

1. POZIOMY JĘZYKÓW PROGRAMOWANIA	2
2 STRUKTURA PROGRAMU	3
3. NAJCZĘŚCIEJ STOSOWANE TYPY ZMIENNYCH:	4
4. PODSTAWOWE POLECENIA JĘZYKA PASCAL	4
5. INSTRUKCJA WARUNKOWA, INSTRUKCJA WYBORU	6
6. PĘTLE	15
7. STANDARDOWE FUNKCJE MATEMATYCZNE	21
8. ZMIENNE LOSOWE	22
9. NADAWANIE ZMIENNYCH PROSTYCH	23
10. RELACJE LOGICZNE	24
11. DZIAŁANIE NA ZMIENNYCH LICZBOWYCH	24
12. ODCZYTYWANIE DANYCH Z KLAWIATURY	25
13. TYPY PORZĄDKOWE	25
14. DEFINIOWANIE TYPÓW	26
15. ZMIANA TYPÓW	26
16. OPERACJE NA ZMIENNYCH ŁAŃCUCHOWYCH	27
17. PROCEDURY	30
18. FUNKCJE	33
19. TABLICE	35
20. RECORDY	41
21. KOLORY WYDRUKU	43
22. KOLOR TŁA	43
23. SYGNAŁY DŹWIĘKOWE	43
24. PRZERYWANIE WYKONANIA PROGRAMU	44
25. ELEMENTY GRAFIKI W TP	44
26. PLIKI.	44
27. PLIKI TEKSTOWE	44
28. PLIKI ELEMENTOWE	47
29. OPERACJE NA PLIKACH DYSKOWYCH	48
30. DOPISYWANIE I USUWANIE TEKSTU	49

o

1. POZIOMY JĘZYKÓW PROGRAMOWANIA

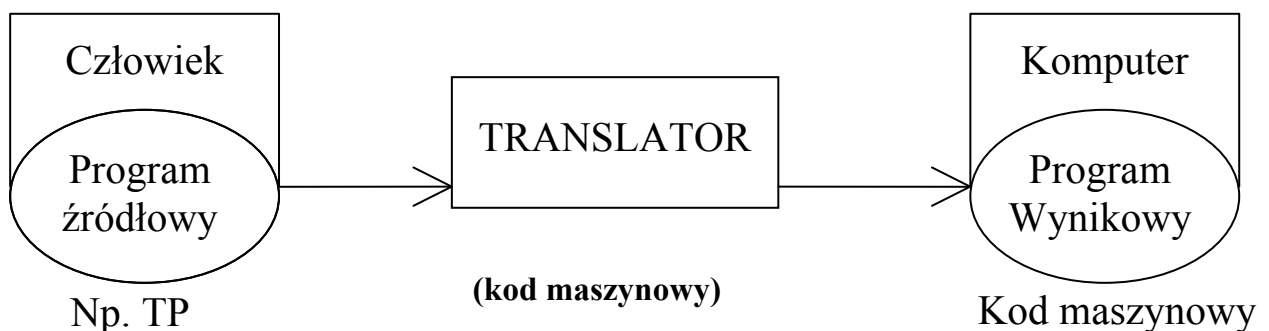
- 1) Język najniższego poziomu (rozказы komputerowe 0,1)
- 2) Język niskiego poziomu (assembler - rozказы zapisywane są w postaci symbolicznej, łatwiejszej do kojarzenia przez człowieka).
- 3) Język wysokiego poziomu - używa sformułowań podobnych do stosowanych w języku naturalnym (np. wzory matematyczne zapisujemy w sposób podobny do stosowanego w matematyce) Wymagają mniejszego nakładu pracy, gdyż jednemu rozkazowi języka wysokiego poziomu może odpowiadać kilka lub kilkanaście rozkazów języka komputerowego.
(Pascal, C++, Delphi)

Wśród opracowanych języków są:

- a) języki problemowe odpowiadające pewnym klasom zastosowań np.: SQL – relacyjne bazy danych.
- b) języki uniwersalne np: Pascal , Basic , Visual Basic ,
Język C służy do opracowania programów i tworzenia systemów operacyjnych np: UNIX. Język C zajmuje pozycję dominującą w profesjonalnych zastosowaniach.
- c) języki związane z konkretnymi programami użytkowymi lub specjalistycznymi.
Np.: dBase - tworzenie programów baz danych.

SYSTEM PROGRAMOWANIA

- a) Translator - program tłumaczący z języka programowego na rozказы komputerowe
- b) Program źródłowy - program napisany w języku programowanym
- c) Program wynikowy (czasami wykonywalny) - program przetłumaczony przez translator



2 STRUKTURA PROGRAMU

Program ▽ nazwa; { nagłówek programu }



begin {słowo kluczowe rozpoczynające blok wykonawczy}

Ciąg instrukcji

end. {słowo kluczowe kończące blok wykonawczy}

a) Nagłówek programu

Po słowie kluczowym program występuje tytuł programu, który może być ciągiem dużych lub małych znaków nie zaczynający się jednak od cyfry, bez znaków zastrzeżonych i spacji. Tytuł pełni rolę informacyjną.

b) Blok deklaracyjny

Po nagłówku jest blok deklaracyjny zawierający określone deklaracje modułu wartości stałych i zmiennych, etykiet, procedur i funkcji.

c) Blok wykonawczy

Słowo kluczowe begin uruchamia program główny zawierający określone instrukcje oraz wywołanie zadanych procedur lub funkcji, które zostały wcześniej zadeklarowane. Słowo kluczowe end z „kropką” zamyka cały program. W bloku wykonawczym mogą występować podbloki wykonawcze zawierające, podobnie jak bloki, pewne instrukcje, procedury i funkcje.

Ogranicznikami podbloku są słowa kluczowe begin, end (bez kropki). Do oddzielenia poszczególnych deklaracji oraz instrukcji służy separator w postaci średnika. Po begin nie stawiamy ;

Wszystkie zmienne w programie powinny być zadeklarowane przez umieszczenie ich nazw i nazw typów za słowem kluczowym VAR w bloku deklaracyjnym.

3. NAJCZĘŚCIEJ STOSOWANE TYPY ZMIENNYCH:

Typ zmienny		Bajty	Wartość minimalna	Wartość maksymalna
Integer	Całkowita	2	-32768	32767
Longint	Całk. długa	4	-214783648	214783647
Shortint	Całk. krótka	1	-128	127
Word	Słowo	2	0	65535
Byte	Bajt	1	0	255
Real	Rzeczywista	4	-3.402·1038	3.402 ·1038
Double	Rzecz.o podw. prec.	8	-1.798·10308	1.798 ·10308
Extended	Rozszerzona	10	-1.1·104932	1.1 ·104932
Complex	Zespolona	8	-9.2·1018	9.2 ·1018

Typy zmiennych nieliczbowych

Boolean	Logiczna	1	True; False	
* String	Łańcuch	Zmienna	0÷255zn.ASCII	
* Char	Znak	1	Znak ASCII	

4. PODSTAWOWE POLECENIA JĘZYKA PASCAL

write – pisz (wyświetl na ekranie)

składnia polecenia **write**

a) write ('dowolny tekst') – jeżeli polecenia tego użyjemy w apostrofach to zostanie wyświetlony na ekranie dowolny tekst zawarty między apostrofami. Cursor zatrzyma się zaraz za wyświetlonym tekstem.

b) writeln ('dowolny tekst') - jeżeli polecenia tego użyjemy w apostrofach to zostanie wyświetlony na ekranie dowolny tekst zawarty między apostrofami. Cursor przejdzie do następnej linii

c) write (a) – jeżeli użyjemy tego polecenia tylko w nawiasach to zostanie wyświetlona zawartość zmiennej a (oczywiście o ile zadaliśmy sobie trudu aby do tej zmiennej wprowadzić jakieś dane). Cursor zatrzyma się zaraz za wyświetlonym tekstem.

d) writeln (a) - jeżeli użyjemy tego polecenia tylko w nawiasach to zostanie wyświetlona zawartość zmiennej a. Cursor przejdzie do następnej linii.

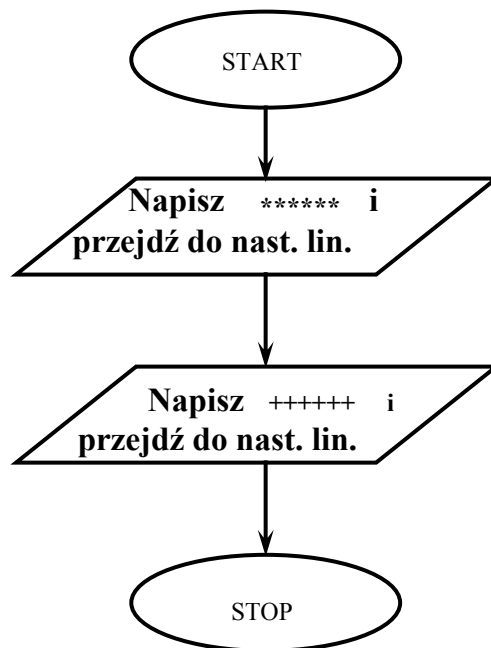
e) instrukcja przypisania := pełni rolę taką jak w matematyce „równa się”. np.

a:=2 {przypisz (podstaw za zmienną a) wartość 2}

- a:= b+c** {przypisz zmiennej a sumę zawartości zmiennych b i c}
- f) **read (a)** – jeżeli użyjemy tego polecenia program przeczyta daną wpisaną z klawiatury i podstawí za zmienną a.
- g) **readln(a)** - jeżeli użyjemy tego polecenia program przeczyta daną wpisaną z klawiatury i podstawí za zmienną a. Cursor przejdzie do następnej linii

```
PROGRAMvznaki1;           {nagł.} {w nazwie nie mogą występować spacje}.
User crt;                 {część deklarac.} {otwarcie pakietu}.
begin                     {rozpoczęcie programu}.
  clrscr                  {instrukcja czyszcząca ekran z pakietu crt}.
  writeln('*****');
  writeln('+++++');       {ciąg instrukcji}.
end.                       {zakończenie programu}.
```

Algorytm zadania znaki1.



