Informatyka 1 (ES1E2009)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia stacjonarne I stopnia

Rok akademicki 2021/2022

Wykład nr 3 (05.04.2022)

dr inż. Jarosław Forenc

Plan wykładu nr 3

- Język C
  - operator warunkowy
  - instrukcja switch
  - pętla for
  - operatory ++ i –
  - pętle while i do…while
- Informatyka i informacja
- Informacja analogowa i cyfrowa
- Systemy liczbowe
  - liczby i cyfry
  - systemy pozycyjne i niepozycyjne

Język C - Operator warunkowy

- Operator warunkowy skład się z dwóch symboli i trzech operandów

```
wyrażenie1 ? wyrażenie2 : wyrażenie3
```

- Najczęściej zastępuje proste instrukcje if-else

```
float akcyza, cena, pojemnosc;
```

```
if (pojemnosc <= 2000)
    akcyza = cena*0.031;    /*  3.1% */
else
    akcyza = cena*0.186;    /* 18.6% */
```

```
akcyza = pojemnosc <= 2000 ? cena*0.031 : cena*0.186 ;
```

Język C - Operator warunkowy

```
if (x < 0)
     y = -x;
 else
     y = x;
```

```
y = x < 0 ? -x : x;
```

- obliczenie modułu liczby x

```
if (a > b)
     max = a;
 else
     max = b;
```

```
max = a > b ? a : b;
```

- wyznaczenie max z dwóch liczb

- Operator warunkowy ma bardzo niski priorytet
- Niższy priorytet mają tylko operatory przypisania (=, +=, -=,...) i operator przecinkowy (,)

Przykład: operator warunkowy

- Studenci chcą dojechać z akademika do sklepu - ile taksówek powinni zamówić? (Jedna taksówka może przewieźć 4 osoby.)

```
#include <stdio.h>

int main(void)
{
    int st, taxi;

    printf("Podaj liczbe studentow: ");
    scanf("%d",&st);

    taxi = st / 4 + (st % 4 ? 1 : 0);

    printf("Liczba taxi: %d\n",taxi);

    return 0;
}
```

```
Podaj liczbe studentow: 23
Liczba taxi: 6
```

Przykład: sprawdzenie parzystości liczby

```
#include <stdio.h>

int main(void)
{
   int x;

   printf("Podaj x: ");
   scanf("%d",&x);

   if (x%2==0)
       printf("Liczba parzysta\n");
   else
       printf("Liczba nieparzysta\n");

   printf("Liczba %s\n",x%2==0 ? "parzysta":"nieparzysta");

   return 0;
}
```

```
Podaj x: -3
Liczba nieparzysta
Liczba nieparzysta
```

Język C - Instrukcja switch

- Instrukcja wyboru wielowariantowego switch

```
switch (wyrażenie)
{
    case wyrażenie_stałe: instrukcje;
    case wyrażenie_stałe: instrukcje;
    case wyrażenie_stałe: instrukcje;
    ...
    default: instrukcje;
}
```

- wyrażenie_stałe - wartość typu całkowitego, znana podczas kompilacji
  - stała liczbowa, np. 3, 5, 9
  - znak w apostrofach, np. 'a', 'z', '+'
  - stała zdefiniowana przez const lub #define

Język C - Instrukcja switch

- Program wyświetlający słownie liczbę z zakresu 1..5 wprowadzoną z klawiatury

```
#include <stdio.h>

int main(void)
{
      int liczba;

      printf("Podaj liczbe (1..5): ");
      scanf("%d",&liczba);
```

Język C - Instrukcja switch

```
switch (liczba)
{
   case 1: printf("Liczba: jeden\n");
           break;
   case 2: printf("Liczba: dwa\n");
           break;
   case 3: printf("Liczba: trzy\n");
           break;
   case 4: printf("Liczba: cztery\n");
           break;
   case 5: printf("Liczba: piec\n");
           break;
   default: printf("Inna liczba\n");
}
```

```
Podaj liczbe: 2
Liczba: dwa
```

```
Podaj liczbe: 0
Inna liczba
```

Język C - Instrukcja switch

```
switch (liczba)
{
    case 1:
    case 3:
    case 5: printf("Liczba nieparzysta\n");
            break;
    case 2:
    case 4: printf("Liczba parzysta\n");
            break;
    default: printf("Inna liczba\n");
}
```

```
Podaj liczbe: 2
Liczba parzysta
```

- Te same instrukcje mogą być wykonane dla kilku etykiet case

Język C - Instrukcja switch

```
switch (liczba)
{
   case 1: case 3: case 5:
           printf("Liczba nieparzysta\n");
           break;
   case 2: case 4:
           printf("Liczba parzysta\n");
           break;
   default: printf("Inna liczba\n");
}
```

```
Podaj liczbe: 2
Liczba parzysta
```

- Etykiety case mogą być pisane w jednym wierszu

Język C - Instrukcja switch

```
switch (liczba%2)
{
    case 1: case -1:
            printf("Liczba nieparzysta\n");
            break;
    case 0:
            printf("Liczba parzysta\n");
}
```

```
Podaj liczbe: 2
Liczba parzysta
```

- Część domyślna (default) może być pominięta

Język C - Instrukcja switch (bez break)

```
switch (liczba)
{
   case 1: printf("Liczba: jeden\n");
   case 2: printf("Liczba: dwa\n");
   case 3: printf("Liczba: trzy\n");
   case 4: printf("Liczba: cztery\n");
   case 5: printf("Liczba: piec\n");
   default: printf("Inna liczba\n");
}
```

```
Podaj liczbe: 2
Liczba: dwa
Liczba: trzy
Liczba: cztery
Liczba: piec
Inna liczba
```

- Pominięcie instrukcji break spowoduje wykonanie wszystkich instrukcji występujących po danym case (do końca switch)

Przykład: suma kolejnych 10 liczb: 1+2+...+10

```
#include <stdio.h>

int main(void)
{
    int suma;

    suma = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;

    printf("Suma wynosi: %d\n",suma);

    return 0;
}
```

```
Suma wynosi: 55
```

