Algorytm

1. **Definicja i cechy algorytmu**

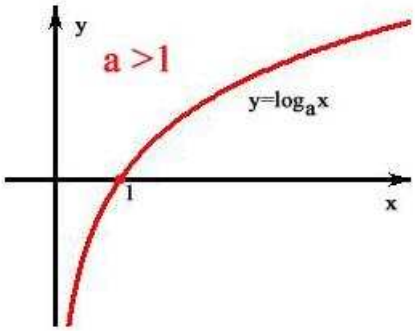
Algorytm - uporządkowany i skończony ciąg dokładnie określonych operacji na obiektach, do rozwiązania dowolnego zadania z określonej ich klasy.

Cechy algorytmu:
- **skończoność** (skończony zbiór operacji),
- **określoność** (operacje i porządek ich wykonania powinny być ściśle określone, niezależnie od obiektów, na których są wykonywane),
- **ogólność** (rozwiązywanie klasy zadań a nie pojedynczego zadania),
- **efektywność** (rozwiązanie najmniejszym kosztem).

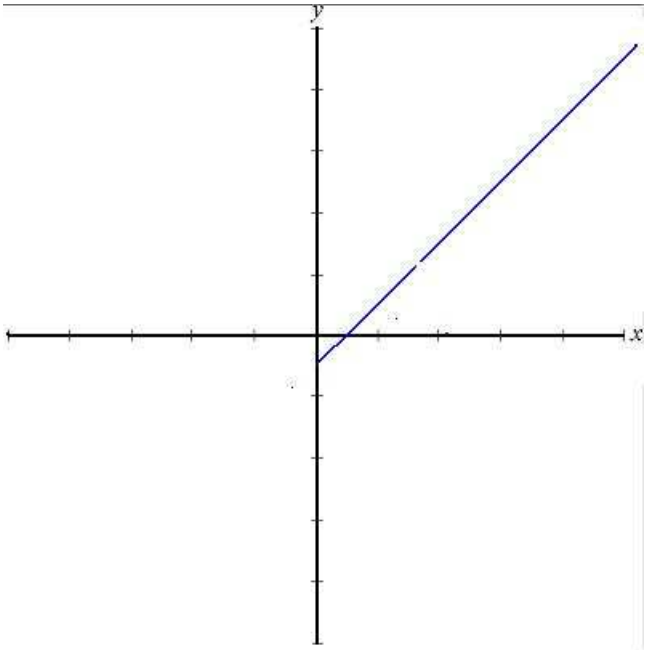
2. **Złożoność obliczeniowa**

- **czasowa T(n):** pesymistyczna, średnia

logarytmiczna T=log(n)

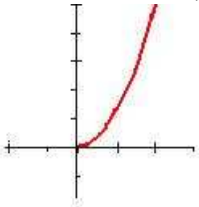


liniowa T=n

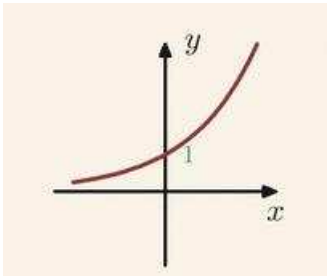


liniowa logarytmiczna T=n*log(n)

wielomianowa $T=n^m$, m=2,3...



wykładnicza $T=2^n$



- **pamięciowa**

3. **Projektowanie algorytmów**

   Do metodologii projektowania należy upraszczanie i wyodrębnianie niezależnych części (procedur, funkcji).
   Wyróżnia się algorytmy szeregowe i równoległe (arch. wieloprocesorowa)

   Projektowanie algorytmów:

   algorytmy zachłanne

   algorytmy wg strategii "dziel i rządź"

   algorytmy oparte na technice rekursji (rekurencji)

   algorytmy oparte na programowaniu dynamicznym (podproblemy)

   algorytmy z powrotem

4. **Struktury danych**

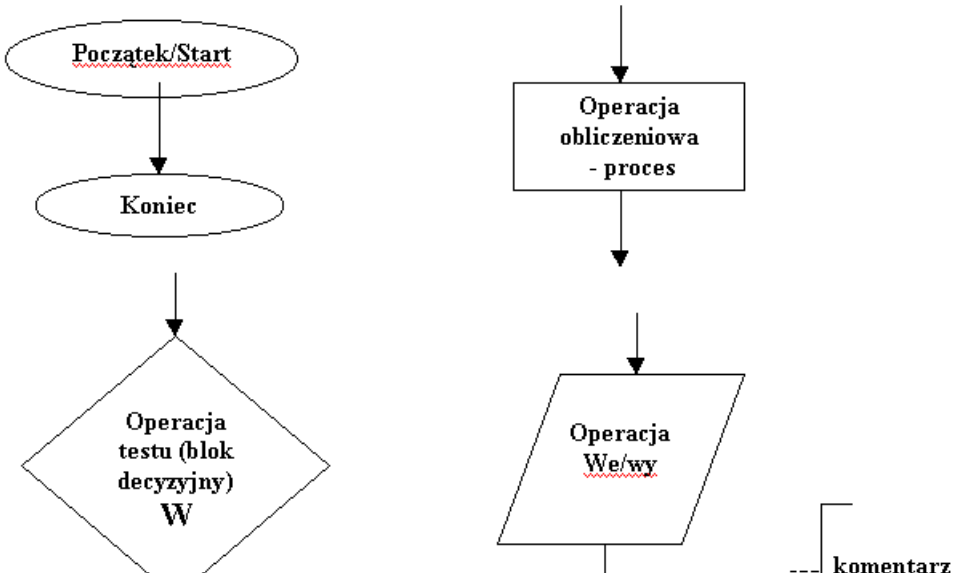
   - tablica
   - lista
   - kolejka (LIFO, FIFO)
   - stos
   - zbiór
   - graf