Przykład programu

Program mnożący 2 liczby rzeczywiste wprowadzane przez użytkownika i wyświetlający wynik na ekranie.

```
Program ▽ mnoz; {nazwa programu}
Uses crt; {użycie pakietu w którym znajduje się polecenie do czyszczenia ekranu}
var a , b, c: real; {deklaracja 3 zmiennych typu rzeczywistego}
  Begin {początek programu}
    clrscr;
    write ('podaj liczbę a = '); {wyświetlenie tekst w apostrofach}
    readln ( a ); {wstawia przeczytaną z klawiatury liczbę za zmienna a}
    write (' podaj liczbę b = '); {wyświetlenie tekst w apostrofach}
    readln ( b ); {wstawia przeczytaną z klawiatury liczbę za zmienna b}
    c := a*b ; {wstawia za zmienną c iloczyn zawartości komórek a i
    writeln ('iloczyn a*b wynosi ', c : 5:2);
    readln;
end.
```

5 cyfr

2 znaki po kropce.

5. INSTRUKCJA WARUNKOWA, INSTRUKCJA WYBORU

W TP istnieją dwie instrukcje warunkowe:

1. IF ... THEN (Jeżeli To)
2. Instrukcja wyboru CASE

Do konstrukcji wyrażenia logicznego stosuje się w TP typ booleon, który posiada dwie stałe TRUE, FALSE. Na zmiennych tych można stosować operacje NOT, AND, OR.

- a) NOT podaje wartość TRUE, jeżeli wyrażenie logiczne przyjmuje wartość FALSE.
- b) AND podaje wartość TRUE, jeżeli obydwa wyrażenia są prawdziwe.
- c) OR podaje wartość TRUE, jeżeli co najmniej jedno wyrażenie jest TRUE.
- d) Kolejność operacji NOT, AND, OR:

Operatory relacji	Znaczenie
=	Równe
<>	Różne
<	Mniejsze
>	Większe
<=	Nie większe
>=	Nie mniejsze

Instrukcja warunkowa ma postać:

IF wyrażenie logiczne THEN instrukcja 1 ELSE instrukcja2

Instrukcje może być pojedynczą instrukcją lub ciągiem instrukcji.

Np.

```
IF a>0 then
    writeln ('a jest liczbą dodatnią')
else
    writeln ('writeln jest liczbą ujemną');
```

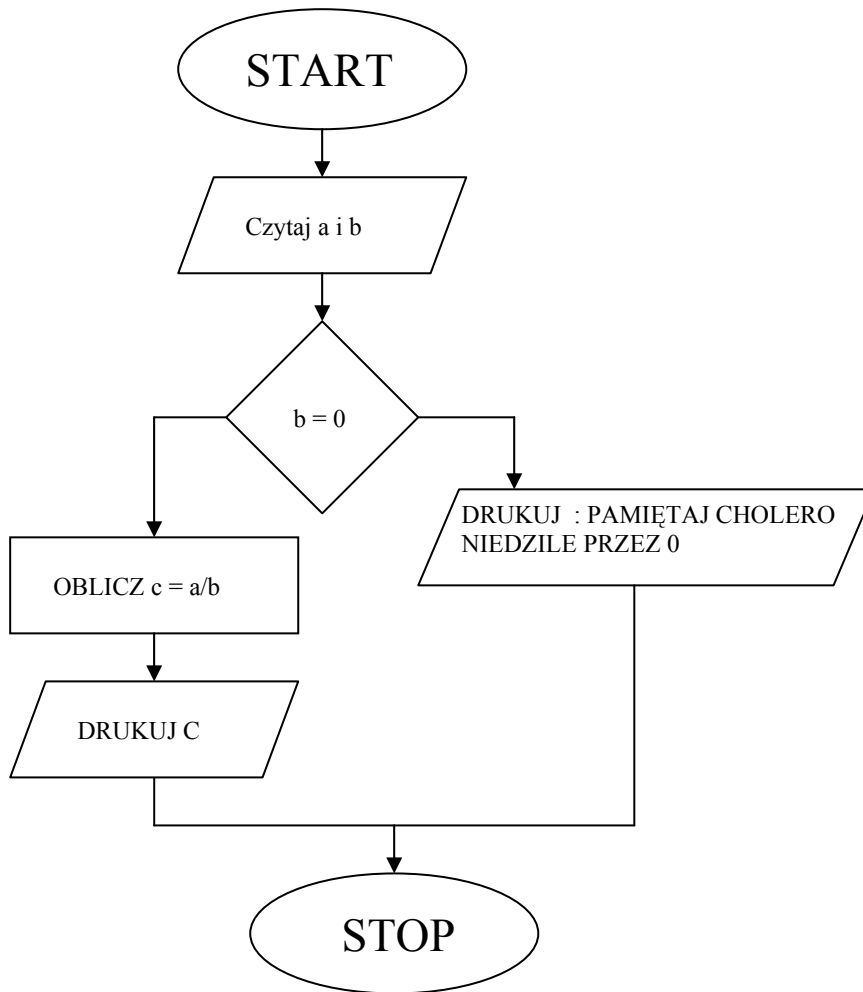
lub np.:

```
IF a>0 then
    begin
        .
        . ciąg instrukcji
        .
    end      {nie ma średnika}
ELSE
    begin
        .
        .
        .
    end;
```

Przykład

1) Rozwiązanie dzielenia dwóch liczb. $c = \frac{a}{b}$

Algorytm programu:



Rozwiązanie :
program zad2;

```
uses crt;  
var a, b, c : real;
```

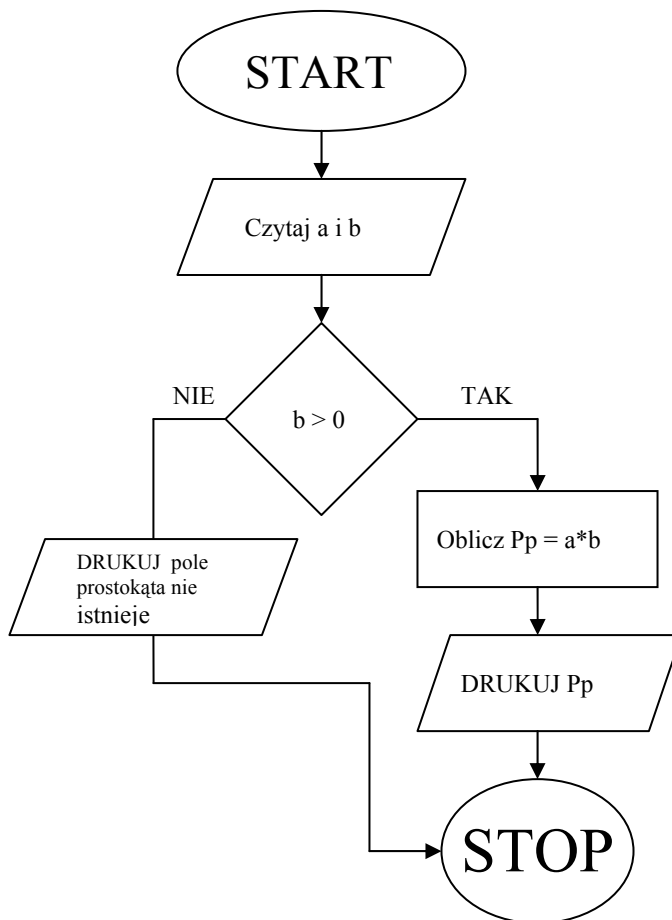
```
begin  
  clrscr;  
  writeln ('program liczący iloraz dwóch liczb a i b');  
  writeln ('podaj a =');  
  readln (a);  
  writeln ('podaj b =');  
  readln (b);
```

```
  IF b = 0 then  
    writeln ('Pamiętaj cholero nie dziel przez zero')  
  else
```

```
begin
  c := a/b;
  writeln ('iloraz liczby a i b wynosi', c : 5 : 2);
end;
  readln
end.
```

Zadanie : Oblicz pole prostokąta o bokach a i b. W przypadku gdy a lub b jest mniejsza od zera program ma sygnalizować to odpowiednim komunikatem.

a) Algorytm programu:



b) rozwiązanie zadania

```
program pole;
  uses crt;
  var a, b, Pp : real;

  begin
    clrscr;
    writeln ('Program liczący pole prostokąta');
    writeln;
```

```
write ('Podaj wartość boku a = ');
readln (a);
writeln;
write ('Podaj wartość boku b =');
readln (b);
  IF (a>0) and (b>0) then
    begin
      Pp := a*b;
      writeln ('Pole prostokąta wynosi Pp =', Pp : 5 : 2);
    else
      writeln ('Taki prostokąt nie istnieje');
    end.
```

INSTRUKCJA WYBORU CASE.

Stosujemy gdy jest konieczność zastosowania większej ilości przypadków dla instrukcji IF ... THEN.

