2. Napisać procedury realizujące następujące zadania:

- łączenie dwóch list,
- zwalnianie wszystkich węzłów listy,
- usuwanie co drugiego elementu listy,
- łączenie dwóch uporządkowanych rosnąco list w jedną listę uporządkowaną rosnąco,
- tworzenie części wspólnej dwóch list,
- tworzenie sumy dwóch list,
- obliczanie sumy elementów o określonych wartościach

- h) obliczanie liczby elementów listy,
- i) zamiana i-tego i k-tego elementu listy,
- j) kopiowanie listy,
- k) drukowanie listy od początku,
- l) drukowanie listy od końca.

3. Napisać procedurę

```
testq(q,x,zm_log),
```

która dla danej listy jednokierunkowej q o elementach typu T oraz dla zmiennej x typu T nadaje zmiennej logicznej zm_log wartość wyznaczoną według wzoru:

$$zm_log = \begin{cases} \text{true} & \text{jeśli lista } q \text{ jest pusta lub } x \text{ występuje w } q, \\ \text{false} & \text{w przeciwnym przypadku.} \end{cases}$$

4. Napisać procedurę, która z danej listy jednokierunkowej o elementach typu

```
type element=record
    / war: integer;
    klucz: lista
end (element)
```

gdzie

```
type lista=^element;
```

usuwa elementy o wartości minimalnej.

5. Dana jest lista jednokierunkowa p . Utworzyć listę jednokierunkową q , zawierającą elementy listy p uporządkowane rosnąco.

6. Zmienić porządek listy jednokierunkowej na przeciwny.

7. Dana jest liniowa lista jednokierunkowa. Sprawdzić, czy zawiera wartość x . Jeśli nie, dopisać x na końcu listy.

8. Utworzyć n -elementową tablicę A zawierającą adresy komórek, w których są zapamiętane liczby całkowite. Obliczyć sumę i iloczyn tych liczb.

9. Napisać procedurę, która dla danej listy jednokierunkowej zadeklarowanej jako

```

type lista=^elem;
  elem=record
    war: integer;
    wsk: lista
  end (elem);

```

drukuje liczby, występujące w liście dokładnie dwa razy.

10. Utworzyć listę dwukierunkową.

11. Dla danej listy dwukierunkowej zdefiniowanej jako

```

type lista=^elem;
  elem=record
    war: integer;
    poprz,nast: lista
  end (elem);

```

napisać procedury:

- usuwania z listy elementów o danej wartości,
- dołączania jednego elementu na początek listy,
- obliczania, ile elementów listy ma daną wartość k ,
- usuwania elementów o wartości maksymalnej,
- porządkowania elementów listy rosnąco,
- drukowania elementów listy od początku,
- drukowania elementów listy od końca.

12. Dane są dwa wielomiany trzech zmiennych x, y, z :

$v(x,y,z)$, $w(x,y,z)$.

Każdy jednomian postaci $cx^i y^j z^k$ ($c > 0$, $i, j, k \geq 0$) zapamiętujemy w postaci rekordu

c		
i	j	k
łącznik		

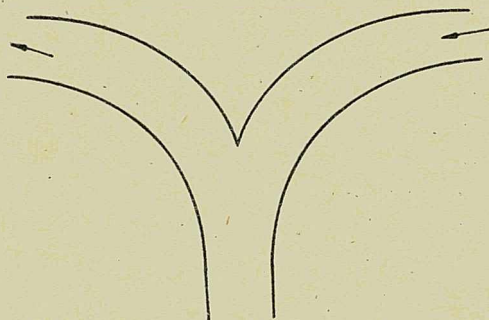
Jednomiany są uporządkowane najpierw według malejących potęg x , następnie względem potęg y , a na końcu z , np.

$$v(x,y,z) = 3x^3 y^4 + 2x^2 y^5 z - y^2 z^3 + z.$$

Obliczyć współczynniki wielomianu $v(x,y,z) + w(x,y,z)$.

13. Zmodyfikować program z przykładu 10.5 tak, by z wyników można było odczytać, czy wydrukowane poddrzewo jest lewym, czy prawym poddrzewem, (z postaci (b) nie wynika, np. czy C jest lewym, czy prawym poddrzewem D):

14. Na wejściu rozjazdu kolejowego przedstawionego poniżej znajduje się n wagonów w kolejności $1, 2, \dots, n$.



Oznaczmy operację pobrania wagonu z wejścia przez D (dołączenie do stosu), natomiast przez U przekazanie na wyjście (usunięcie ze stosu).
Odpowiedzieć na następujące pytania:

1. Jaki ciąg operacji (tzn. ciąg składający się z liter U i D) należy wykonać, aby wagony na wyjściu (dla $n=4$) były ustawione w kolejności 2,4,3,1?

2. Czy na wyjściu takiego rozjazdu można uzyskać taką samą kolejność, co na wejściu? Jeśli tak, to podać odpowiedni ciąg operacji.

3. Czy z ustawienia 1,2,3,4,5 można otrzymać ustawienie

2,3,4,5,1;

1,5,4,2,3;

3,2,5,4,1?

Jeśli tak, to podać odpowiednie ciągi operacji.

15. Dla danego zbioru Z liczb całkowitych utworzyć drzewo binarne, a następnie wydrukować jego elementy w porządku

a) rosnącym,

b) malejącym.

16. Napisać procedurę (z jednym parametrem - wskaźnikiem na korzeń drzewa) zwalniającą wszystkie węzły drzewa binarnego.

17. Poziom korzenia drzewa określamy jako 0. Poziom każdego innego węzła określamy jako poziom o jeden większy niż poziom jego rodzica. Wysokością drzewa jest poziom węzła o najwyższym poziomie. Napisz procedurę:

- a) znalezienia poziomu określonego węzła w drzewie,
- b) znalezienia wysokości drzewa.

18. Wagą drzewa nazywamy liczbę jego węzłów. Napisz procedurę obliczania wagi danego drzewa binarnego.

19. Liściem drzewa nazywamy węzeł, którego wszystkie poddrzewa są puste. Napisz procedurę, która oblicza liczbę liści danego drzewa binarnego.

20. Załóżmy, że drzewo binarne ma węzły postaci

```
type węzeł=record
    k: integer;
    lewe,prawe: drzewo
end;
```

gdzie

```
type drzewo=^węzeł;
```

Napisz procedurę

```
delete(drzewo,k1,k2)
```

usuwania z drzewa wszystkich węzłów, dla których $k_1 \leq k \leq k_2$.

21. Dwa drzewa binarne są symetrycznie podobne, jeśli:

- a) obydwa są puste

lub

b) obydwa są niepuste i lewe poddrzewo pierwszego jest symetrycznie podobne do prawego poddrzewa drugiego drzewa.

Napisz procedurę sprawdzającą, czy dwa drzewa są symetrycznie podobne.

22. Dwa drzewa binarne są podobne, jeśli:

- a) obydwa są puste

lub

b) obydwa są niepuste, ich lewe poddrzewa są podobne i ich prawe poddrzewa są podobne.

Napisz procedurę sprawdzającą, czy dwa drzewa binarne są podobne.

11. ZADANIA RÓŻNE

1. Wykorzystując procedurę obliczania minimum w n -elementowym ciągu liczb rzeczywistych, obliczyć minimum i maksimum m -elementowego ciągu x .

2. Niech

```
var T: array[1..10] of string[15];
```

Jaka będzie postać wyników po wykonaniu instrukcji:

a) for $i:=1$ to n do writeln(' ':5,T[i])

b) for $i:=1$ to n do writeln(T[i]:20)

3. Jaką wartość będą miały elementy tablicy A po wykonaniu następującego fragmentu programu?

```
program haczyk;
var i: integer;
    A: array[1..10] of integer;

procedure xplus10(var x: integer);
var i: integer;
begin
    for i:=1 to 10 do x:=x+1
end {xplus10};

begin
    for i:=1 to 10 do A[i]:=0;
    i:=1;
    xplus10(A[i]);
    for i:=1 to 10 do
        writeln(' A[' ,i:2,' ]= ',A[i]:2)
    end.
```

4. Dana jest liczba naturalna n . Obliczyć 2^n za pomocą mnożenia. Rozpatrzyć n parzyste i nieparzyste.

5. Dana jest macierz D - $n \times n$ o elementach całkowitych. Zastąpić jedynkami te elementy macierzy D , które leżą na głównych przekątnych.

Wydrukować macierz D przed i po zamianie.

6. Napisać procedurę, która dla danej macierzy $X = m \times n$ oblicza iloczyn tych elementów x_{ij} , dla których $i+j$ jest liczbą parzystą. W programie obliczyć takie iloczyny dla danych macierzy $A, B = m \times n$. Wydrukować macierze A, B oraz obliczone iloczyny.

7. Dane są liczby naturalne n, m takie, że n dzieli się przez m . Utworzyć macierz $A = n \times m$ postaci

$$A = \begin{bmatrix} \begin{array}{ccc|ccc} 1 & 2 & & & & \\ & 2 & & & & \\ & & 2 & & & \\ \hline 1 & & & 1 & & \\ & 1 & & 2 & & \\ & & 1 & & 2 & \\ \hline & & & & 1 & \\ & & & & & 1 \\ & & & & & & 1 \end{array} & \begin{array}{c} \overbrace{\hspace{1cm}}^m \\ \vdots \\ \underbrace{\hspace{1cm}}_m \end{array} \end{bmatrix}^n$$

Pozostałe elementy macierzy są zerami.

8. Wypisać wszystkie czynniki pierwsze danej liczby naturalnej n , np. $792 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 11$.

9. Dla danych dwóch liczb a, b znaleźć największy wspólny dzielnik i najmniejszą wspólną wielokrotność.

10. Obliczyć największy wspólny dzielnik danych n liczb naturalnych.

11. Obliczyć najmniejszą wspólną wielokrotność danych n liczb naturalnych.

12. Dana jest tablica $A = n \times n$ o elementach całkowitych. Sprawdzić, czy elementy tablicy spełniają następujące warunki:

- (1) $a_{ij} \in \{1, 2, \dots, n\}$ dla $i, j = 1, \dots, n$;
- (2) $a_{ij} \neq a_{ik}$ dla $i, j, k = 1, \dots, n$ oraz $j \neq k$;
- (3) $a_{ij} \neq a_{kj}$ dla $i, j, k = 1, \dots, n$ oraz $i \neq k$.

13. Napisać program wyznaczający wartość funkcji L określonej wzorem:

$$L(p, q) = \begin{cases} q & \text{dla } p=\text{true}, \\ \sim p & \text{dla } q=\text{true}, \\ \text{true} & \text{dla } p=\text{false}, \\ \text{false} & \text{dla } q=\text{false}. \end{cases}$$

14. Dla liczby rzeczywistej x jest określona funkcja

$$f(x) = \begin{cases} (x-1)^2 - (x^2 + x^3)^2 & \text{jeśli } x < -5, \\ 0 & \text{jeśli } x = -5, \\ (x-1)(x+5)(x^2+1)^3 & \text{jeśli } x > -5. \end{cases}$$

Obliczyć $f(x)$ dla $x = -6, -5, -4, \dots, 4, 5, 6$.

15. Dane $n, (x_i, y_i)$ dla $i=1, \dots, n$. Obliczyć

$$H(x, y) = \begin{cases} x+y & \text{dla } x > 1 \text{ i } |y| \geq 2, \\ x-y & \text{dla } |x| \leq 1 \text{ i } 0 < y < 2, \\ y-x & \text{dla } |x| \leq 1 \text{ i } -2 < y < 0, \\ -y & \text{dla } |x| > 1 \text{ i } y = 0, \\ x & \text{w pozostałych przypadkach.} \end{cases}$$

dla $x_i, y_i, i=1, \dots, n$.

16. Stablicować funkcję

$$f_n(x) = \begin{cases} 1 + \sum_{i=1}^n \frac{x^i}{i!} & \text{dla } x > 0, \\ \prod_{i=1}^n \frac{|x|+i}{\ln(|x|+2i)} & \text{dla } x \leq 0. \end{cases}$$

dla $n=1, 2, \dots, 20$ oraz $x = -1, -\frac{1}{2}, 0, \frac{1}{2}, 1$. Wyniki wydrukować w czytelnej postaci.

17. Funkcja $\lambda(n)$ jest zdefiniowana następująco:

$$\lambda(n) = \begin{cases} 0 & \text{dla } n=1, \\ \lambda(\frac{1}{2}n)+1 & \text{dla } n=2, 4, \dots \\ \min\{\lambda(\frac{1}{2}(n-1)), \lambda(\frac{1}{2}(n+1))\}+2 & \text{dla } n=3, 5, \dots \end{cases}$$

Obliczyć i wydrukować $\lambda(m)$ dla $m=1, 2, \dots, 10$.

18. Dane są liczby rzeczywiste λ, q ($0 \leq q \leq 1, \lambda \geq 0$), oraz liczba naturalna n . Obliczyć

$$P_{j,k} = e^{-\lambda} * \sum_{i=0}^{\min(j,k)} \binom{j}{i} (1-q)^i q^{j-i} \frac{\lambda^{k-i}}{(k-i)!}$$

dla $j, k = 1, \dots, n$.

Wyniki wydrukować w postaci:

$$\begin{matrix} p_{11} & p_{12} & \dots & p_{1n} \\ \vdots & \vdots & & \vdots \\ p_{n1} & p_{n2} & \dots & p_{nn} \end{matrix}$$

19. Dana jest liczba naturalna n oraz ciąg $a_1, \dots, a_n$ ($a_i \neq a_j$ dla $i \neq j$, $i, j = 1, \dots, n$). Obliczyć

$$S = \sum_{i=1}^n \frac{f(a_i)}{\prod_{\substack{j=1 \\ j \neq i}}^n (a_i - a_j)}$$

gdzie

$$f(x) = \begin{cases} e^x & \text{dla } x < -1, \\ \frac{4}{5} & \text{dla } -1 \leq x \leq 1, \\ \cos \sqrt[3]{x} & \text{dla } x > 1. \end{cases}$$

20. Dane są liczby $n, x_1, \dots, x_n$. Obliczyć

$$S = \sum_{i=1}^n \left(\frac{f(x_i)}{i!} \sum_{j=1}^n \frac{f(x_j) + i + j}{j} \right)$$

gdzie

$$f(x) = \begin{cases} \sin\left(\frac{x}{2}\right) & \text{dla } x \leq 0, \\ \frac{\cos(x)}{\ln(x^2) + 1} & \text{dla } 0 < x < 5, \\ x + \frac{5}{2} & \text{dla } x \geq 5. \end{cases}$$

21. Obliczyć wartość G określoną wzorem:

$$G = \begin{cases} \sum_{i=1}^n \frac{a_i}{i!} & \text{jeśli } n < k, \\ \frac{8}{3} & \text{jeśli } n = k, \\ \prod_{i=1}^k (a_i + b_i) & \text{jeśli } n > k, \end{cases}$$

gdzie dane są liczby: $n, k, a_1, \dots, a_n, b_1, \dots, b_k$.

22. Obliczyć przybliżoną wartość całki

$$I = \int_{10}^{20} \frac{dx}{x}$$

stosując wzór

$$\int_{x_0}^{x_n} f(x) dx \approx \sum_{i=1}^{n-1} \frac{h}{2} (f(x_i) + f(x_{i+1})),$$

gdzie $h = x_{i+1} - x_i$. Przyjąć $h = 0.01$.

23. Obliczyć wartość całki metodą trapezów

$$\int_a^b f(x) dx \approx \frac{b-a}{2n} (f(a) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b)).$$

Przyjąć, np. $a=0$, $b=\pi/2$, $f(x) = \frac{\sin(x)}{x}$, $n=25$.