5. **Sposoby zapisu algorytmów**

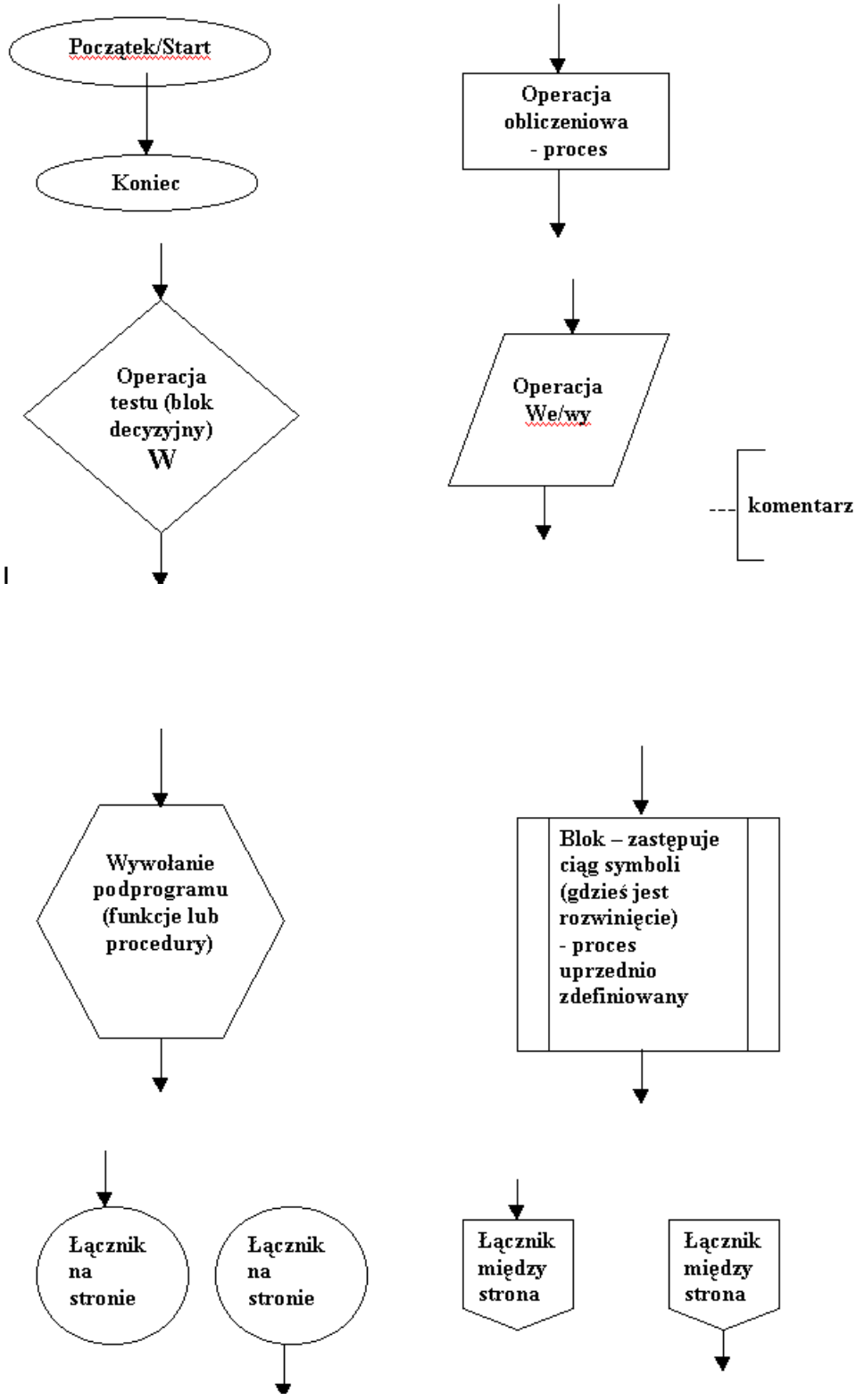
   - w *języku naturalnym (opisowo)*
   - za pomocą *schematów blokowych (schematy działania)*
   - przy pomocy *języków programowania*

Graficzna reprezentacja algorytmu - symbole graficzne

Schematy działania – schematy blokowe – graficzna reprezentacja algorytmu



Schematy działania – schematy blokowe – graficzna reprezentacja algorytmu



Algorytmy - konwencja notacyjna

1) Deklaracja zmiennych
stałe P;
całkowite N;
rzeczywiste N;
Logiczne N;
{gdzie N - lista zmiennych, P - lista podstawień}

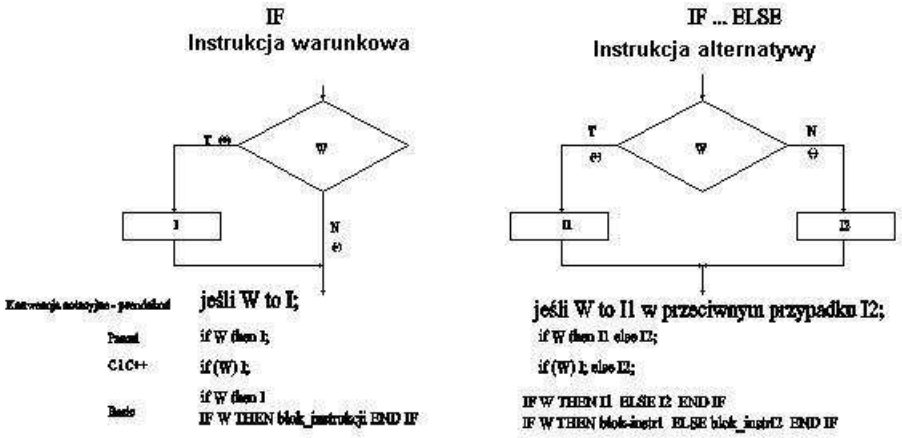
2) Deklaracje tablic:
calkowite tablica N [W1:W2], W[W1:W2, W3:W5];
rzeczywiste tablica N [W1:W2], W[W1:W2, W3:W5];
gdzie: N - nazwa tablicy, W1, W2 W3, W4 - wyrażenia o wrtościach całkowitych wyznaczajace odpowiednio najjniejszy i największy numer elementu tablicy

