Informatyka 1 (EZ1F1002)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr I, studia niestacjonarne I stopnia

Rok akademicki 2024/2025

Wykład nr 4 (08.11.2024)

dr inż. Jarosław Forenc

Plan wykładu nr 4

- Język C
  - instrukcja if
  - operatory relacyjne i logiczne, wyrażenia logiczne
  - operator warunkowy
  - instrukcja switch
- Algorytmy komputerowe
  - definicje, sposoby opisu
  - rekurencja, złożoność obliczeniowa
  - algorytmy sortowania

Przykład: pierwiastek kwadratowy

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

int main(void)
{
    float x, y;

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

    y = sqrt(x);

    printf("Pierwiastek liczby: %f\n",y);

    return 0;
}
```

```
Podaj liczbe: 15
Pierwiastek liczby: 3.872983
```

Język C - Typy i operatory

Nazwa	Rozmiar (bajty)	Zakres wartości
`char`	1	-128 ... 127
`int`	4	-2147483648 ... 2147483647
`float`	4	$-3{,}4 \cdot 10^{38}$ ... $3{,}4 \cdot 10^{38}$
`double`	8	$-1{,}7 \cdot 10^{308}$ ... $1{,}7 \cdot 10^{308}$

Typ	Symbol
Arytmetyczne	`+ - * / %`
Inkrementacji / dekrementacji	`++ --`
Przypisania	`=`
Inne	`() , (typ)`

Język C - Specyfikatory formatu (printf)

Typ w C	Specyfikator	Uwagi
`char`	`%c`	pojedynczy znak
	`%d`	kod ASCII znaku, liczba całkowita
`char *`	`%s`	łańcuch znaków, napis
`int`	`%d %i`	liczba całkowita, dziesiętna
	`%o %O`	liczba całkowita, ósemkowa
	`%x %X`	liczba całkowita, szesnastkowa
`float` `double`	`%f`	liczba rzeczywista
	`%e %E`	liczba rzeczywista, format naukowy
	`%g %G`	liczba rzeczywista (%f lub %e)

Język C - Funkcja scanf()

- Specyfikatory formatu w większości przypadków są takie same jak w przypadku funkcji printf()
- Największa różnica dotyczy typów float i double

Typ w C	Specyfikator	Uwagi
`float`	`%f`	liczba rzeczywista
	`%e` `%E`	liczba rzeczywista, format naukowy
	`%g` `%G`	liczba rzeczywista (%f lub %e)
`double`	`%lf`	liczba rzeczywista
	`%le` `%lE`	liczba rzeczywista, format naukowy
	`%lg` `%lG`	liczba rzeczywista (%f lub %e)

Przykład: pierwiastek kwadratowy

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

int main(void)
{
    float x, y;

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

    y = sqrt(x);

    printf("Pierwiastek liczby: %f\n",y);

    return 0;
}
```

```
Podaj liczbe: 15
Pierwiastek liczby: 3.872983
```

```
Podaj liczbe: -15
Pierwiastek liczby: -1.#IND00
```

Przykład: pierwiastek kwadratowy

```c
#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: 15
Pierwiastek liczby: 3.872983
```

```
Podaj liczbe: -15
Blad! Liczba ujemna
```

Język C - instrukcja warunkowa if

```
if (wyrażenie)
    instrukcja1;
```

- jeśli wyrażenie jest prawdziwe, to wykonywana jest instrukcja1
- gdy wyrażenie jest fałszywe, to instrukcja1 nie jest wykonywana

```
if (wyrażenie)
    instrukcja1;
else
    instrukcja2;
```

- jeśli wyrażenie jest prawdziwe, to wykonywana jest instrukcja1, zaś instrukcja2 nie jest wykonywana
- gdy wyrażenie jest fałszywe, to wykonywana jest instrukcja2, zaś instrukcja1 nie jest wykonywana

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

Język C - instrukcja warunkowa if

```
if (wyrażenie)
    instrukcja;
```

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

```
if (x>0)
    printf("inst1");
```

```
if (x>0)
{
    printf("inst1");
    printf("inst2");
    ...
}
```

Język C - instrukcja warunkowa if

```
if (wyr)
    instr;
```

```
if (wyr)
{
    instr;
    instr;
}
```

```
if (wyr)
    instr;
else
    instr;
```

```
if (wyr)
{
    instr;
    instr;
}
else
{
    instr;
    instr;
}
```

```
if (wyr)
{
    instr;
    instr;
}
else
    instr;
```

```
if (wyr)
    instr;
else
{
    instr;
    instr;
}
```

```
if (wyr)
{
    instr;
}
else
{
    instr;
}
```

Język C - Operatory relacyjne (porównania)

Operator	Przykład	Znaczenie
`>`	`a > b`	a większe od b
`<`	`a < b`	a mniejsze od b
`>=`	`a >= b`	a większe lub równe b
`<=`	`a <= b`	a mniejsze lub równe b
`==`	`a == b`	a równe b
`!=`	`a != b`	a nierówne b (a różne od b)

- Wynik porównania jest wartością typu int i jest równy:
  - 1 - gdy warunek jest prawdziwy
  - 0 - gdy warunek jest fałszywy (nie jest prawdziwy)

Język C - Operatory logiczne

Operator	Znaczenie	Opis
`!`	`NOT, nie`	jednoargumentowy operator negacji logicznej - zmienia argument różny od zera na wartość 0, a argument równy zero na wartość 1
`&&`	`AND, i`	dwuargumentowy operator koniunkcji, iloczyn logiczny
`||`	`OR, lub`	dwuargumentowy operator alternatywy, suma logiczna

- Wynikiem zastosowania operatorów logicznych && i || jest wartość typu int równa 1 (prawda) lub 0 (fałsz)

```
if (x>5 && x<8)
```

```
if (x<=5 || x>8)
```

Język C - Wyrażenia logiczne

- Wyrażenia logiczne mogą zawierać:
  - operatory relacyjne
  - operatory logiczne
  - operatory arytmetyczne
  - operatory przypisania
  - zmienne
  - stałe
  - wywołania funkcji
  - …
- Kolejność operacji wynika z priorytetu operatorów

Operator	Typ operatora
`!`	logiczny
`* / %`	arytmetyczne
`+ -`	arytmetyczne
`> < >= <=`	relacyjne
`== !=`	relacyjne
`&&`	logiczny
`||`	logiczny
`=`	przypisania

Język C - Wyrażenia logiczne

```
int  x = 0, y = 1, z = 2;
```

```
if ( x == 0 )
```

wynik: 1 (prawda)

```
if ( x = 0 )
```

wynik: 0 (fałsz) (!!!)

```
if ( x != 0 )
```

wynik: 0 (fałsz)

```
if ( x =! 0 )
```

wynik: 1 (prawda) (!!!)

```
if ( z > x + y )
```

```
if ( z > (x + y) )
```

wynik: 1 (prawda)

Język C - Wyrażenia logiczne

```
int  x = 0, y = 1, z = 2;
```

```
if ( x>2 && x<5 )
```

```
if ( (x>2) && (x<5) )
```

wynik: 0 (fałsz)

