



Fundusze
Europejskie
Wiedza Edukacja Rozwój



Rzeczpospolita
Polska

Unia Europejska
Europejski Fundusz Społeczny



Wydział Elektryczny
Katedra Elektrotechniki Teoretycznej i Metrologii

Materiały do wykładu z przedmiotu:

Informatyka

Kod: EDS1B1007

WYKŁAD NR 12

Opracował: dr inż. Jarosław Forenc

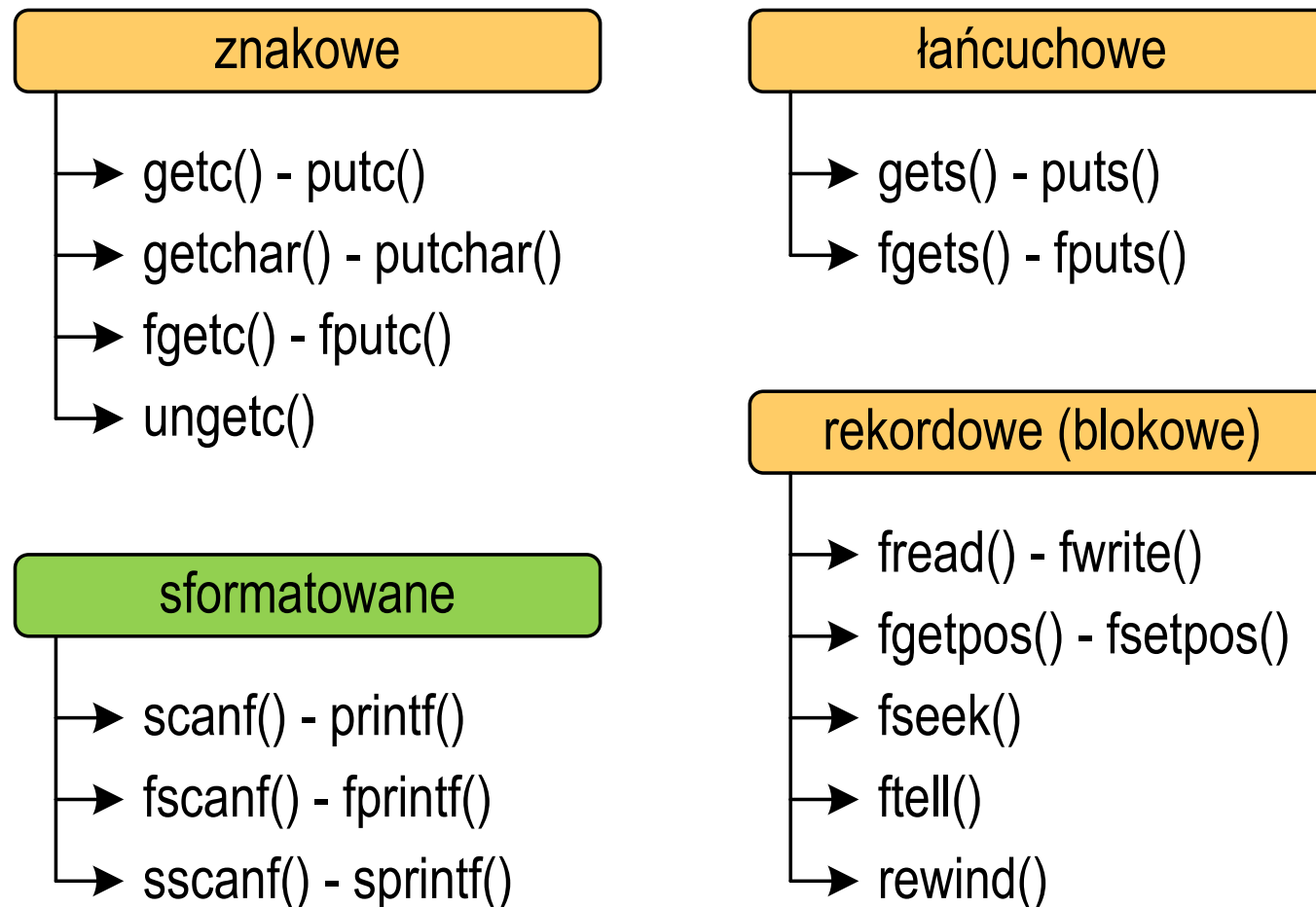
Białystok 2020

Materiały zostały opracowane w ramach projektu „PB2020 - Zintegrowany Program Rozwoju Politechniki Białostockiej” realizowanego w ramach Działania 3.5 Programu Operacyjnego Wiedza, Edukacja, Rozwój 2014-2020 współfinansowanego ze środków Europejskiego Funduszu Społecznego.