INSTRUKCJA MA POSTAĆ:

	lub	
CASE <u>wyrażenia</u> OF		CASE <u>wyrażenie</u> OF
wybór 1: instrukcja;		wybór 1;
wybór 2: instrukcja;		wybór 2;
wybór 3: instrukcja;		wybór 3;
END;		else <u>instrukcje</u> ;
		END;

Przykład

Program liczący pole w zależności od wyboru:

1. Prostokąta
2. Koła
3. Trójkąta

```
program pole_figury;
uses crt;
var figura : integer;
    a, b, P : real;

begin
  clrscr;
```

```
writeln ('wyznacz pole figury');
writeln ('1 – prostokąt, 2 – koło, 3 – trójkąt');
writeln ('Podaj typ figury geometrycznej :');
readln (figura);
CASE figura OF
a: Begin
  writeln ('pole prostokąta');
  write ('podaj bok a = ');
  readln (a)
  write ('Podaj bok b =');
  readln (b);
  P := a*b;
  writeln ('dla boków prostokąta a=', a : 5 : 2' oraz b=', b : 5 : 2);
  writeln ('pole prostokąta P=', P : 6 : 2);
end;

2: Begin
  writeln ('pole kola');
  write ('podaj promien a = ');
  readln (a)
  P := Pi*a*a;
  writeln ('dla promienia a=', a : 5 : 2);
  writeln ('pole kola P=', P : 6 : 2);
end;

3: Begin
  writeln ('pole trojkata');
  write ('podaj wysokosc a = ');
  readln (a)
  write ('Podaj podstawie b =');
  readln (b);
  P := 0,5*a*b;
  writeln ('dla trójkąta a=', a : 5 : 2' oraz b=', b : 5 : 2);
  writeln ('pole trojkataP=', P : 6 : 2);
end
  else
    writeln('podałeś zły znak');
end;
readln
end.
```

Instrukcja IFTHEN	Instrukcja wyboru CASE
IF <u>warunek</u> THEN Begin end ELSE Begin end;	CASE <u>wyrażenia</u> OF Wybór1: Begin ... end; Wybór2: Begin ... end; wybór3: Begin ... end; ELSE <u>instrukcja</u> end;

zadanie:

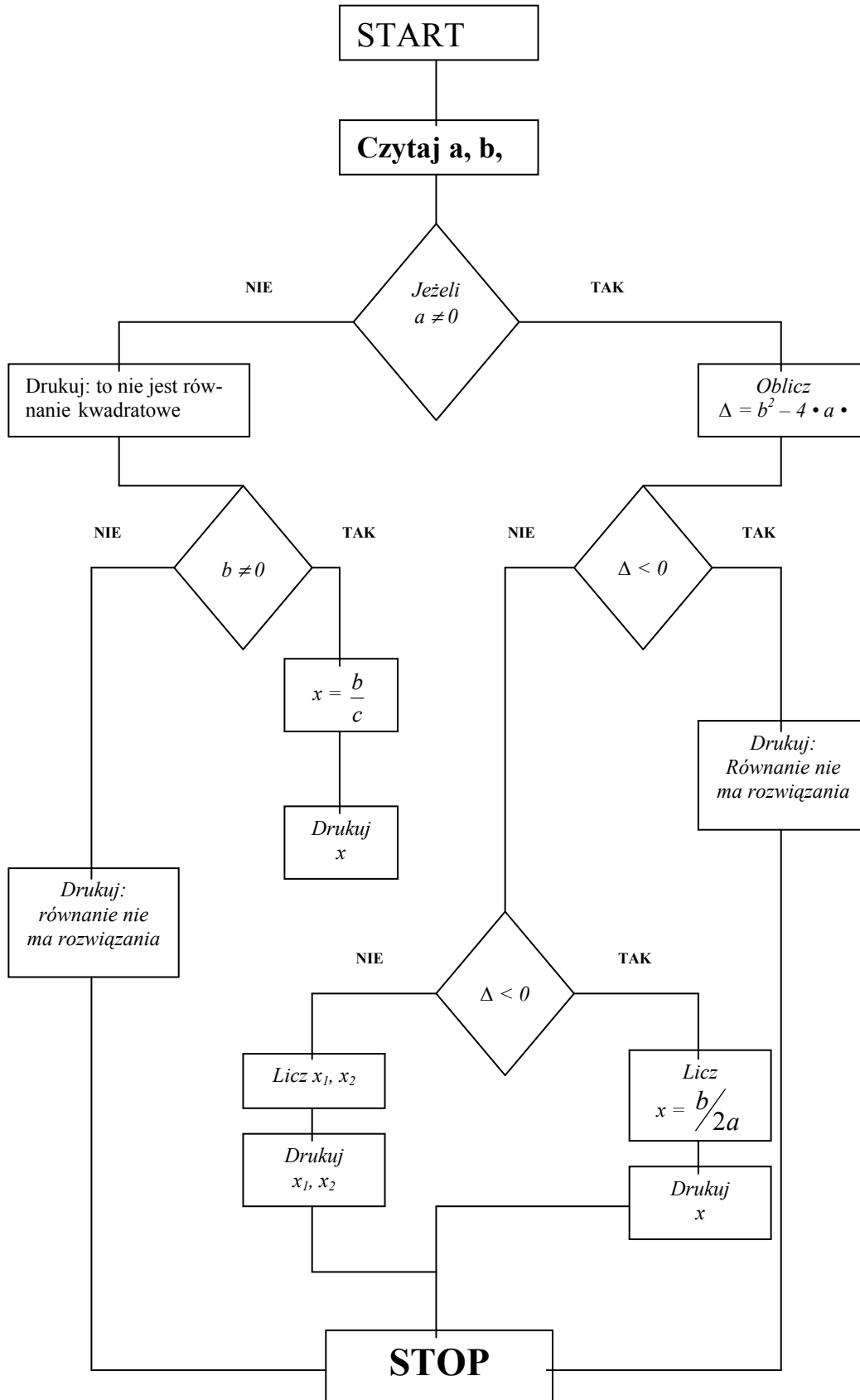
Obliczanie równania kwadratowego.

$Ax^2+bx+c=0$, (jeżeli a różne od 0 to obliczamy $\Delta=b^2-4ac$ i jeżeli $\Delta>0$ to równanie ma 2 rozwiązania, $\Delta=0$ ma jedno rozwiązanie)

Algorytm w postaci znakowej:

1. Odnaleźć wartości liczb a,b,c . Jeżeli $a=0$ to napisz komunikat, że równanie nie jest równaniem kwadratowym. Sprawdź czy b różne od 0. Jeżeli jest to $x=-b/c$, jeżeli $b=0$ to sprzeczne.
2. Oblicz wyróżnik trójmianu i sprawdź jego znak $\Delta=b^2-4ac$
3. Jeżeli $\Delta<0$ to napisz komunikat, że równanie nie ma rozwiązania
4. Jeżeli $\Delta=0$ to napisz komunikat, że równanie ma 1 rozwiązanie
5. jeżeli $\Delta>0$ to napisz komunikat, że równanie ma 2 rozwiązania
6. Zakończ działanie programu

a) Algorytm w postaci sieci działań:



b) Rozwiązanie

Program rownanie_kwadratowe

Uses crt;

Var a,b,c, x₁,x₂,del:real

Begin

Clrscr;

Write('podaj a=');

Readln(a);

Write('podaj b=');

Readln(b);

Write('podaj c=');

Readln(c);

IF a<>0 Then

Begin

Del:=b*b-4*a*c;

If del<0 then

Writeln('równanie nie ma rozwiązania');

Else

Begin

If del=0 Then

Begin

x₁:= -b/2*a;

writeln('x1=',x1:5:2);

end

Begin

x₁:=(-b-sqrt(del))/2*a;

x₂:=(-b+sqrt(del))/2*a;

end;

end;

end

else

begin

writeln('to nie jest równanie kwadratowe');

If b<>0 then

Begin

X1:=-b/c

Writeln('x1=',x1:5:2);

End

Else

Begin

Writeln(',nie ma równania');

End

End.

6. PĘTLE

W TP mówimy o pętlach lub powtórzeniach wtedy gdy określony ciąg instrukcji musi być wielokrotnie powtórzony w celu wykonania zadania. W TP są trzy instrukcje powtórzeń.

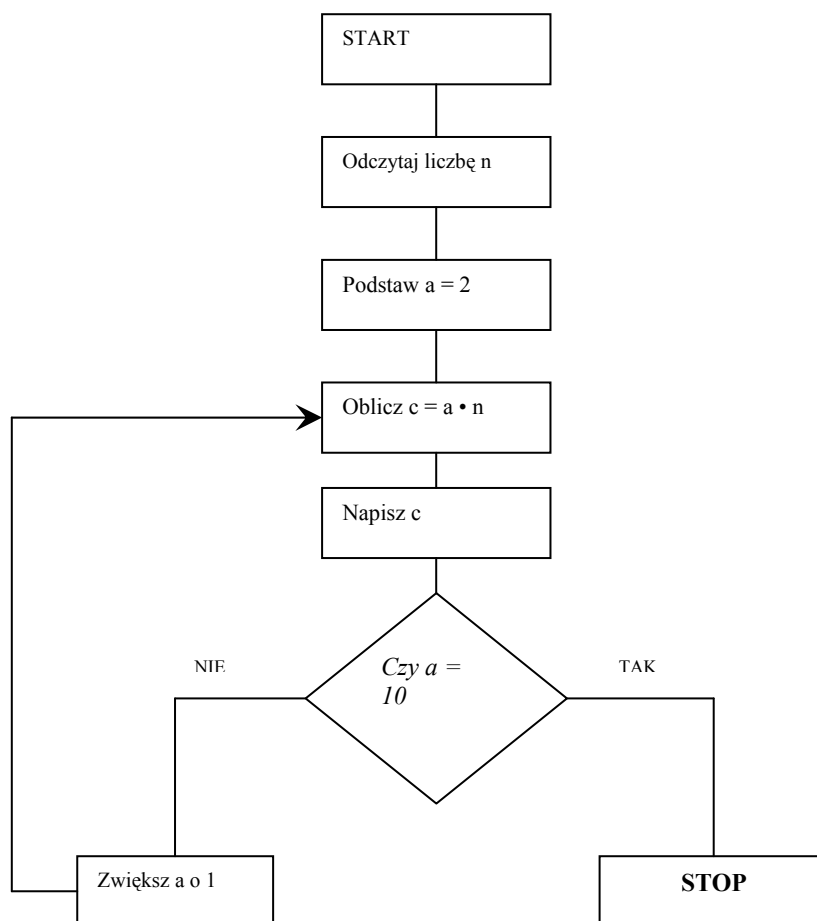
Zadanie

Program ma czytać liczbę n i pomnożyć ją przez liczbę a zwiększając ją o 1 dopóki liczba a nie osiągnie 10. Wartość początkowa $a=2$. Po każdym mnożeniu ma być drukowany wynik.