- Wyrażenia logiczne obliczane są od strony lewej do prawej
- Proces obliczeń kończy się, gdy wiadomo, jaki będzie wynik całego wyrażenia

```
if ( 2 < x < 5 )
```

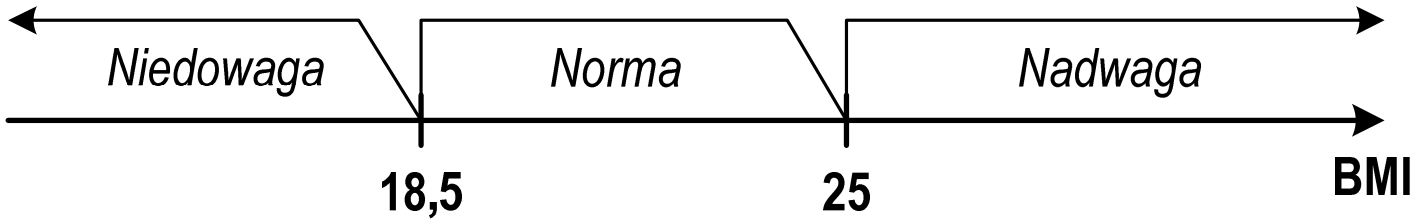
wynik: 1 (prawda) (!!!)

Przykład: obliczanie BMI (Body Mass Index)

- BMI - współczynnik powstały przez podzielenie masy ciała podanej w kilogramach przez kwadrat wzrostu podanego w metrach

$$BMI = \frac{masa}{wzrost^2}$$

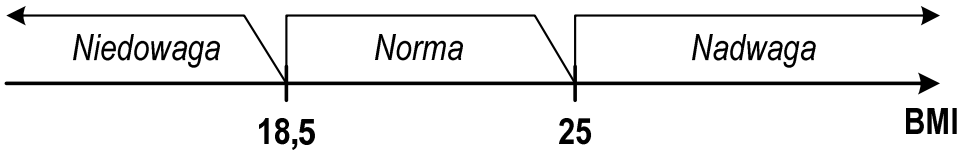
- Dla osób dorosłych:
  - BMI < 18,5 - wskazuje na niedowagę
  - BMI ≥ 18,5 i BMI < 25 - wskazuje na prawidłową masę ciała
  - BMI ≥ 25 - wskazuje na nadwagę



Przykład: obliczanie BMI (Body Mass Index)

```
#include <stdio.h>

int main(void)
{
    double masa, wzrost, bmi;

    printf("Podaj mase [kg]:  "); scanf("%lf",&masa);
    printf("Podaj wzrost [m]: "); scanf("%lf",&wzrost);
    bmi = masa / (wzrost*wzrost);
    printf("bmi: %.2f\n",bmi);

    if (bmi<18.5)
        printf("Niedowaga\n");
    if (bmi>=18.5 && bmi<25)
        printf("Norma\n");
    if (bmi>=25)
        printf("Nadwaga\n");

    return 0;
}
```

```
Podaj mase [kg]:  84
Podaj wzrost [m]: 1.85
bmi: 24.54
Norma
```



Przykład: obliczanie BMI (Body Mass Index)

- Zamiast trzech instrukcji if:

```
if (bmi<18.5)
    printf("Niedowaga\n");
if (bmi>=18.5 && bmi<25)
    printf("Norma\n");
if (bmi>=25)
    printf("Nadwaga\n");
```

można zastosować tylko dwie:

```
if (bmi<18.5)
    printf("Niedowaga\n");
else
    if (bmi<25)
        printf("Norma\n");
    else
        printf("Nadwaga\n");
```

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 != 0 ? 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)

Algorytm - definicje

Definicja 1

- Skończony, uporządkowany ciąg jasno zdefiniowanych czynności, koniecznych do wykonania pewnego zadania

Definicja 2

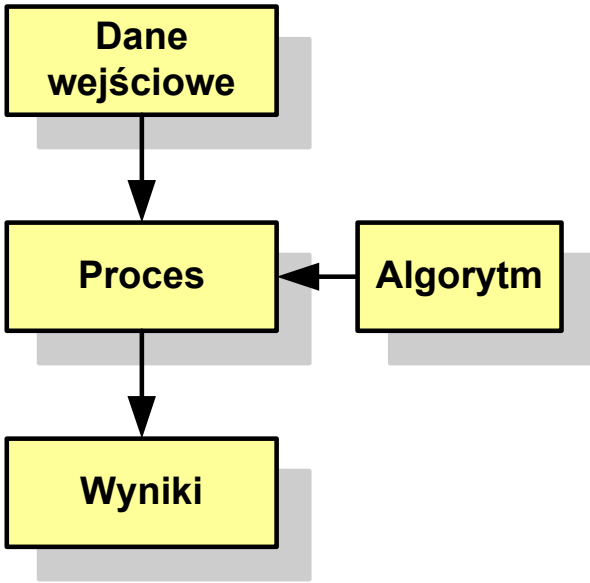
- Opis rozwiązania problemu wyrażony za pomocą operacji zrozumiałych i możliwych do zrealizowania przez wykonawcę

Definicja 3

- Ściśle określona procedura obliczeniowa, która dla właściwych danych wejściowych zwraca żądane dane wyjściowe zwane wynikiem działania algorytmu

Definicja 4

- Metoda rozwiązania zadania



Algorytmy

- Słowo „algorytm” pochodzi od nazwiska matematyka perskiego z IX wieku - Muhammada ibn-Musy al-Chuwarizmiego (po łacinie pisanego jako Algorismus)
- Badaniem algorytmów zajmuje się algorytmika
- „Przetłumaczenie” algorytmu na wybrany język programowania:
  - implementacja algorytmu
  - kodowanie algorytmu
- Sposoby opisu algorytmów
  - opis słowny w języku naturalnym lub lista kroków (opis w punktach)
  - schemat blokowy
  - pseudokod (nieformalna odmiana języka programowania)
  - wybrany język programowania

Opis słowny algorytmu

- Podanie kolejnych czynności, które należy wykonać, aby otrzymać oczekiwany efekt końcowy
- Przypomina przepis kulinarny z książki kucharskiej lub instrukcję obsługi urządzenia, np.

Algorytm:	Tortilla („Podróże kulinarne” R. Makłowicza)
Dane wejściowe:	0,5 kg ziemniaków, 100 g kiełbasy Chorizo, 8 jajek
Dane wyjściowe:	gotowa Tortilla
Opis algorytmu:	Ziemniaki obrać i pokroić w plasterki. Kiełbasę pokroić w plasterki. Ziemniaki wrzucić na gorącą oliwę na patelni i przyrumienić z obu stron. Kiełbasę wrzucić na gorącą oliwę na patelni i przyrumienić z obu stron. Ubić jajka i dodać do połączonych ziemniaków i kiełbasy. Dodać sól i pieprz. Usmażyć z obu stron wielki omlet nadziewany chipsami ziemniaczanymi z kiełbaską.