Przykład: suma kolejnych 100 liczb: 1+2+...+100

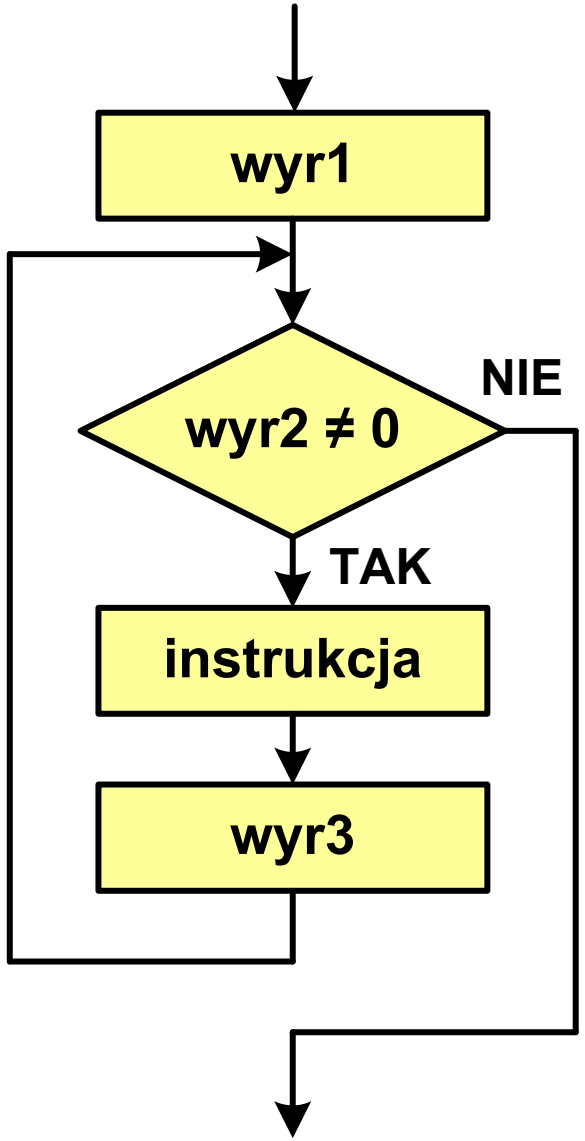
```
#include <stdio.h>

int main(void)
{
    int suma=0, i;

    for (i=1; i<=100; i=i+1)
        suma = suma + i;

    printf("Suma wynosi: %d\n",suma);

    return 0;
}
```

```
Suma wynosi: 5050
```

Język C - pętla for

```
for (wyr1; wyr2; wyr3)
    instrukcja
```

- wyr1, wyr2, wyr3 - dowolne wyrażenia w języku C
- Instrukcja:
  - prosta - jedna instrukcja zakończona średnikiem
  - złożona - jedna lub kilka instrukcji objętych nawiasami klamrowymi



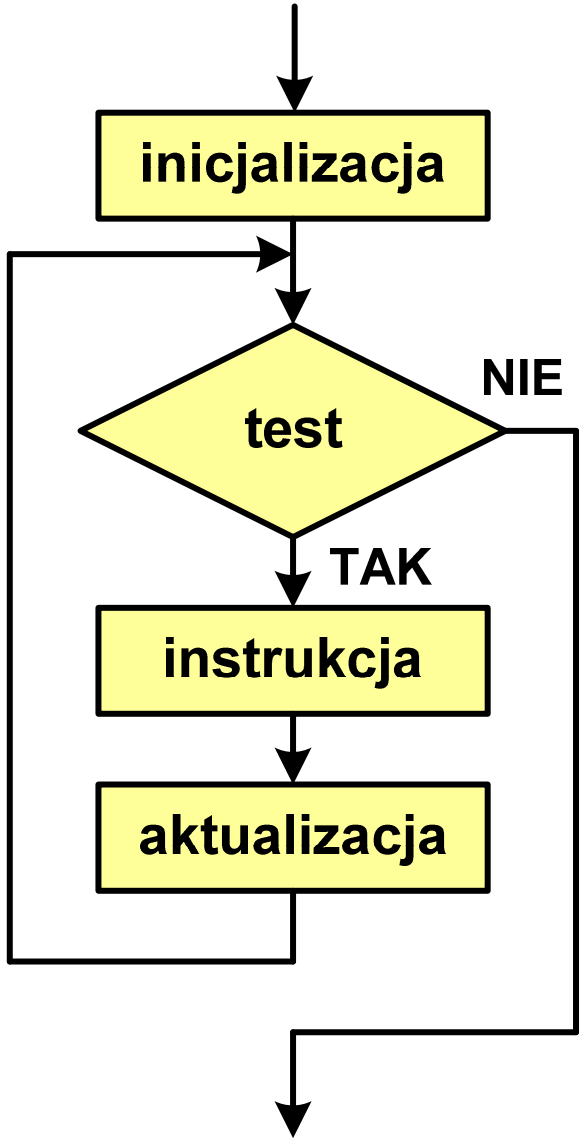
Język C - pętla for

- Najczęściej stosowana postać pętli for

```
int i;
for (i = 0; i < 10; i = i + 1)
    instrukcja
```

- Instrukcja zostanie wykonana 10 razy (dla i = 0, 1, 2, … 9)
- Funkcje pełnione przez wyrażenia

```
for (inicjalizacja;test;aktualizacja)
    instrukcja
```



Przykład: wyświetlenie tekstu 5 razy

```
#include <stdio.h>

int main(void)
{
    int i;

    for (i=0; i<5; i=i+1)
        printf("Programowanie nie jest trudne\n");

    return 0;
}
```

```
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
```

Przykład - suma liczb: 1 + 2 + ... + N

```
#include <stdio.h>
#define N 1234

int main(void)
{
    int i, suma=0;

    for (i=1; i<=N; i++)
        suma = suma + i;

    printf("Suma %d liczb to %d\n", N, suma);

    return 0;
}
```

```
Suma 1234 liczb to 761995
```

Język C - pętla for (przykłady)

```
for (i=0; i<10; i++)
    printf("%d ",i);
```

```
0 1 2 3 4 5 6 7 8 9
```

```
for (i=0; i<10; i++)
    printf("%d ",i+1);
```

```
1 2 3 4 5 6 7 8 9 10
```

```
for (i=1; i<=10; i++)
    printf("%d ",i);
```

```
1 2 3 4 5 6 7 8 9 10
```

Język C - pętla for (przykłady)

```
for (i=1; i<10; i=i+2)
    printf("%d ",i);
```

```
1 3 5 7 9
```

```
for (i=10; i>0; i--)
    printf("%d ",i);
```

```
10 9 8 7 6 5 4 3 2 1
```

```
for (i=-9; i<=9; i=i+3)
    printf("%d ",i);
```

```
-9 -6 -3 0 3 6 9
```

Język C - pętla for (break, continue)

- W pętli for można stosować instrukcje skoku: break i continue

```
int i;

for (i=1; i<10; i++)
{
    if (i%2==0)
        continue;
    if (i%7==0)
        break;
    printf("%d\n",i);
}
```

```
1 3 5
```

- continue przerywa bieżącą iterację i przechodzi do obliczania wyr3
- break przerywa wykonywanie pętli

Język C - pętla for (najczęstsze błędy)

- Postawienie średnika na końcu pętli for

```
int i;
for (i=0; i<10; i++);
printf("%d ",i);
```

```
10
```

- Przecinki zamiast średników pomiędzy wyrażeniami

```
int i;
for (i=0, i<10, i++)
    printf("%d ",i);
```

Błąd kompilacji!