Algorytm znakowy

1. Odczytaj liczbę n
2. nadaj zmiennej a wartość 2 (musi być nadana wartość początkowa)
3. oblicz iloczyn $c=a \cdot n$ i przedstaw wynik
4. zwiększ zmienną a o 1. Jeżeli a jest równe 10 zakończ działanie w przeciwnym razie przejdź do kroku 3

Algorytm sieciowy



Konstrukcje pętli while

(dopóki) (rób)
while warunek **do**
 begin
 {ciąg instrukcji}
 end

```
Program petla;  
Uses crt;  
Var c, a, n : integer;  
    Begin  
        Clrscr;  
        Write('podaj liczbę całkowitą');  
        Readln(n);  
        a:=2  
while a<=10 do  
    begin  
        c := a*n;  
        writeln('wynika dla a=',a,' jest równy c=',c);  
        a := a+1;  
    end;  
readln;  
end.
```

Konstrukcja pętli for

for warunek początkowy **to** warunek końcowy **do**
 Begin
 {ciąg instrukcji}
 end;

```
Program cos;  
Uses crt;  
Var a, b, c : real  
    Begin  
        Read(a);  
        Read(b);  
        For i:=1 to b do  
            Begin  
                C := a+1;  
                Writeln('wynik c=',c);
```


Modyfikacje pętli

```
FOR wart.początkowa   DOWN TO   wart.końcowa DO
  Begin
  .....
  .....
  .....
end;
```

W instrukcji FORTO licznik zwiększa się o 1
W instrukcji FORDOWN TO licznik zmniejsza się o 1

Jako zmienne w liczniku mogą występować liczby całkowite lub litery
'a'.....'z' gdyż znaki te mają swoją kolejność którą TP rozróżnia, w liczniku nie mogą występować liczby rzeczywiste.

INSTRUKCJE WARUNKOWE ZAGNIEŻDŻONE

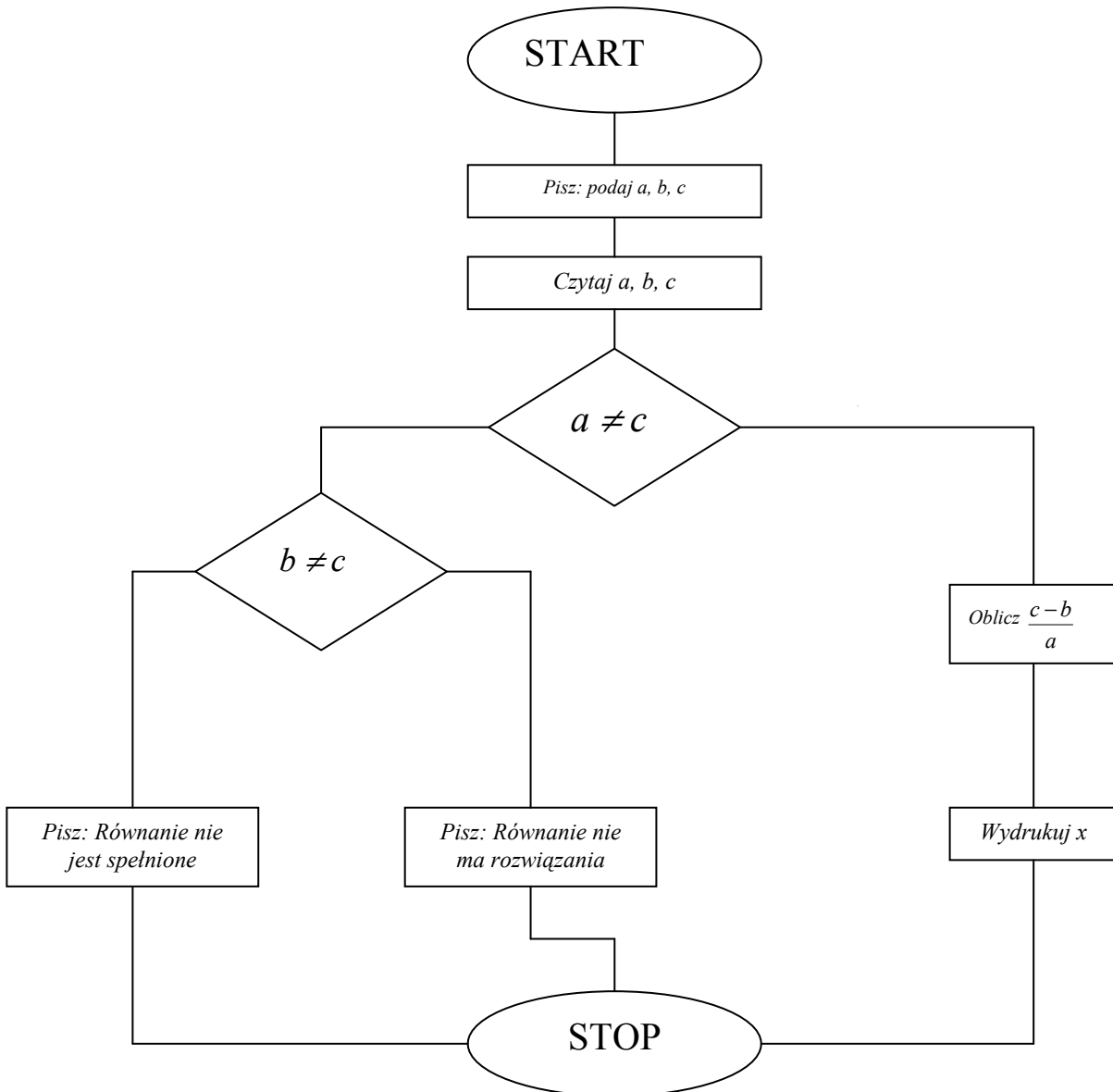
Wśród instrukcji wykonywalnych warunkowo mogą być zawarte następne instrukcje warunkowe. Mogą być umieszczone bezpośrednio lub wewnątrz bloków.

np.

```
IF a = 0 THEN IF b = c THEN
  writeln ('sprzeczne')
ELSE
  writeln ('prawdziwe');
```

np.

```
ax + b = c
program dwa;
var a, b, c : real;
begin
  writeln ('podaj a b c oddzielając je spacjami')
  readln (a,b,c);
  writeln('równanie',a : b: 2, 'x+',b : 6 : 2, '= ', c : 6 : 2);
IF a <> 0 THEN
  writeln ('ma rozwiązanie x =',(c-b) / a : 6 : 2);
ELSE
IF b <> c THEN
  writeln ('nie ma rozwiązania')
ELSE
  writeln ('jest zawsze spełnione');
END.
```



ZAGNIEŹDŻONA PĘTLA FOR

Pojedyncza pętla umożliwia nadanie kolejnych wartości jednej zmiennej to pętla umieszczona wewnątrz pętli umożliwia systematyczne zmienianie dwóch zmiennych.

np. tabliczka mnożenia

program ▼ tabliczka;

var i, j, il : integer;

begin

for I := 2 to 5 do

for j:=2 to 7 do

begin

il := i*j;

writeln(i , 'x' , j , '=' ,il);

end

```
readln  
end.
```

```
program ▽ petla ;  
uses crt;  
Var c, a, n: integer;  
Begin  
Clrscr;  
Write ('Podaj liczbę całkowitą ');  
readln (n);  
a:=2 ;  
WHILE a<=10 DO  
BEGIN  
c:=a * n ;  
writeln (' wynik dla a = ', a, ' jest równy c=', c);  
a := a+1 ;  
END ;  
readln  
end.
```

7. STANDARDOWE FUNKCJE MATEMATYCZNE

pierwiastek kwadratowy	SQRT(zm)	np.	SQRT(a+b)
kwadrat liczby	SQR(zm)		SQR(a+b)
wartość bezwzględna	ABS(zm)		ABS(a+b)
funkcja trygonometryczna	SIN(zm),COS(zm)		
logarytm naturalny	LN(zm)		
funkcja wykładnicza	EXP(zm)		
	ARCTAN(zm)		wartością jest kąt w radianach
działania na zmiennych całkowitych			
dzielenie całkowite	DIV	np. 7 DIV 2	wynik 3
Reszta z dzielenia całkowitego	MOD	np. 7 MOD 2	wynik 1

np.

Należy pamiętać o nawiasach i stosować je w miejscach nawet gdy jest ich za dużo.

np.

$$x = \frac{-b - \sqrt{\Delta}}{2a}$$

x:=(b-SQRT(delta))/(2*a)

np.

$$x^y \Leftrightarrow e^{y \ln(x)}$$

$$\text{EXP}(y * \text{LN}(x)) = x^y$$

8. ZMIENNE LOSOWE

W TP istnieje funkcja RANDOM generująca liczby losowe. Za każdym użyciem ma ona inną wartość. Wartością funkcji RANDOM bez parametru jest liczba rzeczywista nieujemna mniejsza od 1. Funkcja RANDOM z parametrem o wartości całkowitej (typu Word) ma wartość liczba przedziału od 0 do wartości o 1 mniejszej od parametru.

np.

RANDOM (100) daje liczby od 0 do 99. Należy wcześniej zainicjować parametr.