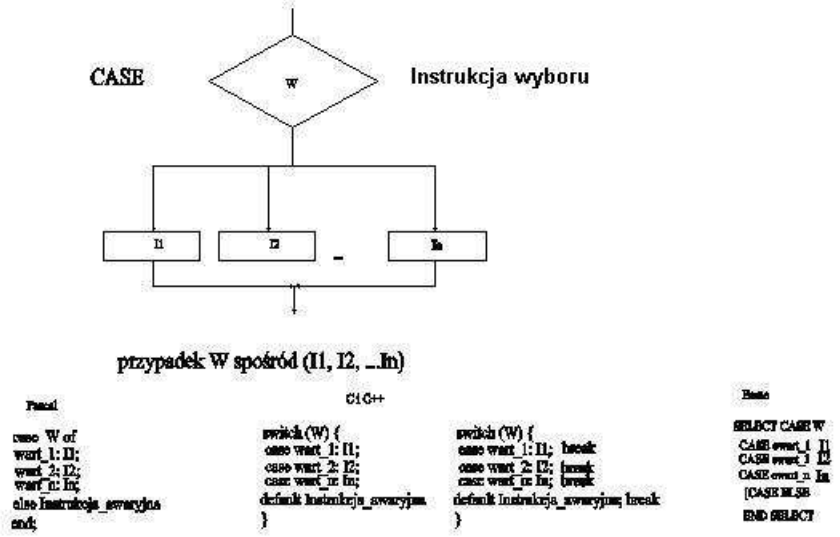
3) Instrukcja przypisania
n:=W; lub I:

n:=W; lub I;
gdzie N - nazwa zmiennej, W- wyrazenie;, I - instrukcja prosta

4) Instrukcje z wyborem
jesli W to I;
jeśli W to I1 w przeciwnym przypadku I2;
przypadek N spośród (I1, I2, ...In);
gdzie W - warunek,
I, I1, ...In - instrukcje proste lub złożone,
N - wyrazenie przyjmujące wartości jedynie z przedziału [1, n]

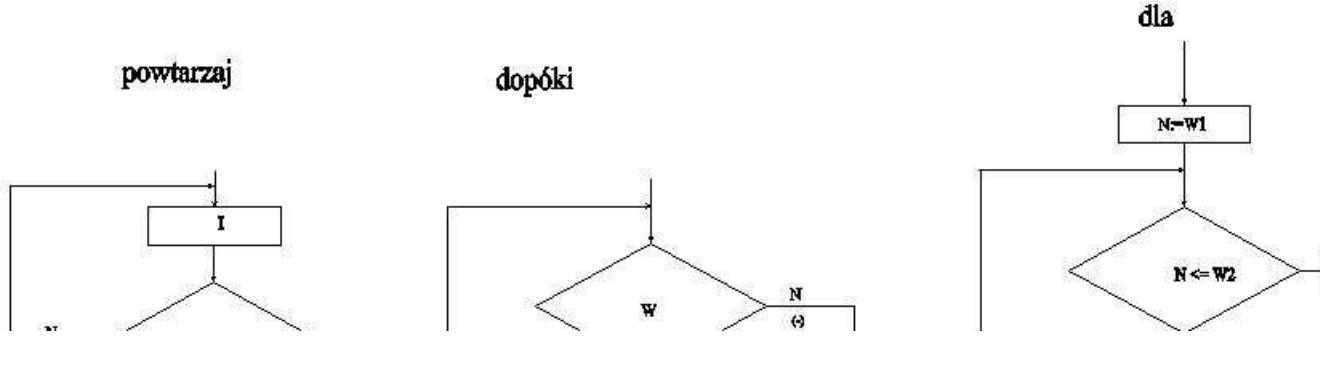
Instrukcjom tym odpowiadają schematy blokowe

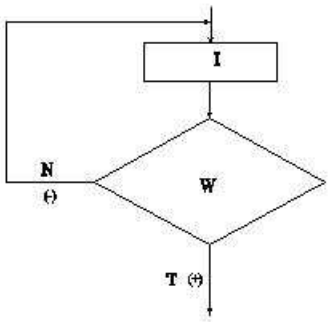


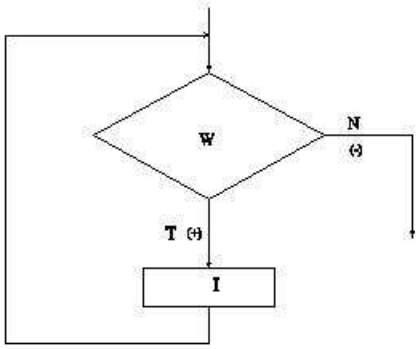


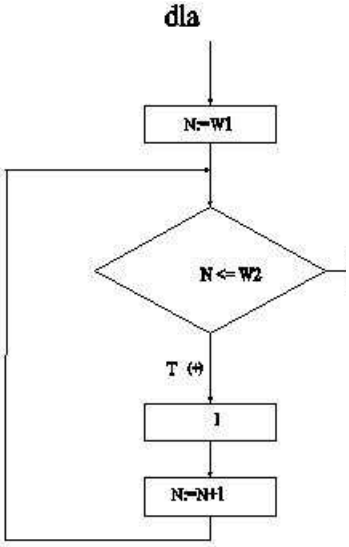
Schematy blokowe instrukcji wyboru

5) Instrukcje iteracji
powtarzaj I aż do W;
dopóki W wykonuj I;
dla N:=W1 do W2 wykonuj I;
gdzie:
I - instrukcja prosta lub złożona
W - wariant kontynuacji lub zakończenia iteracji
N - nazwa zmiennej
W1, W2 - wyrażenia









Pseudokod	powtarzaj I aż do W;	dopóki W wykonuj I;	dla N:=W1 do W2 wykonuj I;
Pascal	repeat I; until W;	while W do I;	for N:=W1 to W2 do I;

C, C++	do
	I;
	while (!W);
		while (W) I;	for (N:=W1; N<=W2; N++) I;
			równoważna instrukcji
			N=1;
			while (N<W2)
			{
			I;
			N=N+1;
			}

Basic	do until W I; loop do I; loop until W	Przykład w Basic'u k = 0 DO UNTIL k = 5 k = k + 1 PRINT k LOOP	do while W I loop do I loop while W	Przykład w Basic'u k = 0 PRINT "1" DO WHILE k < 5 k = k + 1 PRINT k LOOP	for N:=W1 to W2 I next N

6) Instrukcje wejścia/wyjscia
czytaj (N); pisz (W);
gdzie N - nazwa zmiennej;
W - wyrazenie

7) Instrukcja procedury
Deklaracja: procedura P(W); S;
treść;
gdzie:
P - nazwa procedury,
W - wykaz parametrów formalbycg,
S - specyfikacj (deklaracja) parametrów formalnych
Wywołania: P(WA);
gdzie: P - nazwa procedury,
WA - wykaz parametrów aktualnych

8) Instrukcja procedury funkcyjnej
Deklaracja:
całkowita procedura P(W); S;
rzeczywista procedura P(W); S;
logiczna procedura P(W); S;
gdzie:
P - nazwa procedury
W - wylkaz parametrów formalnych,
S - specyfikacja parametrów formalnych.
Wywołanaie: N:= W1 + P(WA);
gdzie:
P - nazwa procedury,
N - nazwa zmiennej,
W1 - wyrażenia,
WA - wykaz parametrów aktualnych.