24. Dla danej liczby rzeczywistej c oraz danych argumentów $x_i = i \frac{\pi}{4}$ dla $i=1, \dots, n$ obliczyć wartość funkcji

$$f(x_i) = \frac{\sin(c+x_i) + \cos(c-x_i)}{\sin(c+x_i) * \cos(c-x_i)}.$$

Wyniki wydrukować w czytelnej postaci.

25. Obliczyć wartość funkcji

$$f(x) = \frac{10}{\sqrt{\pi}} \sin(x) (1 - \cos(x))$$

dla x zmieniającego się z krokiem h w przedziale $[a, b]$.

26. Obliczyć

$$\sum_{i=1}^4 k_i (x+y)^i,$$

gdzie:

$$k_1 = f(x, y),$$

$$k_2 = f\left(\frac{1}{2}x, y+k_1\right),$$

$$k_3 = f\left(\frac{1}{4}x, y+k_1+k_2\right),$$

$$k_4 = f\left(\frac{1}{8}x, y+k_1+k_2+k_3\right).$$

Przyjąć

$$f(x,y) = \frac{\sin(x+y) * \cos(x-y)}{\sin(x+y) + \cos(x-y)}.$$

27. Obliczyć

$$\sum_{i=1}^4 k_i(x+y),$$

gdzie:

$$k_1 = f(x,y)$$

$$k_2 = f\left(\frac{1}{2}x, y + \frac{1}{2}k_1\right),$$

$$k_3 = f\left(\frac{1}{4}x, y + \frac{1}{2}k_1 + \frac{1}{4}k_2\right),$$

$$k_4 = f\left(\frac{1}{8}x, y + \frac{1}{2}k_1 + \frac{1}{4}k_2 + \frac{1}{8}k_3\right).$$

Przyjąć

$$f(x,y) = \frac{\sin(x+y)*\cos(x-y)}{\sin(x+y)+\cos(x-y)}.$$

28. Wydrukować potęgi od 1 do 10 dla liczb 2, 3, 4.

29. Napisać program drukujący za pomocą n różnych znaków napis LALA powiększonymi literami

```

* *      * * * * *  * *      * * * * *
* *      * * * * *  * *      * * * * *
* *      * *      * *      * *      * *
* *      * * * * *  * *      * * * * *
* * * * *  * *      * * * * *  * *      * *
* * * * *  * *      * * * * *  * *      * *

```

30. Obliczyć wartość współczynnika dwumianu Newtona

$$\binom{n}{k} = \frac{n * (n-1) * \dots * (n+1-m)}{1 * 2 * \dots * m},$$

gdzie

$$m = \min\{k, n-k\} \text{ oraz } 0 \leq k \leq n.$$

31. Napisać iteracyjną i rekurencyjną funkcję, która dla danej liczby naturalnej n i dla danej liczby rzeczywistej x oblicza wartość wielomianu Hermite'a $H_n(x)$ ze wzorów:

$$H_0(x) = 1,$$

$$H_1(x) = 2x,$$

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x) \text{ dla } n > 1.$$

32. Dla danej liczby naturalnej n oraz liczby rzeczywistej x obliczyć wartość wielomianu $P_n(x)$ ze wzorów:

$$P_0(x) = 1,$$

$$P_1(x) = x,$$

$$P_{r+1}(x) = \frac{2r+1}{r+1} x P_r(x) - \frac{r}{r+1} P_{r-1}(x) \quad \text{dla } r=1,2,\dots,n-1.$$

(bez użycia tablic).

33. Znaleźć stopień wielomianu $H_n(x)$, dla którego spełniona jest nierówność

$$|H_n(x_0) - y_0| < \varepsilon,$$

gdzie

x_0, y_0, ε - dane liczby rzeczywiste,

$$H_0(x) = \frac{1}{2}, \quad H_1(x) = \sqrt[3]{x},$$

$$H_k(x) = 2x^2 H_{k-1}(x) - H_{k-2}(x) \quad \text{dla } k=2,3,\dots$$

Podać stopień n oraz wartość $H_n(x)$.

34. Wydrukować wartości wielomianów $P_0(x), P_1(x), \dots, P_n(x)$, gdzie

x - dana liczba rzeczywista,

n - dana liczba naturalna,

$$P_0(x) = 1,$$

$$P_1(x) = 2x,$$

$$P_{k+1}(x) = \begin{cases} \frac{k+1}{n} x P_k(x) - \frac{2k+3}{k+1} P_{k-1}(x) & \text{dla } x \leq 0, \\ \frac{k}{4} x P_k(x) - \frac{2k}{5} P_{k-1}(x) & \text{dla } x > 0, \end{cases}$$

dla $k=1,2,\dots,n-1$.

35. Dla danych liczb rzeczywistych a, b, c, d obliczyć wartość sumy, iloczynu, ilorazu i modułu liczb zespolonych x, y , gdzie

$$x = a + i*b, \quad y = c + i*d.$$

36. Danych jest n liczb zespolonych postaci

$$z_k = x_k + i*y_k \quad \text{dla } k=1,\dots,n.$$

Funkcji unit nadać wartość true, jeśli moduły wszystkich liczb zespolonych z_k ($k=1,\dots,n$) są mniejsze od 1.

37. Napisać rekurencyjną procedurę porządkowania n -elementowego ciągu w ciąg rosnący.

38. Napisać iteracyjną i rekurencyjną procedurę szukania liczby x w uporządkowanej rosnąco tablicy a . Porównać napisane procedury.

39. Funkcja $\text{comm}(n,k)$ wyznacza liczbę różnych zespołów k -osobowych utworzonych spośród n osób, np. $\text{comm}(4,3)=4$ ponieważ dla danych osób A, B, C, D są możliwe następujące 4 zespoły 3-osobowe:

ABC, ABD, ACD, BCD .

Napisać procedurę rekurencyjną obliczania $\text{comm}(n,k)$ dla $n, k \geq 1$, przy czym wiadomo, że

$$\text{comm}(n,k) = \text{comm}(n-1,k) + \text{comm}(n-1,k-1) \quad \text{dla } n, k \geq 1,$$

$$\text{comm}(0,k) = 0 \quad \text{dla } k \geq 1,$$

$$\text{comm}(n,0) = 1 \quad \text{dla } n \geq 0.$$

40. Niech A będzie tablicą o n elementach całkowitych. Napisać rekurencyjne procedury obliczające:

a) $\max_{1 \leq i \leq n} A_i,$

b) $\min_{1 \leq i \leq n} A_i,$

c) $\sum_{i=1}^n A_i,$

d) $\prod_{i=1}^n A_i.$

41. Dane są liczby naturalne k, n, m, x, y, z . Obliczyć wartość

$$A = \sqrt[3]{S(n,x) + S(k,z) + S(m,y)},$$

gdzie

$$S(p,t) = \prod_{i=1}^p t * \left(\sum_{j=1}^i \frac{t}{j!} \right).$$

42. Zaprogramować obliczanie wielkości

$$\frac{z_1}{z_2} \quad \text{i} \quad |z|,$$

gdzie z_1, z_2, z - liczby zespolone postaci $z_k = x_k + i*y_k$.

Wykorzystać wzory:

$$\frac{z_1}{z_2} = \begin{cases} \frac{x_1 + y_1 p}{x_2 + y_2 p} + i \frac{y_1 - x_1 p}{x_2 + y_2 p}, & p = \frac{y_2}{x_2} \quad \text{dla } |x_2| \geq |y_2|, \\ \frac{x_1 p + y_1}{x_2 p + y_2} + i \frac{y_1 p - x_1}{x_2 p + y_2}, & p = \frac{x_2}{y_2} \quad \text{dla } |x_2| < |y_2|, \end{cases}$$

oraz

$$|z| = \begin{cases} 0 & \text{dla } x=y=0, \\ |x| \sqrt{1+(y/x)^2} & \text{dla } |x| \geq |y| > 0, \\ |y| \sqrt{1+(x/y)^2} & \text{dla } |y| > |x| > 0. \end{cases}$$

43. Dla danych liczb całkowitych n, m wydrukować wszystkie potęgi liczby n mniejsze niż m .

44. Dodawanie w słupkach. Dane są dwie liczby całkowite nieujemne reprezentowane za pomocą tablic liczb całkowitych. Napisać program, który dla n -cyfrowych liczb całkowitych

$$u_1 u_2 \dots u_n$$

$$v_1 v_2 \dots v_n$$

oblicza ich sumę $w_0 w_1 w_2 \dots w_n$, gdzie w_0 jest przeniesieniem.

45. Napisać program, który oblicza $n!$ dla dużych n (np. $n > 10$) i podaje dokładny wynik.

46. Napisać funkcję $\text{cyfra}(n, k)$, która podaje k -tą cyfrę licząc od prawej strony liczby n , np.

$$\text{cyfra}(17892, 4) = 7,$$

$$\text{cyfra}(13, 5) = 0.$$

47. Napisać funkcję, która dla danej liczby rzeczywistej x oraz liczby całkowitej m zaokrągla liczbę x do m miejsc po przecinku, np. dla $x=127.5762$, $m=2$, wynik: 127.58.

48. Dla danej liczby naturalnej n sprawdzić, czy ma dokładnie dwie takie same cyfry.

49. Jeśli $p > 0$ jest liczbą pierwszą, to liczba

$$2^p - 1 \text{ jest liczbą pierwszą} \iff L_{p-2} = 0,$$

gdzie

$$L_0 = 4,$$

$$L_{i+1} = (L_i^2 - 2) \bmod (2^p - 1) \quad \text{dla } i=0, \dots, p-3.$$

Dla dziesięciu początkowych liczb pierwszych sprawdzić, która liczba postaci $2^p - 1$ jest liczbą pierwszą.

50. Napisać program, który dla danej liczby naturalnej n znajduje pierwszy n -elementowy ciąg kolejnych liczb naturalnych, wśród których nie ma liczby pierwszej.

51. Zamienić liczby całkowite p , q , $p+q$ oraz $p \cdot q$ z układu dwójkowego na układ dziesiętny.

52. Napisać procedurę zamieniającą liczbę całkowitą dodatnią z układu dziesiętnego na układ dwójkowy.

53. Napisać procedurę, która daną liczbę całkowitą (w systemie dziesiętnym) zapisuje w systemie szesnastkowym wykorzystując 16 znaków

0, 1, 2, ..., 9, A, B, C, D, E, F,

np. $(16)_{10} = (10)_{16}$, $(12)_{10} = (C)_{16}$.

54. Dane są liczby naturalne a , b w układzie ósemkowym. Obliczyć ich iloczyn w układzie dwójkowym i wynik podać w układzie dziesiętnym.

55. Opracować procedury zamiany liczb

- a) z układu dwójkowego na czwórkowy,
- b) z układu dwójkowego na ósemkowy,
- c) z układu trójkowego na dziewiątkowy,
- d) z układu ósemkowego na dwójkowy.

56. Zbiór P na początku zawiera dwie liczby całkowite: 1 i 2.

Utworzyć zbiór P zawierający wszystkie liczby (mniejsze niż dopuszczalne ograniczenie górne komputera) postaci

$$3x + y,$$

gdzie x , y są różnymi elementami P .

Tak więc zbiór P zawiera liczby

1, 2, 5, 7, 8, 10, 11, 13, 14, 16, 17, ...

57. Liczbami bliźniaczymi nazywamy dwie liczby pierwsze p , q ($p < q$) takie, że $q = p + 2$. Napisać program znajdujący k pierwszych par liczb bliźniaczych.

58. Napisać program, który drukuje wszystkie sześćoelementowe zbiory liczb naturalnych $a_1, a_2, a_3, a_4, a_5, a_6$ większych od zera takich, że

$$a_1^2 + a_2^2 + a_3^2 = a_4^2 + a_5^2 + a_6^2,$$

gdzie

$$a_1 \leq a_2 \leq a_3 \leq 20,$$

$$a_1 < a_4 \leq a_5 \leq a_6 \leq 20.$$

Wskazówka. Wygenerować wszystkie możliwe sumy kwadratów i wykorzystując procedurę sortowania, znaleźć sumy powtarzające się.

59. Znaleźć k pierwszych liczb piramidalnych, które są liczbami trójkątnymi, gdzie

- liczby trójkątne są elementami ciągu określonego wzorami:

$$t_1 = 1,$$

$$t_{n+1} = t_n + n + 1,$$

- liczby piramidalne są elementami ciągu określonego wzorami:

$$T_1 = 1,$$

$$T_{n+1} = T_n + t_{n+1},$$

np. dla $k=3$

Liczby	n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
trójkątne		1, 3,	6,	10,	15,	21,	28,	36,	45,	55,	66,	78,	91,	105,	120	
piramidalne		1, 4,	10,	20,	35,	56,	84,	120								

60. Liczba automorficzna, to liczba znajdująca się na końcu swego kwadratu, np. $5^2=25$, $6^2=36$, $25^2=625$. Znaleźć n liczb automorficznych.

61. Znaleźć liczby od 1 do 3000000, które są równe sumie silni swoich cyfr, np. $145=1!+4!+5!$

62. Dana jest liczba $n_{\max}$. Wyznaczyć liczby Stirlinga drugiego rodzaju dla $n=0,1,\dots,n_{\max}$; $m=0,\dots,n$, ze wzorów:

$$S(0,n) = S(n,0) = 0 \quad \text{dla } n>0,$$

$$S(n,n) = 1 \quad \text{dla } n \geq 0,$$

$$S(n,m) = mS(n-1,m) + S(n-1,m-1) \quad \text{dla } n>0, m=1,\dots,n-1..$$

Obliczyć

$$P_n = \sum_{m=1}^n S(n,m) \quad \text{dla } n=0,1,\dots,n_{\max}.$$

63. Dla ciągu liczbowego $a_1, \dots, a_n$ można wykonać cykliczne przesunięcie o:

- 0, wówczas ciąg wejściowy nie ulega zmianie,
- 1, wówczas jest utworzony ciąg $a_2, \dots, a_n, a_1$,
- -1, wówczas jest utworzony ciąg $a_n, a_1, \dots, a_{n-1}$,
- $k \neq 0$, wówczas wykonuje się $|k|$ -krotne przesunięcie o $\text{sgn}(k)$.

Napisać program przesuwający dany n -elementowy ciąg liczb rzeczywistych o daną liczbę całkowitą k .

64. Dla liczb naturalnych k z przedziału $[1, kk]$ (kk -dane) można utworzyć ciąg liczbowy według algorytmu:

- (1) jeśli $k=1$, to stop,
- (2) jeśli k jest parzyste, to $k:=k/2$,
- (3) jeśli k jest nieparzyste, to $k:=3k+1$.

Na przykład: dla $k=21$ otrzymujemy ciąg liczb:

21, 64, 32, 16, 8, 4, 2, 1.

Napisać program, który:

a) tworzy takie ciągi liczb dla $k \in [1, kk]$ i drukuje:

k ,

n_k - liczba wykonanych kroków algorytmu,

$\max k$ - maksymalna wartość w ciągu liczb dla wartości k .

Na przykład: jeśli $k=21$, to $n_{21}=7$, $\max k=64$.

Wyniki wydrukować w postaci: ($kk=4$)

k	$n(k)$	$\max(k)$
1	0	1
2	1	2
3	7	16
4	2	4

Uwaga. Dotychczas nie wiadomo, czy dla każdej liczby naturalnej algorytm jest skończony.

b) dla n od 1 do danej liczby $n_{\max}$ wyznacza zbiory $Z(n)$ liczb, dla których wykonuje się n kroków algorytmu.