Język C - pętla for (najczęstsze błędy)

- Błędny warunek - brak wykonania instrukcji

```
int i;
for (i=0; i>10; i++)
    printf("%d ",i);
```

- Błędny warunek - pętla nieskończona

```
int i;
for (i=1; i>0; i++)
    printf("%d ",i);
```

```
1 2 3 4 5 6 7 8 9 ...
```

Język C - pętla nieskończona

```
for (wyr1; wyr2; wyr3)
    instrukcja
```

- Wszystkie wyrażenia (wyr1, wyr2, wyr3) w pętli for są opcjonalne

```
for ( ; ; )
    instrukcja
```

- pętla nieskończona

- W przypadku braku wyr2 przyjmuje się, że jest ono prawdziwe

Język C - zagnieżdżanie pętli for

- Jako instrukcja w pętli for może występować kolejna pętla for

```
int i, j;
for (i=1; i<=3; i++)          // pętla zewnętrzna
    for (j=1; j<=2; j++)      // pętla wewnętrzna
        printf("i: %d   j: %d\n",i,j);
```

```
i: 1   j: 1
i: 1   j: 2
i: 2   j: 1
i: 2   j: 2
i: 3   j: 1
i: 3   j: 2
```

Język C - operator inkrementacji (++)

- Jednoargumentowy operator ++ zwiększa wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator ++ może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
`++x`	preinkrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
`x++`	postinkrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

Język C - operator inkrementacji (++)

- Przykład

```
int x = 1, y;
y = 2 * ++x;
```

```
int x = 1, y;
y = 2 * x++;
```

- Kolejność operacji

```
++x                 x = 2
2 * ++x             2 * 2
y = 2 * ++x         y = 4
```

```
2 * x               2 * 1
y = 2 * x           y = 2
x++                 x = 2
```

- Wartości zmiennych

```
x = 2     y = 4
```

```
x = 2     y = 2
```

Język C - operator inkrementacji (++)

- Miejsce umieszczenia operatora ++ nie ma znaczenia w przypadku instrukcji typu:

```
x++;
++x;
```

równoważne

```
x = x + 1;
```

- Nie należy stosować operatora ++ do zmiennych pojawiających się w wyrażeniu więcej niż jeden raz

```
x = x++;
x = ++x;
```

- Zgodnie ze standardem języka C wynik powyższych instrukcji jest niezdefiniowany

Język C - operator dekrementacji (--)

- Jednoargumentowy operator -- zmniejsza wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator -- może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
`--x`	predekrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
`x--`	postdekrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

Język C - priorytet operatorów ++ i --

Priorytet	Operator / opis
1	`++` `--` (przyrostki) `()` `[]` `.` `->`
2	`++` `--` (przedrostki) `sizeof` `(typ)` `+` `-` `!` `~` `*` `&` (jednoargumentowe)
3	`*` `/` `%`
4	`+` `-` (dwuargumentowe)
5	`<<` `>>`
6	`<` `>` `<=` `>=`
7	`==` `!=`
8	`&` (bitowy)
9	`^`

Język C - miesięczny kalendarz

- Napisz program wyświetlający miesięczny kalendarz. Wczytaj liczbę dni w miesiącu i dzień tygodnia, od którego zaczyna się miesiąc.
- Przykład działania programu:

```
Liczba dni w miesiącu: 31
Pierwszy dzien tygodnia (1-Pn, 2-Wt, ...): 4
---------------------------------------
 Pn Wt Sr Cz Pt So  N
            1  2  3  4
  5  6  7  8  9 10 11
 12 13 14 15 16 17 18
 19 20 21 22 23 24 25
 26 27 28 29 30 31
```

Język C - miesięczny kalendarz

```c
#include <stdio.h>

int main()
{
   int ile_dni, dzien_tyg, i;

   printf("Liczba dni w miesiacu: "); scanf("%d",&ile_dni);
   printf("Pierwszy dzien tygodnia (1-Pn, 2-Wt, ...): ");
   scanf("%d",&dzien_tyg);

   printf("-----------------------------------\n");
   printf(" Pn Wt Sr Cz Pt So  N\n");

   for (i=1; i<dzien_tyg; i++) printf("   ");
   for (i=0; i<ile_dni; i++)
   {
       printf("%3d",i+1);
       if ((i+dzien_tyg)%7==0) printf("\n");
   }
   printf("\n"); return 0;
}
```

Język C - miesięczny kalendarz

```
Liczba dni w miesiacu: 30
Pierwszy dzien tygodnia (1-Pn, 2-Wt, ...): 5
-----------------------------------------
 Pn Wt Sr Cz Pt So  N
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

```
#include <st

int main()
{
   int ile_d

   printf("L
   printf("P
   scanf("%d

   printf("-
   printf("

   for (i=1; i<dzien_tyg; i++) printf("    ");
   for (i=0; i<ile_dni; i++)
   {
       printf("%3d",i+1);
       if ((i+dzien_tyg)%7==0) printf("\n");
   }
   printf("\n"); return 0;
}
```

Przykład: pierwiastek kwadratowy

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float x, y;

    printf("Podaj liczbe: ");
    scanf("%f",&x);

    if (x>=0)
    {
        y = sqrt(x);
        printf("Pierwiastek liczby: %f\n",y);
    }
    else
        printf("Blad! Liczba ujemna\n");

    return 0;
}
```

```
Podaj liczbe: -3
Blad! Liczba ujemna
```

```
Podaj liczbe: 3
Pierwiastek liczby: 1.732051
```

Przykład: pierwiastek kwadratowy (pętla while)