WA - wykaz parametrów aktualnych.

9) Instrukcja prosta
Każda z instrukcji opisanych w punktach 3 do 8.

10) Instrukcja złożona
Jest to ciąg instrukcji prostych lub złożonych , określonych następujaci:
poczatel I1; I2; ... In koniec
gdzie: Ii - instrukcja prosta lub złożona

Przykład

Dany jest zbiór n liczb X1, X2, ..., Xn. Należy obliczyć iloczyn y=P Xi = X1*X2*...*Xn.

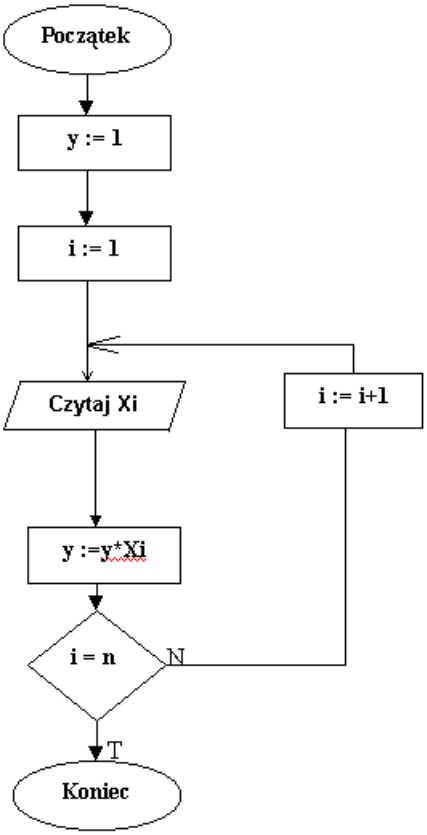
Algorytm

1) Język naturalny:

A. Przyjmij wartość y równą 1 i skocz do B
B. Przyjmij wartość i równą 1 i przejdź do C
C. Przyjmij y równe iloczynowi y*Xi i skocz do B
D. Jeżeli i=n to Koniec. W przeciwnym razie przyjmij i równe i+1 i skocz do C

1. y :=1
2. i := 1
3. y := y*Xi
4. Jeżeli i=n to Koniec. W przeciwnym razie i := i+1 i skocz do 3

2) Schemat blokowy



Uwaga: Jeśli chcemy uwzględnić ciąg kolejnych liczb naturalnych, bez wprowadzania Xi to należałoby zmodyfikować algorytm:
Pominąc moduł Czytaj Xi
Zmodyfikowac wzór na obliczenie y
y := y* i;

Konstrukcje programu spotykane w schematach blokowych:

Konstrukcje programu spotykane w schematach blokowych:

sekwencja instrukcji (blok), test, instrukcja wyboru (case, switch), instrukcje iteracji (while, repeat, do while, for), instrukcje we/wy, procedury, funkcje

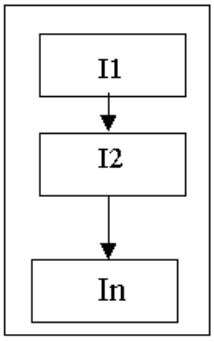
- zapis w Pascalu i C/C++.

Sekwencja instrukcji

Konstrukcje programu

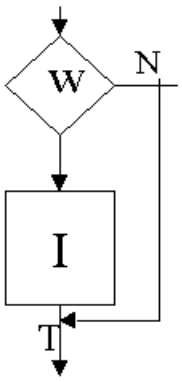
Zapis w językach

Sekwencja instrukcji:



Pascal	*C/C++*	**Basic**
begin	{	' Poczatek programu glównego - brak 'wyróżnienia
I1 ;	I1 ;	
I2;	I2;	I1 : I2 ' 2 instrukcje w jednej linii
.....	...	
In;	In;	In ' instrukcja pojed.
end;	}	END ' Nie musi być jeśli po nim nie ma 'definicji procedur lub funkcji

Operacje testu



if W then I;	if (W) I;	IF W THEN I:

Struktura programów w językach Basic, Pascal, C:

' Basic

' Poczatek programu glównego - brak wyróżnienia

' Deklaracje zmiennych, funkcji, podprogramow

' DECLARE function funkcja1 ()
' DECLARE SUB podprogram1()

I1 : I2

In

END ' Nie musi być jeśli po nim nie m a definicji procedur lub funkcji

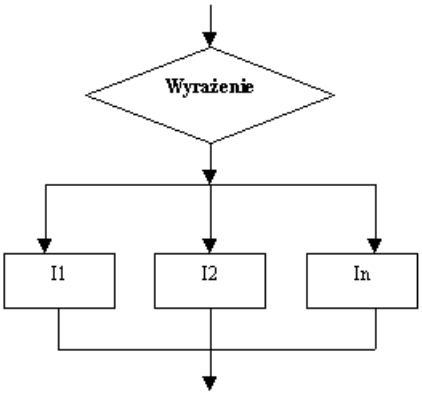
{Pascal:}

```
Program Nazwa;
{Deklaracje}

Begin
{Instrukcje}
I1; I2;
In;

End.
```

```
// C
// Deklaracje, definicje funkcji
void funkcja()
{
};

main()
{
// Instrukcje
// Sekwencja instrukcji - poczatek
{I1; I2;
In;
} // Sekwencja instrukcji - koniec
}
```

Instrukcje wyboru

Instrukcja wyboru:



Instrukcje Case w Pascalu,
SELECT CASE w Basic'u
i
switch w C

Pozwala dokonywać wyboru spośród wielu różnych możliwości

PASCAL:

```
Case wyrażenie od
Wartość_1: Instrukcja_1;
Wartość_2: Instrukcja_2;
Wartość_n: Instrukcja_n
Else Instrukcja_awaryjna
End
```

C:

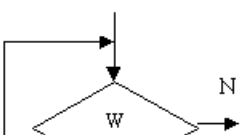
```
switch (wyrażenie)
{
 case wyrażenie_stałe_1: Lista_instrukcji_1;
break;
case wyrażenie_stałe_2: Lista_instrukcji_2;
break;
case wyrażenie_stałe_n: Lista_instrukcji_n;
break;
default : awaryjna_lista_instrukcji;
}
```