Plan wykładu nr 12

- Typy operacji wejścia-wyjścia
 - sformatowane
 - rekordowe (blokowe)
- Algorytmy komputerowe
 - definicje
 - sposoby opisu

Sformatowane operacje wejścia-wyjścia



Sformatowane operacje wejścia-wyjścia

```
FILE *fp; char txt[30];  
/* ... */  
printf("Witaj świecie"); // na ekran  
fprintf(fp, "Witaj świecie"); // do pliku  
sprintf(txt, "Witaj świecie"); // do tablicy znaków
```

```
FILE *fp; char txt[30] = "15 3.14";  
int x; float y;  
/* ... */  
scanf("%d %f", &x, &y); // z klawiatury  
fscanf(fp, "%d %f", &x, &y); // z pliku  
sscanf(txt, "%d %f", &x, &y); // z tablicy znaków
```

Odczytanie zawartości pliku tekstowego

- Jak odczytać liczby z pliku tekstowego nie wiedząc ile ich jest?

```
3.830073
70.848717
99.322487
19.812616
7.132175
49.134800
10.238960
18.668173
8.914456
69.258705
```

Obsługa błędów wejścia-wyjścia

FEOF

stdio.h

```
int feof(FILE *fp);
```

- Sprawdza, czy podczas ostatniej operacji wejścia dotyczącej strumienia **fp** został osiągnięty koniec pliku
- Zwraca wartość różną od zera, jeśli podczas ostatniej operacji wejścia został wykryty koniec pliku, w przeciwnym razie zwraca wartość **0** (zero)

Przykład: odczytanie liczb z pliku tekstowego

```
#include <stdio.h>

int main(void)
{
    FILE *fp; float x;

    fp = fopen("liczby.txt", "r");

    fscanf(fp, "%f", &x);
    while (!feof(fp))
    {
        printf("%f\n", x);
        fscanf(fp, "%f", &x);
    }

    fclose(fp);

    return 0;
}
```

```
3.830073
70.848717
99.322487
19.812616
7.132175
49.134800
10.238960
18.668173
8.914456
69.258705
```

Przykład: odczytanie liczb z pliku tekstowego

- Sposób zapisu liczb w pliku wejściowym nie ma znaczenia dla prawidłowości ich odczytu
- Liczby powinny być oddzielone od siebie znakami spacji, tabulacji lub znakiem nowego wiersza

```
3.830073
70.848717
99.322487
19.812616
7.132175
49.134800
10.238960
18.668173
8.914456
69.258705
```

```
3.830073    70.848717
99.322487   19.812616
7.132175    49.134800
10.238960   18.668173
8.914456    69.258705
```

```
3.830073    70.848717    99.322487
19.812616    7.132175     49.134800
10.238960    18.668173    8.914456
69.258705
```


Przykład: odczytanie danych z pliku tekstowego

- Odczytanie danych różnych typów z pliku tekstowego

```
Nowak Grzegorz 15-12-2000  
Kowalski Wojciech 03-05-1997  
Jankowska Anna 23-05-1995  
Mazur Krzysztof 14-01-1990  
Krawczyk Monika 03-11-1995  
Piotrowska Maja 12-06-1998  
Dudek Piotr 31-12-1996  
Pawlak Julia 01-01-1997
```

Grzegorz	Nowak	wiek: 20
Wojciech	Kowalski	wiek: 23
Anna	Jankowska	wiek: 25
Krzysztof	Mazur	wiek: 30
Monika	Krawczyk	wiek: 25
Maja	Piotrowska	wiek: 22
Piotr	Dudek	wiek: 24
Julia	Pawlak	wiek: 23

Przykład: odczytanie danych z pliku tekstowego

```
#include <stdio.h>

int main()
{
    FILE *fp;
    char naz[20], im[20];
    int d, m, r;

    fp = fopen("osoby.txt", "r");
    fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);
    while (!feof(fp))
    {
        printf("%-12s %-12s wiek: %d\n", im, naz, 2020-r);
        fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);
    }
    fclose(fp);

    return 0;
}
```