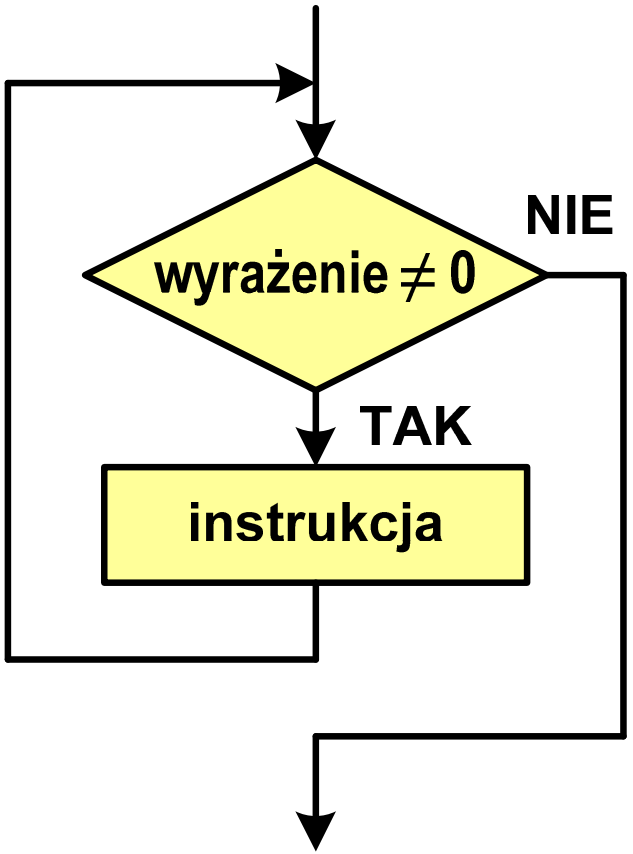
```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float x, y;

    printf("Podaj liczbe: ");
    scanf("%f",&x);
    while (x<0)
    {
        printf("Blad! Liczba ujemna\n\n");
        printf("Podaj liczbe: ");
        scanf("%f",&x);
    }
    y = sqrt(x);
    printf("Pierwiastek liczby: %f\n",y);

    return 0;
}
```

```
Podaj liczbe: -3
Blad! Liczba ujemna

Podaj liczbe: -5
Blad! Liczba ujemna

Podaj liczbe: 3
Pierwiastek liczby: 1.732051
```

Język C - pętla while

```
while (wyrażenie)
    instrukcja
```

- „dopóki wyrażenie w nawiasach jest prawdziwe wykonuj instrukcję”



- Wyrażenie w nawiasach:
  - prawdziwe - gdy jego wartość jest różna od zera
  - fałszywe - gdy jego wartość jest równa zero
- Jako wyrażenie najczęściej stosowane jest wyrażenie logiczne

Język C - pętla while

```
while (wyrażenie)
    instrukcja
```

- Instrukcja:
  - prosta - jedna instrukcja zakończona średnikiem
  - złożona - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
int x = 10;
while (x>0)
    x = x - 1;
```

```
int x = 10;
while (x>0)
{
    printf("%d\n",x);
    x = x - 1;
}
```

Przykład: suma liczb dodatnich

```c
#include <stdio.h>

int main(void)
{
    int x, suma = 0;

    printf("Podaj liczbe: ");
    scanf("%d",&x);

    while(x>0)
    {
        suma = suma + x;
        printf("Podaj liczbe: ");
        scanf("%d",&x);
    }
    printf("Suma liczb: %d\n",suma);

    return 0;
}
```

```
Podaj liczbe: 4
Podaj liczbe: 8
Podaj liczbe: 2
Podaj liczbe: 3
Podaj liczbe: 5
Podaj liczbe: -2
Suma liczb: 22
```

Język C - pętla while

- Program pokazany na poprzednim slajdzie zawiera typowy schemat przetwarzania danych z wykorzystaniem pętli while

```
printf("Podaj liczbe: ");
scanf("%d",&x);

while(x>0)
{
    suma = suma + x;

    printf("Podaj liczbe: ");
    scanf("%d",&x);
}
```

wczytanie danych

operacje na danych

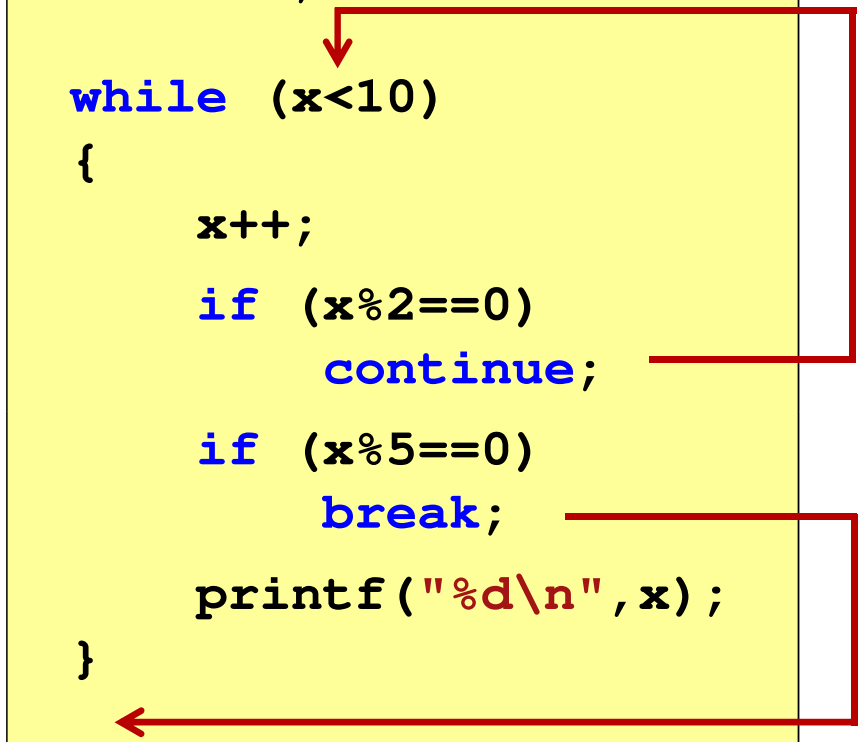
wczytanie danych

- Dane mogą być wczytywane z klawiatury, pliku, itp.

Język C - pętla while (break, continue)

- break i continue są to instrukcje skoku

```
int x=0;

while (x<10)
{
    x++;
    if (x%2==0)
        continue;
    if (x%5==0)
        break;
    printf("%d\n",x);
}
```



- continue przerywa bieżącą iterację
- break przerywa wykonywanie pętli

Język C - pętla while (najczęstsze błędy)

- Postawienie średnika po wyrażeniu w nawiasach powoduje powstanie pętli nieskończonej - program zatrzymuje się na pętli

```
int x = 10;
while (x>0);
    printf("%d ",x--);
```

- Brak aktualizacji zmiennej powoduje także powstanie pętli nieskończonej - program wyświetla wielokrotnie tę samą wartość