Basic

```
SELECT CASE wyrazenie
CASE wyrazenie1
  [Lista_instrukcji_-1]
[CASE expressionlist2
  [Lista_instrukcji_2]...
[CASE ELSE
  [Lista_instrukcji_n]]
END SELECT
```

Organizacja obliczeń cyklicznych - instrukcje iteracji

Dopóki W wykonuj I – instrukcja while

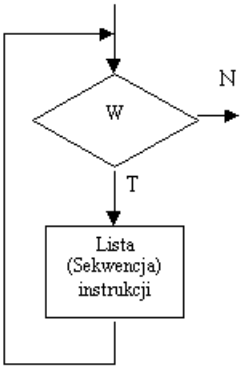


Dopóki zachodzi warunek W to wykonuj sekwencję instrukcji I

Pascal:

Organizacja obliczeń cyklicznych - instrukcje iteracji

Dopóki W wykonuj I – instrukcja while

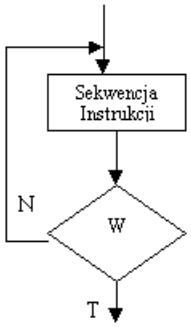


Dopóki zachodzi warunek W to wykonuj sekwencję instrukcji I

Pascal:
While W do Lista_instrukcji;

C:
while (W) lista_instrukcji;

Powtarzaj I aż zajdzie warunek W : repeat...until; do...while

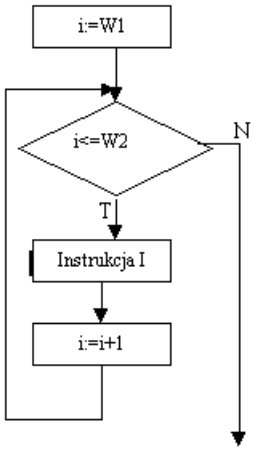


Pascal:
Repeat
Instrukcja_1;
Instrukcja_2;
.....
Instrukcja_n;
Until W;

C:
do {Lista_instrukcji} while (!W);

Instrukcja for – dla

Instrukcja jest stosowana, gdy z góry można określić liczbę niezbędnych powtórzeń obliczeń



Dla zmiennej i w granicach W1 do W2 wykonuj instrukcję I
Pascal:
For zmienna:=wartość_pocz to wartość_końcowa do Instrukcja;
FOR zm_ster:=Wyr1 to Wyr2 do
lub
FOR zm_ster:=Wyr1 downto Wyr2 do
For i:=W1 to W2 do I;
For i:=W1 to W2 do Begin I1; I2; .... In; End;
Lub
For zmienna:=wartość_pocz downto wartość_końcowa do Instrukcja;
For i:=W1 downto W2 do I;

C:
for (wyrażenie1; wyrażenie2; wyrażenie3) Instrukcja;
Co jest równoważne konstrukcji:
Wyrażenie1;
while (wyrażenie2) {instrukcja; wyrażenie3; }
for (i=W1; i<=W2; i++) I;

Instrukcje wejścia /wyjścia (we/wy)

Instrukcje wejścia – czytania:

Czytaj(N); - N – nazwa zmiennej

W Pascalu:

Read(lista_argumentów) lub **Readln**(lista_argumentów);

Np.

```
a,b,c : integer;
```

Np.

```
a,b,c : integer;
read(a,b,c);
```

W języku C:

Funkcja: **scanf**

scanf(łańcuch_formatu, lista_argumentów);

np. scanf(“%f”,&promien);

Funkcja **getchar**() umożliwia wprowadzenie pojedynczego znaku do pamięci komputera:

c=getchar(); gdzie c jest zmienną typu char: char c;

Funkcja **gets** umożliwia wprowadzenie jednej linii tekstu.

Przykład:

```
char linia[80];
gets(linia{;
```

W języku C++

Wykorzystanie strumienia wejściowego **cin** i operatora >>, np.

```
int a, b, c;
cin >> a >> b >> c;
```

W języku Basic

```
INPUT odczytuje wprowadzenie z klawiatury lub z pliku. LINE INPUT czyta
linię do 255 znaków z klawiatury lub z pliku.
 INPUT [;] ["komunikat"{; | ,}] lista_zmiennych
LINE INPUT [;] ["komunikat";] zmienna$
INPUT #filenumber%, lista_zmiennych
LINE INPUT #numer_pliku%, zmienna$
     • komunikat        Opcjonalny ciąg znaków, który jest wyświetlany
                       zanim użytkownik wprowadzi dane. Średnik po komuni-
                       kacie dodaje do niego znak zapytania.
     • lista_zmiennych  Jedna lub więcej zmiennych oddzielonych przecinkami,
                       w których są przechowywane dane wprowadzane z klawia-
                       tury lub wczytywane z pliku. Nazwy zmiennych mogą
                       składać się z maksimum 40 znaków i muszą zaczynać się
                       od litery. Akceptowane znaki to A-Z, 0-9, i kropka (.).
     • zmienna$          Przechowuje linię znaków wprowadzoną z klawiatury
                       albo odczytaną z pliku.
     • numer_pliku%     Numer otwartego pliku.
     • INPUT używa przecinka jako separatora pomiędzy wprowadzeniami.
     LINE INPUT czyta wszystkie znaki do znaku końca wiersza.
     • Dla wpisu z klawiatury - średnik bezpośrednio po INPUT utrzymuje
     kursor w tej samej linii po naciśnięciu Enter przez użytkownika.

Przykłady:

Przykład 1)
```

```
CLS
PRINT "Obliczenie pierwiastka z podanej liczby metoda iteracyjna"
PRINT
INPUT "Podaj dokladnosc obliczenia "; e#
INPUT "Wprowadz liczbe "; a#
' e# = .0000005#
r# = 1    ' Wartosc startowa pierwiastka
i% = 0    ' Iteracja

DO       ' Poczatek petli
 i% = i% + 1
 PRINT "Iteracja "; i%; " q1="; q#; "  r1="; r#
 q# = a# / r#
 r# = (r# + q#) / 2
 PRINT "q2="; q#; " r2="; r#; " r-q = "; (r# - q#)
LOOP WHILE ABS(r# - q#) > e#    ' Koniec petli

PRINT
PRINT "Pierw obsl z "; i%; " iteracji = "; r#    '
PRINT "Pierw    =    SQR("; a#; ") = "; SQR(a#);
```

```
Przykład 2)
```