Na przykład:

$Z(0)=\{1\}$,

$Z(1)=\{2\}$,

$Z(2)=\{4\}$,

$Z(3)=\{8\}$,

$Z(4)=\{16\}$.

Zbiór $Z(n)$ można tworzyć rekurencyjnie w następujący sposób:

$$Z(0) = \{1\}.$$

Jeśli $Z(n) = \{k_1, k_2, \dots, k_n\}$, to $Z(n+1)$ zawiera:

- dla każdej liczby k_i liczbę $2k_i$,
- jeśli liczba $(k_i - 4)/6$ jest liczbą naturalną dodatnią, to liczbę $(k_i - 1)/3$.

c) znajduje ciągi kolejnych liczb $k, k+1, \dots, k+r$, dla których

$$n_k = n_{k+1} = \dots = n_{k+r}.$$

65. Dla danego ciągu liczb całkowitych policzyć ile jest podciągów zawierających co najmniej trzy liczby dodatnie obok siebie.

66. Dany jest tekst zakończony znakiem * napisany małymi literami. Wyprowadzić ten tekst wielkimi literami.

67. Obliczyć przybliżoną wartość liczby π do n miejsc po kropce

$$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots)$$

68. Obliczyć długość krzywej $y = f(x)$ na odcinku $[a, b]$ (należy aproksymować ją krzywą łamaną i obliczyć długość tej łamanej).
Przyjąć $f(x) = \sin(x)$, $a = 0$, $b = \pi$.

69. Napisać procedurę sprawdzającą, czy dana formuła logiczna jest tautologią, tzn. czy dla każdych wartości zmiennych logicznych formuła jest prawdziwa. Tautologiami są na przykład formuły

$$p \vee \neg p, \quad \neg(p \wedge \neg p).$$

70. Wśród n osób przeprowadzono ankietę zawierającą 4 pytania:

1) Wiek:

- a) poniżej 20 lat
- b) 20 - 29 lat
- c) 30 - 39 lat
- d) 40 - 49 lat
- e) 50 - 59 lat
- g) 60 lub więcej lat

2) Palisz:

- a) tylko fajkę
- b) mniej niż jedną paczkę papierosów dziennie
- c) 1 lub 2 paczki papierosów dziennie

- d) więcej niż 2 paczki papierosów dziennie
- e) nie palę wcale

3) Przynależność polityczna:

- a) partyjny
- b) bezpartyjny

4) Zarobek:

- a) poniżej 150 tys. zł.
- b) 150 - 250 tys. zł włącznie
- c) 250 - 350 tys. zł włącznie
- d) 350 - 450 tys. zł
- e) powyżej 450 tys. zł

Dane są podawane jako n ciągów 4-elementowych zawierających numery odpowiedzi do kolejnych pytań. Opracować następujące wyniki ankiety:

- a) średni przedział wieku ankietowanych,
- b) liczbę osób palących, które mają mniej niż 20 lat,
- c) procent osób bezpartyjnych,
- d) w którym przedziale zarobkowym było najwięcej ankietowanych.

71. Chodzenie losowe po kole. Dany jest promień R okręgu o środku $(0,0)$. Startujemy z punktu $x=0$, $y=0$ w czasie $t=0$. Dla kolejnych $t=1,2,3,\dots$ możemy wykonać jeden z czterech ruchów

1. $x:=x-1$,
2. $x:=x+1$,
3. $y:=y-1$,
4. $y:=y+1$,

przy czym ruchy te są wybierane losowo. Chodzenie kończy się, jeśli

$$x^2 + y^2 \geq R^2.$$

- a) Wydrukować liczbę wykonanych ruchów (czyli t) oraz numery wykonywanych ruchów.
- b) Wydrukować układ współrzędnych oraz kolejne punkty (x,y) .

72. Gra "zgadnij liczbę". Napisać program, który losowo generuje dowolną cyfrę. Zadaniem gracza jest odgadnięcie tej liczby maksymalnie w trzech próbach. Po każdej nieudanej próbie program powinien drukować komunikat informujący, czy liczba podana przez gracza jest mniejsza czy większa od liczby wygenerowanej.

73. Napisać program, który sprawdza wiadomości ucznia szkoły podstawowej z zakresu dodawania, odejmowania, mnożenia lub dzielenia.

74. Zapisać procedurę wyznaczającą najlepszy ruch w następującej grze: W puli jest umieszczona pewna liczba sztonów. Dwóch graczy na zmianę bierze jeden lub dwa sztony z puli. Gracz, który bierze ostatni, przegrywa.

75. Dana jest liczba całkowita R oznaczająca rok. Sprawdzić, czy dany rok jest rokiem przestępnym (rok jest przestępny, jeśli dzieli się przez 4 i nie dzieli się przez 100 lub dzieli się przez 400).

76. Napisać procedurę - wieczny kalendarz - określając dzień tygodnia w zależności od daty (trzech liczb całkowitych: rok, miesiąc, dzień).

77. Dane są trójki liczb naturalnych. Napisać program informujący, która trójka określa właściwą datę. Ponadto dla poprawnych dat program powinien podać, który to dzień w roku i ile dni zostało do końca roku.

78. Zebrania pewnego towarzystwa odbywają się w każdy pierwszy czwartek miesiąca. Napisać program, który podaje daty zebrań w wybranym roku.

79. Dane są dwie liczby naturalne godz i min ($0 < \text{godz} \leq 12$, $0 \leq \text{min} < 60$) oznaczające aktualną godzinę i minuty. Podać, ile czasu musi upłynąć, by wskazówki: godzinowa i minutowa na zegarze

a) pokryły się,

b) ustawiły się pod kątem 90° .

80. Zaprogramować grę w kości pięcioma kostkami. Wynik rzutu, zwany figurą, symulowany jest jednokrotnym losowaniem 5 liczb. Uwzględnić następujące figury:

Figura	Wynik rzutu	Premia
SEKWENCJA	1,2,3,4,5 lub 2,3,4,5,6	60
SASIEDNIE	n, n, n, k, k lub n, n, k, k, k , gdzie $k = n+1$	40
DWIE_PARY	n, n, k, k (piąta liczba dowolna)	30
POKER	wszystkie jednakowe	100
NIE_FIGURA	układ nie wymieniony wyżej	-

Ocena figury równa jest sumie oczek plus premia.

Opracować procedury o nazwach SEKWENCJA, SASIEDNIE, DWIE_PARY, POKER sprawdzające, czy zawartość 5-elementowej tablicy jest figurą o danej nazwie i obliczające ocenę wyniku rzutu. Sprawdzenie będzie

łatwiejsze, jeżeli elementy tablicy zostaną najpierw uporządkowane.

Napisać program symulujący przebieg opisanej gry (N rzutów) z dwoma uczestnikami.

81. Gra w życie. W czasie t pole żyje, jeśli albo:

a) było puste w czasie $t-1$ i żyły dokładnie trzy pola sąsiednie, albo

b) żyło w czasie $t-1$ i żyły dwa lub trzy pola sąsiednie.

W pozostałych przypadkach pole jest puste.

Sąsiedztwo pola y stanowi osiem pól x

x	x	x
x	y	x
x	x	x

Zasymulować konfigurację 15×15 pól przy żyjących początkowo wybranych polach. Zasymulować k cykli, np. dla $k=5$ i początkowo żyjących pól:

(3,8), (4,7), (5,7), (5,8), (5,9), (10,7), (10,8), (10,9), (11,7), (12,8),
(3,2), (4,3), (5,1), (5,2), (5,3), (9,1), (9,2), (9,3), (10,3), (11,2).

82. Napisać program opisujący następującą grę: komputer generuje n -elementowy zbiór cyfr. Gracz wprowadza z klawiatury swój n -elementowy zbiór cyfr. Gra kończy się, jeśli zbiór gracza jest identyczny ze zbiorem komputera. Jeśli ten zbiór nie jest identyczny, to komputer informuje, ile cyfr jest wspólnych w obu zbiorach.

83. W grupie n kobiet i n mężczyzn każda osoba podaje 5 osób płci przeciwnej najbardziej jej odpowiadających. Napisać program dopasowujący pary małżeńskie.

84. N osób głosuje na m przebojów. Wydrukować listę przebojów.

85. Wieże Hanoi. Dane są trzy pałeczki: A, B, C oraz n krążków różnej wielkości. Przenieść wszystkie krążki z pałeczki A na pałeczkę C (wykorzystując pałeczkę B) według zasad:

a) krążki przenosi się pojedynczo,

b) większy krążek nie może znaleźć się nad krążkiem mniejszym.

Wydrukować n oraz liczbę wykonanych przeniesień.

Opracować taki sposób drukowania wyników, aby można było odczytać, który krążek, z której pałeczki i na którą pałeczkę był przenoszony w danym kroku.

86. Wydrukować szachownicę.

87. Rozmieścić 8 królowek na polu szachowym tak, by żadna z nich nie biła innych.

88. Napisać program, który wyznacza drogę skoczka szachowego przez wszystkie 64 pola tak, by

- a) przez żadne pola nie przechodzić więcej niż raz,
- b) przejść przez maksymalną liczbę pól bez przecinania własnej drogi.

89. Zaprogramować grę w kółko i krzyżyk.

90. Napisać program, który wczytuje dany ciąg znaków pomijając ewentualne spacje (przed lub między znakami) a następnie sprawdza, czy wczytany ciąg

- a) zawiera tylko znaki +, -, *, /, (,), _, litery i cyfry,
- b) zawiera tyle samo okrągłych nawiasów otwierających co zamykających,
- c) nie zawiera znaków +, -, *, / obok siebie,
- d) rozpoczyna się znakiem -, (, cyfrą lub nazwą,
- e) po każdym znaku (zawiera znak -, cyfrę lub nazwę,
- f) po znaku) nie występuje cyfra, litera ani znak (,
- g) nie zawiera liter występujących bezpośrednio po cyfrach.

91. Napisać procedury, które sprawdzają, czy dany ciąg znaków reprezentuje

- a) liczbę całkowitą,
- b) liczbę rzeczywistą,
- c) nazwę, tzn. czy jest to ciąg liter, cyfr i/lub znaku podkreślenia, przy czym zaczyna się literą.

92. Napisać program, który czyta i oblicza wartość wyrażenia, w którym:

- a) liczby są całkowite,
- b) każda liczba jest poprzedzona znakiem plus lub minus,
- c) wyrażenie jest zakończone średnikiem,

tzn. wyrażenia powinny być w postaci np. +20-4-3+169; -17+14;

93. Gra w TOTOLOTKA. N graczy wybiera po 6 liczb od 1 do 49. Następnie jest losowanych 6 liczb. Wydrukować nazwiska osób, które trafnie podały

- a) sześć liczb,
- b) pięć liczb,
- c) cztery liczby.

94. Napisać program symulujący działanie automatu do wypłacania pieniędzy: n złotych należy wypłacić minimalną liczbą monet k -złotowych i 1-złotowych. Jeżeli jest to niemożliwe, program powinien podać najbliższą mniejszą kwotę, jaką można wypłacić.

95. Dana jest lista tytułów książek i artykułów. Napisać program, który sprawdza, czy dany tytuł występuje na tej liście. Jeśli tak, to wydrukować potrzebne informacje o danej pozycji.

96. Napisać program obliczający dla danej książeczki PKO procent, jaki należy wypłacić właścicielowi. Uwzględnić daty wpłat i wypłat, kwoty wpłacane i wypłacane, wysokość oprocentowania książeczki, datę ostatniego oprocentowania, datę złożenia książeczki do oprocentowania itp.

97. W pewnym systemie katalogów wszystkie katalogi są zorganizowane jako drzewa binarne. Każdy korzeń drzewa zawiera m.in. nazwę katalogu i datę ostatniego odwołania do katalogu. Napisać procedurę usuwającą te katalogi, do których ostatnie odwołanie było przed daną datą.

12. ROZWIĄZANIA ZADAŃ

ROZDZIAŁ 4

```
25. program słownie;
    var cyfra: 0..9;
    begin
        clrscr;
        write('Podaj cyfrę: '); readln(cyfra);
        if (cyfra<0) or (cyfra>9) then writeln('Złe dane')
            else begin
                write(cyfra:2);
                case cyfra of
                    0: writeln('- zero');
                    1: writeln('- jeden');
                    2: writeln('- dwa');
                    3: writeln('- trzy');
                    4: writeln('- cztery');
                    5: writeln('- pięć');
                    6: writeln('- sześć');
                    7: writeln('- siedem');
                    8: writeln('- osiem');
                    9: writeln('- dziewięć');
                end {case}
            end {else}
    end.
```

```
34. program Sx;
    var i,n,x,S: integer;
    begin
        readln(n);
        for i:=1 to n do
            begin
                readln(x);
```



```

    if not odd(x) then S:=sqr(x)+3*abs(x)
        else if x>0 then S:=(x+7) div 2
            else S:=(x-7) div 2;
        writeln('x=',x:5,' S=',S:7)
    end {i}
end.

```

```

35. program abcd;
    var a,b,c,d,x,y,x2,y2: real;
    begin
        readln(a,b,c,d);
        if (b<=a) or (c<=0)
            then writeln('Zle dane')
            else
                begin
                    x:=ln(c)+cos(a+b);
                    x2:=sqr(x);
                    y:=sqrt(abs(a*x2*x+b*x+c));
                    y2:=sqr(y);
                    x2:=x2+y2;
                    if x2<=a
                        then x2:=exp(x)+abs(y*y2)
                        else if x2<b
                            then x2:=sin(x)+sqr(d)
                            else x2:=12.0*sqr(y2)+exp(ln(x)/3.0);
                    writeln('z=',x2:10:5)
                end {else}
    end.

```

```

37. program Sm;
    var a,b,c,S: real;
        m: integer;
    begin
        readln(a,b,c,m);
        case m of
            1: begin
                    S:=a;
                    if b>S then S:=b;
                    if c>S then S:=c
                end;
            2: begin
                    S:=a;

```

```

        if b<S then S:=b;
        if c<S then S:=c
    end;
3: S:=(a+b+c)/3.0;
4: S:=sqr(a)+sqr(b)+sqr(c);
5: S:=abs(a)+abs(b)+abs(c)
else S:=0.0
end {case};
writeln('a=',a:5:2,' b=',b:5:2,' c=',c:5:2,
        ' m=',m:3,' S=',S:8:3)
end.

```

```

40. program dwiesumy;
var x,wyrac,suma1,suma2: real;
    i,n: integer;
begin
    clrscr;
    write('x=');
    readln(x);
    if x=0
    then writeln('suma1=suma2=1.0')
    else
        begin
            write('Podaj n: ');
            readln(n);
            suma1:=1.0;
            wyrac:=1.0;
            for i:=1 to n do
                begin
                    wyrac:=wyrac*x/i;
                    suma1:=suma1+wyrac
                end {i};
            suma2:=wyrac;
            for i:=n downto 2 do
                begin
                    wyrac:=wyrac*i/x;
                    suma2:=suma2+wyrac
                end {i};
            writeln('suma1=',suma1:15:8);
            writeln('suma2=',suma2:15:8)
        end {x<>0}
    end.

```

```

46. program suma;
   var licznik,x,suma: real;
       i,n: integer;
begin
   clrscr;
   write('Podaj n: ');
   readln(n);
   x:=2.0;
   repeat
      suma:=0.0;
      licznik:=-1.0;
      for i:=2 to n+1 do
         begin
            licznik:=-licznik*x;
            suma:=suma+licznik/i
         end {i};
         writeln('n=',n:3,' x=',x:5:2,' suma=',suma:16:4);
         x:=x+3.0
      until x>20.0
   end.