Przykład programu

```
program ▼ liczby_losowe;
```

```
uses crt ;
```

```
begin
```

```
clrscr;
```

```
RANDOMIZE {uruchomienie generatora}
```

```
writeln (Random : 9 : 4);
```

```
writeln (Random : 9 : 4);
```

```
writeln (Random (100) : 4);
```

```
writeln (Random (100) : 4);
```

```
readln
```

```
end.
```

Przykład zadania (Test z tabliczki mnożenia) Generuje losowo dowolne czynniki (od 2 : 9)

Użytkownik decyduje czy chce kontynuować test czy go zakończyć. Liczona jest ilość poprawnych odpowiedzi.

ALGORYTM LICZBOWY

1. Nadaj wartości zerowe zmiennym LICZ i DOBRE ODP.
2. Nadaj wartości losowe zmiennym CZ 1 i CZ 2. Pomnóż je przez siebie i wartość wyniku nadaj zmiennej IL.
3. Odczytaj liczbę . Jeżeli jest równa wartości zmiennej IL to przejdź do ludu 5?
4. Napisz tekst stwierdzający że odpowiedź jest błędna i podając wartość zmiennej IL jako odpowiedź prawidłową. Przejdź do kralm 6.
5. Napisz tekst stwierdzający, że odpowiedź jest poprawna. Zwiększ wartość DOBRE odpowiedzi o jeden.
6. Zwiększ wartość zmiennej LICZ o jeden. Odczytaj decyzję użytkownika. Jeżeli chce kontynuować test przejdź do ludu 2.

7. Podaj wynik końcowy, liczbę pytań LICZ i prawidłowych odpowiedzi DOBRE ODP.
8. Zakończ.

Program tabliczka;

```
uses crt;
```

```
var lin, dobreodp, cz1, cz2, il : integer;  
    jeszczejdn : string;
```

```
begin
```

```
  clrscr;
```

```
  randomize;
```

```
  Licz := 0;
```

```
  Dobre odp. := 0;
```

```
  Repeat {wybór rodzaju pętli}
```

```
    Cz1 := 2+random(9); {wartość min.2, wartość max.2+8=0}
```

```
    Cz2 := 2+random(9);
```

```
    Write ('Podaj wynik mnożenia; cz1, 'przez' ,cz2, '=');
```

```
    Readln (il);
```

```
    If cz1*cz2 := il Then
```

```
      Begin
```

```
        Dobreodp := Dobre odp.+1;
```

```
        Writeln('Odpowiedz poprawnie');
```

```
      End
```

```
    Else
```

```
      Writeln('Odpowiedź błędna. Iloczyn jest równy' ,cz1*cz2);
```

```
      Licz :=Licz+1
```

```
    Writeln ('czy jeszcze jedno pytanie? (t/n)');
```

```
    Readln (jeszcze jdn);
```

```
    Until jeszcze jdn <> 't' ;
```

```
    Write('Na' , Licz, ' zadanych pytań');
```

```
    Writeln ('udzieliłeś' , dobre odp. , 'prawidłowych odp.');
```

```
    Readln
```

```
END
```

9. NADAWANIE ZMIENNYCH PROSTYCH

Zadanie

Obliczanie wartości funkcji $f(x)=-2x^2+8x$ w przedziale 2 do 3 z krokiem 0,01.

Program ▽ funkcji_kwadratowej;

```
Uses crt;
```

```
Var i : integer; {i – licznik}.
```

```

y,x,krok : real;
Begin
  x :=2
  Krok :=0,01;
  For i :=0 to 100 do
  Begin
    x :=x+krok;
    y :=-2*x*x+8x*x;
    writeln (x : 15 : 10);
    writeln (y : 20 : 10);
  end;
  readln
end.

```

10. RELACJE LOGICZNE

Relacja		Przykład	
Nazwa	Symbol	Relacja zachodzi	Relacja nie zachodzi
Równe	=	3 = 3 'Ala' = 'Ala'	5 = 5.01 'A' = 'a'
Mniejsze	<	3 < 5 'c' < 'F'	5 < 3 'g' < 'b'
Większe	>	5.01 > 5.001	- 5 > - 3
Nie mniejsze	>=	3 ≥ 3 'Ewa' ≥ 'Ela'	3 ≥ 3.001 'A' ≥ 'a'
Nie większe	<=	'Jan' ≤ 'Jan' 'Tak' ≤ 'tak'	'Janek' ≤ 'Jan' 'nie' ≤ 'Tak'
Nie równe	<>	'Tak' ≠ 'tak'	3 ≠ 3'

11. DZIAŁANIE NA ZMIENNYCH LICZBOWYCH

Działanie	Symbol	Przykład	Wynik	Typ wyniku w przykł.
Dodawanie	+	3 + 5,5	8,5	rzeczywisty
		3 + 5	8	całkowity
Odejmowanie	-	17 - 12	5	całkowity
Mnożenie	*	3 * 17	51	całkowity
		30 * 1,7	51,0	rzeczywisty
Dzielenie rzeczywiste	/	9/4	2,25	rzeczywisty

Dzielenie całkowite	DIV	11 DIV 4	2	całkowity
Reszta z dzielenia całkowitego	MOD	11 MOD 4	3	całkowity

12. ODCZYTYWANIE DANYCH Z KLAWIATURY

Proste instrukcje służące do odczytywania znaku READ, READWV do odczytywania danych z klawiatury READKEY, KEYPRESSED. Obie funkcje są dostępne w pakiecie crt

- a) Wartością funkcji READKEY jest znak (char) odpowiadający naciśniętemu klawiszowi

np.:

```
a: = readkey
```

spowoduje przypisanie **a** dowolnego znaku z klawiatury (zmienna **a** powinna być typu znakowego)

- b) Funkcja KEYPRESSED informuje czy jakikolwiek klawisz został naciśnięty. Jeżeli tak to jej wartością jest PRAWDA.

np.: repcat

```
until KeyPressed
```

Naciśnięcie: **Alt, Ctrl, Shift, CapsLock, PrintScreen, Pause** – nie jest traktowane jako naciśnięcie klawisza.

13. TYPY PORZĄDKOWE

Do tych typów należą typy znaczkowe i całkowite.

W trybie porządkowym elementy są uporządkowane i ponumerowane.

- a) Succ (zm) – następnik zmiennej zm (następnik ostatniego jest nieokreślony).

np. succ (4) = 5

succ ('a') = 'b' {wartością succ ('a') jest 'b'}

- b) Pred (zm) – poprzednik zmiennej zm (poprzednik pierwszego jest nieokreślony)

Pred (- 5) = - 6

Pred ('Y') = 'x'

c) Ord (zm) – numer typu porządkowego

Ord (- 8) = - 8

Ord ('A') = 65'

14. DEFINIOWANIE TYPÓW

Typy standardowe: char, integer, real;

możemy definiować własne typy, które możemy używać w programie.

Definicja typów musi być podana przed deklaracją zmiennych.

np. definicje typu w postaci łańcucha o określonej długości

```
type s8 = string[8]          TYPE_nazwa = okr.typu
```

a następnie deklarować w programach:

np.

```
program
  type lancuch8 = string[8];
  var aa, bb, cc : lancuch8;
  begin
    aa:='Grzegorz'
    bb:='Maciej'
    cc:='Nowak'
    writeln (aa, ' ', bb, ' ', cc);
    readln
  end.
```

Nie jest konieczna definicja typu w tym przypadku wystarczyłaby aa, bb, cc:

```
string[8].
```

15. ZMIANA TYPÓW

Zamianie liczby rzeczywistej na całkowitą przez zaokrąglenie w dół służy funkcja

TRUNC (truncate – obetnij)

TRUNC(7.6) = 7, TRUNC(7.1) = 7

Zaokrągleniu liczby w górę lub w dół w zależności od błędu służy funkcja

ROUND

ROUND(7.6) = 8, ROUND(7.1) = 7

Funkcja przyporządkowująca kodowi znaku 0÷255 sam znak ASCII jest funkcją CHR.

Funkcją odwrotną jest ORD

np.:

CHR(67) = C

CHR(65) = A

program drukowanie; {drukuje zestaw znaków}

uses crt;

var kod: integer;

begin

cliser;

for kod:= 32 to 255 do n

write (Chr(kod), ' ');

readln

end.

16. OPERACJE NA ZMIENNYCH ŁAŃCUCHOWYCH

Przedmiotem działań mogą być także teksty i stałe tekstowe i zmienne.

Podstawowym działaniem jest łączenie tekstów (dodawanie). Dodawanie zmiennych łańcuchowych powoduje połączenie obu tekstów w jednym bez zaznaczenia przerwy

np.:

'Ala' + 'ma' jest równe 'Alama' a nie 'Ala_ma'

Funkcje wykonujące operacje na zmiennych łańcuchowych:

a) **LENGTH** podaje długość łańcucha będącego jej argumentem

np. LENGTH ('Beata') jest 5

b) **COPY** ma trzy argumenty

- 1) zmienna łańcuchowa
- 2) liczby całkowite od której
- 3) liczby całkowite do której

wartością COPY ('Turbo_Pascal', 4, 6); jest łańcuch 'bo_Pas'

c) **POS** ma dwa argumenty będące łańcuchami. Jej wartością jest pozycja drugiego łańcucha począwszy od której jest on zgodny z pierwszym łańcuchem.

np.: wartość funkcji POS('Pascale, Turbo_Pascal') jest 7

przykład:

```
program                                                    Im1
– pierwsze słowo
var Im1, Im2, SumaIm, Fragm : string                      Im2 – drugie słowo
                                                         SumaIm – suma
                                                         Fragm – fragment
dwóch słów
Begin                                                    Fragm – fragment
łańcucha sumy
  write('Napisz pierwsze słowo');
  Readln(Im1);
  write('Napisz drugie słowo');
  Readln(Im2);
  SumaIm:= Im1+Im2 {suma dwóch słów}
  writeln('Długość słow=', length(Im1), ' ', length(Im2)); {drukuje długość po-
szczególnych słów}
  writeln('suma słów=', SumaIm); {drukuje sumę obu słów}
  Fragm:= copy (sumaIm,6,13) ); {wyznacza fragment łańcucha su-
my}
  writeln('Fragment sumy=', Fragm); {drukuje słowo które jest fragment}
  writeln('pozycja fragmentu w sumie=', pos (Fragm, SumaIm));
  readln
end.

Im1 'Bogdan'
Im2 'mówi'
Im3 'Bankowy'
Suma Im BogdanMówiBankowy
```

length(Im1) = 6
length(Im2) = 4
length(Im3) = 7

Fragm:= copy(SumaIm,6,13 ≡ (BogdanMówiBankowy) ≡ MówiBankowy

POS(Fragm, SumaIm) ≡ (MówiBankowy, BogdanMówiBankowy) ≡ 6

- d) funkcja **INSERT** umożliwia wstawianie jednego łańcucha wewnątrz drugiego
- e) **DELETE** kasująca fragment łańcucha.