Przykład: odczytanie danych z pliku tekstowego

```
#include <stdio.h>

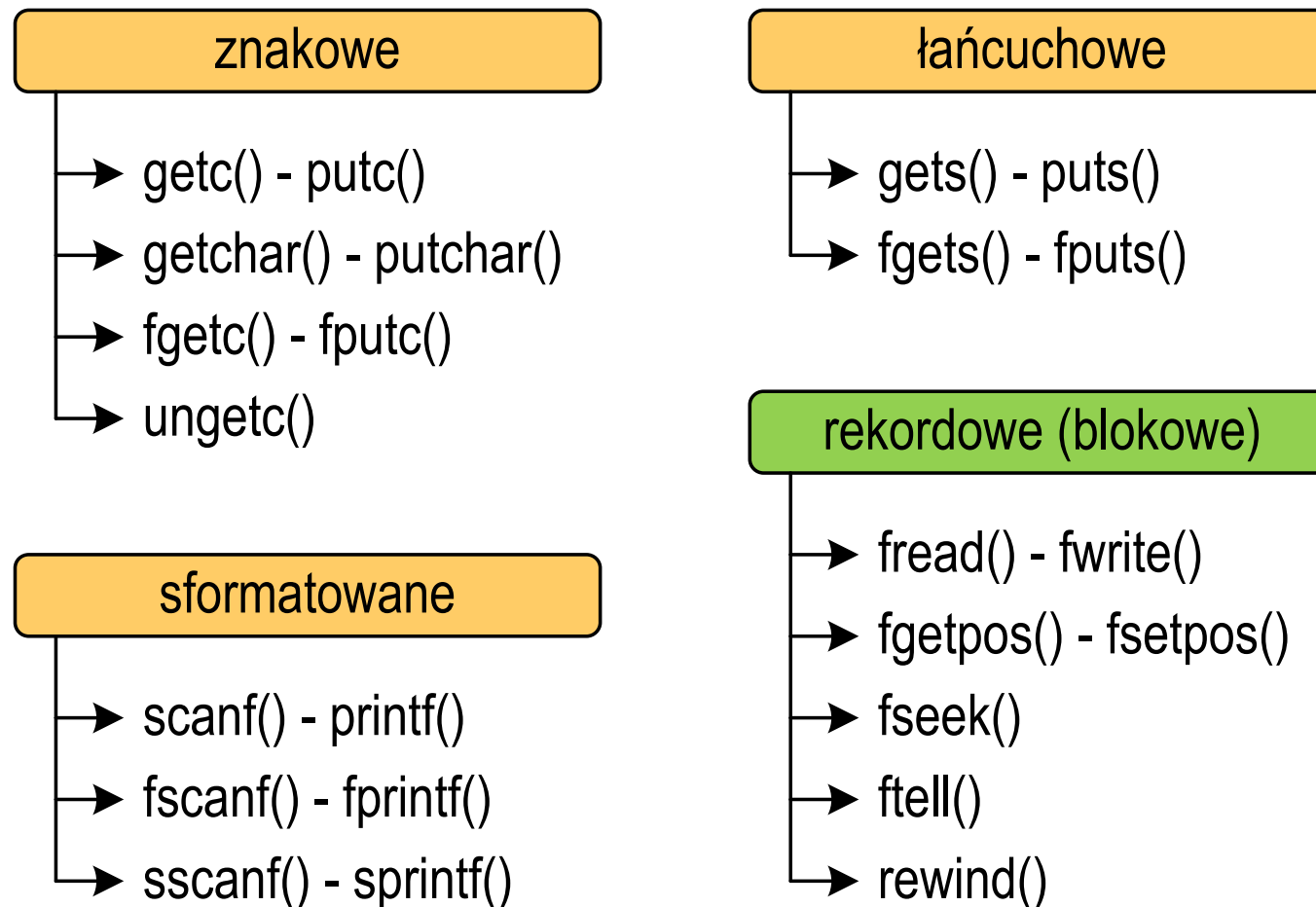
int main()
{
    FILE *fp;
    char naz[20], im[20];
    int d, m, r;

    fp = fopen("osoby.txt", "r");
    fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);
    while (!feof(fp))
    {
        printf("%-12s %-12s wiek: %d\n", im, naz, 2020-r);
        fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);
    }
    fclose(fp);

    return 0;
}
```

Grzegorz	Nowak	wiek: 20
Wojciech	Kowalski	wiek: 23
Anna	Jankowska	wiek: 25
Krzysztof	Mazur	wiek: 30
Monika	Krawczyk	wiek: 25
Maja	Piotrowska	wiek: 22
Piotr	Dudek	wiek: 24
Julia	Pawlak	wiek: 23

Rekordowe (blokowe) operacje wejścia-wyjścia



Rekordowe (blokowe) operacje wejścia-wyjścia

FWRITE

stdio.h

```
size_t fwrite(const void *p, size_t s, size_t n,  
             FILE *fp);
```

- Zapisuje **n** elementów o rozmiarze **s** bajtów każdy, do pliku wskazywanego przez **fp**, biorąc dane z obszaru pamięci wskazywanego przez **p**
- Zwraca liczbę zapisanych elementów - jeśli jest ona różna od **n**, to wystąpił błąd zapisu (brak miejsca na dysku lub dysk zabezpieczony przed zapisem)

Przykład: zapisanie danych do pliku binarnego

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int    x = 10, tab[5] = {1,2,3,4,5};
    float  y = 1.2345f;

    fp = fopen("dane.dat", "wb");
    fwrite(&x, sizeof(int), 1, fp);
    fwrite(tab, sizeof(int), 5, fp);
    fwrite(tab, sizeof(tab), 1, fp);
    fwrite(&y, sizeof(float), 1, fp);
    fclose(fp);