Lista kroków

- Uporządkowany opis wszystkich czynności, jakie należy wykonać podczas realizacji algorytmu
- Krok jest to pojedyncza czynność realizowana w algorytmie
- Kroki w algorytmie są numerowane, operacje wykonywane są zgodnie z rosnącą numeracją kroków
- Jedynym odstępstwem od powyższej reguły są operacje skoku (warunkowe lub bezwarunkowe), w których jawnie określa się numer kolejnego kroku
- Przykład (instrukcja otwierania wózka-specerówki):

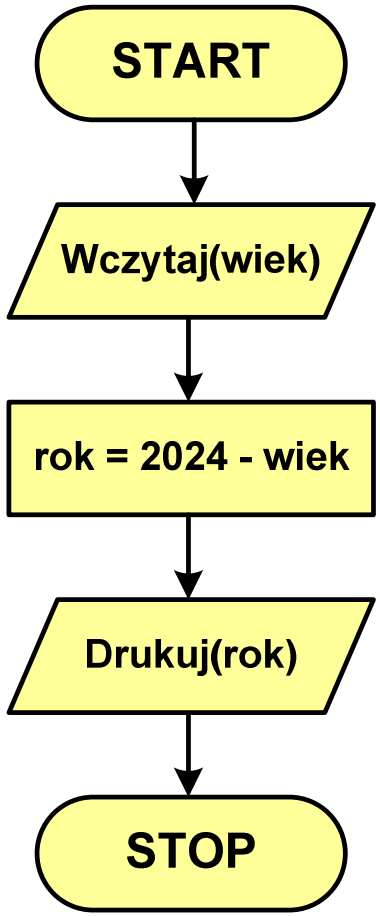
  Krok 1: Zwolnij element blokujący wózek

  Krok 2: Rozkładaj wózek w kierunku kółek

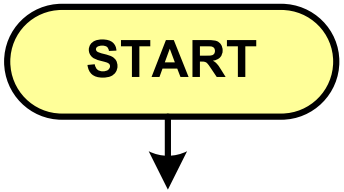
  Krok 3: Naciskając nogą dolny element blokujący aż do zatrzaśnięcia, rozłóż wózek do pozycji przewozowej

Schemat blokowy

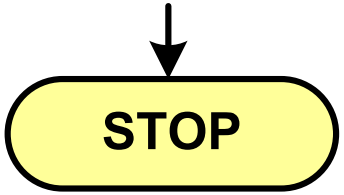
- Zawiera plan algorytmu przedstawiony w postaci graficznej
- Na schemacie umieszczane są bloki oraz linie przepływu (strzałki)
- Blok zawiera informację o wykonywanej operacji
- Linie przepływu (strzałki) określają kolejność wykonywania bloków algorytmu
- Przykład: wyznaczanie roku urodzenia na podstawie wieku (algorytm liniowy)



Schemat blokowy - symbole graficzne

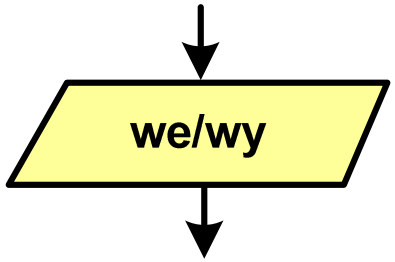


- blok startowy, początek algorytmu
- wskazuje miejsce rozpoczęcia algorytmu
- ma jedno wyjście
- może występować tylko jeden raz

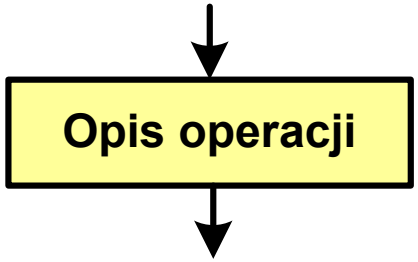


- blok końcowy, koniec algorytmu
- wskazuje miejsce zakończenia algorytmu
- ma jedno wejście
- musi występować przynajmniej jeden raz

Schemat blokowy - symbole graficzne

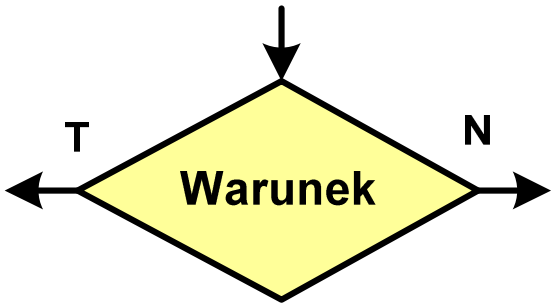


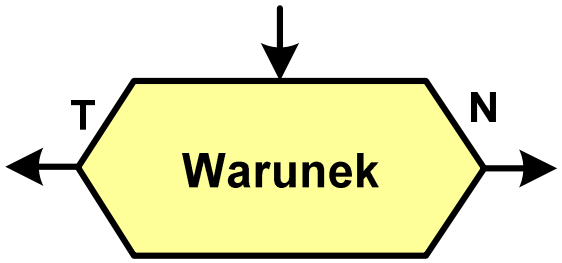
- blok wejścia-wyjścia
- poprzez ten blok wprowadzane są (czytane) dane wejściowe i wyprowadzane (zapisywane) wyniki
- ma jedno wejście i jedno wyjście



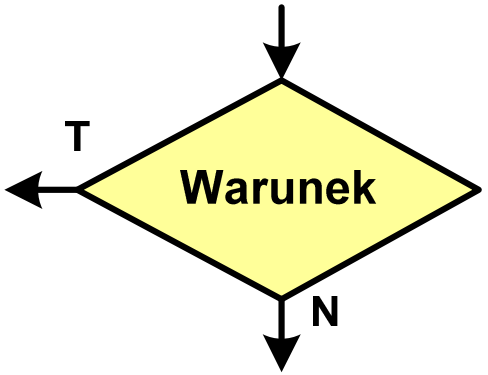
- blok wykonawczy, blok funkcyjny, opis procesu
- zawiera jedno lub kilka poleceń (elementarnych instrukcji) wykonywanych w podanej kolejności
- instrukcją może być np. operacja arytmetyczna, podstawienie
- ma jedno wejście i jedno wyjście

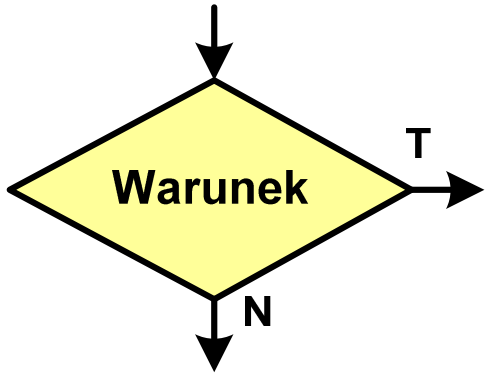
Schemat blokowy - symbole graficzne



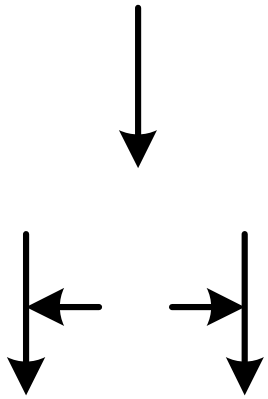


- blok warunkowy (decyzyjny, porównujący)
- wewnątrz bloku umieszcza się warunek logiczny
- na podstawie warunku określana jest tylko jedna droga wyjściowa
- połączenia wychodzące z bloku:
  - T lub TAK - gdy warunek jest prawdziwy
  - N lub NIE - gdy warunek nie jest prawdziwy
- wyjścia mogą być skierowane na boki lub w dół