Przykład 2)

```
    CLS
    OPEN "LIST" FOR OUTPUT AS #1
    DO
        INPUT "   IMIE:       ", Imie$  'Czyta wpisy z klawiatury.
        INPUT "   WIEK:       ", Wiek$
        WRITE #1, Imie$, Wiek$
        INPUT "Dodac nowy wpis"; R$
    LOOP WHILE UCASE$(R$) = "T"
    CLOSE #1
    'zwrotne echo z pliku
    OPEN "LIST" FOR INPUT AS #1
    CLS
    PRINT "Zapisy w pliku:": PRINT
    DO WHILE NOT EOF(1)
        LINE INPUT #1, REC$  'Czyta zapisy z pliku.
        PRINT REC$           'Wypisuje je na ekranie.
    LOOP
    CLOSE #1
    KILL "LIST"
```

```
REM abcq3.bas
REM Program liczy sume liczb od podanej wartosci N1 do podanej N2
CLS
PRINT "Program liczy sume liczb od N1 do N2"
PRINT
INPUT "Podaj n1, n2 "; n1, n2
sum = 0
FOR k = n1 TO n2
sum = sum + k
NEXT k
PRINT
PRINT "Suma liczb od "; n1; " do "; n2; " = "; sum
END
```

Instrukcje wyjścia – pisania

Pisz(W); - W – wyrażenie

W języku Pascal:

Write(lista_argumentów);
Writeln(lista_argumentów);

Np.

```
a,b,c : integer;
read(a,b,c); {wczytanie danych}
writeln('a=',a:3, ' b=',b:3, 'c=',c:3); {wypisanie danych na ekranie}
```

W języku C:

Funkcja **printf**(łancuch_formatu, lista argumentów);

Przykład:

```
int a, b, c;
scanf("%d %d %d",a,b,c); /* wczytanie danych */
printf("a=%d b=%d c=%d",a,b,c); /*wydruk*/
```

Funkcja **putchar** wyprowadza na zewnątrz jeden znak.

Przykłady:

```
int znak='a';
putchar('\n'); putchar(znak);
```

Funkcja **puts** wyprowadza cały łańcuch znaków.

Przykład:

```
char napis[]="ABCDEF";
puts(napis);
puts("\nTo jest napis");
```

W C++ do wyświetlania stosuje się strumień wyjściowy **cout** oraz operator <<

Np.

```
cout << a << b;
cout << "Podaj dane";
```

Np.

```
cout << a << b;
cout << “Podaj dane”;
```

W języku Basic

PRINT USING, LPRINT USING

```
PRINT USING wypisuje tekst sformatowany na ekran lub do pliku.
LPRINT USING drukuje tekst sformatowany na drukarce LPT1.

PRINT [#numer_pliku%,] USING łańcuch_formatujący$; lista_wyrażeń [{; | ,}]
LPRINT USING łańcuch_formatujący$; lista_wyrażeń [{; | ,}]
      • numer_pliku%     Numer otwartego pliku sekwencyjnego.
      • łańcuch_formatujący$;    Wyrażenie łańcuchowe zawierające jeden
                        lub wiecej znaków formatujących.
      • lista_wyrażeń    Lista jednego lub więcej wyrażeń numerycznych lub
                        łańcuchowych do wypisania, oddzielonych perzecinkami,
                        średnikami, apcjami lub tabulatorami.
      • {; | ,}          Określa, gdzie rozpocznie się następny wpis:
                           ; oznacza drukowanie bezpośrednio po ostatniej
                             wartości.
                           , oznacza drukowanie od początku następnej strefy
                             drukowania. Strefy drukowania mają szerokość
                             14 znaków.
 Zobacz także:    PRINT, LPRINT    WIDTH

Przykład:
    a = 123.4567
    PRINT USING "###.##"; a
    LPRINT USING "+###.####"; a
    a$ = "ABCDEFG"
    PRINT USING "!"; a$
    LPRINT USING "\ \"; a$
```

Znaki, które formatują wyrażenie numeryczne

#	Pozycja cyfry.	-	Umieszczony po cyfrze, drukuje wiodący minus
.	Pozycja kropki dziesiętnej.		dla liczb ujemnych.
,	Umieszczony na lewo od kropki dziesiętnej, drukuje przecinki co każde trzy cyfry.	$$	Drukuje wiodący $.
		**	Wypełnia gwiazdkami (*)
+	Pozycja znaku liczby .		wiodace spacje.
^^^^	Drukuje w postaci wykładniczej	**$	Kombinacja ** and $$.

Znaki używane do formatowania wyrażenia łańcuchowego

&	Drukuje cały łańcuch.	\ \	Drukuje pierwszych n znaków,
!	Drukuje tylko pierwszy znak łańcucha.		gdzie n jest liczbą spacji miedzy \ \ plus 2.

Znaki używane do druku literałów

_	Drukuje następny znak formatujacy jako literał	Każdy znak nie zamieszczony w tej tabeli jest drukowany jako literał.

Podprogramy

Podprogram jest to wyróżniona część programu komunikująca się z pozostałą częścią w ściśle określony sposób. Do komunikacji wykorzystuje się parametry, w definicji podprogramu nazywane parametrami formalnymi, a przy wywołaniu podprogramu parametrami aktualnymi. Podprogram może być wielokrotnie wywoływany w części głównej programu lub innych podprogramów. Wywołanie podprogramu polega na podaniu jego nazwy oraz w nawiasach parametrów.

W języku Pascal istnieją 2 rodzaje podprogramów: procedury i funkcje.

Procedury

Ogólna postać procedury (w języku Pascal)

Procedure nazwa (lista_parametrów_formalnych);
{deklaracje}
begin
{treść procedury}
end;

Przykład:

Przykład:

```
Procedure kwadrat(w:integer);
Begin
Writeln('Liczba: ',w);
Writeln('Kwadrat: ',w*w);
End;
```

Wywołanie procedury:
nazwa(lista_parametrów_aktualnych);
np. kwadrat(2);

Parametry można przekazywać przez wartość (podprogram nie zmienia wartości zmiennych wejściowych) i przez zmienną – przed parametrem przekazywanym przez adres musi być słowo var (zwraca zmienione wartości parametrów).