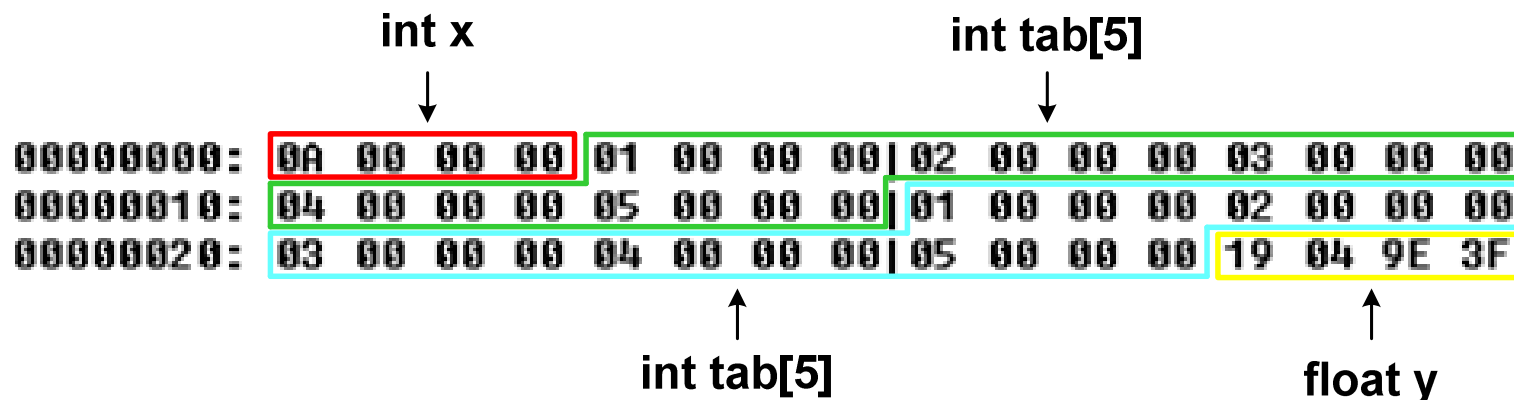
    return 0;
}
```

Przykład: zapisanie danych do pliku binarnego

- Czterokrotne wywołanie funkcji `fwrite()`

```
fwrite(&x, sizeof(int), 1, fp); // int x = 10;  
fwrite(tab, sizeof(int), 5, fp); // int tab[5] = {1,2,3,4,5};  
fwrite(tab, sizeof(tab), 1, fp); // int tab[5] = {1,2,3,4,5};  
fwrite(&y, sizeof(float), 1, fp); // float y = 1.2345;
```

spowoduje zapisanie do pliku 48 bajtów:



Rekordowe (blokowe) operacje wejścia-wyjścia

FREAD

stdio.h

```
size_t fread(void *p, size_t s, size_t n,  
             FILE *fp);
```

- Pobiera **n** elementów o rozmiarze **s** bajtów każdy, z pliku wskazywanego przez **fp** i umieszcza odczytane dane w obszarze pamięci wskazywanym przez **p**
- Zwraca liczbę odczytanych elementów - w przypadku gdy liczba ta jest różna od **n**, to wystąpił błąd końca strumienia (w pliku było mniej elementów niż podana wartość argumentu **n**)

Przykład: odczytanie liczb z pliku binarnego

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int x, ile = 0;

    fp = fopen("liczby.dat", "rb");
    fread(&x, sizeof(int), 1, fp);
    while (!feof(fp))
    {
        ile++; printf("%d\n", x);
        fread(&x, sizeof(int), 1, fp);
    }
    fclose(fp);
    printf("Odczytano: %d liczb\n", ile);
    return 0;
}
```

```
37
31
83
27
6
62
31
50
Odczytano: 8 liczb
```

Przykład: odczytanie liczb z pliku binarnego

- Po otwarciu pliku wskaźnik pozycji pliku pokazuje na jego początek

↓
25 00 00 00 1F 00 00 00 | 53 00 00 00 1B 00 00 00 | %■■■■■■■■S■■■■■■■■
06 00 00 00 3E 00 00 00 | 1F 00 00 00 32 00 00 00 | ■■■■>■■■■■■■■2■■■

- Po odczytaniu jednej liczby: `fread(&x,sizeof(int),1,plik);`
wskaźnik jest automatycznie przesuwany o `sizeof(int)` bajtów

↓
25 00 00 00 1F 00 00 00 | 53 00 00 00 1B 00 00 00 | %■■■■■■■■S■■■■■■■■
06 00 00 00 3E 00 00 00 | 1F 00 00 00 32 00 00 00 | ■■■■>■■■■■■■■2■■■