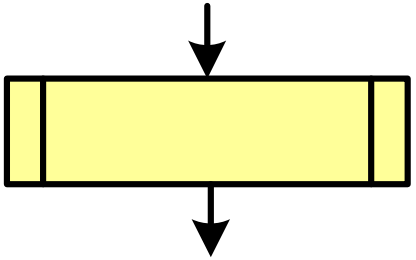




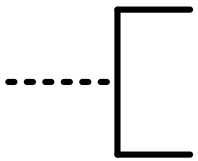
Schemat blokowy - symbole graficzne



- linia przepływu, połączenie, linia
- występuje w postaci linii zakończonej strzałką
- określa kierunek przemieszczania się po schemacie
- linie pochodzące z różnych części algorytmu mogą zbiegać się w jednym miejscu



- podprogram
- wywołanie wcześniej zdefiniowanego fragmentu algorytmu (podprogramu)
- ma jedno wejście i jedno wyjście



- komentarz
- dodanie do schematu dodatkowego opisu

Pseudokod i język programowania

Pseudokod:

- Pseudokod (pseudojęzyk) - uproszczona wersja języka programowania
- Często zawiera zwroty pochodzące z języków programowania
- Zapis w pseudokodzie może być łatwo przetłumaczony na wybrany język programowania

Opis w języku programowania:

- Zapis programu w konkretnym języku programowania
- Stosowane języki: Pascal, C, C++, Matlab, Python
  (kiedyś - Fortran, Basic)

Największy wspólny dzielnik - algorytm Euklidesa

- NWD - największa liczba naturalna dzieląca (bez reszty) dwie (lub więcej) liczby całkowite

NWD(1675,3752) = ?

Algorytm Euklidesa - przykład

a	b	Dzielenie większej liczby przez mniejszą	Zamiana
1675	3752	b/a = 3752/1675 = 2 reszta 402	b = 402
1675	402	a/b = 1675/402 = 4 reszta 67	a = 67
67	402	b/a = 402/67 = 6 reszta 0	b = 0
67	0	KONIEC	

NWD(1675,3752) = 67

Algorytm Euklidesa - lista kroków

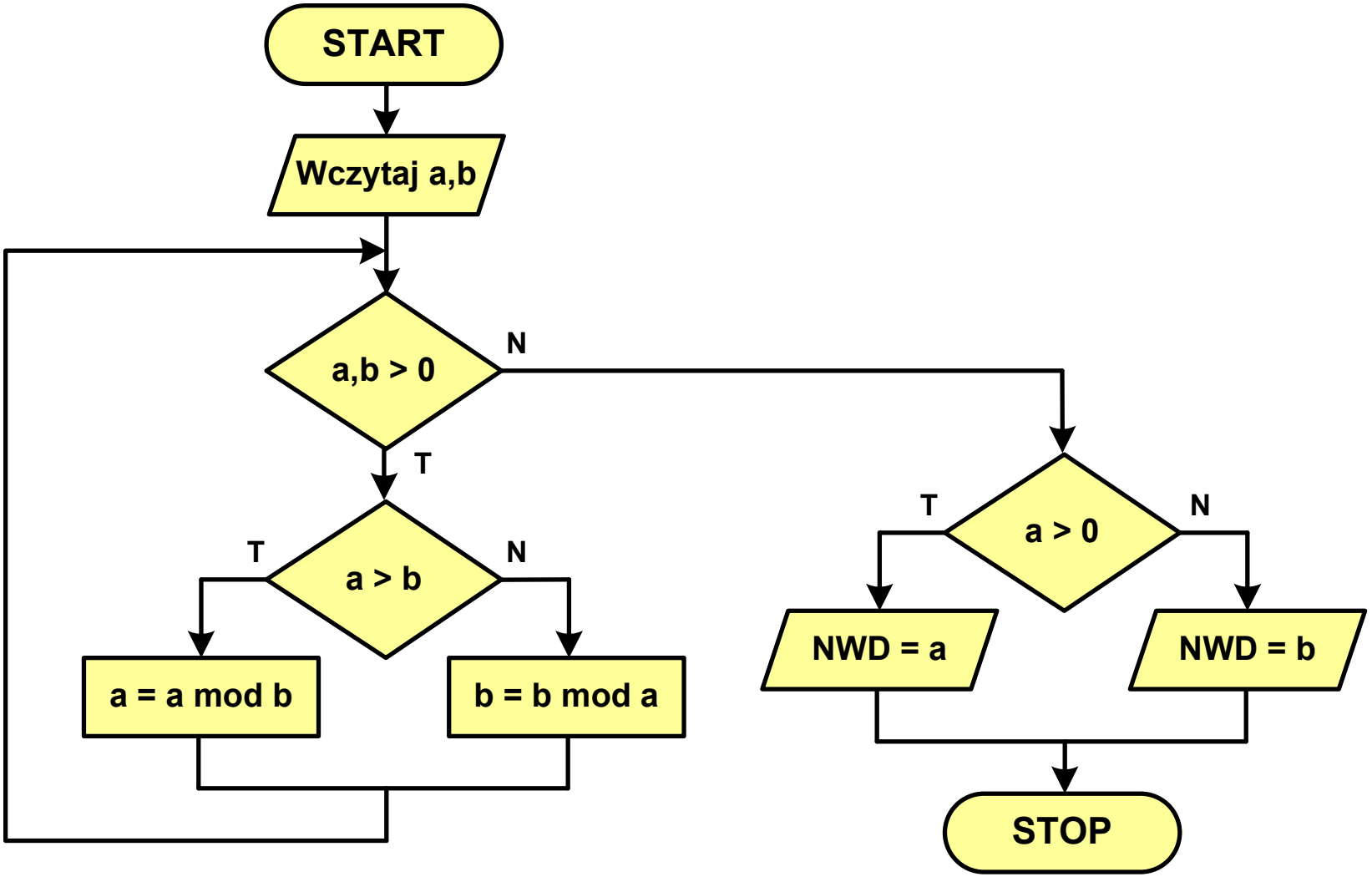
Dane wejściowe: niezerowe liczby naturalne a i b

Dane wyjściowe: NWD(a,b)

Kolejne kroki:

1. Czytaj liczby a i b
2. Dopóki a i b są większe od zera, powtarzaj krok 3,
   a w przeciwnym przypadku przejdź do kroku 4
3. Jeśli a jest większe od b, to weź za a resztę z dzielenia a przez b,
   w przeciwnym przypadku weź za b resztę z dzielenia b przez a
4. Przyjmij jako największy wspólny dzielnik tę z liczb a i b,
   która pozostała większa od zera
5. Drukuj NWD(a,b)