```
int x = 10;
while (x>0)
    printf("%d ",x);
```

```
10 10 10 10 10 ...
```

Język C - pętla while (pętla nieskończona)

- W pewnych sytuacjach celowo stosuje się pętlę nieskończoną (np. w mikrokontrolerach)

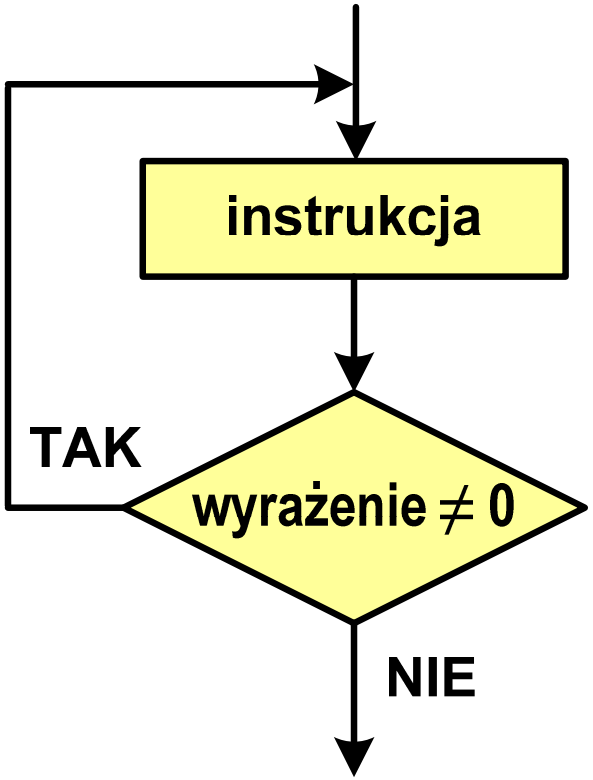
```
while (1)
{
     instrukcja
     instrukcja
     ...
}
```

- W układach mikroprocesorowych program działa aż do wyłączenia zasilania

Język C - pętla do ... while

```
do
    instrukcja
while (wyrażenie);
```

- „wykonuj instrukcję dopóki wyrażenie w nawiasach jest prawdziwe”



- Wyrażenie w nawiasach:
  - prawdziwe - gdy jego wartość jest różna od zera
  - fałszywe - gdy jego wartość jest równa zero

Język C - pętla do ... while

```
do
    instrukcja
while (wyrażenie);
```

- Instrukcja:
  - prosta - jedna instrukcja zakończona średnikiem
  - złożona - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
int x = 10;
do
    x = x - 1;
while (x>0);
```

```
int x = 10;
do
{
    printf("%d\n",x);
    x = x - 1;
}
while (x>0);
```

Język C - pętla do ... while (break, continue)

- break i continue są to instrukcje skoku

```
int x=0;

do
{
    x++;
    if (x%5==0)
        break;
    if (x%2==0)
        continue;
    printf("%d\n",x);
}
while (i<10);
```

- break przerywa wykonywanie pętli
- continue przerywa bieżącą iterację

Przykład: suma liczb < 100

```c
#include <stdio.h>

int main(void)
{
    int x, suma = 0;

    do
    {
        printf("Podaj liczbe: ");
        scanf("%d",&x);
        suma = suma + x;
    }
    while(suma<100);

    printf("Suma liczb: %d\n",suma);

    return 0;
}
```

```
Podaj liczbe: 34
Podaj liczbe: 9
Podaj liczbe: 26
Podaj liczbe: -8
Podaj liczbe: 67
Suma liczb: 128
```

Informatyka

- Informatyka (ang. computer science)
  - dziedzina nauki i techniki zajmująca się gromadzeniem, przetwarzaniem i wykorzystywaniem informacji
  - w języku polskim termin informatyka zaproponował w październiku 1968 r. prof. Romuald Marczyński na konferencji poświęconej „maszynom matematycznym”
  - wzorem nazwy były francuskie informatique i niemieckie Informatik

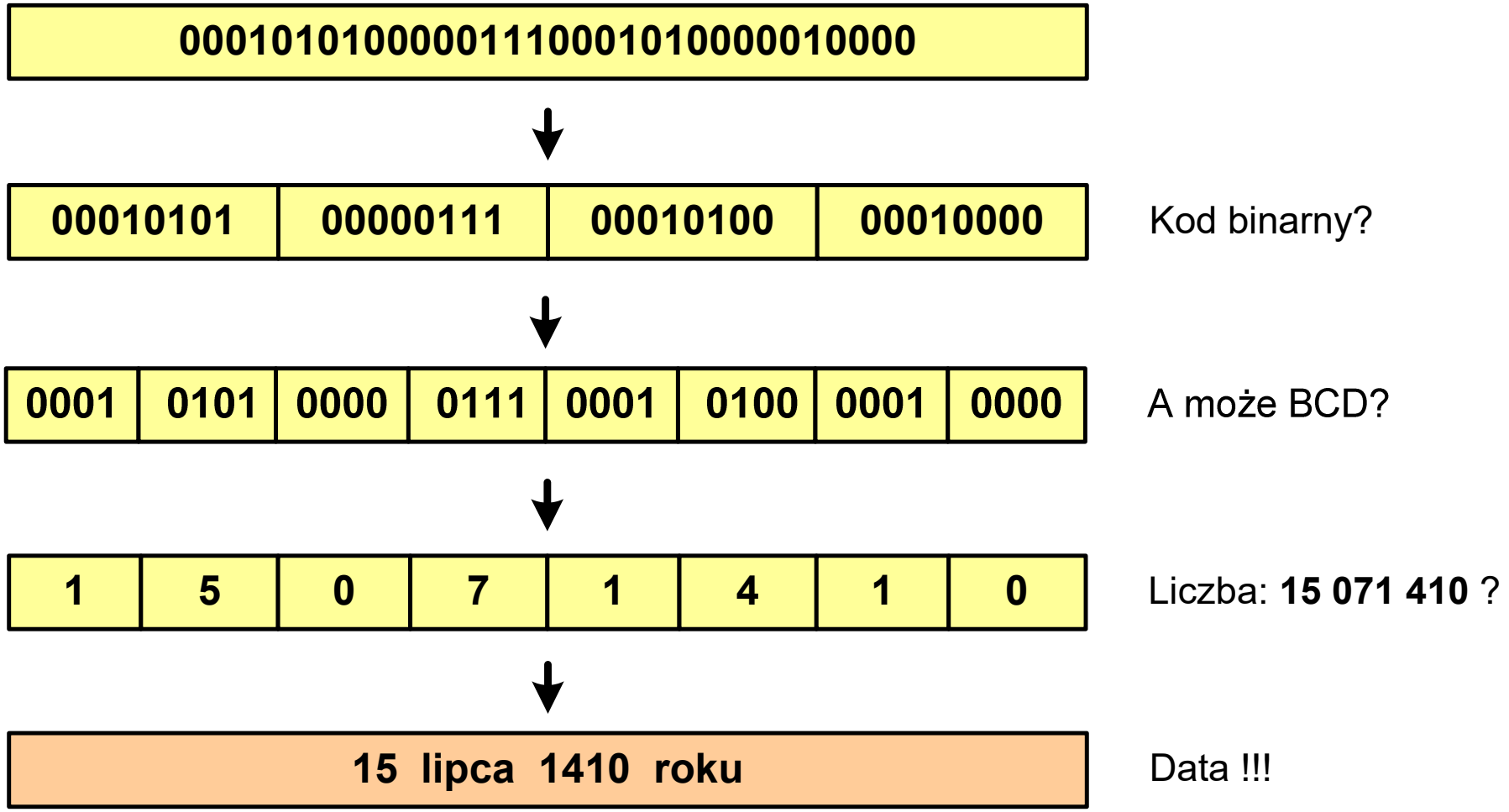
- Informatykę można rozpatrywać jako:
  - samodzielną dyscyplinę naukową
  - narzędzie wykorzystywane przez inne nauki
  - gałąź techniki
  - przemysł wytwarzający sprzęt (hardware) i oprogramowanie (software)