- Po odczytaniu kolejnej liczby: `fread(&x,sizeof(int),1,plik);`
wskaźnik jest ponownie przesuwany o `sizeof(int)` bajtów

↓
25 00 00 00 1F 00 00 00 | 53 00 00 00 1B 00 00 00 | %■■■■■■■■S■■■■■■■■
06 00 00 00 3E 00 00 00 | 1F 00 00 00 32 00 00 00 | ■■■■>■■■■■■■■2■■■

- Plik binarny zawiera liczby: 37 31 83 27 6 62 31 50

Rekordowe (blokowe) operacje wejścia-wyjścia

REWIND

stdio.h

```
void rewind(FILE *fp);
```

- Ustawia wskaźnik pozycji w pliku wskazywanym przez **fp** na początek pliku

FTELL

stdio.h

```
long int ftell(FILE *fp);
```

- Zwraca bieżące położenie w pliku wskazywanym przez **fp** (liczbę bajtów od początku pliku)

Przykład: ile razy występuje w pliku wartość max

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int x, max, ile = 0;

    fp = fopen("dane.dat", "rb");

    fread(&x, sizeof(int), 1, fp);
    max = x;
    while(!feof(fp))
    {
        if (x > max) max = x;
        fread(&x, sizeof(int), 1, fp);
    }
    printf("Wartosc max: %d\n", max);
}
```

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

Przykład: ile razy występuje w pliku wartość max

```
rewind(fp);  
  
fread(&x, sizeof(int), 1, fp);  
while(!feof(fp))  
{  
    if (x == max) ile++;  
    fread(&x, sizeof(int), 1, fp);  
}  
printf("Wystąpienia max: %d\n", ile);  
  
fclose(fp);  
  
return 0;  
}
```

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

```
Wartosc max: 9  
Wystapienia max: 4
```

Rekordowe (blokowe) operacje wejścia-wyjścia

FSEEK

stdio.h

```
int fseek(FILE *fp, long int offset, int mode);
```

- Pozwala przejść bezpośrednio do dowolnego bajtu w pliku wskazywanym przez **fp**
- **offset** określa wielkość przejścia w bajtach, zaś **mode** - punkt początkowy, względem którego określane jest przejście (**SEEK_SET** - początek pliku, **SEEK_CUR** - bieżąca pozycja, **SEEK_END** - koniec pliku)
- Gdy wywołanie jest poprawne, to funkcja zwraca wartość **0** gdy wystąpił błąd (np. próba przekroczenia granic pliku), to funkcja zwraca wartość **-1**

Rekordowe (blokowe) operacje wejścia-wyjścia

FGETPOS

stdio.h

```
int fgetpos(FILE *fp, fpos_t *pos);
```

- Zapamiętuje pod zmienną **pos** bieżące położenie w pliku wskazywanym przez **fp**; zwraca **0**, gdy wywołania jest poprawne i wartość niezerową, gdy wystąpił błąd

FSETPOS

stdio.h