Funkcje w języku Pascal

Ogólna postać funkcji:

Function nazwa (lista_parametrów_formalnych): typ_wyniku;
{deklaracje stałych, zmiennych i typów}
begin
{treść funkcji}
end;

W treści funkcji musi być przypisanie nazwa:=wynik;

Wywołanie funkcji:

Zmienna:=nazwa(lista_parametrów_aktualnych);

Przykład:

```
Function kwadr(bok:integer):integer;
Begin
Kwadr:=bok*bok;
End;
```

Wywołanie:
Writeln('Pole kwadratu o boku 5 = ',kwadr(5));

Język C/C++

W języku C są tylko funkcje

Definicja dowolnej funkcji jest następująca:

typ nazwa(deklaracje_parametrów)
{
instrukcje
}

Przykłady:

```
int kwadr(int a) /* analogia do funkcji w języku Pascal */
{
return (a*a);
}
```

Wywołanie:

printf("Pole kwadratu o boku 5 = %d",kwadr(5));

Obliczenie sumy kwadratów liczb:

```
Int SumaKwadratow(int n)
{
int i, suma=0;
for (i=1; i<=n; i++)
suma += i*i;
return (suma);
}
```

Basic

Basic

Procedury

SUB nazwa_proc [(par1, [par2,...])]
[daklaracje_rodzaju_zmennych]
Instrukcje
[EXIT SUB]
END SUB

Wywołanie:
CALL nazwa_proc(parametry);

Przykład:

```
REM p263
REM Program rozwiazuje rownanie kwadratowe
REM  a x^2 + b x  + c  =  0
REM del   wyznacznik rownania
REM  x1, x2  pierwiastki
REM  PODPROGRAMY
REM  dane()  czytania danych
REM  delta()  liczy del
REM  pierw()  oblicza x1, x2
REM  wyniki()  druk wynikow

REM PROGRAM GLOWNY
  DECLARE SUB dane (a, b, c)
  DECLARE SUB delta (a, b, c, del)
  DECLARE SUB pierw (a, b, sqrdel, x1, x2)
  DECLARE SUB wyniki (a, b, c, x1, x2)
start:
 CLS
 LOCATE 12, 20
 PRINT "Program rozwiazuje rownanie kwadratowe"
  LOCATE 14, 30
  PRINT "a*x^2+b*x+c=0"
  INPUT "START - Enter "; st$
  CALL dane(a, b, c)
  CALL delta(a, b, c, del)
    IF del < 0 THEN
       CLS
       LOCATE 10, 20
       PRINT "delta < 0,  nie ma rozwiazania "
       GOTO koniec
    END IF
  sqrdel = SQR(del)
  CALL pierw(a, b, sqrdel, x1, x2)
  CALL wyniki(a, b, c, x1, x2)
koniec:
  LOCATE 22, 50
  INPUT "Nastepne rownanie ?  T / N "; tn$
  IF tb$ = "T" OR tn$ = "t" THEN GOTO start
  END

 SUB dane (a, b, c)
   INPUT "a = "; a
   INPUT "b = "; b
   INPUT "c = "; c
END SUB

SUB delta (a, b, c, del)
  del = b ^ 2 - 4 * a * c
END SUB

SUB pierw (a, b, sqrdel, x1, x2)
  mian = 2 * a
  x1 = (-b - sqrdel) / mian
  x2 = (-b + sqrdel) / mian
END SUB

SUB wyniki (a, b, c, x1, x2)
  CLS
  LOCATE 6, 20
  PRINT "Rownanie "; a; "x^2 + "; b; "x + "; c; " =  0"
  LOCATE 8, 22
  PRINT " x1="; x1, "x2="; x2
END SUB
```

Funkcje

Defines a FUNCTION procedure.

```
FUNCTION name [(parameterlist)] [STATIC]
   [statementblock]
  name = expression
   [statementblock]
END FUNCTION
```

- name — The name of the function and the data type it returns, specified by a type-declaration character (%, &, !, #, or $).
- parameterlist — One or more variables that specify parameters to be passed to the function when it is called:

  variable[( )] [AS type] [, variable[( )] [AS type]]...

  Variable is a BASIC variable name.
  Type is the variable type (INTEGER, LONG, SINGLE, DOUBLE, STRING, or a user-defined type).
- STATIC — Specifies that the values of the function's local variables are saved between function calls.
- expression — The return value of the function.

Example:

```
DECLARE FUNCTION ThirdOf (x)
  FOR i = 3 TO 12
      PRINT i, ThirdOf(i)
  NEXT i
  END
  FUNCTION ThirdOf (x)
      ThirdOf = x / 3
  END FUNCTION
```

Przykład:

```
REM p27.bas
REM Zastosowanie funkcji – FUNCTION nazwa (parametry)
DECLARE FUNCTION sumprz (a, b, c)
CLS
INPUT "Podaj 3 liczby "; a, b, c
PRINT "Suma dlug. przek prostop a,b,c= "; sumprz(a, b, c)
END

FUNCTION sumprz (a, b, c)
p1 = SQR(a ^ 2 + b ^ 2)
p2 = SQR(b ^ 2 + c ^ 2)
p3 = SQR(a ^ 2 + c ^ 2)
sumprz = 2 * (p1 + p2 + p3)
END FUNCTION
```

Reprezentacja informacji w komputerze

Zmienne binarne - przechowują znaki binarne 0, 1 - bity

Wektory zmiennych binarnych - przechowują wektory znaków binarnych

Długość słowa	Wektor- nazwa polska	Nazwa angielska
1	bit	bit
4	kęs BCD	nibble
8	bajt	byte
16	słowo 16 bitowe	word
24	sówo 24 bitowe	word
32	słowo 32 bitowe	word

Słowo - wielkość wektora, która przesyłana jest (wymieniana z pamięcią komputera) w wyniku wykonania jednej operacji czytania lub pisania.

Blok informacji cyfrowej - wielkość informacji wymieniana z pamięcia zewnetrzna.

Blok informacji cyfrowej - wielkość informacji wymieniana z pamięcią zewnętrzną.

1KB=1024B=2^10B

1MB=1024KB=2^10KB=2^20B=1048576 bajtów

1GB=1024MB=2^30B

Kodowanie, dekodowanie

Dla człowieka naturalnym sposobem liczenia jest korzystanie z systemu dziesiętnego, a dla komputera korzystanie z zapisu dwójkowego. Przepływowi prądu odpowiada wartość bitu 1, a brakowi przepływu cyfra binarna 0. W urządzeniach komputerowych, które oparte są na zastosowaniu elementów elektronicznych dwójkowy (binarny) sposób reprezentacji (inaczej kodowania) danych jest najprostszy. Polega na pojawianiu się w kolejnych odstępach czasu impulsów elektrycznych (1) lub ich braku (0). Mogą one reprezentować liczbę, znak, rozkaz lub adres komórki pamięci.