Algorytm Euklidesa - schemat blokowy



Algorytm Euklidesa - pseudokod

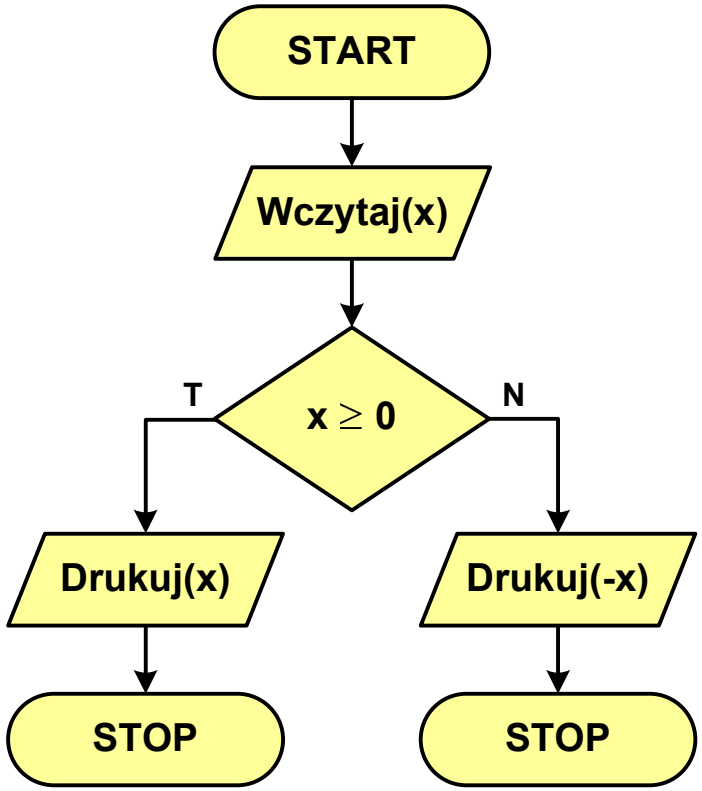
```
NWD(a,b)
while a>0 i b>0
 do if a>b
       then a ← a mod b
       else b ← b mod a
if a>0
   then return a
   else return b
```

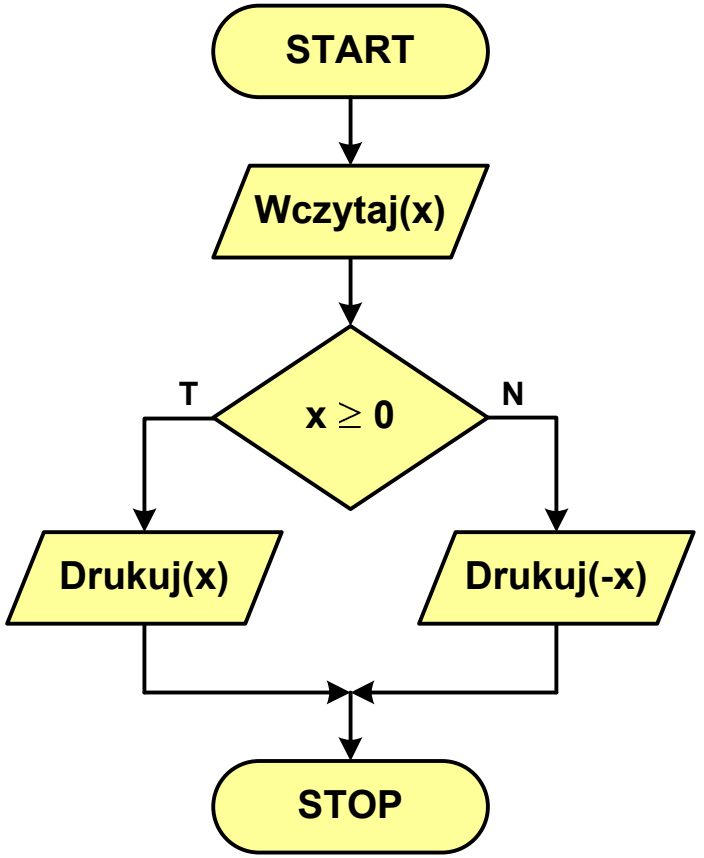
Algorytm Euklidesa - język programowania (C)

```c
#include <stdio.h>

int main(void)
{
    int a = 1675, b = 3752, NWD;

    while (a>0 && b>0)
        if (a>b)
            a = a % b;
        else
            b = b % a;

    if (a>0)
        NWD = a;
    else
        NWD = b;

    printf("NWD = %d\n",NWD);
}
```

Wartość bezwzględna liczby - schemat blokowy

$$|x| = \begin{cases} x \ \ dla \ \ x \geq 0 \\ -x \ \ dla \ \ x < 0 \end{cases}$$