```
int fsetpos(FILE *fp, const fpos_t *pos);
```

- Przechodzi do położenia **pos** w pliku wskazywanym przez **fp**; zwraca **0**, gdy wywołania jest poprawne i wartość niezerową, gdy wystąpił błąd

Algorytm - definicje

Definicja 1

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

Definicja 2

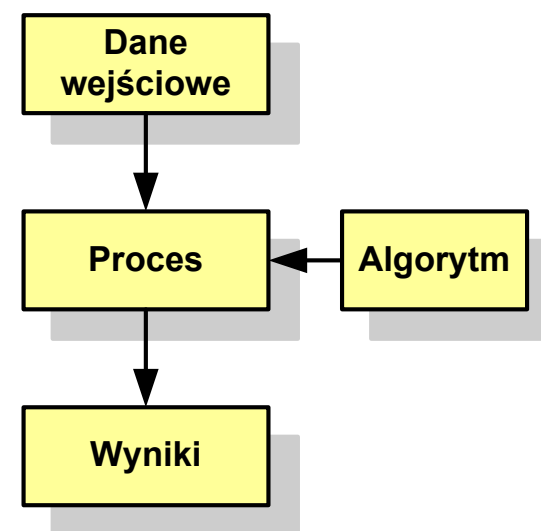
- Opis rozwiązania problemu wyrażony za pomocą operacji zrozumiałych i możliwych do zrealizowania przez wykonawcę

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 Muhammada ibn-Musy **al-Chuwarizmiego** (po łacinie pisanego jako **Algorismus**), matematyka perskiego z IX wieku
- Badaniem algorytmów zajmuje się **algorytmika**
- Algorytm może zostać **zaimplementowany** w postaci **programu komputerowego**
- Przetłumaczenie algorytmu na wybrany język programowania nazywane jest też **kodowaniem** algorytmu
- Ten sam algorytm może być zaimplementowany (zakodowany) w różny sposób przy użyciu różnych języków programowania.

Podstawowe cechy algorytmu

- Posiada dane wejściowe (w ilości większej lub równej zero) pochodzące z dobrze zdefiniowanego zbioru
- Zwraca wynik
- Jest precyzyjnie zdefiniowany (każdy krok algorytmu musi być jednoznacznie określony)
- Poprawność (dla każdego z założonego dopuszczalnego zestawu danych wejściowych)
- Kończy działanie po skończonej liczbie kroków (powinna istnieć poprawnie działająca reguła stopu algorytmu)
- Efektywność (jak najkrótszy czas wykonania i jak najmniejsze zapotrzebowanie na pamięć).

Sposoby opisu algorytmów

1. Opis słowny w języku naturalnym lub w postaci listy kroków (opis w punktach)
2. Schemat blokowy
3. Pseudokod (nieformalna odmiana języka programowania)
4. 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

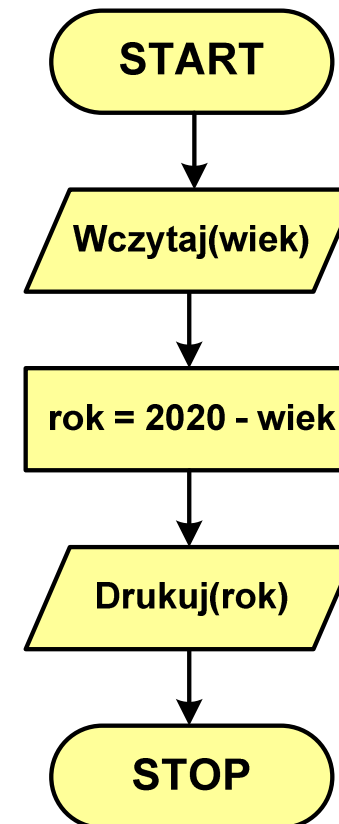
Opis algorytmu: Ziemniaki obrać i pokroić w plasterki. Kiełbasę pokroić w plasterki. Ziemniaki wrzucić na gorącą oliwę na patelni i przyrumienić z obu stron. Kiełbasę wrzucić na gorącą oliwę na patelni i przyrumienić z obu stron. Ubić jajka i dodać do połączonych ziemniaków i kiełbasy. Dodać sól i pieprz. Usmażyć z obu stron wielki omlet nadziewany chipsami ziemniaczanymi z kiełbaską.

Lista kroków

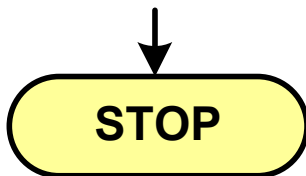
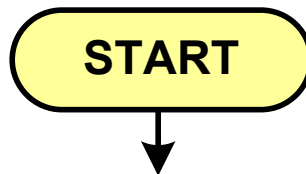
- Uporządkowany opis wszystkich czynności, jakie należy wykonać podczas realizacji algorytmu
- **Krok** jest to pojedyncza czynność realizowana w algorytmie
- Kroki w algorytmie są numerowane, operacje wykonywane są zgodnie z rosnącą numeracją kroków
- Jedynym odstępstwem od powyższej reguły są operacje 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: wyznaczenie roku urodzenia na podstawie wieku (**algorytm liniowy**)