Informacja

- Informatyka (ang. computer science)
  - dziedzina nauki i techniki zajmująca się gromadzeniem, przetwarzaniem i wykorzystywaniem informacji
- Informacja - wielkość abstrakcyjna, która może być:
  - przechowywana w pewnych obiektach
  - przesyłana pomiędzy pewnymi obiektami
  - przetwarzana w pewnych obiektach
  - stosowana do sterowania pewnymi obiektami
- Dane - surowe fakty i liczby
- Przetwarzanie danych - logicznie powiązany zespół czynności pozwalających na uzyskanie z danych niezbędnych informacji

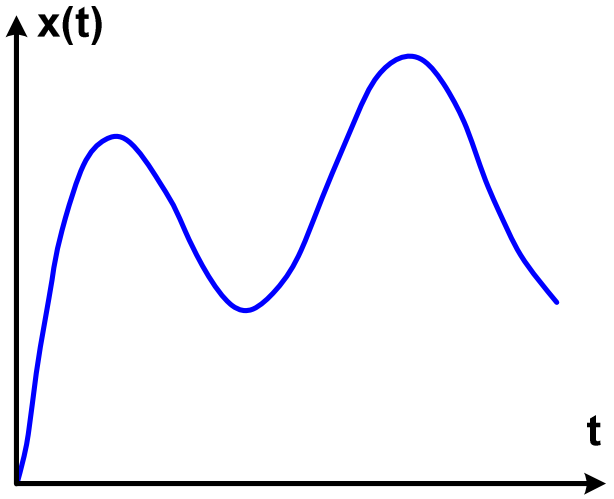
Informacja

- Co oznaczają poniższe dane?

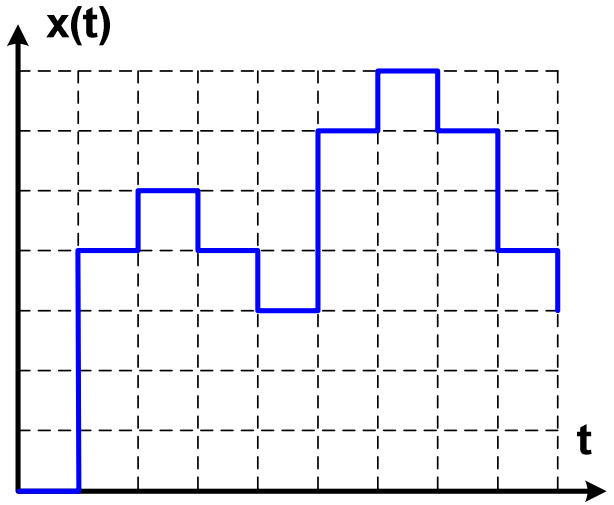


Informacja analogowa i cyfrowa

- Sygnał analogowy
  - może przyjmować dowolną wartość z ciągłego przedziału (nieskończonego lub ograniczonego zakresem zmienności)
  - wartości mogą zostać określone w każdej chwili czasu dzięki funkcji matematycznej opisującej dany sygnał



- Sygnał cyfrowy
  - dziedzina i zbiór wartości są dyskretne
  - sygnał ciągły, który może zmieniać swoją wartość tylko w określonych chwilach czasu i może przyjmować tylko określone wartości



Informacja analogowa i cyfrowa

- Zalety sygnałów cyfrowych:
  - odporne na zakłócenia
  - powtarzalne (np. kopia filmu na DVD i VHS)
  - możliwość przesyłania na duże odległości
  - możliwość szyfrowania sygnału (kryptografia)
  - niższe koszty przetwarzania
- Wady sygnałów cyfrowych:
  - ograniczenie częstotliwości próbkowania (sygnał analogowy zamieniony na cyfrowy i ponownie na analogowy nie jest już tym samym sygnałem)

Liczby i cyfry

- Liczba - pojęcie abstrakcyjne, abstrakcyjny wynik obliczeń, wartość
  - umożliwia wyrażenie wyniku liczenia przedmiotów oraz mierzenia wielkości
- Cyfra - umowny znak (symbol) stosowany do zapisu liczby
  - liczba znaków służących do zapisu jest zależna od systemu liczbowego i przyjętego sposobu zapisu
  - system dziesiętny - 10 znaków
  - system szesnastkowy - 16 znaków
  - system rzymski - 7 znaków
- Cyfry rzymskie