Równanie kwadratowe - schemat blokowy

$$ax^2 + bx + c = 0$$

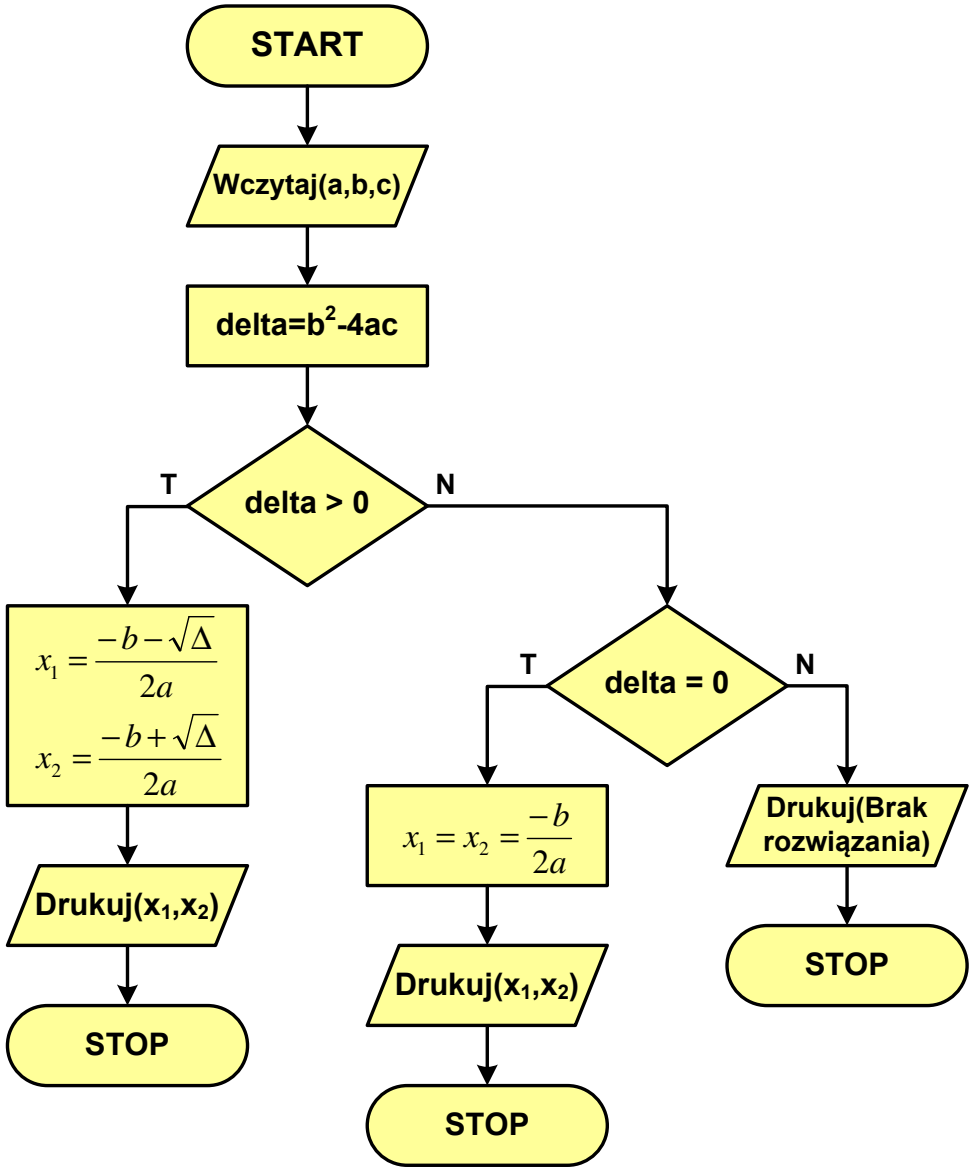
$$\Delta = b^2 - 4ac$$

$\Delta > 0$:

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a}, \quad x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

$\Delta = 0$:

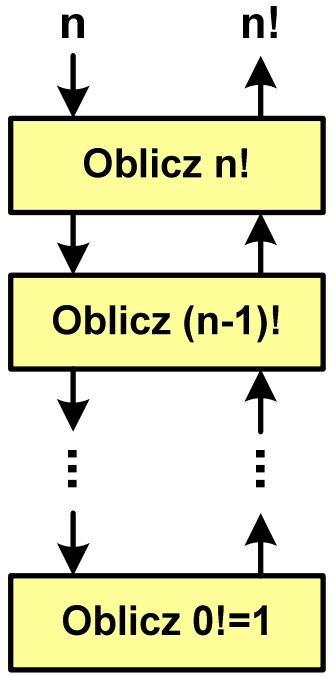
$$x_1 = x_2 = \frac{-b}{2a}$$



Rekurencja

- Rekurencja lub rekursja - jest to odwoływanie się funkcji lub definicji do samej siebie
- Rozwiązanie danego problemu wyraża się za pomocą rozwiązań tego samego problemu, ale dla danych o mniejszych rozmiarach
- W matematyce mechanizm rekurencji stosowany jest do definiowania lub opisywania algorytmów
- Silnia:

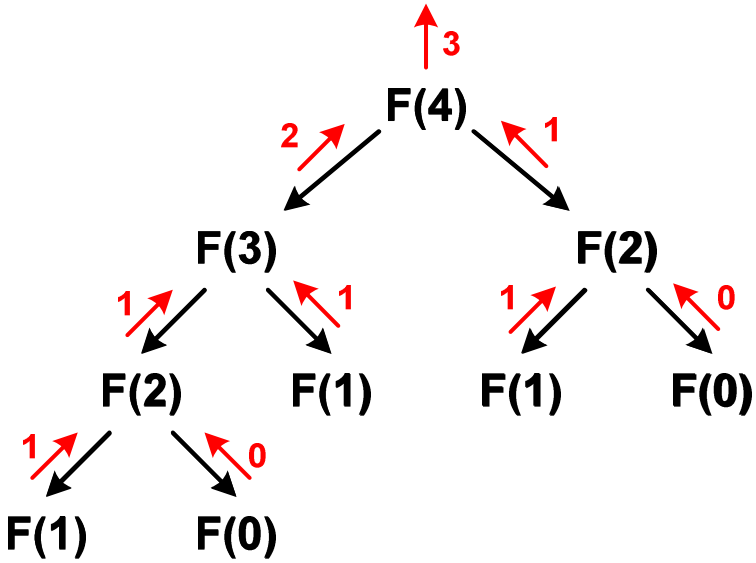
$$n! = \begin{cases} 1 & dla \quad n = 0 \\ n(n-1)! & dla \quad n \geq 1 \end{cases}$$




```
int silnia(int n)
{
    return n==0 ? 1 : n*silnia(n-1);
}
```

Rekurencja - ciąg Fibonacciego

$$F_n = \begin{cases} 0 & dla \quad n = 0 \\ 1 & dla \quad n = 1 \\ F_{n-1} + F_{n-2} & dla \quad n > 1 \end{cases}$$




```
int F(int n)
{
    if (n==0) return 0;
    if (n==1) return 1;
    return F(n-1) + F(n-2);
}
```

Złożoność obliczeniowa

- W celu rozwiązania danego problemu obliczeniowego szukamy algorytmu najbardziej efektywnego czyli:
  - najszybszego (najkrótszy czas otrzymania wyniku)
  - o możliwie małym zapotrzebowaniu na pamięć
- Problem: Jak ocenić, który z dwóch różnych algorytmów rozwiązujących to samo zadanie jest efektywniejszy?
- Do oceny efektywności służy złożoność obliczeniowa algorytmu (koszt algorytmu) czyli ilość zasobów potrzebnych do jego działania (czas, pamięć)
- Miarą złożoności czasowej jest liczba podstawowych (dominujących) operacji (porównanie, podstawienie, operacja arytmetyczna) - pozostałe operacje są pomijane
- Miarą złożoności pamięciowej jest liczba wykorzystanych komórek pamięci (bajty lub liczba zmiennych określonego typu)

Złożoność obliczeniowa

- Złożoność obliczeniowa algorytmu jest funkcją opisującą zależność między liczbą danych a liczbą operacji wykonywanych przez ten algorytm
- W praktyce stosuje się oszacowanie powyższej funkcji - są to tzw. notacje (klasy złożoności):
  - O (duże O) - najbardziej popularna
  - Ω (omega)
  - Θ (theta)

Notacja O („duże O”)

- Wyraża złożoność matematyczną algorytmu
- Do wyznaczenia złożoności bierze się pod uwagę liczbę dominujących operacji wykonywanych w algorytmie
- Przykład zapisu: $O(n^2)$
  - po literze O występuje wyrażenie w nawiasach zawierające literę n, która oznacza liczbę elementów, na których działa algorytm
- W funkcji opisującej złożoność bierze się pod uwagę tylko najistotniejszy składnik, np.