```

```

58. program fx;
   var a,h,x: real;

   function f(x: real): real;
   begin
      if x<=0 then f:=abs(x)
         else f:=ln(x)
      end {f};
   begin
      repeat
         write('Podaj liczby rzeczywiste a>0 oraz h>0: ');
         readln(a,h);
         until (a>0.0) and (h>0.0);
         writeln;
         writeln('      a          f(a)');
         x:=-a;
         repeat
            writeln(x:6:2,f(x):17:4);
            x:=x+h
         until x>a
      end.

```

```

60. program bisekcja;
    const maxiter=500;      {maksymalna liczba iteracji}
    var xp,xk,x0,eps,Gp,Gk,G0: real;
        i: integer;

    function G(x: real): real;
    begin
        G:=sqr(x)-4.0
    end {G};

    begin
        clrscr;
        readln(xp,xk,eps);
        Gp:=G(xp);
        Gk:=G(xk);
        x0:=xp+0.5*(xk-xp);
        G0:=G(x0);
        i:=0;
        while (abs(G0)>eps) and (i<maxiter) do
        begin
            if G0*Gp<0 then begin
                xk:=x0;
                Gk:=G0
            end
            else begin
                xp:=x0;
                Gp:=G0
            end;
            x0:=xp+0.5*(xk-xp);
            G0:=G(x0);
            i:=i+1
        end {while};
        writeln('G(',x0:10:6,')=',G0:10:6)
    end.

```

```

66. program ciag;
    var u0,u1,eps: real;
        k: integer;

    begin
        readln(eps);
        u1:=0.0;
        k:=0;
        writeln('u(',k:4,')=',u1:10:6);

```



```

repeat
  u0:=u1;
  u1:=(u0+1.0)/(u0+2.0);
  k:=k+1;
  writeln('u(',k:4,')=',u1:10:6)
until (u1-u0)<eps
end.

```

77. program pierwsze;

```

var np,nk,i: integer;

function jest(n: integer): boolean;
var i,sqrtn: integer;
    p: boolean;
begin
  p:=true;
  i:=2;
  sqrtn:=trunc(sqrt(n));
  while (i<=sqrtn) and p do
    begin
      if (n mod i=0) then p:=false;
      i:=i+1
    end {while};
    jest:=p
  end {jest};
begin
  readln(np,nk);
  for i:=np to nk do
    begin
      write(i:5);
      if jest(i) then writeln('- liczba pierwsza')
        else writeln('- to nie jest liczba pierwsza')
      end {i}
    end.

```

78. program centy;

```

var kwota,s,c25,c10,c5,c1: integer;

procedure licz(var s,c: integer; m: integer);
begin
  c:=s div m;
  s:=s mod m
end {licz};

```

```

begin
  readln(kwota);
  s:=kwota;
  c25:=0; c10:=0; c5:=0; c1:=0;
  licz(s,c25,25);
  licz(s,c10,10);
  licz(s,c5,5);
  writeln(kwota,'c=',c25,'*25c+',c10,'*10c+',c5,'*5c+',s,'*1c')
end.

```

79. function wspliniowe(x1,x2,x3,y1,y2,y3: real): boolean;

```

begin
  if x1=x2
  then
    if x1=x3 then wspliniowe:=true
    else wspliniowe:=false
  else
    if (y1-y2)*(x3-x1)=(y3-y1)*(x1-x2)
    then wspliniowe:=true
    else wspliniowe:=false
  end {wspliniowe}
end.

```

80. program dzien;

```

var dz: 1..7;
begin
  clrscr;
  write('Podaj dzien (1-7): ');
  readln(dz);
  writeln;
  write('Dzisiaj jest ');
  case dz of
    1: writeln('poniedzialek');
    2: writeln('wtorek');
    3: writeln('sroda');
    4: writeln('czwartek');
    5: writeln('piatek');
    6: writeln('sobota');
    7: writeln('niedziela')
  end {case}
end.

```

83a program trójkąt;

```
var i,j,k,m,n: integer;
    zn: char;
begin
    readln(n,m);
    for k:=1 to n do
    begin
        clrscr;
        zn:=chr(33+random(93)); { wybór znaku }
        { drukowanie trójkąta }
        for i:=1 to m do
        begin
            for j:=1 to 24+m-i do write(' ');
            for j:=1 to 2*i-1 do write(zn) ;
            writeln
        end {i};
        delay(3000)
    end {k}
end.
```

85. program oceny;

```
const szablon='.....';
var i: integer;
begin
    clrscr;
    writeln;
    for i:=1 to 80 do write('-');
    writeln;
    writeln;
    write('                ');writeln(szablon);
    write('                ');writeln('nazwisko i imię ucznia');
    writeln;
    writeln('  1.Sprawowanie          ',szablon);
    writeln('  2.Język polski           ',szablon);
    writeln('  3.Matematyka             ',szablon);
    writeln('  4.Środowisko             ',szablon);
    writeln('  5.Muzyka                  ',szablon);
    writeln('  6.Wych.plastyczne         ',szablon);
    writeln('  7.Praco-technika          ',szablon);
    writeln('  8.Wych.fizyczne           ',szablon);
    writeln;
    for i:=1 to 20 do write(' ');
```

```

writeln(szablon);
for i:=1 to 20 do write(' ');
writeln('podpis wychowawcy');
writeln;
writeln;
for i:=1 to 80 do write('-');
writeln;
writeln
end.

```

87. program druk;

```

const kr20='.....';
      kr10='.....';
var i: integer;
begin
  gotoxy(1,6);
  for i:=1 to 40 do write(' ');
  writeln(kr20);
  writeln;
  writeln('                                ZAWIADOMIENIE');
  writeln;
  writeln('  Uprzejmie zawiadamiamy ob.',kr20,',');
  writeln;
  writeln('  że dnia',kr20,'roku',kr10);
  writeln;
  writeln('  o godz.',kr10,'w budynku',kr20);
  writeln;
  writeln('  w sali',kr10,'odbędzie się zebranie');
  writeln;
  writeln('  ',kr20,kr20,kr20);
  writeln;
  for i:=1 to 40 do write(' ');
  writeln(kr20); writeln;
  for i:=1 to 40 do write(' ');
  writeln(kr20)
end.

```

98. program obszar;

```

const spacje='      -';
      tekst='  leży w zakreskowanym obszarze');
var ax,x,y,r,r2,r3,rkw: real;
    i,n: integer;

```



```

procedure pisztak;
begin
    writeln(spacje,tekst)
end {pisztak};

procedure pisznie;
begin
    writeln(spacje,' nie',tekst)
end {pisznie};

begin
    clrscr;
    write('Podaj r>0: ');
    readln(r);
    r2:=r+r;
    r3:=r2+r;
    rkw:=sqr(r);
    write('Podaj n: ');
    readln(n);
    for i:=1 to n do
    begin
        write('Podaj x,y');
        readln(x,y);
        if y>0
        then
            begin
                ax:=abs(x);
                if (y<=r3) and (r<=ax) and (ax<=r2)
                then pisztak
                else
                    if (y<=r2) and (ax<=r) and (sqr(x)+sqr(y)>=rkw)
                    then pisztak
                    else pisznie
                end {y>0}
            else
                begin
                    ax:=sqr(x)+sqr(y);
                    if (rkw<=ax) and (ax<=4*rkw)
                    then pisztak
                    else pisznie
                end {y<=0}
            end {i}
        end.

```

ROZDZIAŁ 5

29. program parzyste;

```

type tab=array[1..101 of integer;
var z,i,lparz,lnparz,n: integer;
    a: tab; { w tablicy a zostaną umieszczone na początku
              elementy parzyste a za nimi elementy nieparzyste }
begin
  clrscr;
  write('Podaj n: '); readln(n);
  lparz:=0;
  lnparz:=0;
  for i:=1 to n do
  begin
    write('Podaj element ciągu: ');
    readln(z);
    if odd(z) then
      begin
        lnparz:=lnparz+1;
        a[lnlnparz+1]:=z
      end
    else
      begin
        lparz:=lparz+1;
        a[lparz]:=z
      end
    end {i};
  writeln;
  writeln('Elementy ciągu po ustawieniu:');
  for i:=1 to n do
    write(a[i]:5);
    writeln('Liczba parzystych: ',lparz:5);
    writeln('Liczba nieparzystych: ',lnparz:5)
  end.

```

30. program kosz;

```

const m=10;
var j,n,liczba: integer;
    a: array[0..m] of integer;

```

```

begin
  readln(n);
  for j:=0 to m-1 do
    a[j]:=0;
  for j:=1 to n do
    begin
      readln(liczba);
      a[liczba]:=a[liczba]+1
    end {j};
    { drukowanie bez powtórzeń }
  writeln('Elementy ciągu po uporządkowaniu:');
  for j:=0 to m-1 do
    if a[j]>0 then write(j:5)
  end.

```

```

46. program ui;
  var u,x,suma: real;
      i,m: integer;
  begin
    readln(m,x);
    suma:=0.0;
    u:=1.0;
    writeln('Ciąg u:');
    for i:=1 to m do
      begin
        u:=i*x*u;
        suma:=suma+u;
        writeln('u[',i:2,']= ',u:10:3)
      end {i};
    writeln('Suma elementów u[i] wynosi: ',suma:10:3)
  end.

```

```

55. program twciąg;
  const nmax=20;
  type tablica=array[1..nmax] of real;
  var a,b,c: tablica;
      i,n,m: integer;

  procedure czytaj(var x: tablica; var n: integer; nazwa: char);
  var i: integer;
  begin
    write('Podaj liczbę elementów ciągu: '); readln(n);

```

```

    writeln('Podaj elementy ciągu ',nazwa,':');
    for i:=1 to n do read(x[i]);
    writeln
end {czytaj};

procedure drukuj(x: tablica; n: integer; nazwa: char);
var i: integer;
begin
    writeln('Elementy ciągu ',nazwa,':')
    for i:=1 to n do write(x[i]:8:2);
    writeln
end {drukuj};

procedure cnm(x,y: tablica; n: integer; var z: tablica);
var i,l: integer;
begin
    l:=1;
    for i:=1 to n do
        begin
            z[l]:=x[i];
            z[l+1]:=y[i];
            l:=l+2
        end {i}
    end {cnm};

procedure twórz_ciąg(x,y: tablica; n,m: integer; var z: tablica);
var i: integer;
begin
    if n<m then
        begin
            cnm(x,y,n,z);
            for i:=n+1 to m do z[n+i]:=y[i]
        end {n<m}
    else
        begin
            cnm(x,y,m,z);
            if n>m then
                for i:=m+1 to n do z[m+i]:=x[i]
            end {n>m}
        end {twórz_ciąg};
begin
    czytaj(a,n,'a');
    czytaj(b,m,'b');

```



```

twórz_ciąg(a,b,n,m,c);
writeln;
drukuj(a,n,'a');
drukuj(b,m,'b');
drukuj(c,n+m,'c')
end.

```

63. program fragmenty;

```

const nmax=20;
type tab=array[1..nmax] of real;
var a: tab;
    jest: boolean;
    n,nk,i: integer;

procedure spr(x: tab; np,nk: integer; var s: boolean);
var i: integer;
begin
    s:=true;
    i:=np+1;
    while s and (i<=nk) do
    begin
        if x[i]<x[i-1] then s:=false;
        i:=i+1
    end {while}
end {spr};

procedure druk(x: tab; np,nk: integer; s: boolean);
var i: integer;
begin
    writeln;
    writeln('Czy poniższy ciąg jest rosnący?');
    for i:=np to nk do
        write(x[i]:5:1);
    writeln;
    if s then writeln('Ciąg jest rosnący')
    else writeln('Ciąg nie jest rosnący')
end {druk};

begin
    readln(n);
    for i:=1 to n do readln(a[i]);
    writeln; writeln('Ciąg a:');
    for i:=1 to n do write(a[i]:5:1);
    writeln;

```

```

spr(a,1,n,jest);
druk(a,1,n,jest);
if not jest then
    begin
        i:=1;
        while i<n do
            begin
                nk:=i+4;
                if nk>n then nk:=n;
                spr(a,i,nk,jest);
                druk(a,i,nk,jest);
                i:=i+5
            end {while}
        end {if}
    end.

```

67. program clog;

```

const maxn=20;
var a: array[1..maxn] of integer;
    b: array[1..maxn] of boolean;
    i,l,n,pom: integer;
begin
    clrscr;
    gotoxy(10,5);
    write('Podaj n: '); readln(n);
    writeln(' Podaj elementy ciągu a (liczby całkowite):');
    pom:=0;
    for i:=1 to n do
        begin
            readln(a[i]);
            if a[i]<0 then pom:=pom+1
        end {i};
    writeln(' Podaj elementy ciągu b (wartości 0 lub 1):');
    for i:=1 to n do
        begin
            readln(l);
            if l=1 then begin
                b[i]:=true;
                pom:=pom-1
            end {if}
            else b[i]:=false
        end {i};
    end.

```

```

case pom of
  0: begin
    l:=1;
    for i:=1 to n do
      if b[i] then
        begin
          while a[l]>=0 do l:=l+1;
          if i<>l then
            begin
              pom:=a[i];
              a[i]:=a[l];
              a[l]:=pom
            end {i<>l};
          l:=l+1
        end {if};
      writeln(' Elementy ciągu b:');
      for i:=1 to n do write(b[i]:6);
      writeln;
      writeln(' Elementy ciągu a (po uporządkowaniu):');
      for i:=1 to n do write(a[i]:6);
      writeln
    end {pom=0}
  else writeln('Złe dane')
end {case}
end.

```

68. program p3;

```

var xk,k,n,min,i: integer;
begin
  clrscr;
  write(' Podaj n: ');
  readln(n);
  i:=0;
  repeat
    i:=i+1;
    write(' Podaj liczbę naturalną: ');
    readln(min);
    k:=min mod 3
  until (k=0) or (i=n);
  if k<>0
    then writeln(' Nie ma liczby podzielnej przez 3')
    else

```

```

begin
  if i<>n then
    for k:=i+1 to n do
      begin
        write(' Podaj liczbę naturalną: ');
        readln(xk);
        if (xk mod 3=0) and (xk<min) then min:=xk
        end {k};
      writeln('Minimalna liczba podzielna przez 3=',min:5)
    end {k=0}
  end.

```

70. program ciągu;

```

type tablica=array[1..50] of integer;
var m,n,i,j,k,l: integer;
    x,y,z: tablica;
procedure czytaj(var x: tablica; var n: integer; znak: char);
begin
  write('Podaj liczbę elementów ciągu: '); readln(n);
  writeln('Podaj elementy ciągu ',znak,':');
  for i:=1 to n do readln(x[i])
end {czytaj};

procedure drukuj(x: tablica; n: integer, znak: char);
var i: integer;
begin
  writeln;
  writeln('Elementy ciągu ',znak,':');
  for i:=1 to n do
    write(x[i]:3);
  writeln
end {drukuj};

begin
  czytaj(x,n,'x');
  czytaj(y,m,'y');
  if x[n]<=y[1]
  then
    begin
      for i:=1 to n do z[i]:=x[i];
      for j:=1 to m do z[n+j]:=y[j]
    end
  else

```



```

if y[m]<=x[l]
then
begin
for j:=1 to m do z[j]:=y[j];
for i:=1 to n do z[m+i]:=x[i]
end
else
begin
l:=1; i:=1; j:=1;
repeat
if x[i]<y[j] then
begin
z[l]:=x[i];
l:=l+1;
i:=i+1
end
else
begin
z[l]:=y[j];
l:=l+1;
j:=j+1
end
until (i>n) or (j>m);
if i<=n
then
for k:=i to n do
begin
z[l]:=x[k];
l:=l+1
end {k}
else
for k:=j to m do
begin
z[l]:=y[k];
l:=l+1
end {k}
end {else};
drukuj(x,n,'x');
drukuj(y,m,'y');
drukuj(z,n+m,'z')
end.