I	V	X	L	C	D	M
1	*5*	*10*	*50*	*100*	*500*	*1000*

Liczby i cyfry

- Cyfry arabskie (pochodzą z Indii)
  - arabskie, standardowe europejskie

1	2	3	4	5	6	7	8	9	0

  - indyjsko-arabskie

١	٢	٣	٤	٥	٦	٧	٨	٩	٠
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*0*

  - wschodnio-indyjsko-arabskie

١	٢	٣	۴	۵	۶	٧	٨	٩	٠
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*0*

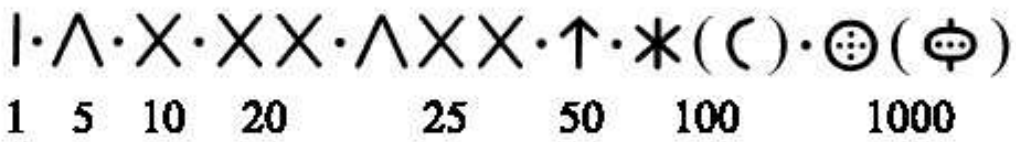
- W niektórych systemach jako cyfry stosowane są litery, np.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*

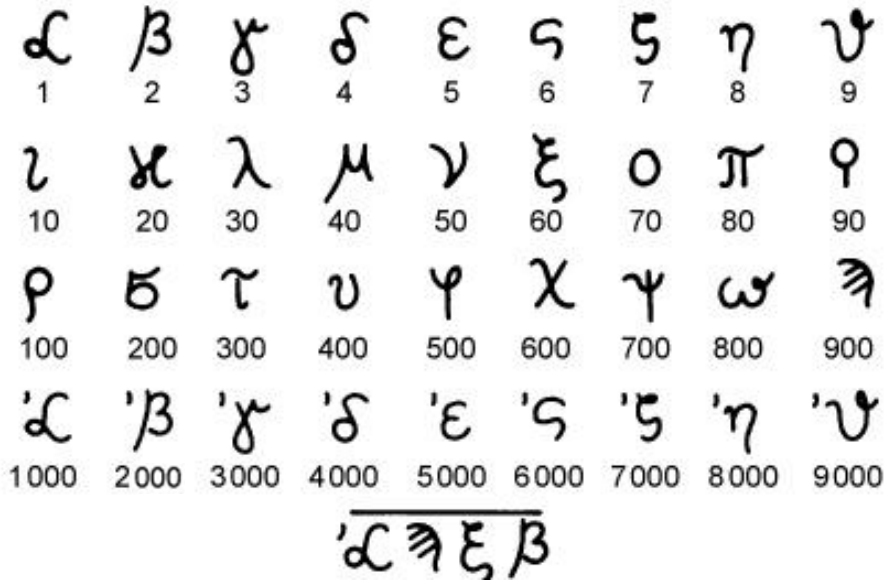
Liczby i cyfry

- Inne przykłady zapisu cyfr i liczb:

cyfry etruskie



cyfry grecko-jońskie

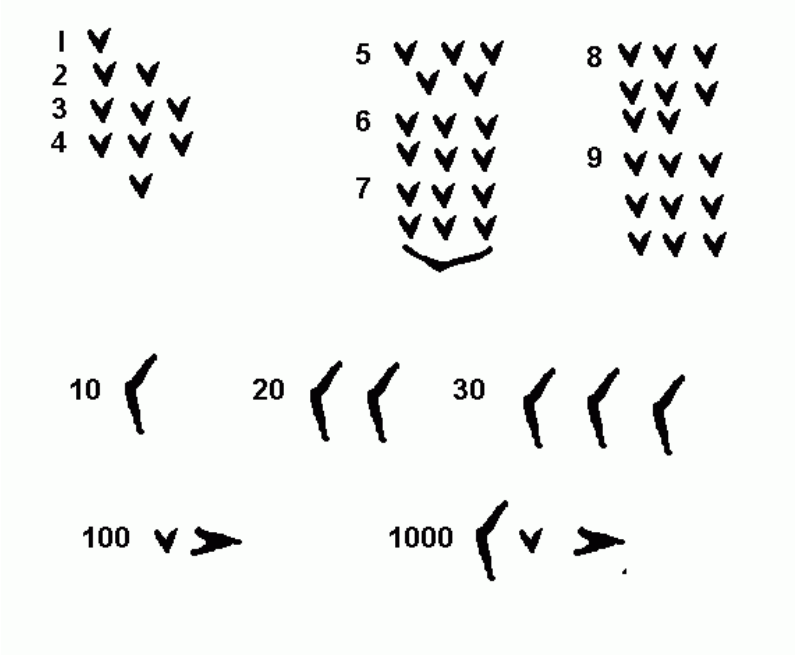


cyfry w pisowni chińskiej

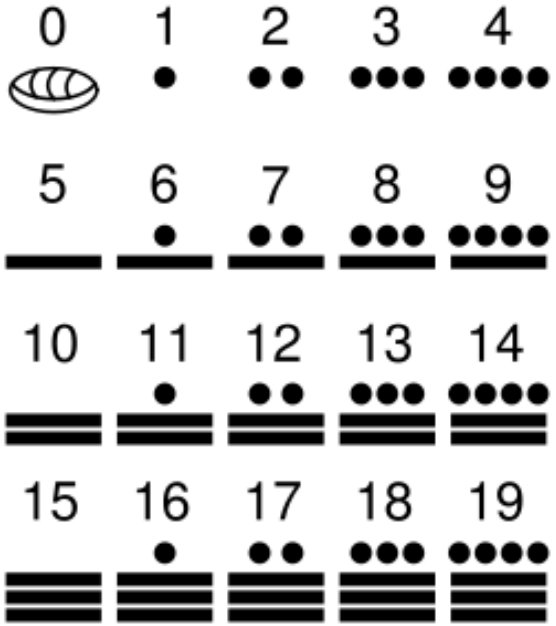
Liczby i cyfry

- Inne przykłady zapisu cyfr i liczb:

liczby w piśmie klinowym (Babilończycy)