  $f(n) = n^2+2n \rightarrow O(n^2)$ $\quad\quad$ $f(n) = n^2+n-5 \rightarrow O(n^2)$

- W powyższych przykładach dla dużego n wpływ składnika liniowego i stałego na wartość funkcji jest nieistotny w porównaniu ze składnikiem głównym n^2

Notacja O („duże O”)

- Porównanie najczęściej występujących złożoności:

Elementy (n)	O(log n)	O(n)	O(n log n)	$O(n^2)$	$O(n^3)$	$O(2^n)$
10	3	10	33	100	1 000	1024
100	7	100	664	10 000	1 000 000	$1{,}27 \cdot 10^{30}$
1 000	10	1 000	9 966	1 000 000	10^9	$1{,}07 \cdot 10^{301}$
10 000	13	10 000	132 877	10^8	10^{12}	$1{,}99 \cdot 10^{3010}$

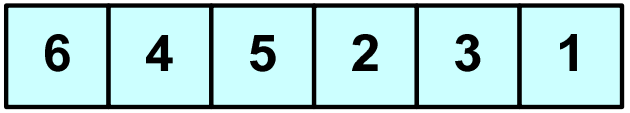
O(log n) - logarytmiczna (np. przeszukiwanie binarne)
O(n) - liniowa (np. porównywanie łańcuchów znaków)
O(n log n) - liniowo-logarytmiczna (np. sortowanie szybkie)
$O(n^2)$ - kwadratowa (np. proste algorytmy sortowania)
$O(n^3)$ - sześcienna (np. mnożenie macierzy)
$O(2^n)$ - wykładnicza (np. problem komiwojażera)

Sortowanie

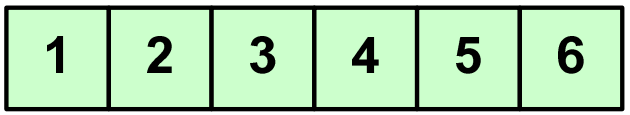
- Sortowanie polega na uporządkowaniu zbioru danych względem pewnych cech charakterystycznych każdego elementu tego zbioru (wartości każdego elementu)
- W przypadku liczb, sortowanie polega na znalezieniu kolejności liczb zgodnej z relacją ≤ lub ≥

Przykład:

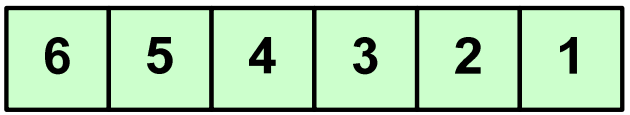
- Tablica nieposortowana:



- Tablica posortowana zgodnie z relacją ≤ (od najmniejszej do największej liczby):



- Tablica posortowana zgodnie z relacją ≥ (od największej do najmniejszej liczby):



Sortowanie

- W przypadku słów sortowanie polega na ustawieniu ich w porządku alfabetycznym (leksykograficznym)

Przykład:

- Tablica nieposortowana:

Paweł	Piotr	Adrian	Ela	Ola	Henryk

- Tablice posortowane:

Adrian	Ela	Henryk	Ola	Paweł	Piotr

Piotr	Paweł	Ola	Henryk	Ela	Adrian

Sortowanie

- W praktyce sortowanie sprowadza się do porządkowanie danych na podstawie porównania - porównywany element to klucz

Przykład:

- Tablica nieposortowana (imię, nazwisko, wiek):

Piotr	Ola	Paweł	Jan	Ela	Magda
Kowalski	Nowak	Wójcik	Kamiński	Król	Mazur
25	18	23	20	22	15

- Tablica posortowana (klucz - nazwisko):

Jan	Piotr	Ela	Magda	Ola	Paweł
Kamiński	Kowalski	Król	Mazur	Nowak	Wójcik
20	25	22	15	18	23

- Tablica posortowana (klucz - wiek):

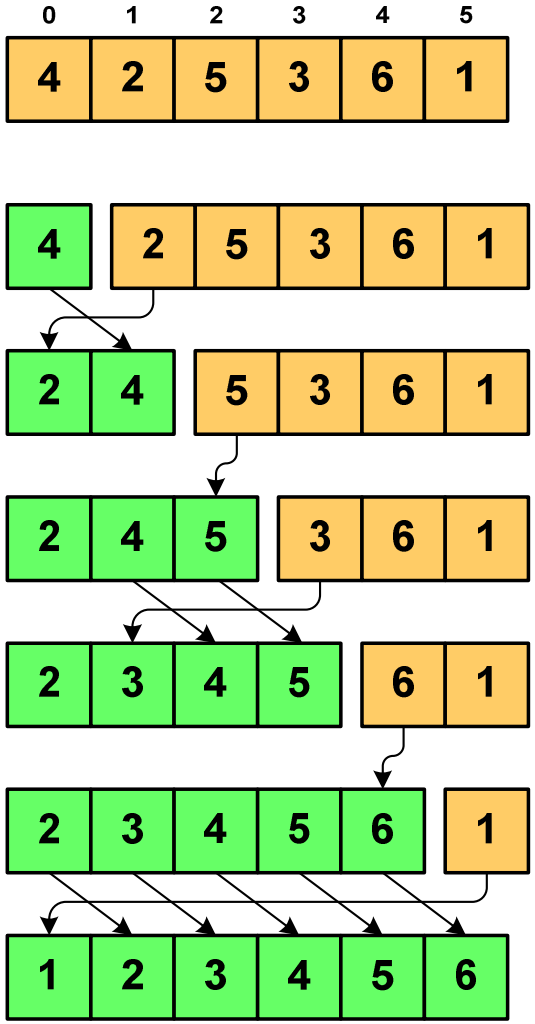
Magda	Ola	Jan	Ela	Paweł	Piotr
Mazur	Nowak	Kamiński	Król	Wójcik	Kowalski
15	18	20	22	23	25

Sortowanie

- Po co stosować sortowanie?
  - posortowane elementy można szybciej zlokalizować
  - posortowane elementy można przedstawić w czytelniejszy sposób
- Przykładowe algorytmy sortowania
  - proste wstawianie (insertion sort)
  - proste wybieranie (selection sort)
  - bąbelkowe (bubble sort)
  - szybkie (quick sort)
  - przez scalanie (merge sort)
  - kubełkowe / przez zliczanie (bucket sort)

Proste wstawianie (insertion sort)

Przykład:

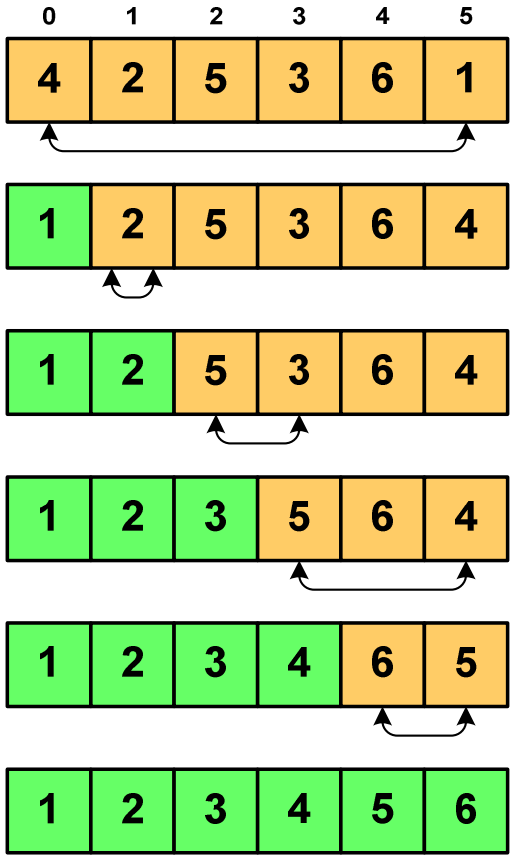


Program w języku C:

```
int main(void)
{
   int tab[N],i,j,tmp;
   // ...
   for (i=1; i<N; i++)
   {
      j=i;
      tmp=tab[i];
      while (tab[j-1]>tmp && j>0)
      {
         tab[j]=tab[j-1];
         j--;
      }
      tab[j]=tmp;
   }
}
```