Podstawową jednostką informacji w komputerach jest bajt równy 8 bitom. Jeśli do zakodowania jednego znaku wykorzystuje się 1 bajt, to można zaadresować 2^8 (2 do potęgi 8) = 256 różnych znaków. Mogą to być litery alfabetu, cyfry, operatory matematyczne, znaki specjalne.

Aby możliwa była wymiana informacji między różnymi komputerami, opracowano standardowy kod wymiany informacji ASCII (American Standard Code for Information Interchane), w którym każdemu znakowi przyporządkowano liczbę (kod) od 0 do 255.

Kody od 0 do 31 to znaki specjalne nie mające odpowiednika w alfabecie. Są to znaki sterujace ekranem i drukarką.
Kody od 32 do 127 to m.in. kody cyfr (od 48 (0) do 57 (9)), kody dużych liter alfabetu (od 65 (A) do 90 (Z)), kody małych liter alfabetu (od 97 (a) do 122 (z)). Rozszerzenie ASCII zawiera definicje znaków o kodach od 128 do 255. Mogą być wykorzystane m.in. do definicji znaków narodowych. Np. znane są standardy polskich liter: Mazovia, DHN, Latin 2.

Przy kodowaniu (zapisie) informacji mamy 2 sposoby: **zapis prosty** i **zapis kodowany**. Zapis prosty - jednemu stanowi /sytuacji odpowiada jedna zmienna binarna. W zapisie kodowanym każdej sytuacji odpowiada konfiguracja zmiennych binarnych. Zapis prosty nieoszczędny - dużo zmiennych binarnych.

Zapis	Liczba sytuacji	Liczba zmiennych binarnych
prosty	n	n
kodowany	n	2^log(n) (przy podstawie 2)

Przejście z zapisu prostego na kodowany - **kodowanie**, odwrotnie - **dekodowanie**.

Układ kodujący: 1..2^n -> 1..n

Reprezentacja wartości alfanumerycznych (litery, cyfry, znaki specjalne, dodatkowe):

Kody powszechnie stosowane do reprezentowania wartości alfanumerycznych:

- **ISO-7** (powstał w Europie - 7-bitowy)
- **ASCII** - 128 znaków lub rozszerzony 256 znaków (0..255)
- **UNICODE** - 16 bitów do kodowania, 64 tys. znaków do umieszczenia, np. OFFICE 97

Reguły: znaki alfabet. w kolejności, cyfry w kolejności, litery duże i małe (stałe przesunięcie), 0-31 - znaki sterujące transmisją, 32-47 - pomocnicze, 48-57 - cyfry 0..9, 58-64 znaki pomocznicze, 65-90 duże literu A..Z, 91-96 znaki specjalne, 97-122 małe litery a..z, 123-126 znaki pomocnicze, 127 - znak sterujacy DEL.

Kody liczbowe

Kod **binarny prosty** - NKB - dane bez znaku - stosowany do kodowania liczb całkowitych dodatnich, np. prezentacji adresów komórek, numerów rejestrów.

a[i] = {0,1} - cyfry 0, 1

n elementowy wektor

a[n-1]	a[n-2]		a[1]	a[0]

a[n-1] - MSB (most significant byte), a[0] - LSB

L=2^0*a[0]+2^1*a[1]+...2^(n-1)*a[n-1] / a[i]={0.1}

Np. 1001 = 1*2^0+0*2^1+0*2^2+1*2^3

By skrócić zapis liczby stosuje się zapisy: ósemkowy (a[i]=0..7)i szestnastkowy (a[i]=0..F).

W zapisie 8-m trzy cyfry binarne zastępuje jedna cyfra ósemkowa. a[i] = {0..7}

W zapisie 16-m cztery cyfry binarne są zastąpione przez jedną cyfrę 16-ową.

W zapisie 16-m cztery cyfry binarne są zastąpione przez jedną cyfrę 16-ową.

Zapis dwójkowo-dziesiętny BCD - każdej cyfrze układu 10-go przyporządkowuje się 4 znaki binarne.

Cyfra dziesiętna	Kod BCD (binarny)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Zapis liczb całkowitych ujemnych

3 kody:

- znak, wartość bezwzględna
- kod uzupełnieniowy do 2 (U2)
- kod uzupełnieniowy do 1 (U1)

Zapis w postaci **znak wartość bezwzględna** : a[n] a[n-1] ..a[0].
W a[n] znak liczby: 0 gdy dodatnia, 1 gdy ujemna

L = Suma (2^i*a^i); i=0..n-1 gdy a[n]=0

L = -Suma (2^i*a^i); i=0..n-1 gdy a[n]=1

Zapis w postaci **kodu uzupełnieniowego do 2**

L = -2^n*a^n*a[n] + Suma (2^i*a^i); i=0..n-1

Przykład:

6 00110
-6 11010
Suma 100000

Zapis w postaci **kodu uzupełnieniowego do 1**

L = -(2^n-1)*a^n*a[n] + Suma (2^i*a^i); i=0..n-1

Przykład:

6 00110
-6 11011
Suma 11111

Kody liczb ułamkowych (rzeczywistych)

Zapis stałoprzecinkowy i zmiennoprzecinkowy.

W zapisie stałoprzecinkowym położenie kropki rozdzielającej jest dokładnie ustalone.

a[n] a[n-1] ... a[k] . a[k-1] ... a[0]

Zalety - operacje na liczbach całkowitych szybkie. Wada - mały zakres liczbowy.

Zapis zmiennoprzecinkowy

znak mantysy . mantysa m cecha c

|m| < 1
L = m*2^c

Zalety: duży zakres wartości, wady: mała liczba cyfr znaczących, operacje trwają dłużej.

Kody Gray'a

Cel - żeby tylko na jednej pozycji była zmiana cyfry

	a1\a0	0	1
0	0	00=0	01=1
2	1	10=3	11=2

a1	a0	Liczba dzies.
0	0	0
0	1	1
1	1	2
1	0	3

Kod Gray'a	Cyfra dzies.
000	0
001	1
011	2
010	3
110	4
111	5
101	6
100	7