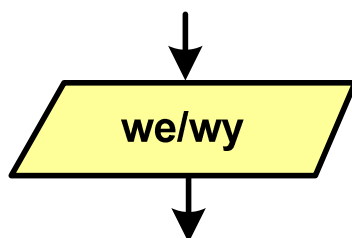


Schemat blokowy - symbole graficzne

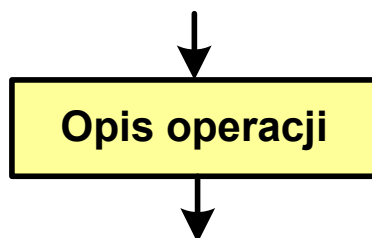


- **blok startowy**, początek algorytmu
 - wskazuje miejsce rozpoczęcia algorytmu
 - ma jedno wyjście
 - może występować tylko jeden raz
-
- **blok końcowy**, koniec algorytmu
 - wskazuje miejsce zakończenia algorytmu
 - ma jedno wejście
 - musi występować przynajmniej jeden raz

Schemat blokowy - symbole graficzne

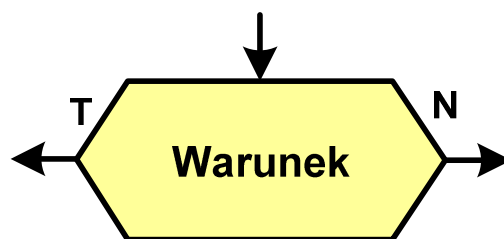


- **blok wejścia-wyjścia**
- poprzez ten blok wprowadzane są (czytane) dane wejściowe i wyprowadzane (zapisywane) wyniki
- ma jedno wejście i jedno wyjście

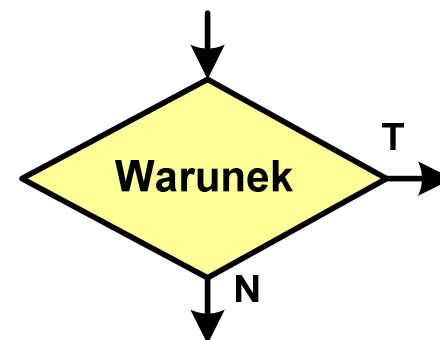
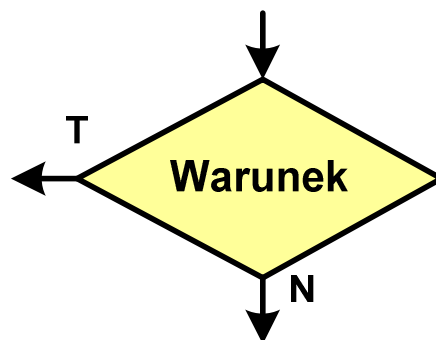


- **blok wykonawczy**, blok funkcyjny, opis procesu
- zawiera jedno lub kilka poleceń (elementarnych instrukcji) wykonywanych w podanej kolejności
- instrukcją może być np. operacja arytmetyczna, podstawienie
- ma jedno wejście i jedno wyjście

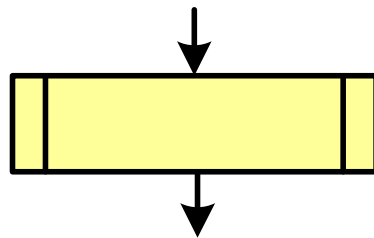
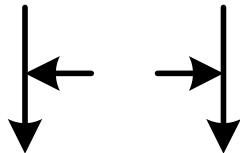
Schemat blokowy - symbole graficzne



- **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

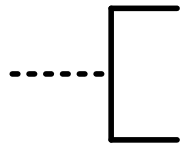


Schemat blokowy - symbole graficzne

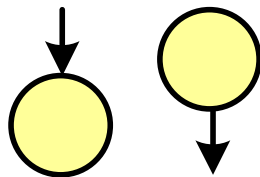


- **linia przepływu**, połączenie, linia
- występuje w postaci linii zakończonej strzałką
- określa kierunek przemieszczania się po schemacie
- łączy inne bloki występujące na schemacie
- linie pochodzące z różnych części algorytmu mogą zbiegać się w jednym miejscu
- **podprogram**
- wywołanie wcześniej zdefiniowanego fragmentu algorytmu (podprogramu)
- ma jedno wejście i jedno wyjście

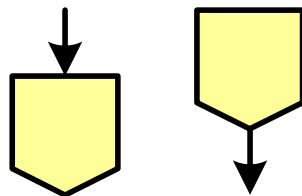
Schemat blokowy - symbole graficzne



- **komentarz**
- dodanie do schematu dodatkowego opisu



- **łącznik stronicowy** (wewnętrzny)
- połączenie dwóch odrębnych części schematu znajdujących się na tej samej stronie
- łączniki opisywane są etykietami



- **łącznik międzystronicowy** (zewnątrzny)
- połączenie dwóch odrębnych części schematu znajdujących się na różnych stronach
- łączniki opisywane są etykietami

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

$$\text{NWD}(1675, 3752) = ?$$

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	

$$\text{NWD}(1675, 3752) = 67$$

Algorytm Euklidesa - lista kroków

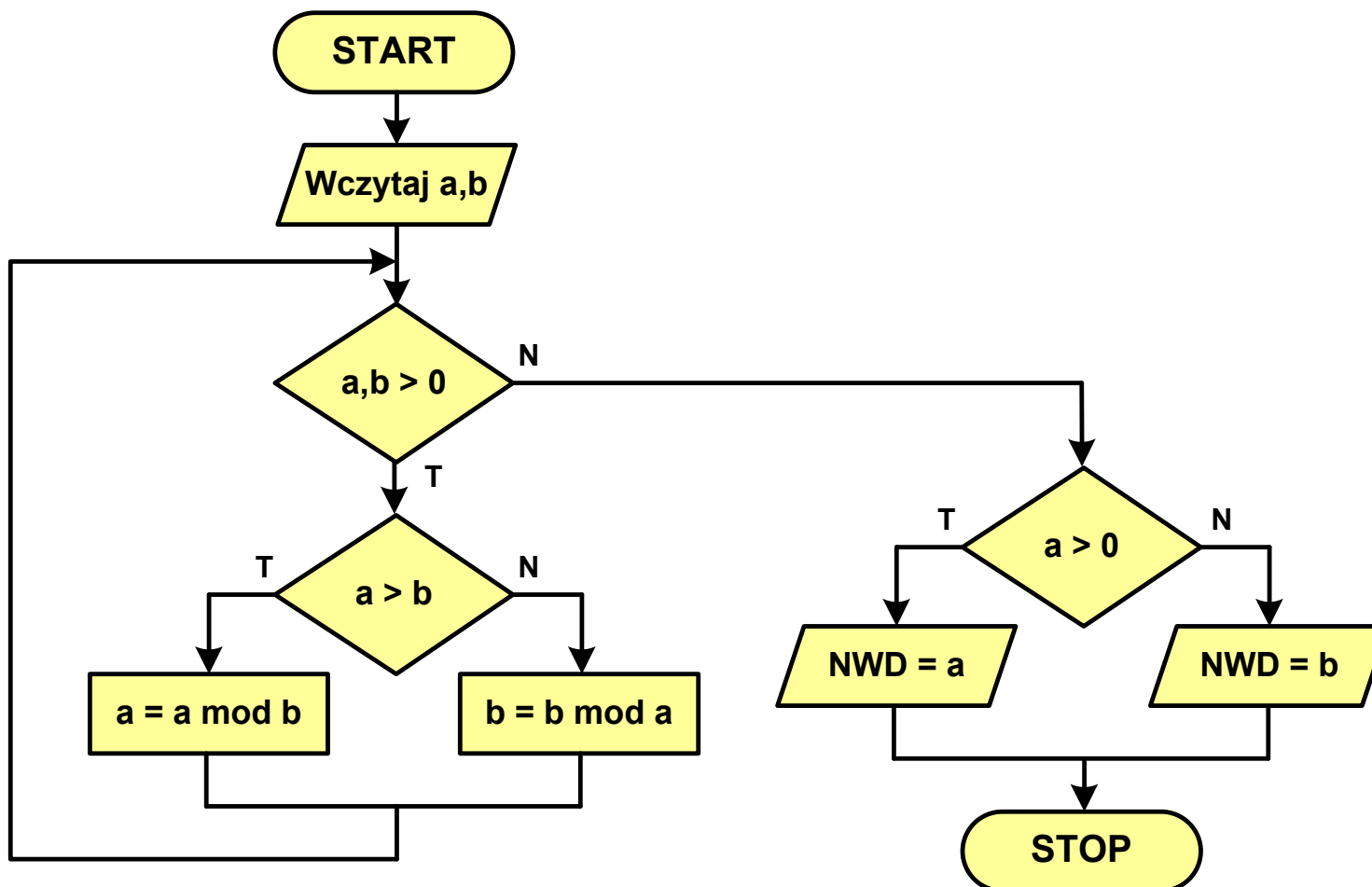
Dane wejściowe: niezerowe liczby naturalne a i b

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 następnie przejdź do **kroku 4**
3. Jeśli a jest większe od b , to weź za a resztę z dzielenia a przez b , w przeciwnym razie 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)