Proste wybieranie (selection sort)

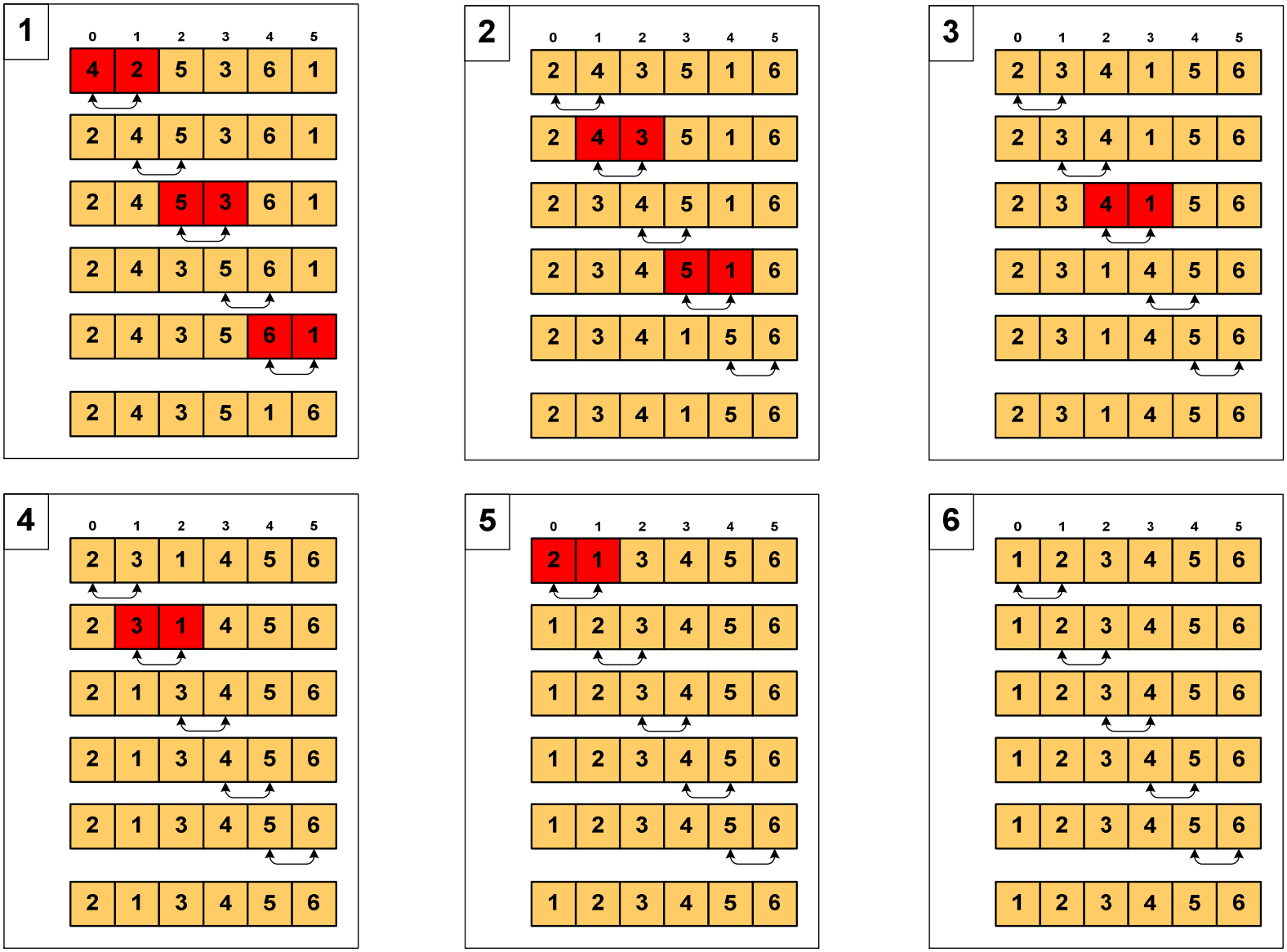
Przykład:



Program w języku C:

```
int main(void)
{
    int tab[N],i,j,k,tmp;
    // ...
    for (i=0;i<N-1;i++)
    {
        k=i;
        for (j=i+1; j<N; j++)
            if (tab[k]>=tab[j])
                k = j;
        tmp = tab[i];
        tab[i] = tab[k];
        tab[k] = tmp;
    }
}
```

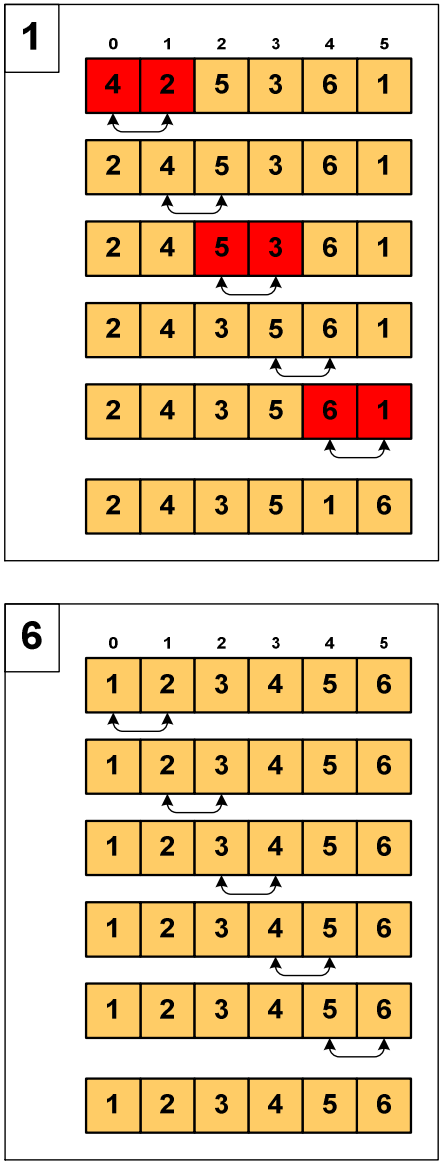
Bąbelkowe (bubble sort)



Bąbelkowe (bubble sort)

Program w języku C:

```
int main(void)
{
    int tab[N],i,j,tmp,koniec;
    // ...
    do {
        koniec=1;
        for (i=0;i<N-1;i++)
            if (tab[i]>tab[i+1])
            {
                tmp=tab[i];
                tab[i]=tab[i+1];
                tab[i+1]=tmp;
                koniec=0;
            }
    } while (!koniec);
}
```



Koniec wykładu nr 4

Dziękuję za uwagę!