```

```

75. program wektor;
    type tablica=array[1..10] of real;
    var a,b,c: tablica;
        i,n: integer;
        k: real;

    function ilskal(x,y: tablica; n: integer): real;
    var i: integer;
        S: real;
    begin
        S:=0.0;
        for i:=1 to n do
            S:=S+x[i]*y[i];
        ilskal:=S
    end {ilskal};

    begin
        readln(n);
        for i:=1 to n do readln(a[i]);
        for i:=1 to n do readln(b[i]);
        k:=ilskal(a,a,n);
        if k=0.0 then writeln('Zle dane')
            else
                begin
                    k:=-ilskal(a,b,n)/k;
                    writeln('k=',k:8:2);
                    writeln('Wektor c=b+k*a:');
                    for i:=1 to n do
                        begin
                            c[i]:=b[i]+k*a[i];
                            write(c[i]:8:2)
                        end {i};
                    writeln;
                    writeln('Iloczyn skalarny wektorów a i c wynosi:',
                        ilskal(a,c,n):8:5)
                end {else}
        end.

```

```

118. program zeruj;
    var n,i,j: integer;
        a: array[1..10,1..10] of integer;
    begin
        write('Podaj n: '); readln(n);

```

```

for i:=1 to n do
  for j:=1 to n do
    readln(a[i,j]);
  for i:=1 to n-2 do
    for j:=1 to n-i-1 do a[i,j]:=0;
  for i:=3 to n do
    for j:=n-i+3 to n do a[i,j]:=0;
    {drukowanie macierzy wynikowej}
  writeln('Macierz A po wyzerowaniu:');
  for i:=1 to n do
    begin
      for j:=1 to n do write(a[i,j]:3);
      writeln
    end {i}
  end.

```

124. program wierszk;

```

var m,n,i,j,k: integer;
    a: array[1..10] of array [1..10] of integer;
begin
  write(' Podaj n,m,k: '); readln(n,m,k);
  if (1<=k) and (k<=n) then
    begin
      writeln(' Podaj elementy macierzy wierszami:');
      for i:=1 to n do
        begin
          for j:=1 to m do
            begin
              read(a[i,j]); write(' ':3)
            end {j};
          writeln
        end {i};
      for i:=k to n-1 do a[i]:=a[i+1];
      writeln('Macierz A po usunięciu',k:3,'-go wiersza:');
      for i:=1 to n-1 do
        begin
          writeln;
          for j:=1 to m do write(a[i,j]:5)
        end {i}
      end {then}
      else writeln('Złe k')
    end.

```

```

130. procedure twórz_macierz(var A: tablica; n: integer);
    var i,j: integer;
begin
    for i:=1 to n do
        begin
            A[i,1]:=i;
            for j:=2 to n do
                A[i,j]:=A[i,j-1]+j
            end {i}
        end {twórz_macierz};
end.

163. program zmiany;
    var i,j,k,l,n,m: integer;
        a: array[1..10,1..10] of integer;
begin
    clrscr;
    writeln(' Podaj wymiary macierzy:');
    write('   n= ');
    read(n);
    write('   m= ');
    readln(m);
    writeln(' Podaj ',n*m,' elementów macierzy (wierszami):');
    for i:=1 to n do
        for j:=1 to m do
            readln(a[i,j]);
        end {j};
        writeln;
        writeln(' nr wiersza      liczba zmian znaku');
        for i:=1 to n do
            begin
                k:=0;
                for j:=2 to n do
                    begin
                        l:=a[i,j]*a[i,j-1];
                        if l<0
                            then k:=k+1
                        else if l=0 then if (a[i,j]<0) or (a[i,j-1]<0)
                            then k:=k+1;
                    end {j};
                    writeln(i:7,' ':15,k:3)
                end {i}
            end.

```


ROZDZIAK 6

```

1. program alfabet;
   type tab=string[15];
   var nazw: array[1..20] of tab;   { tablica nazwisk }
       pom: tab;                     { zmienna pomocnicza }
       i,j,k,n: integer;
begin
   readln(n);
   for i:=1 to n do readln(nazw[i]);
   for j:=1 to n-1 do
begin
   k:=j;
   for i:=j+1 to n do
      if nazw[i]<nazw[k] then k:=i;
   pom:=nazw[k];
   nazw[k]:=nazw[j];
   nazw[j]:=pom;
end {j};
   writeln('NAZWISKA W PORZĄDKU ALFABETYCZNYM:');
   for i:=1 to n do
      writeln('      ',nazw[i])
end.

```

```

17. program spacje;
   var znak: char;
       sp: boolean;
       tekst: string[50];
       i: integer;
begin
   writeln(' Podaj tekst zakończony kropką:');
   readln(tekst); writeln;
   sp:=false;
   znak:=tekst[1];
   i:=1;
   while znak<> '.' do
begin
   if znak<> ' ' then
begin

```

```

        write(znak);
        sp:=false
    end
    else
        if not sp then
            begin
                write(znak);
                sp:=true
            end;
        i:=i+1;
        znak:=tekst[i];
    end {while};
    write('.')
end.

```

18. program dwójspacje;

```

var tekst: string[20];
    i: integer;
begin
    clrscr;
    writeln(' Podaj tekst:');
    readln(tekst);
    writeln;
    for i:=1 to length(tekst) do
        write(tekst[i], ' ')
    end.

```

19. program cz;

```

var tekst: string[20];
    znak: char;
    i,licznik: integer;
begin
    clrscr;
    writeln('Podaj tekst zakończony kropką');
    readln(tekst);
    i:=1;
    licznik:=0;
    znak:=tekst[i];
    while znak<>'.' do
        begin
            if znak='c' then
                if tekst[i+1]='z' then

```

```

begin
    licznik:=licznik+1;
    i:=i+1
end;

i:=i+1;
znak:=tekst[i]
end {while};
write('Dwugłoska "cz" wystąpiła',1:4);
if l=1 then writeln(' raz')
    else writeln(' razy')
end.

```

```

20. program zdania;
var tekst: string[100];
    i: integer;
begin
    clrscr;
    writeln(' Podaj ciąg zdań');
    readln(tekst);
    for i:=1 to length(tekst) do
        begin
            write(tekst[i]);
            if tekst[i]='.' then writeln
        end {i}
    end.

```

ROZDZIAŁ 7

```

16. program dni_życia;
const nmax=10;
type data=record
    dzień: 1..31;
    miesiąc: 1..12;
    rok: 1900..1990
end {data};
opis=record
    nazwisko,imię: string[15];
    data_ur: data;
    wiek: integer
end {opis};

```



```

        4,6,9,11: dni:=dni+30;
        2: dni:=dni+dz
    end {case}
end {with};
dni:=dni-ur.dzień
end {licz_dni};

begin
    clrscr; gotoxy(2,2);
    writeln('DZISIEJSZA DATA');
    with akt_data do
    begin
        gotoxy(2,3); write('DZIEN      '); read(dzień);
        gotoxy(2,4); write('MIESIĄC    '); read(miesiąc);
        gotoxy(2,5); write('ROK       '); read(rok)
    end {with};
    clrscr;
    gotoxy(2,2);
    write('PODAJ LICZBE OSÓB: '); read(n);
    gotoxy(2,3);
    for i:=1 to n do
    begin
        write('OSOBA ',i);
        with OSOBY[i],data_ur do
        begin
            gotoxy(2,4); write('NAZWISKO-'); read(nazwisko);
            gotoxy(2,5); write('      IMIĘ-'); read(imię);
            gotoxy(2,6); write('DATA URODZENIA');
            gotoxy(2,7); write('DZIEN   '); read(dzień);
            gotoxy(2,8); write('MIESIĄC '); read(miesiąc);
            gotoxy(2,9); write('ROK     '); read(rok);
            clrscr
        end {with}
    end {i};
    for i:=1 to n do
        with OSOBY[i] do
            licz_dni(data_ur,akt_data,OSOBY[i].wiek);
    end {i};
    for i:=1 to n-1 do
    begin
        k:=i;
        min:=OSOBY[i].wiek;
        for j:=i+1 to n do

```

```

    if OSOBY[j].wiek < min then begin
        k:=j;
        min:=OSOBY[j].wiek
    end {if};

    if k > i then
        begin
            pom:=OSOBY[k];
            OSOBY[k]:=OSOBY[i];
            OSOBY[i]:=pom
        end {k>i}
    end {i};
    writeln;
    with akt_data do
    begin
        writeln('D Z I S I E J S Z A   D A T A');
        writeln(dzień:2,'.',miesiąc:2,'.',rok:4)
    end {with};
    writeln('-----');
    for i:=1 to n do
        with OSOBY[i],data_ur do
        begin
            writeln;
            writeln(nazwisko,' ',imię);
            write(dzień:2,'.',miesiąc:2,'.',rok:4);
            write('   W I E K   ',wiek:8);
            if wiek=1 then writeln(' dzień')
                else writeln(' dni');
            writeln('* * * * *');
        end {with}
    end.

```

18. program stawka;

```

const nmax=10;
type pracownik=record
    imię,nazwisko: string[15];
    stawka: integer;
    czaspracy: array[1..6] of real
end;
var osoby: array[1..nmax] of pracownik;
    k,i,n: integer;
    zarobek,godz: real;
    { godz-liczba przepracowanych godzin w tygodniu }

```

```

begin
  clrscr;
  write(' Podaj liczbę pracowników: ');
  readln(n); { n-liczba pracowników }
  for i:=1 to n do
    with osoby[i] do
      begin
        write(' Podaj imię: '); readln(imię);
        write(' Podaj nazwisko: '); readln(nazwisko);
        write(' Podaj stawkę: '); readln(stawka);
        writeln(' Podaj czas pracy w kolejne dni tygodnia:');
        for k:=1 to 6 do readln(czaspracy[k])
      end {i};
    writeln;
    writeln('   NAZWISKO           IMIĘ           STAWKA           ZAROBEK?');
    for i:=1 to n do
      begin
        with osoby[i] do
          begin
            godz:=0.0;
            for k:=1 to 6 do godz:=godz+czaspracy[k];
            if godz<=40.0 then zarobek:=godz*stawka
              else zarobek:=1.5*godz*stawka-20.0*stawka;
            writeln(nazwisko:10,imię:11,stawka:10,zarobek:16:2)
          end {with}
        end {i}
      end.

```

ROZDZIAŁ 8

```

1. program zbiory;
   type zakres=4..10;
      zbiór=set of zakres;
   var A,B,C: zbiór;
       m,n,i: integer;
       k: zakres;

   procedure czytaj(var X: zbiór; var n: integer);
   var i: integer;
       k: zakres;

```

```

begin
  write('Podaj liczbę elementów zbioru: ');
  readln(n);
  X:=[];
  writeln('Podaj elementy zbioru:');
  for i:=1 to n do
    begin
      readln(k); X:=X+[k]
    end {i}
  end {czytaj};

  procedure drukuj(X: zbiór);
  var k: zakres;
  begin
    for k:=4 to 10 do if k in X then write(k:3);
    writeln
  end {drukuj};

  begin
    czytaj(A,n);
    czytaj(B,m);
    C:=A+B;
    writeln;
    writeln('Zbiór A:'); drukuj(A);
    writeln('Zbiór B:');drukuj(B);
    writeln;
    writeln('Suma zbiorów A+B:'); drukuj(C)
  end.

```

5. program WielkieLitery;

```

var i,licznik: integer;
    tekst: string[20];
begin
  clrscr;
  writeln(' Podaj tekst:'); readln(tekst);
  licznik:=0;
  for i:=1 to length(tekst) do
    if tekst[i] in ['A'..'Z']
      then begin
        writeln(tekst[i], ' - wielka litera');
        licznik:=licznik+1
      end;
  write(' Podany tekst zawiera ',licznik:5);

```



```

case licznik of
  1: writeln(' wielką literę');
  2..4: writeln(' wielkie litery')
  else writeln(' wielkich liter')
end (case)
end.

```

```

6. program częstość;
var litera: array['A'..'Z'] of integer;
    znak: char;
    tekst: string[30];
    i: integer;
begin
  for znak:='A' to 'Z' do
    litera[znak]:=0;
  clrscr;
  writeln(' Podaj tekst');
  readln(tekst);
  for i:=1 to length(tekst) do
  begin
    znak:=tekst[i];
    if znak in ['A'..'Z']
      then litera [znak]:=litera[znak]+1
    end (i);
  writeln;
  writeln(' LITERA      LICZBA WYSTĄPIEN');
  writeln;
  for znak:='A' to 'Z' do
    if litera[znak] > 0 then writeln(znak:5,litera[znak]:16)
  end.

```

```

7. program głoski;
var znak: char;
    licznik: integer;
begin
  licznik:=0;
  repeat
    readln(znak);
    if znak in ['a','e','i','o','u','y'] then licznik:=licznik+1
  until znak='.';
  writeln('Liczba głosek w tekście wynosi ',licznik:5)
end.

```

```

9. program sito;
   const maxn=100;
   var liczby: set of 2..maxn;
       i,m: integer;
begin
   clrscr;
   writeln(' Liczby pierwsze mniejsze od ',maxn:5,' :');
   writeln;
   liczby:=[2..maxn];
   for i:=2 to maxn do
     if i in liczby
       then begin
           writeln(i:4);
           for m:=1 to (maxn div i) do
             liczby:=liczby-[i*m]
           end
         end
   end.

```

ROZDZIAŁ 9

```

1. program piszliczby;
   var f: file of real;
       i: integer;
       x,eps: real;
begin
   assign(f,'dane');
   rewrite(f);
   clrscr;
   write('Podaj eps: ');
   readln(eps);
   i:=0;
   repeat
     i:=i+1;
     x:=(i-0.1)/(i*sqr(i)+abs(sin(2*i)/cos(2*i)));
     write(f,x)
   until x<eps;
   close(f)
end.

```

ROZDZIAK 10

```
2b procedure zwolnij(var a: lista);
```

```
begin
```

```
  if a <> nil then
```

```
    begin
```

```
      zwolnij(a^.wsk);
```

```
      dispose(a)
```

```
    end
```

```
end {zwolnij};
```

```
8. program wsk;
```

```
const nmax=10;
```

```
type tab=array[1..nmax] of ^integer;
```

```
var A: tab;
```

```
  i,n,suma,iloczyn: integer;
```

```
begin
```

```
  clrscr;
```

```
  write(' Podaj n: ');
```

```
  readln(n);
```

```
  for i:=1 to n do
```

```
    begin
```

```
      new(A[i]);
```

```
      write(' Podaj wartość kolejnego elementu: ');
```

```
      readln(A[i]^)
```

```
    end {i};
```

```
    suma:=0;
```

```
    iloczyn:=1;
```

```
    for i:=1 to n do
```

```
      begin
```

```
        suma:=suma+A[i]^;
```

```
        iloczyn:=iloczyn*A[i]^
```

```
      end {i};
```

```
      writeln('suma=',suma);
```

```
      writeln('iloczyn=',iloczyn)
```

```
end.
```

ROZDZIAŁ 11

28. program potęgi;

```

const nmax=10;
var p2,p3,p4: real;
    n: integer;
begin
  clrscr;writeln;
  writeln(' k          2          3          4');
  writeln('-----');
  p2:=1.0; p3:=1.0; p4:=1.0;
  writeln(' 0',p2:11:1,p3:11:1,p4:11:1);
  for n:=1 to nmax do
  begin
    p2:=p2+p2;    p3:=3.0*p3;    p4:=4.0*p4;
    writeln(n:2,p2:11:1,p3:11:1,p4:11:1)
  end {n}
end.