Przypomnienie

Program znaki1;

```
Begin
    Writeln(' * * * * * ');
    Writeln(' - - - - - ');
    Writeln(' + + + + + ');
End.
```

Ten sam program z użyciem procedur.

Program znaki2;

```
Uses crt;
Procedure plusy;
Begin
    Writeln(' + + + + + ');
End;
Procedure minusy;
Begin
    Writeln(' - - - - - ');
End;
Procedure gwiazdki;
Begin
    Writeln(' * * * * * ');
End.
Begin {program}
    Clrscr;
    Gotoxy(32,7);
    Gwiazdki;
```

```
Gotoxy(32,8);  
Plusy;  
Gotoxy(32,9);  
Minusy;  
End.
```

17. PROCEDURY

Procedury są podprogramami realizującymi określone zadania. Przy konstrukcji większych programów stosowanie procedur jest konieczne. Struktura procedury jest podobna do struktury programu.

```
Procedure nazwa (lista parametrów formalnych); {może być, ale nie musi}  
  Var  
  .  
  .      {deklaracja zmiennych lokalnych}  
  .  
  Begin  
      {treść procedury}  
  
  End;
```

- Wywołanie procedury polega na podaniu nazwy procedury wraz z listą parametrów aktualnych (o ile takie istnieją)
- Treść procedury musi być w programie umieszczona po deklaracji zmiennych globalnych a przed rozpoczęciem programu głównego
- Zmienne lokalne są to zmienne używane tylko wewnątrz procedury
- Zmienne globalne (zmienne całego programu) mogą funkcjonować we wszystkich procedurach, programie głównym
- Parametry formalne – przy deklaracji procedury
Parametry aktualne – przy wywołaniu procedury

Użycie procedury:

Zadanie polegające na obliczaniu pola kwadratu i koła:

- a- zmienna, bok kwadratu i promień koła
- c1- pole kwadratu
- c2- pole koła

```
Program obliczanie;  
  Uses crt;  
  Var a :integer;      {zmienna globalna}  
Procedure kolo;  
  Var c2: real; {zmienna lokalna}  
  Begin  
    C2:=Pi*a*a  
    Writeln('pole koła wynosi P=',c2:5:2);  
  End;  
  
Procedure kwadrat;  
  Var c1: real; {zmienna lokalna}  
  Begin  
    c1:= a * a;  
    Writeln('Pole kwadratu wynosi P=',c1:5:2);  
  End;  
Begin {program}  
Clrscr;  
Writeln('podaj liczbę będącą r koła i a kwadratu');  
write('a=');  
  Readln (a);  
  Kolo;      {wywołanie procedury kolo}  
  Kwadrat;   {wywołanie procedury kwadrat}  
End.        {programu}
```

Np. gdybyśmy użyli w programie głównym np. `writeln(c1)` to program nie zrozumiałby, o co chodzi, ponieważ nie rozumie, co to jest `c1`.

Zastosowanie procedur z parametrami formalnymi

```
Program obliczanie2;  
  Uses crt;  
  Var n, wynik: integer;  
  Procedure szescian (a: integer);  
    Begin      {brak zmiennych lokalnych}  
      Wynik:= a *a* a;  
      Write('Dla n=', a, ' wynik =', wynik);  
    End;  
  Begin {rozpoczęcie programu głównego}  
    Clrscr;  
    Writeln('obliczanie sześciangu liczby naturalnej n');  
    Write('podaj liczbę naturalną n=');  
    Readln (n);
```


Parametr `il` został poprzedzony słowem kluczowym `var`. Oznacza to, że jest on parametrem przekazywanym przez zmienną i służy do wyprowadzenia wyniku obliczeń do bloku zewnętrznego (programu głównego). Parametry aktualne muszą być nazwami zmiennych odpowiedniego typu.

18. FUNKCJE

Wynikiem funkcji jest wartość.

Funkcje są podprogramami podobnie jak procedury

Ogólna postać funkcji

Function nazwa (lista parametrów formalnych):typ wyniku;

```
Ver {deklaracja stałych,}
    {zmiennych, typów}
begin
    {treść funkcji}
end;
```

np.

a) W treści funkcji musi być umieszczone przypisanie `nazwa:=wynik;` {Przypisanie to nadaje nazwę zmienną lub stałą określającą przekazywaną wartość.

b) wywołanie funkcji polega na przypisaniu pewnej zmiennej nazwy funkcji wraz z list

parametrów aktualnych.

`zmienna := nazwa (lista par. formalnych)`

lub umieszczenie nazwy funkcji w określonym wyróżnieniu umożliwia to przekazanie wartości funkcji w miejsce wywołania tej funkcji

Procedury i funkcje są bardzo do siebie podobne. Jedne i drugie mogą się posługiwać parametrami formalnymi. Różnica :

- Funkcje mają zawsze określony rezultat o typie określonym przez typ wyniku funkcji

Przykład :

Napisać program zawierający funkcję wyznaczenia mniejszej liczby z dwóch podległych liczb. Funkcja wykorzystuje zmienne globalne . Funkcja będzie czytać mierne dopóki wczytane dwie zmienne nie będą sobie równe.

program funkmn;

uses crt;

var a, b, min:longint

```
zn:char;
function mniejsza:longint;
begin
  repeat
  clrscr;
  writeln('wyznaczenie liczby mniejszej');
  writeln('podaj dwie liczby naturalne')
  write('podaj liczbę a=');
  readln(a);
  write('podaj liczbę b=');
  readln(b);
  until a<>b
  IF a>b
  else min:=a;
  mniejsza :=min; {nadanie funkcji wartości}
end; {funkcji}
begin {programu}
  repeat
  clrscr;
  written
  c:=mniejsza; {przypisanie wyniku funkcji – zmiennej}
  writeln('z liczb',a,' i ',b,'liczba mniejsza to',mniej);
  writeln('jeszcze raz');
  zn:= readkey
end.
```

Zastosowano zmienne globalne zadeklarowane w bloku programu głównego a, b, min, c, zn.

Podobnie jak program czy też procedura funkcja może posiadać swoje własne struktury (zmienne, stałe, typy, procedury i funkcje). Deklaruje się je tak jak w programie.

Rekurencja :

Rekurencja jest to odwołanie się w funkcji do funkcji z parametrem mniejszym o jeden.

Przykład:

funkcję $n!$ można określić rekurencyjnym:

$$0! = 1$$

$$n! = n*(n-1)!$$

1) Program obliczy silnię :

program silniarek;

```
uses crt;
var a:byte
    zn:char;
function silnia(n:byte):longint;
begin
    IF n=0 THEN
        silnia:=1
    IF n>0 THEN
        silnia:=n*silnia(n-1);
end;
begin {program}
repeat
clrscr;
writeln('obliczenia wyznaczenia a!');
writeln('podaj liczbe z zakresu 1-10 a=');
readln(a);
write(a,'!=',silnia(a));
writeln('jeszcze raz');
zn:=readkey
until not (zn='t')
end.
```

zmienne przekazywane przez wartość

! Jeżeli jest więcej parametrów należy pamiętać aby kolejność parametrów formalnych odpowiadała kolejności parametrów lokalnych

19. TABLICE

Tablica jest to struktura danych zawierająca pewien uporządkowany zbiór elementów tego samego typu.

a) Tablica jednowymiarowa

2	3	5	7	2	0	7	-9	7	-6	4	5
i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10	i=11	i=12

Deklaracja tablicy

TYPE tab=ARRAY [1..10] OF integer

{zadeklarowany typ tablicowy o nazwie tab o max. indeksie 10 w której są liczby całkowite}

deklaracja typu następuje w części deklaracyjnej przed VAR

po deklaracji typu należy zadeklarować zmienna tego typu

np. **VAR T1 : tab;**

Wpisywanie danych do tablicy odbywa się za pomocą pętli
np.

```
FOR i:=1 to 10 do  
  begin  
    T1[i]:=4;  
  end;
```

efekt

4	4	4	4	4	4	4	4	4	4
i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10

lub

```
FOR i:=1 to 10 do  
  begin  
    write ('podaj liczbe = ');  
    readln (T1[i]);  
  end;
```

Wyświetlanie zawartości tablicy

```
FOR i:=1 to 10 do  
  begin  
    write (T1[i], ' ');  
  end;
```

operacje matematyczne na tablicach

Dodawanie

T3[i]:=T1[i]+t2[i];

Mnożenie

T3[i]:=T1[i]*t2[i];

b) tablice wielowymiarowe

Deklaracja tablicy dwuwymiarowej

```
TYPE tab2=ARRAY [1..10,1..5] OF integer;
```

{zadeklarowano typ o nazwie tab2 tablicowy o ilości wierszy 10 i ilości kolumn 5 zawierającej liczby całkowite}

```
FOR i:=1 to 10 do
  begin
    for j:=1 to n do
      begin
        write ('podaj liczbe = ');
        readln (T1[i,j]);
      end;
    end;
  end;
```

Wyświetlanie zawartości tablicy

```
FOR i:=1 to 10 do
  begin
    for j:=1 to n do
      begin
        write (T1[i,j]:4);
      end;
    end;
  end;
```

Tablica jest to struktura danych zawierająca pewien uporządkowany zbiór elementów tego samego typu. Tablice jednowymiarowe odpowiadają wektorom natomiast tablice dwuwymiarowe macierzom. elementy tablicy oznakowane są za pomocą indeksów. tablica stanowi więc pewien typ strukturalny, zwany typem tablicowym. Tablicę tworzymy za pomocą słowa kluczowego ARRAY zdefiniowanie typu tablicowego ma postać :

TYPE nazwa tabeli = ARRAY [indeks1,..,indeks n]of typ zmiennej;

Daną nazwę tabeli można wykorzystać w deklaracji VAR

np.