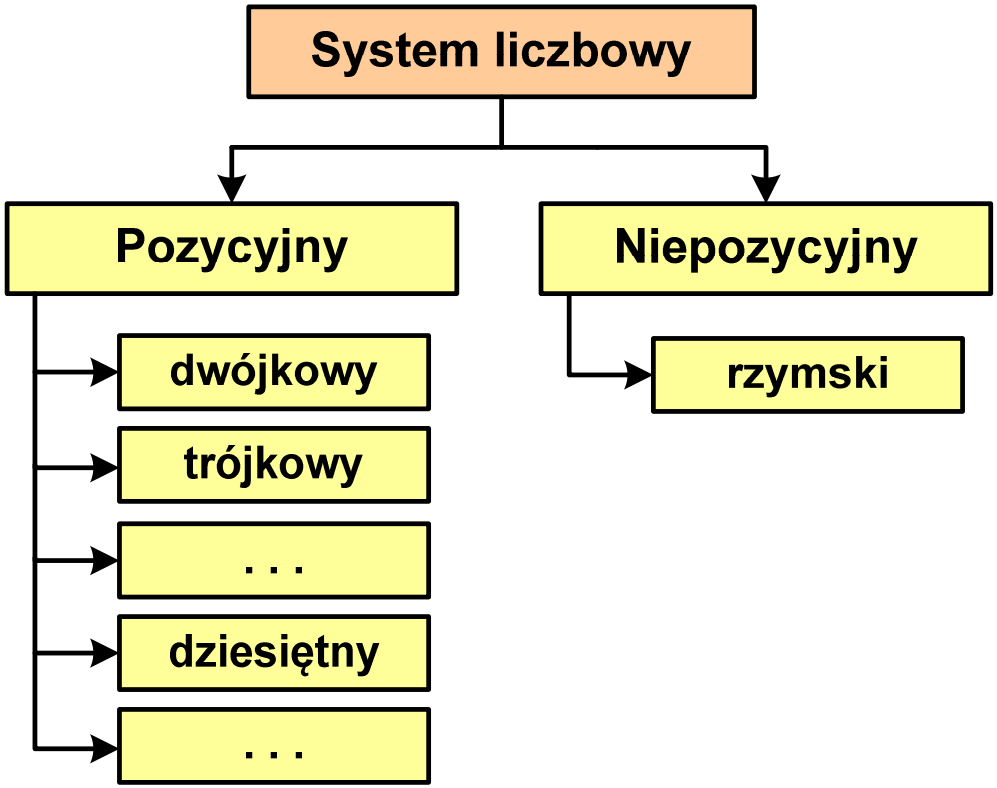


system prekolumbijski



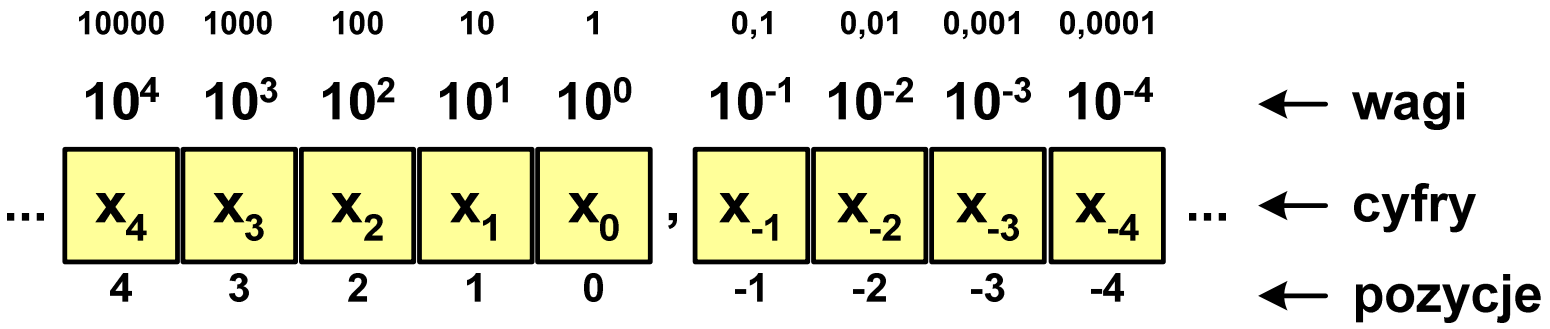
Systemy liczbowe

- System liczbowy - zbiór zasad umożliwiających zapis liczb za pomocą cyfr oraz wykonywanie działań na tych liczbach

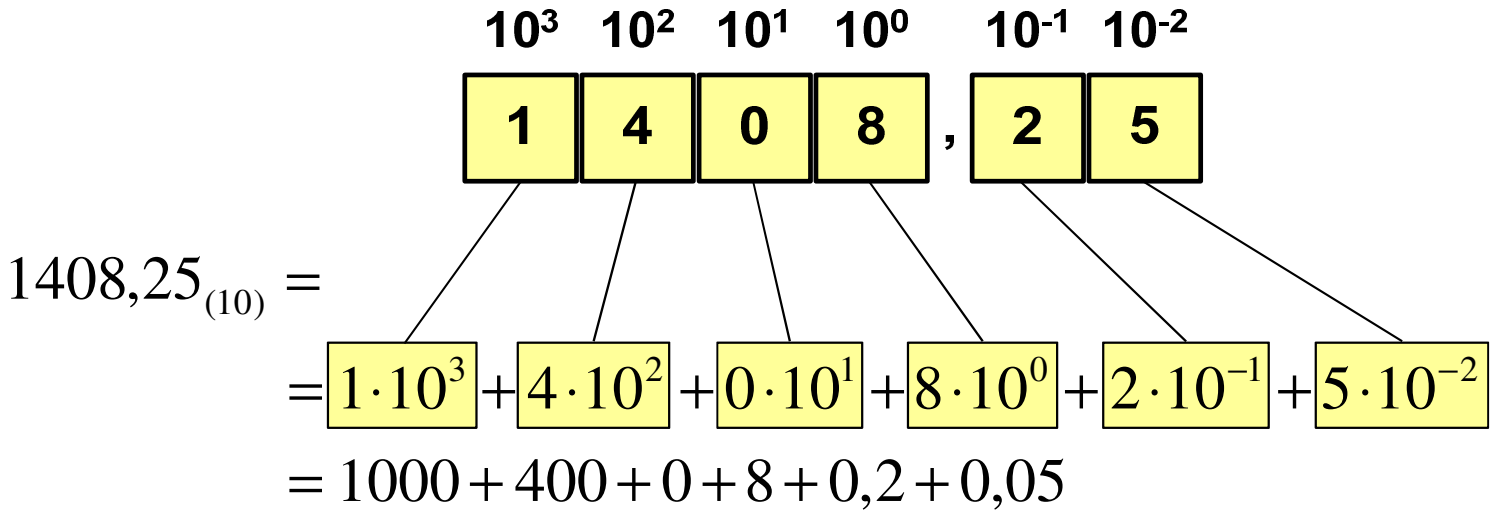


- Pozycyjny - znaczenie cyfry jest zależne od miejsca (pozycji), które zajmuje ona w liczbie
  - system dziesiętny - liczba 111 (każda cyfra ma inne znaczenie)
- Niepozycyjny - znaczenie cyfry jest niezależne od miejsca położenia w liczbie
  - system rzymski - liczba III

System dziesiętny (ang. decimal)



- p - podstawa systemu pozycyjnego, D - zbiór dozwolonych cyfr
- p = 10, D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}



Koniec wykładu nr 3

Dziękuję za uwagę!