```

63. program przesun;

```

type tablica=array[1..10] of real;
var a: tablica;
    ka,k,n,i: integer;

procedure cykl1(n: integer; var x: tablica);
var i: integer;
    z: real;
begin
  z:=x[1];
  for i:=1 to n-1 do x[i]:=x[i+1];
  x[n]:=z
end {cykl1};

procedure cyklm1(n: integer; var x: tablica);
var i: integer;
    z: real;
begin
  z:=x[n];
  for i:=n downto 2 do x[i]:=x[i-1];
  x[1]:=z
end {cyklm1};

```



```

begin
  clrscr;
  write('Podaj n: ');
  readln(n);
  writeln('Podaj elementy ciągu:');
  for i:=1 to n do
    readln(a[i]);
  write('Podaj k: '); readln(k);
  writeln;
  writeln('Ciąg a:');
  for i:=1 to n do write(a[i]:5:1);
  writeln;
  ka:=abs(k);
  if ka>n then ka:=ka mod n;
  if k>0 then for i:=1 to ka do cykl1(n,a)
    else if k<0 then for i:=1 to ka do cyklm1(n,a);
  writeln('Ciąg a po przesunięciu o',k:5);
  for i:=1 to n do write(a[i]:5:1)
end.

```

64a program m3d1;

```

const kk=10; {kk-k końcowe}
var i,k,nk,maxk: integer;
begin
  clrscr;
  writeln('    k    n(k)    max(k)');
  for i:=1 to kk do
    begin
      k:=i;
      nk:=0;
      maxk:=k;
      while k<>1 do
        begin
          if odd(k) then k:=3*k+1
            else k:=k div 2;
          nk:=nk+1;
          if k>maxk then maxk:=k
        end {k<>1};
      writeln(i:4,nk:6,maxk:9)
    end {i}
end.

```

64b program zbiorcy;

```

const nmax=10;
var Z,noweZ: array[1..nmax] of integer;
    j,n,s,licznik: integer; {s-liczba elementów w Z}
                                {licznik-liczba elementów w noweZ}
    pom:          real; {j,n,pom-zmienne      pomocnicze}
begin
    writeln('  s      n      elementy z(n)');
    n:=0;
    s:=1;
    Z[1]:=1;
    write(s:3,n:5,z[1]:6);
    for n:=1 to nmax do
    begin
        writeln;writeln;
        licznik:=s;
        for j:=1 to s do
        begin
            noweZ[j]:=2*Z[j];
            pom:=(Z[j]-4)/6.0;
            if (frac(pom)=0.0) and (pom>0.0)
                {wynikiem frac jest część ułamkowa liczby rzeczywistej}
            then
            begin
                licznik:=licznik+1;
                noweZ[licznik]:=(Z[j]-1) div 3
            end {if}
        end {j};
        s:=licznik;
        write(s:3,n:5);
        for j:=1 to s do
        begin
            write(noweZ[j]:6);
            Z[j]:=noweZ[j]
        end {j}
    end {n}
end.
```

65. program trzydod;

```

var a: array[1..10] of integer;
    n,i,l,licznik: integer;
begin
```

```

write(' Podaj n: '); readln(n);
for i:=1 to n do readln(a[i]);
i:=1; licznik:=0;
while i<=n do
begin
  if a[i]>0 then
    begin
      l:=0;
      repeat
        l:=l+1;
        i:=i+1
      until (i>n) or (a[i]<=0);
      if l>=3 then licznik:=licznik+1
    end {a[i]>0};
    i:=i+1
  end {while};
  writeln(' Ciąg a:');
  for i:=1 to n do write(a[i]:4);
  writeln;
  writeln(' Liczba podciągów=',licznik:3)
end.

```

92. program wyrażenie;

```

type tekst=string[30];
var wyrażenie: tekst;
    wartość,i: integer;

function liczba(var i: integer; a: tekst): integer;
{funkcja wyznacza wartość liczby występującej w łańcuchu a od
pozycji i}
var t,p: integer;
begin
  p:=1; {p-potęgi 10}
  t:=0; {t-wartość liczby}
  i:=i+1;
  while a[i] in ['0'..'9'] do
  begin
    t:=t*p+ord(a[i])-ord('0');
    i:=i+1;
    p:=10*p
  end {while};
  liczba:=t

```

```
end {liczba};

begin
  clrscr;
  writeln(' Podaj wyrażenie:');
  readln(wyrażenie);
  wartość:=0;
  i:=1;
  while wyrażenie[i]<>';' do
    case wyrażenie[i] of
      '+': wartość:=wartość+liczba(i,wyrażenie);
      '-': wartość:=wartość-liczba(i,wyrażenie)
    end {case};
    writeln(' Wartość wyrażenia = ',wartość:8)
  end.
```


DODATEK A

SKŁADNIA JĘZYKA TURBO PASCAL 3.0 (DIAGRAMY SYNTAKTYCZNE)

Uwagi

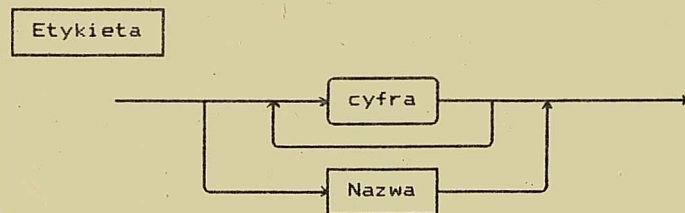
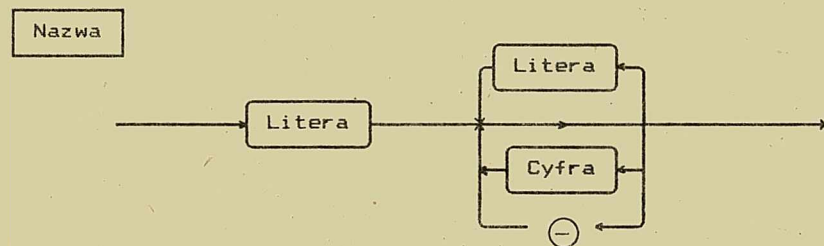
1. Prostokąty oznaczają diagramy syntaktyczne, natomiast kółka i owale oznaczają symbole języka Turbo Pascal 3.0.

2. Między kolejnymi symbolami może wystąpić dowolna liczba znaków rozdzielających z jednym wyjątkiem: żadne znaki rozdzielające nie mogą występować wewnątrz nazw i liczb.

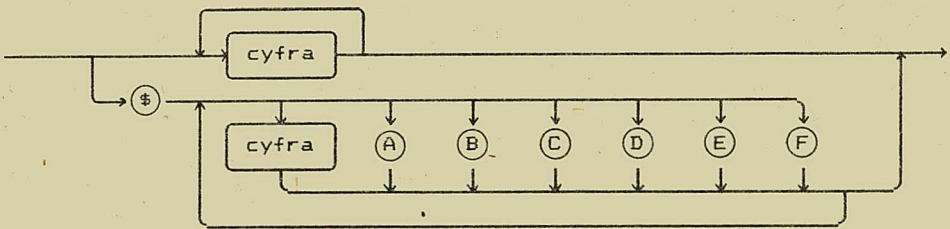
3. Przynajmniej jeden znak rozdzielający musi wystąpić między następującymi po sobie nazwami (lub liczbami) i symbolami podstawowymi (takimi jak np. begin, end).

4. Znakami rozdzielającymi mogą być spacje, komentarze i koniec wiersza. Komentarzem jest dowolny ciąg znaków ujęty w nawiasy klamrowe { oraz } lub (* i *).

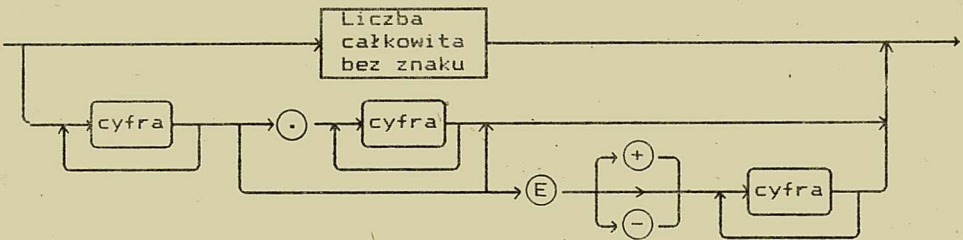
5. Słowo "nazwa" oznacza nazwę stałej, typu, zmiennej, funkcji lub procedury.



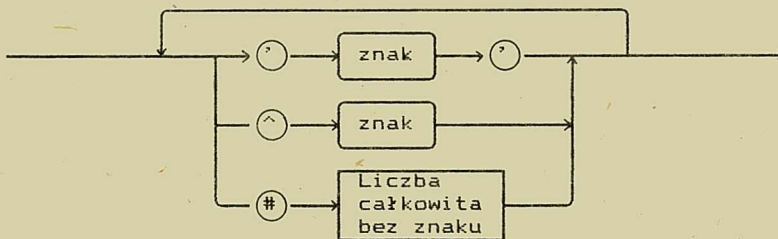
Liczba całkowita bez znaku



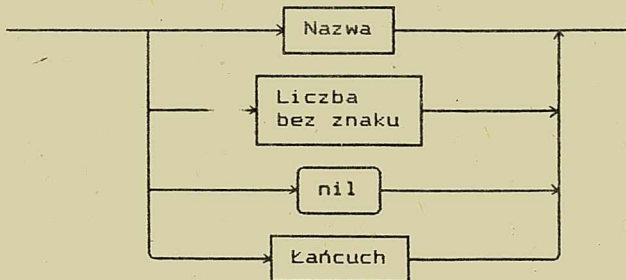
Liczba bez znaku



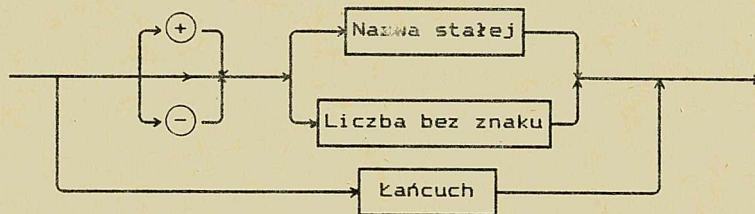
Łańcuch



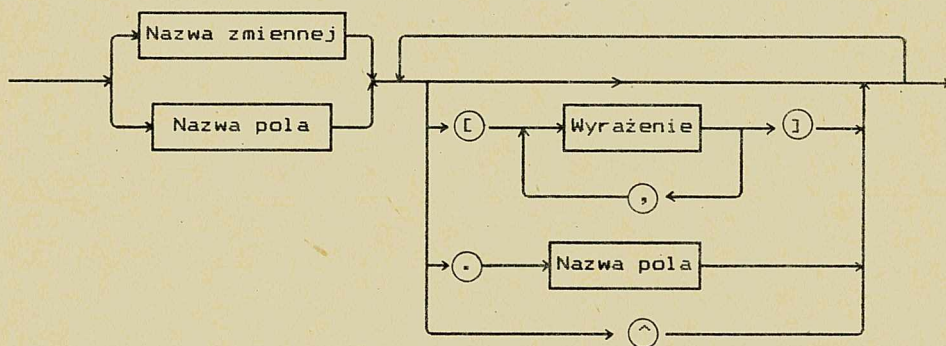
Stała bez znaku



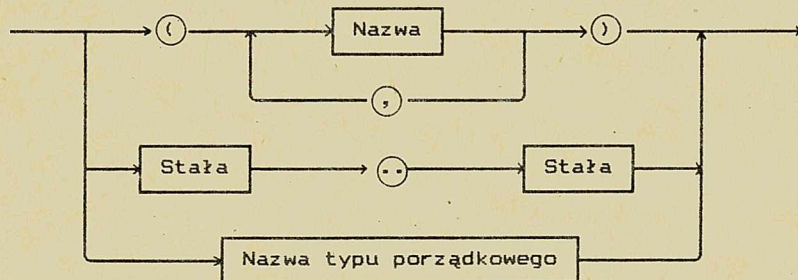
Stała

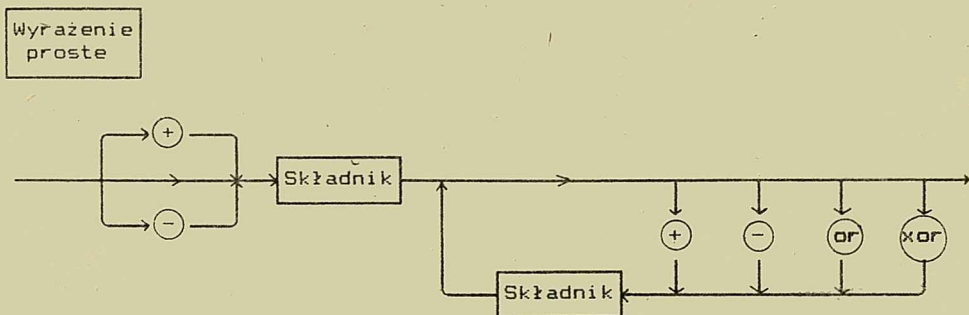
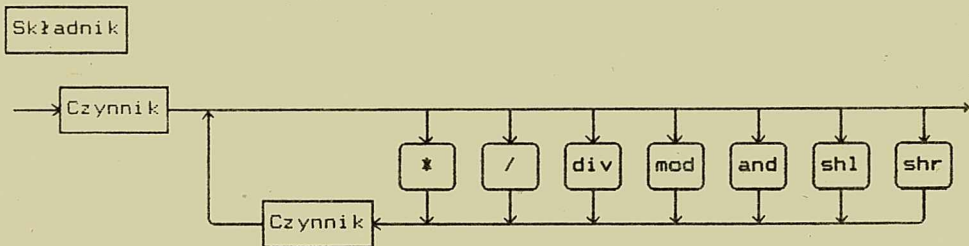
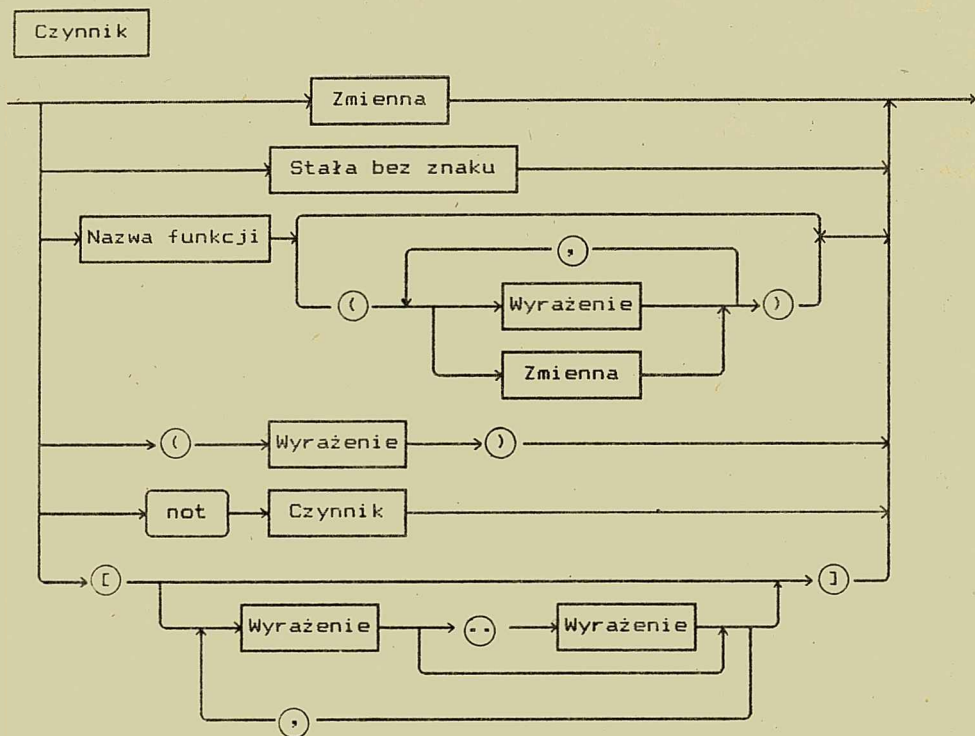