T1:nazwa tabeli

indeks oznacza liczbę elementów. Zawiera ona kres górny i dolny czyli zakres tablicy. Jeśli podany jest jeden indeks to jest to tablica jednowymiarowa

np.

```
TYPE Dane= ARRAY[1..10]OF INTEGER;
```

Zdefiniowano typ tablicy identyfikujący jednowymiarową zawierającą 10 elementów

1	2	3	4	5	6	7	8	9	10

wewnątrz są elementy typu całkowitego.

Dany identyfikator tablicy DANE wykorzystany został w dek. zmiennych
VAR Tab1:DANE {deklaruje zmienną TAB1 typu DANE}

- 2) Jeżeli tablica zawiera więcej indeksów np. 2 to jest to tablica dwuwymiarowa

TYPE dane2=ARRAY [1..5,1..7]OF Real;
zadeklarowany typ tablicowy zawierający 5 wierszy(1..5) i 7 kolumn(1..7) z elementami w postaci liczb rzeczywistych

Var Liczby:Dane2;
{zdeklarowano zmienną liczby typu Dane2}.

3)Identyfikator tablicy

Dla określenia elementu znajdującego się w tablicy dwuwymiarowej oznaczamy wiersze jako zmienne i (i=1...i) oraz kolumny jako zmienną j (j=1...j) to element w tablicy na przecięciu i-tego wiersza i j-tej kolumny ma oznaczenie nazwa [i,j] np..
Dane2 [2,3]

4)Nadawanie wartości elementom tablicy odbywa się przez przypisanie

Dane2[2,3]:=25
elementowi tablicy Dane2 znajdującego się na przecięciu 3 wiersza i 5 kolumny przypisano wartość 25 (często używa się do tego instancję FOR)
Przykład

Program realizujący wprowadzenie liczb całkowitych do tablicy jednowymiarowej o „n” elementach a następnie wydrukowanie jej zawartości. Przyjmując dla tablicy maksymalny rozmiar max=20 elementów, ale wprowadzanie i drukowanie elerych z tablicy zastosować procedury.

```
program tablica1;  
uses crt;  
const max=20; {deklaracje stałej}  
type Dane=ARRAY[1..max]of integer;  
var n:integer  
    T1:Dane;  
    procedure wyprowadzanie(var T:Dane);  
        var i:integer; {zm. licznikowa}.  
begin  
    writeln('wprowadzenie danych o tabeli'0;  
    write('podaj liczbę elementów (n<=20)n=');  
    readln(n)
```

```
far i:=1 to n DO
  begin
    write('element nr=',i,'T1[',i,']=');
    readln(T[i]);
  end;
end;
procedure odczyt(var T:Dane);
var i:integer;
begin
  writeln('odczyt elementów z tabeli')
  For i:=1 to n do
    begin
      write('element nr=',i,'T1[',i,']=');
      writeln(T[i]);
    end;
  end;
begin {program}
  clrsrc;
  wprowadzenie(T1)
  odczyt(T1)
end.
```

b) tablice dwuwymiarowe

Napisać program realizujący wprowadzenie liczb całkowitych do tablicy o wymiarach $m \times n$ (m - liczba wierszy, n - liczba kolumn) oraz wydrukowanie jej zawartości. Przyjąć $w_{max}=10$, $k_{max}=10$. Do wprowadzenia i wyprowadzenia wyniku zastosować procedury.

```
program tablica2
  uses crt;
  const wmax=10 {maksymalna liczba wierszy}
        kmax=10 {maksymalna liczba kolumn}
  type Dane=ARRAY[1..wmax,1..kmax]OF INTEGER
  var w,k:integer {aktualna liczba wierszy I kolumn}
  tab:Dane;
  procedure wprowadzenie(ver T:Dane)
  var i,j:integer {zmiany licznikowe}
  begin
    writeln('wprowadzenie danych do tablicy');
    write('podaj liczbę wierszy W<=10 w=');
    readln(W);
    write('podaj liczbę kolumn k<=10 k=');
```

```
readln(k);
writeln('podaj elementy tablicy w kolejności wiersz po wierszu');
  For i:=1 to w do
    For j:=1 to k do
      begin
        write ('w=',i,' k',j,'TAB[' ,i,' ',j,']=')
        writeln(T[i,j]);
      end;
    end;
begin
clrscr;
wprowadzenie (TAB)
odczyt(TAB)
end.
```

Zadanie:

Napisać program wyznaczający wszystkie k-elementowe kombinacje bez powtórzeń ze zbioru n-elementowego.

```
program kombinacje;
uses crt;
const max=100
type tablica=array[1..max]of integer;
var n,k:integer
tab:tablica;
zn:clear
procedure wyświetlanie(k:integer);
var i:integer;
begin
  For i:=1 to n-k+a do
    begin
      tab[a]:=I;
      IF a<k:=I
        kombinacja(a+1,i+1);
      else
        wyświetlanie(k);
    end;
  end;
begin
while n<=k do
  begin
    clrscr;
```

```
writeln('wyznaczenie kombinacji n po k');
write('podaj liczbę n=')
readln(n);
write('podaj liczbę k=');
readln(k);
end;
  IF n>=k then
    kombinacja(1,1);
end.
```

20. RECORDY

Typ rekordowy umożliwia łączenie danych różnych typów w jedną złożoną strukturę zwana rekordem. Poszczególne elementy rekordu nazywamy polami danego rekordu.

Rekordy należy deklarować w części deklaracyjnej programu.

Deklaracja typu rekordowego ma postać:

```
TYPE nazwa_rekordu=RECORD
                                lista-pola1:typ1;
                                lista-pola2:typ2;
```

```
END;
```

W części deklaracyjnej musimy zadeklarować zmienną np. dane, typu rekordowego:

```
VAR dane:nazwa_rekordu
```

Definiowanie typu rekordowego zaczyna się słowem kluczowym RECORD, po którym umieszczają się listy poszczególnych pól z opisującymi je typami.

Po słowie VAR zadeklarowano zmienną rekordową dane.

Przykład

```
TYPE pracownik=RECORD
    nazwisko:STRING[20]
    imie:STRING[20]
    nr_osoby:INTEGER
    adres:STRING[20]
    telefon:STRING[20]
    plec:CHAR
    dochody:REAL
    wiek:BYTE
END;
VAR osoba:pracownik    {deklaracja zmiennej osoba typu pracownik}
```

Każde pole rekordu może zostać wybrane do dalszego przetwarzania za pomocą wskazania przez tzw. deskryptor pola. Ma on postać:

nazwa_rekordu.nazwa.pola

np. do poprzedniego

osoba.imie

wskazuje imie rekordu osoba

Do wprowadzania danych do pól korzystamy z instrukcji READ(ln). Do czytania danych z rekordu stosujemy instrukcję WRITE(ln).

Przykład programu

Napisać program umożliwiający wprowadzenie danych osobowych dla 5 pracowników. Rekord ma zawierać następujące pola: nr.ew, nazwisko, imie, plec.

```
Program dane_osobowe;
  uses crt;
  TYPE dane_personalne=RECORD
      nrew:BYTE
      nazwisko:STRING[30]
      imie:STRING[20]
      plec:CHAR
  END;
  VAR dane:dane_personalne;
  licznik:BYTE;
  BEGIN
    for licznik:=1 to 5 do
      BEGIN
        writeln('Podaj nrew',licznik,'-ej osoby:');
        readln(dane.nrew);
        writeln('Podaj nazwisko',licznik,'-ej osoby');
        readln(dane.nazwisko);
        write('Podaj imie',licznik,'-ej osoby');
        readln(dane.imie);
        write('Podaj plec m/k',licznik,'-ej osoby');
        readln(dane.plec);
        writeln;
      END;
    readln;
```

END.

{zapisywanie do pliku}

Do wybrania odpowiedniego pola rekordu należy wskazać to pole za pomocą de-skryptora. W Turbo Pascalu istnieje instrukcja wiążąca WITH, umożliwiająca wskazanie całej grupy pól danego rekordu. Instrukcja WITH ma postać:

```
WITH nazwa_rekordu DO
Instrukcje
```

Instrukcja jest najczęściej instrukcją złożoną. Zawarte w niej nazwy pól odnoszą się do rekordu wyszczególnionego w instrukcji WITH.

Zamiast zapisać

```
write(dane.nrew,'');
write(dane.nazwisko,'');
```

można zapisać

```
WITH dane DO
BEGIN
write(nrew);
write(nazwisko);
END;
```

Identyfikatory nrew i nazwisko będą traktowane jako pola rekordu dane.

21. KOLORY WYDRUKU

Procedura TEXTCOLOR określa kolor napisu. Ma jeden parametr np. blink-mruganie, red-czerwony, yellow-żółty).

TextColor(4) lub TextColor(red)

TextColor(14+blink) lub TextColor(yellow+blink)

22. KOLOR TŁA

Procedura określająca kolor tła TEXTBACKGROUND również z jednym parametrem.

TextBackGround(green)

23. SYGNAŁY DŹWIĘKOWE

Procedura SOUND powoduje wydanie dźwięku o określonej wysokości (częstotliwość w hercach). Dźwięk trwa do czasu wykonania bezparametrowej procedury NOSOUND. Czas trwania dźwięku można też określić za pomocą procedury DELAY z parametrem określającym czas wykonywania tej procedury

Wyrażonym w milisekundach. Procedury te są dostępne w module CRT np.

Sound(1200); Delay(500); 1 dźwięk o $f=1200\text{Hz}$ $t=0,5\text{s}$

ASSIGN(nazwa pliku, nazwa plik dyskowy) np.:

ASSIGN(plik1,dane1.dan) - Skojarzenie zmiennej plikowej z plikiem dyskowym o rozsz.dan

Plik1- nazwa zmiennej plikowej w programie

Dane1.dan - nazwa rzeczywista pliku na dysku

Następnie należy otworzyć plik za pomocą jednej z procedur RESET, REWRITE, APPEND

RESET(nazwa pliku) – otwiera już tylko istniejący plik

REWRITE(nazwa pliku) – otwiera zarówno nowy plik jak i istniejący (kasujący zapisane dane w tym pliku)

APPEND(nazwa pliku) – otwiera plik tylko do zapisu. Dane są zapisywane na końcu pliku.

Do zamykania pliku stosuje się procedurę CLOSE w postaci

CLOSE(nazwa pliku)

Zapisywanie i odczytywanie danych z plików tekstowych dokonuje się za pomocą instrukcji READ, READLN, WRITE, WRITELN uzupełniających o nazwę pliku i list argumentów.

PRZYKŁAD:

Napisać program pozwalający zapisywać i odczytywać dane z pliku tekstowego.

Do pliku należy wpisać liczby naturalne od 10 do 30

f- zmienna plikowa

liczby.dan – nazwa pliku dyskowego

program plik tekstowy;

uses crt;

var f:tekst; (zmienna plikowa)

i: integer;

begin;

clrscr;

ASSIGN(f,'liczby.dan'); (skojarzenie zmiennej plikowej f z rzeczywistą nazwą pliku na dysku liczby.dan)

REWRITE(f) (otwarcie pliku)

FOR i:=10 to 30 do

WRITE(f,i:4); (Zapisanie liczby do pliku)

CLOSE(f) (zamknięcie dostępu do pliku)

WRITELN('Plik liczby.dan zawiera liczby od 10 do 30');

Repeat until keypressed;

```
WRITELN('odczytanie plku liczby.dan');
RESET(f);           (otwarcie już istniejącego pliku
  WHILE NOT EOF(f) DO (dopóki nie ma końca pliku f)
    Begin
      READ(f,i);    (czytanie liczb z pliku)
      WRITE(i:5);   (wyświetlenie liczby)
    End;
  Close(f);         (zamknięcie dostępu do pliku)
End.
```

*EOF(end of File) – koniec pliku

Napisać procedurę przyjmującą nazwę osobę

Procedura przyjmowanie;

```
Begin
  Clrscr;
  ASSIGN(listap,'akta.dan');
  RESET(listap);
  WRITELN('Rezygnacja – naciśnij N+ ENTER);
  REPEAT
  WRITE('nazwisko:'); READLN(dane.nazwisko);
  IF ((dane.nazwisko<>'n')and(dane.nazwisko<>'N')) then
    Begin
      WRITE('imie');
      READLN(dane.imie);
      WRITE('rok urodzenia');
      READLN('dane.rok.or');
      WRITE('pleć m/k');
      READLN('dane.plec);
      WRITE('nr.osoby');
      READLN('dane.nr.osoby);
      SEEK('listap,FileSize(listap));
      Dane.NR.osob:=FilePos(listap);
      WRITE(listap,dane);
    End
  UNTIL(dane.nazwisko='N')or(dane.nazwisko='n');
  CLOSE(listap);
End.
```

Pierwsza pozycja w każdym pliku oznacza jest numerem 0. Po wykonaniu procedury

SEEK(listap,FileSize(listap))

Zmiana listap wskazuje pozycja FileSize(listap)

Wynikiem funkcji FileSize(listap) jest liczba wszystkich rekordów pliku.

Zmienna listap wskazywać będzie zawsze na ostatni element pliku. Takie ustawienie zmiennej ma na celu umożliwienie dopisania na końcu pliku danych personalnych nowego pracownika.

27. PLIKI ELEMENTOWE

Pliki elementowe – zdefiniowane, są to pliki, w których przechowywana informacja występuje w postaci zakodowanej. Nie ma więc możliwości bezpośredniego ich odczytania.

DEKLARACJA ZMIENNEJ PLIKOWEJ:

Nazwa pliku: FILE OF typ złożony

typ prosty – np. real, integer

typ złożony – (typ, który sobie utworzyliśmy, np. array)

Przyporządkowanie nazwy pliku dyskowego do danej zmiennej plikowej dokonuje się za pomocą procedury ASSIGN (jak dla txt).

Do otwierania plików procedury RESET, REWRITE (bez APPEND).

Pliki zamyka się procedurą CLOSE.

ZADANIE:

Napisać program pozwalający wyznaczyć liczbę elementów w pliku elementowym.

Program ma tworzyć plik dysk. dane.den :

Program wyznaczenie;

Uses crt;

Var Ile: integer; {ilość elem.pliku}

X:real;

F: File of real {plik typu elem.}

Nazwa:string;

Begin

Clrscr;

Writeln('Wyznaczanie liczby elementów w pliku');

Writeln('podaj nazwę pliku z danymi');

Readln(nazwa);

If length(nazwa)=0 then nazwa:='rzecz.den'

ASSIGN(f,nazwa);

RESET(f);

Ile := 0

While not eof(f) do

Begin

```
        Read(f,x);
        Ile:=ile+1;
    End;
    Close(f);
    Writeln('w pliku:',nazwa,' jest',ile,'liczb rzeczywistych');
    Readln;
End.
```

28. OPERACJE NA PLIKACH DYSKOWYCH

Do wyboru określonego elementu z pliku elementowego służy tzw. Wskaźnik pliku. Do ustawienia tego wskaźnika pliku na określonym elemencie służy procedura SEEK. Elementy są numerowane od zera.

Procedura SEEK ma postać:

SEEK(VAR f, poz:longint);

Np.

SEEK (f,5) oznacza ustawienie wskaźnika pliku w pliku identyfikowanym zmienną plikową f na pozycji 6

W celu określenia pozycji wskaźnika pliku w pliku identyfikowanym zmienną plikową f stosuje się funkcję FilePos

Funkcja FilePos ma postać:

FilePos (var f):longint;

Np.

Poz := FilePos(f)

Przy dzieleniu liczby elementów w pliku identyfikowanym zmienna plikowa f stosujemy funkcję FilePos.

FUNKCJA Filepos ma postać;

Filepos (var f):longint

Np.:

Liczba := filesize(f);

Jeżeli plik jest pusty podawana jest wartość zero.

Przykład

Napisać program pozwalający zapisywać dane do pliku elementowego w postaci liczb naturalnych od 1 do 10. Następnie zamienić 3 elementy ciągu na 23.

Program zmiana;

Uses crt;

Var f : file of INTEGER;

Y,X:INTEGER;

CH:CHAR;

Begin

Clrscr;

```
ASSIGN(F,'plik.dok');  
REWRITE(f);  
FOR i:=1 to 10 do  
Begin  
    Write(f,x);  
    Writeln(' ',x:2,'-',x:2);  
End;  
SEEK(f,2): {ustawienie wskaźnika na 3.elemencie}  
X:=23; {ustawienie nowej wartości}  
Write(f,x): {zapisanie do pliku}  
SEEK(f,0){ustaw wskaźnik na 1.elemencie}  
For i:=1 to 10 do  
    Begin  
        Read(f,y);  
        Writeln(' ',x:2,'-',y:2);  
    End;  
Ch:=readkey  
End.
```

29. DOPISYWANIE I USUWANIE TEKSTU

INS – włączanie lub wyłączanie trubu dopisywania,
Ctrl N – wstawianie pustego wiersza,
Ctrl Y – usunięcie wiersza,
Backspace – usunięcie znaku przed kursorem,
DEL – usunięcie znaku za kursorem.

Operacje na blokach tekstu

Ctrl K B – zaznaczenie początku bloku
Ctrl K K – zaznaczenie końca bloku
Ctrl K C – skopiowanie zaznaczonego bloku
Ctrl K V – przeniesienie zaznaczonego bloku
Ctrl K Y – usunięcie zaznaczonego bloku
Ctrl K H – zgaszenie podświetlenia bloku
Ctrl K R – wczytanie bloku z dysku
Ctrl K w – zapisanie bloku na dysku
Ctrl Ins – skopiowanie zaznaczonego bloku do bufora
Shift Ins – przeniesienie zawartosci z bufora do aktywnego okna

Wykaz pewnych błędów tłumaczenia programu

Nr	znaczenie komunikatu
3	nazwa niezadeklarowana lub użyta poza zakresem

8	brak w napisie symbolu
20	spodziewana nazwa zmiennej
26	niezgodność typów
36	spodziewany symbol begin
37	spodziewany symbol end
38	spodziewany wyrażenie typu całkowego
40	spodziewany wyrażenie typu logicznego
42	błąd w wyrażeniu
85	brak symbolu;
87	brak symbolu,
88	brak symbolu(
89	brak symbolu)
90	brak symbolu=
91	brak symbolu:=
92	brak symbolu[
93	brak symbolu]
94	brak symbolu.
95	brak symbolu..
97	błędna zmienna sterująca instrukcji for

Wykaz pewnych błędów wykonania programu

Nr	znaczenie komunikatu
106	błąd w liczbie
200	dzielenie przez zero
201	wartość wyrażenia spoza dopuszczalnego przedziału
205	nadmiar zmiennopozycyjny