Zmienna

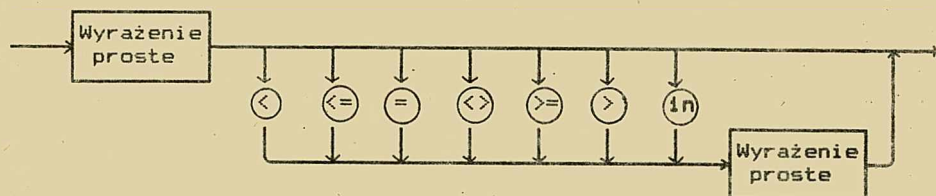


Typ porządkowy

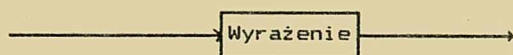
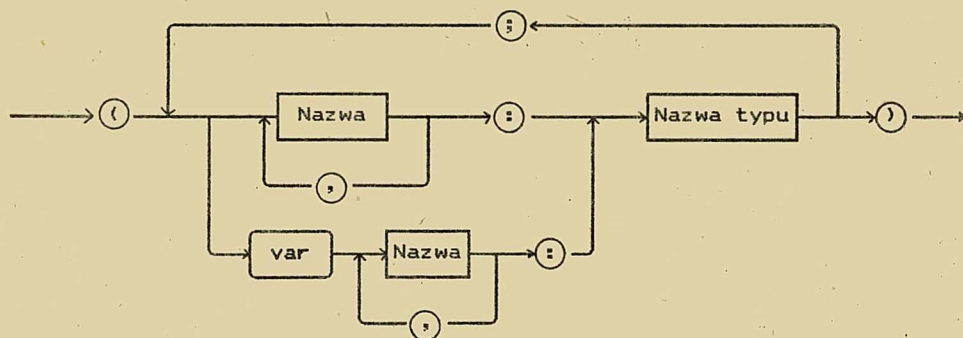


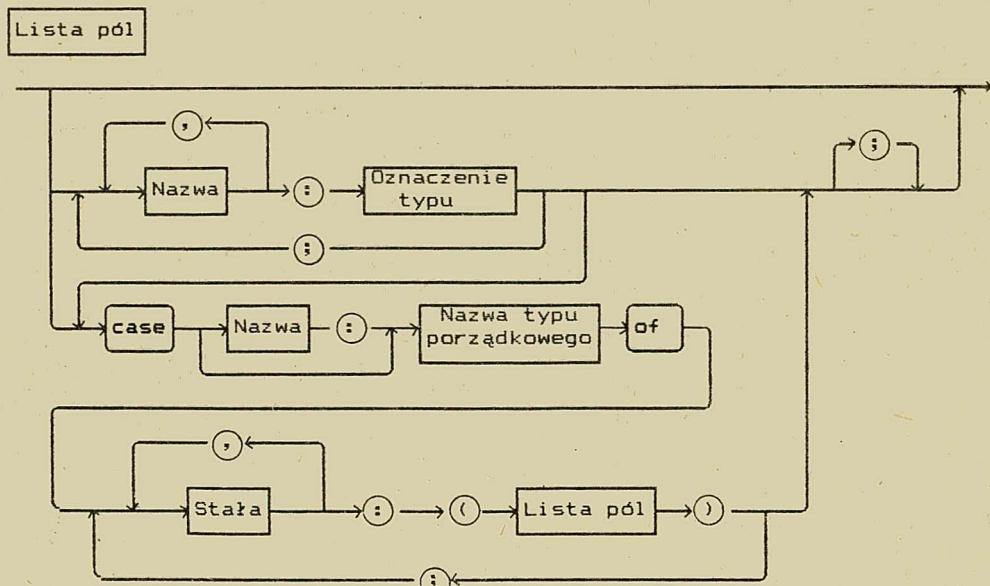
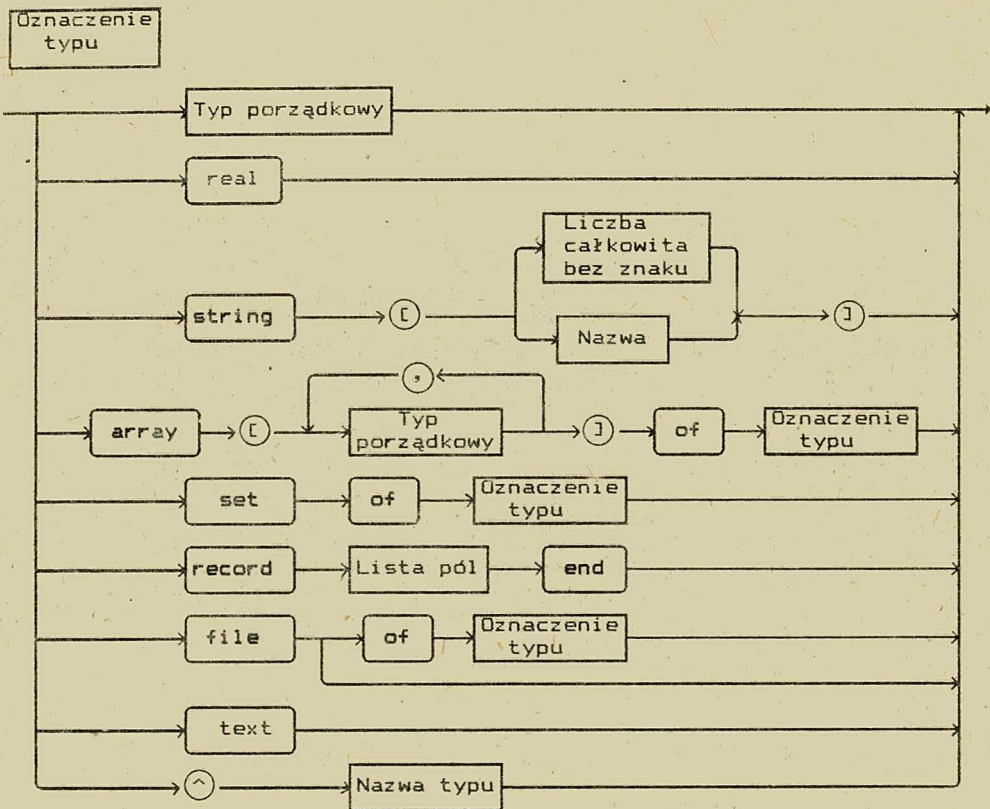


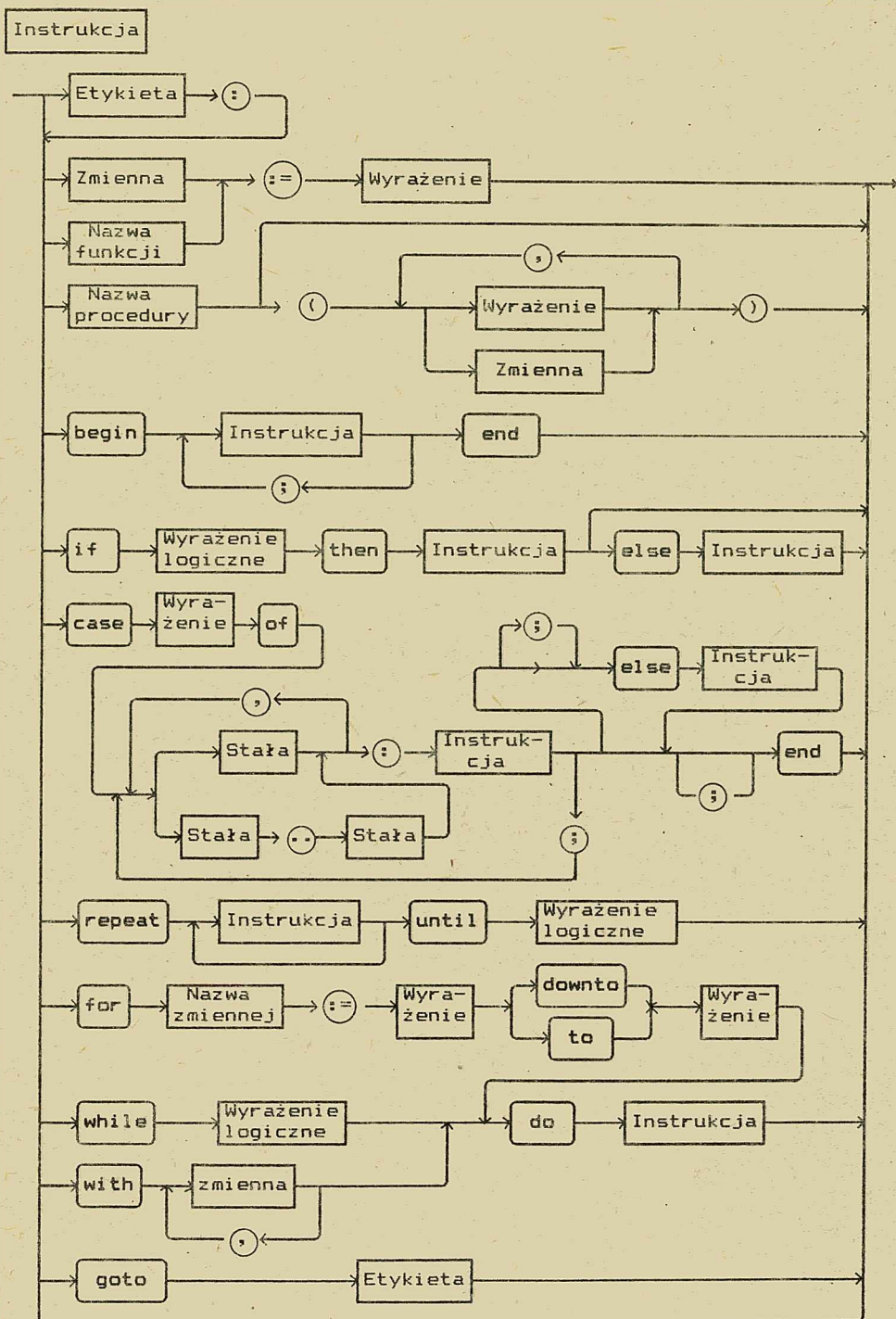
Wyrażenie



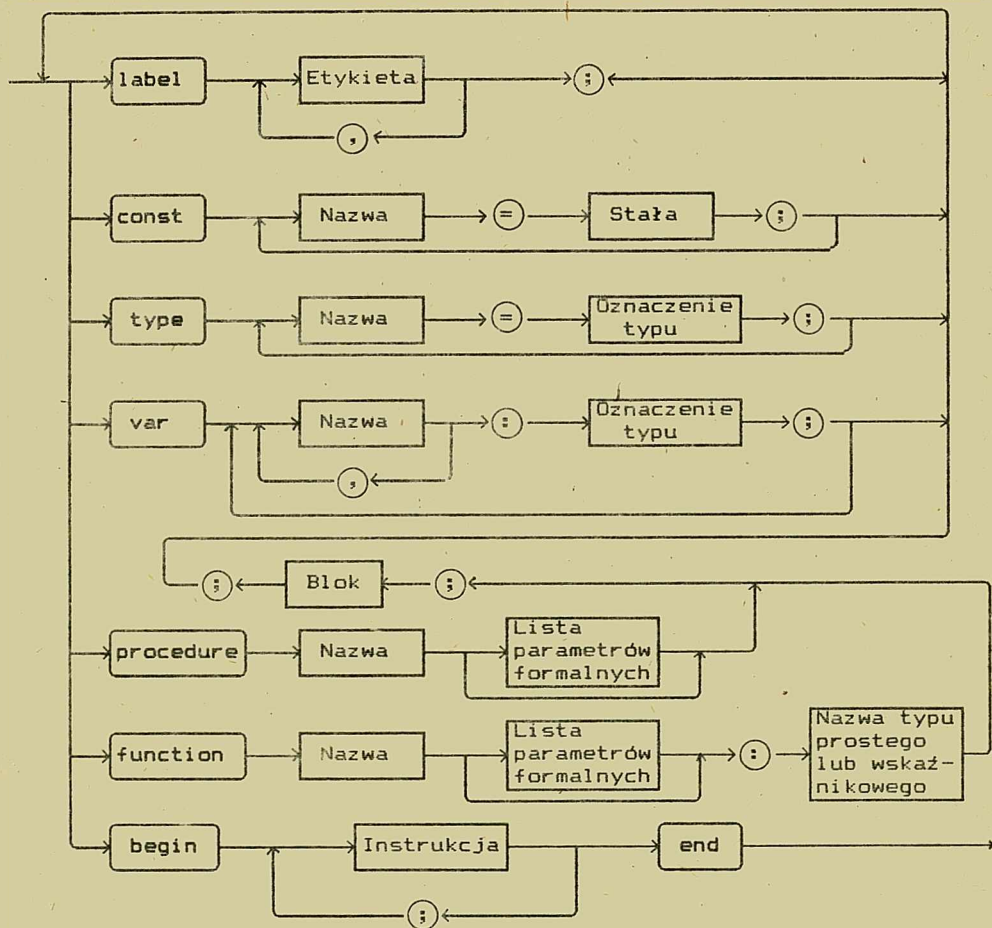
Wyrażenie logiczne

Lista
parametrów
formalnych

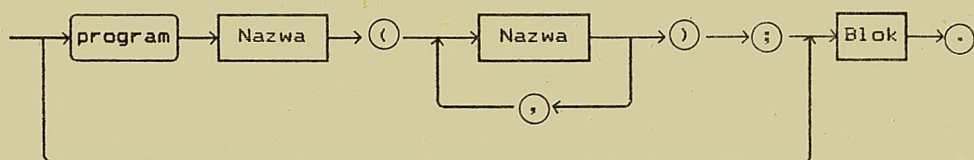




Blok



Program



DODATEK B

ZESTAWIENIE OBIEKTÓW STANDARDOWYCH W JĘZYKU TURBO PASCAL 3.0

W rozdziale tym zebrano wszystkie stałe, zmienne, funkcje i procedury standardowe w języku Turbo Pascal 3.0.

STAŁE STANDARDOWE

false, true	p. 4.4.1
maxint	p. 4.4.1, 4.5
pi	p. 4.5

TYPY STANDARDOWE

integer, byte, real	}	p. 4.4.1
char, boolean		
text		podrozdz. 9.1

ZMIENNE STANDARDOWE

input, output	podrozdz. 9.1
---------------	---------------

FUNKCJE STANDARDOWE

abs(x)	}	wartość bezwzględna x,
sqr(x)		kwadrat x, x - typu integer lub real, typ wyniku jak typ argumentu;
arctan(x)	}	arcus tangens x,
cos(x)		cosinus x,
exp(x)		e^x ,
frac(x)		część ułamkowa x,
int(x)		część całkowita x,
ln(x)		logarytm naturalny x,
sin(x)		sinus x,
sqr(x)		pierwiastek kwadratowy x, x - typu integer lub real, typ wyniku - real;

round(x) }
trunc(x) }

odd(x)

pred(x) }
succ(x) }

chr(x)

ord(x)

hi(x)

lo(x)

random

random(x)

keypressed

paramcount

paramstr(x)

zaokrąglenie x do najbliższej liczby całkowitej,
część całkowita x,
x - typu real, typ wyniku - integer;
p. 4.4.1

nieparzystość x,

$$\text{odd}(x) = \begin{cases} \text{true} & \text{jeśli } x \text{ - nieparzyste,} \\ \text{false} & \text{w przeciwnym przypadku,} \end{cases}$$
 x - typu integer,
typ wyniku - boolean

poprzednik x,
następnik x,
x - typu porządkowego,
typ wyniku taki jak typ argumentu;
podrozdz. 4.4

znak o kodzie x,
x - typu integer, typ wyniku - char;
podrozdz. 4.4

numer porządkowy x,
x - typu porządkowego,
typ wyniku - integer;
podrozdz. 4.4

wartość bardziej znaczącego bajtu x,
wartość mniej znaczącego bajtu x,
x - typu integer, typ wyniku - integer;
wartość losowa z przedziału [0,1),
typ wyniku - real;

wartość losowa z przedziału [0,x-1],
x - typu integer, typ wyniku - integer;
sprawdzenie, czy w buforze wejściowym konsoli znajduje się nie wprowadzony jeszcze znak,
typ wyniku - boolean;

liczba parametrów przekazanych do programu w chwili jego wywołania,
typ wyniku - integer;

łańcuch reprezentujący parametr o numerze x, przekazany programowi w chwili jego wywołania,
x - typu integer, typ wyniku - string;

<code>sizeof(x)</code>	liczba bajtów przydzielonych danej <code>x</code> lub potrzebnych do reprezentowania danej typu <code>x</code> , <code>x</code> – nazwa zmiennej lub nazwa typu, typ wyniku – integer;
<code>swap(x)</code>	przypisanie danej, której bardziej znaczący bajt ma wartość <code>lo(x)</code> a mniej znaczący bajt ma wartość <code>hi(x)</code> , <code>x</code> – typu integer, typ wyniku – integer;
<code>upcase(x)</code>	zamiana małej litery na odpowiednią wielką, <code>x</code> , wynik – typu char; p. 4.4.1
<code>concat(x1,...,xn)</code> <code>copy(x,n,k)</code> <code>length(x)</code> <code>pos(x1,x2)</code>	funkcje dotyczące zmiennych typu <code>string</code> , podrozdz. 6.1
<code>eof(f)</code> <code>eoln</code> <code>filesize(f)</code> <code>filepos(f)</code> <code>seekeof(f)</code> <code>seekeoln(f)</code>	
<code>maxavail</code>	rozmiar największego spójnego obszaru pamięci operacyjnej dostępnego na stercie, typ wyniku – integer;

PROCEDURY STANDARDOWE

<code>randomize</code>	zainicjowanie generatora liczb losowych;
<code>clreol</code>	zastąpienie spacjami wszystkich znaków wiersza, w którym znajduje się kursor;
<code>clrscr</code>	czyszczenie ekranu;
<code>crtnit</code> <code>crtexit</code>	skierowanie do monitora łańcucha znaków inicjujących (kończących), określonego podczas instalowania systemu Turbo Pascal;
<code>delline</code>	usunięcie wiersza, w którym znajduje się kursor, pozycja kursora nie ulega zmianie;

insline

wstawienie pustego wiersza w miejscu wskazywanym przez kursor, pozycja kursora nie ulega zmianie;

gotoxy(x,y)

ustawienie kursora w wierszu o numerze $y \bmod n$ i kolumnie o numerze $x \bmod m$, przy czym lewy górny narożnik ekranu ma współrzędne (1,1), x zmienia się od lewej do prawej a y od góry do dołu;

lowvideo

poddawanie inwersji znaków wyprowadzanych na ekran;

normvideo

powrót do normalnego wyprowadzania znaków na ekran;

delay(t)

wstrzymanie wykonywania programu na t milisekund (jeśli $t < 0$, to przyjmuje się $t = 0$);

fill(x,k,z)

umieszczenie w obszarze, którego początek przydzielono zmiennej x , k danych bajtowych o takiej samej reprezentacji jak zmienna bajtowa z , gdzie
 x - nazwa zmiennej dowolnego typu,
 k - wyrażenie typu integer,
 z - wyrażenie typu char lub byte;

exit

zakończenie wykonywania bloku, w którym tę procedurę wywołano;

halt

zakończenie wykonywania programu;

move(x,y,k)

przepisanie zawartości k bajtów z obszaru pamięci, którego początek przydzielono zmiennej x do obszaru, którego początek przydzielono zmiennej y , gdzie
 x, y - nazwy zmiennych dowolnego typu,
 k - wyrażenie typu integer;

delete(x,n,k)

insert(x1,x2,n)

str(w,x)

str(w:m,x)

str(w:m:n,x)

val(x,y,k)

procedury dotyczące łańcuchów,
 podrozdz. 6.1

<pre> assign(f,x) reset(f) rewrite(f) read(f,x1,...,xn) readln(f) readln(f,x1,...,xn) write(f,x1,...,xn) writeln(f,x1,...,xn) writeln(f) seek(f,n) close(f) rename(f,x) erase(f) </pre>	<pre> } </pre> procedury dotyczące zmiennych typu plikowego; rozdz. 9
<pre> blockread(f,x,n,licz) blockread(f,x,n) blockwrite(f,x,n,licz) blockwrite(f,x,n) </pre>	<pre> } </pre> procedury dotyczące plików bez typu elementów; podrozdz. 9.2
<pre> new(x) dispose(x) mark(x) release(x) getmem(x,k) freemem(x,k) </pre>	<pre> } </pre> procedury związane z dynamicznym zarządzaniem pamięcią operacyjną; rozdz. 10

TABLICA KODU ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	^@ NUL	32	20		64	40	@	96	60	`
1	01	^A SOH	33	21	!	65	41	A	97	61	a
2	02	^B STH	34	22	"	66	42	B	98	62	b
3	03	^C ETX	35	23	#	67	43	C	99	63	c
4	04	^D EOT	36	24	\$	68	44	D	100	64	d
5	05	^E ENQ	37	25	%	69	45	E	101	65	e
6	06	^F ACK	38	26	&	70	46	F	102	66	f
7	07	^G BEL	39	27	'	71	47	G	103	67	g
8	08	^H BS	40	28	(	72	48	H	104	68	h
9	09	^I HT	41	29	)	73	49	I	105	69	i
10	0A	^J LF	42	2A	*	74	4A	J	106	6A	j
11	0B	^K VT	43	2B	+	75	4B	K	107	6B	k
12	0C	^L FF	44	2C	,	76	4C	L	108	6C	l
13	0D	^M CR	45	2D	-	77	4D	M	109	6D	m
14	0E	^N SO	46	2E	.	78	4E	N	110	6E	n
15	0F	^O SI	47	2F	/	79	4F	O	111	6F	o
16	10	^P DLE	48	30	0	80	50	P	112	70	p
17	11	^Q DC1	49	31	1	81	51	Q	113	71	q
18	12	^R DC2	50	32	2	82	52	R	114	72	r
19	13	^S DC3	51	33	3	83	53	S	115	73	s
20	14	^T DC4	52	34	4	84	54	T	116	74	t
21	15	^U NAK	53	35	5	85	55	U	117	75	u
22	16	^V SYN	54	36	6	86	56	V	118	76	v
23	17	^W ETB	55	37	7	87	57	W	119	77	w
24	18	^X CAN	56	38	8	88	58	X	120	78	x
25	19	^Y EM	57	39	9	89	59	Y	121	79	y
26	1A	^Z SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	^L ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	^N FS	60	3C	<	92	5C	\	124	7C	
29	1D	^J GS	61	3D	=	93	5D	]	125	7D	}
30	1E	^^ RS	62	3E	>	94	5E	^	126	7E	~
31	1F	^_ US	63	3F	?	95	5F	_	127	7F	DEL

LITERATURA

- [1] ALAGIC S., ARBIB M. A., Projektowanie programów poprawnych i dobrze zbudowanych, Warszawa, WNT, 1982.
- [2] BIELECKI J., Turbo Pascal wersja 3.0, Warszawa, WNT, 1987.
- [3] GROGONO P., Programming in Pascal, Addison-Wesley Publishing Company, INC. 1978.
- [4] IGLEWSKI M., MADEY J., MATWIN S., Pascal, Język wzorcowy, Pascal 360, wyd. 4, Warszawa, WNT, 1986.
- [5] KOTT R. K., Programowanie w języku Pascal, Warszawa, WNT, 1988.

SKOROWIDZ

A

abs 233
 algorytm 7
 alternatywa 29
 and 29, 228
 apostrof 104
 arctan 233
 array 61, 230
 assign 141, 145

B

bajt 27
 begin 16
 bit 27
 blok 16, 21, 232
 błędy kompilacji 13
 - wykonania 13
 boolean 29, 233
 blockread 147
 blockwrite 147
 byte 28, 233

C

case 19, 230, 231
 char 29, 233
 chr 30
 close 142, 145
 clreol 235
 clrscr 20, 235
 concat 106
 const 31
 copy 106
 cos 233
 crtinit 235

crtexit 235

częściowy porządek 63
 część definiująca 16, 21
 - deklarująca 16, 21
 - wykonawcza 16
 - zmienna 121
 czynnik 228
 czytelność programu 7, 8

D

definicja funkcji 21
 - procedury 21
 - stałej 31
 deklaracja zmiennej 32
 delay 20, 236
 delete 107, 236
 delline 20, 235
 diagramy syntaktyczne 16, 225
 dispose 153
 div 27, 228
 długość łańcucha 104
 - pliku 141
 do 18
 downto 18
 drzewo 159
 - binarne 160

E

efektywność programu 7, 8
 else 18
 end 16
 eof 143, 146
 eoln 146
 erase 142

etykieta 20, 225

exit 236

exp 233

F

false 29, 233

file 141

filepos 143

filesize 143

fill 236

for 18, 231

frac 233

freemem 154, 237

function 21

funkcja 21

- rekurencyjna 24

- standardowa 106, 233

G

getmem 154, 237

goto 20, 231

gotoxy 20, 236

H

halt 218

hi 216

I

if 18, 231

in 132, 229

input 145, 233

insert 107, 236

inline 236

instrukcja 16, 231

- case 19, 231

- for 18, 231

- if 18, 231

- iteracyjna 18

- przypisania 17

- pusta 18

- repeat 18, 231

- skoku 20

- warunkowa 18

- while 18, 231

- with 20, 117, 231

- wiążąca 20, 117

- wyboru 19

- złożona 17

int 233

integer 27, 233

K

keypressed 18, 234

kod ASCII 238

kolejka 159

komentarz 15, 18, 225

kompilacja programu 13

kompilator 13

koniunkcja 29

konkatenacja 105

korzeń drzewa 159

kropka 16

L

label 232

length 106

liczba automorficzna 180

- Stirlinga 180

liczby bliźniacze 179

- piramidalne 180

- trójkątne 180

lista 152

- cykliczna 159

- dwukierunkowa 159

- jednokierunkowa 152

- nazw rekordów 20

- parametrów aktualnych 20

- formalnych 21, 229

- pól 115, 230

- stałych wyboru 19

liść drzewa 169

ln 233

lo 234

lowvideo 236

Ł

łańcuch 104, 226
- pusty 105

M

mark 154, 237
maxavail 154, 235
maxint 27, 32, 233
metoda połowienia przedziału 38
mod 27, 228
move 236

N

nałówek funkcji 21
- procedury 21
- programu 16
nazwa 16, 30, 225
negacja 29
new 153
nil 152
normvideo 236
not 29, 228

O

odd 234
of 19, 61
operacje na łańcuchach 105
- - plikach 141, 237
operator relacyjny 26, 106
or 27, 29, 228
ord 26, 28, 29, 131, 234
otwarcie pliku 142
output 145, 233
oznaczenie typu 230

P

paramcount 234
parametr aktualny 21
- formalny 21
paramstr 234
pi 32, 233
plik 141

- bez typu elementów 146
- pusty 141
- standardowy 144
- tekstowy 144
poddrzewo 159
podprogram 21
poprawność programu 7
porządek 161
- częściowy 63
- poprzeczny 161
- wsteczny 161
- wzdłużny 161
pos 106
poziom drzewa 169
pred 26, 28, 29, 234
procedura 20, 21
- rekurencyjna 24
- standardowa 107, 141, 154, 235
procedure 21
program 16, 232
programowanie strukturalne 7
przekazywanie parametrów 22
- - przez funkcję 23
- - - procedurę 23
- - - wartość 22, 23
- - - zmienną 22, 23

R

random 234
randomize 237
read 142, 145, 236
readln 142, 145, 236
real 28, 233
record 115
rekord 1115
- z wariantami 120
relacja 63
- antysymetryczna 63
- przechodnia 63
- zwrotna 63
release 154, 237
rename 142, 237

repeat 18, 231
 reset 142, 145, 237
 rewrite 142, 145, 237
 round 28, 234

S

seek 142, 237
 seekeof 146, 235
 seekedln 146, 235
 set 131
 shl 27, 228
 shr 27, 228
 sin 233
 sizeof 235
 składnia 16, 225
 składnik 228
 sqr 233
 sqrt 233
 stała 152, 227
 sterta 154
 stos 159
 str 107, 236
 string 104
 succ 26, 28, 234
 swap 235
 symbol specjalny 29
 szkic programu 7

T

tablica 61
 - znakowa 62
 text 144, 230, 233
 then 18
 to 18
 true 29, 233
 trunc 28, 234
 typ 25
 - bajtowy 25, 28
 - bazowy 131
 - całkowity 25, 27
 - danej 25
 - logiczny 25, 29

- łańcuchowy 25, 104
 - okrojony 26, 31
 - plikowy 25, 141
 - porządkowy 25, 26, 227
 - prosty 25, 26
 - rekordowy 25, 115
 - rzeczywisty 25, 28
 - standardowy 25, 26
 - strukturalny 25, 26
 - tablicowy 25, 26, 61
 - wskaźnikowy 25, 152
 - wyliczeniowy 26, 30
 - wyniku funkcji 21
 - zbiorowy 25, 131
 - złożony 25
 - znakowy 25, 29
 type 32

U

until 18
 upcase 29, 235
 uniwersalność programu 7
 urządzenie zewnętrzne 114, 145

V

var 32
 val 108, 236

W

waga drzewa 169
 wariant 120
 wartość funkcji 21
 - indeksu 61
 - początkowa 33
 węzeł drzewa 159
 while 18
 with 20, 117
 write 142, 146, 237
 writeln 142, 146, 237
 wstępująca metoda 9
 wykonanie programu 13
 wyrażenie 229

- logiczne 229
- proste 228
- wysokość drzewa 169

X

xor 29, 228

Z

- zakres deklaracji 32
- zamknięcie pliku 142
- zbiór 131
 - bazowy 131
 - danych 141
 - potęgowy 131
- zmienna 25, 227
 - dynamiczna 27, 152
 - lokalna 21, 23
 - standardowa 233
 - statyczna 27
- znak końca wiersza 144
- rozdzielający 225
- sterujący 32
- zstępująca metoda 9
- zupełność programu 7, 8



263892 ⁰/₂

Cena zł 2500,-

Wydawnictwa Politechniki Wrocławskiej
są do nabycia w:
P.P. „Dom Książki”
Księgarni Wr 49
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
oraz
Wojewódzkiej Księgarni Technicznej
ul. Świdnicka 8, 50-067 Wrocław