```
#include <stdio.h>

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

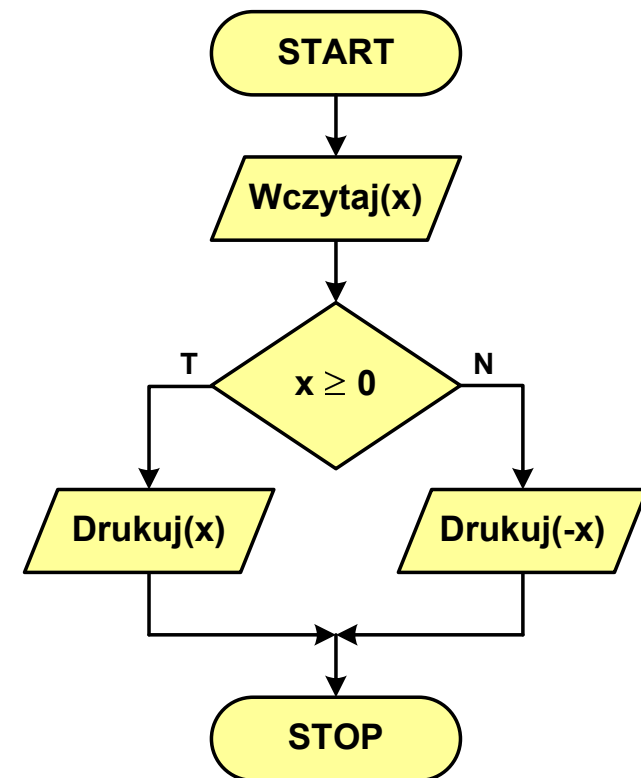
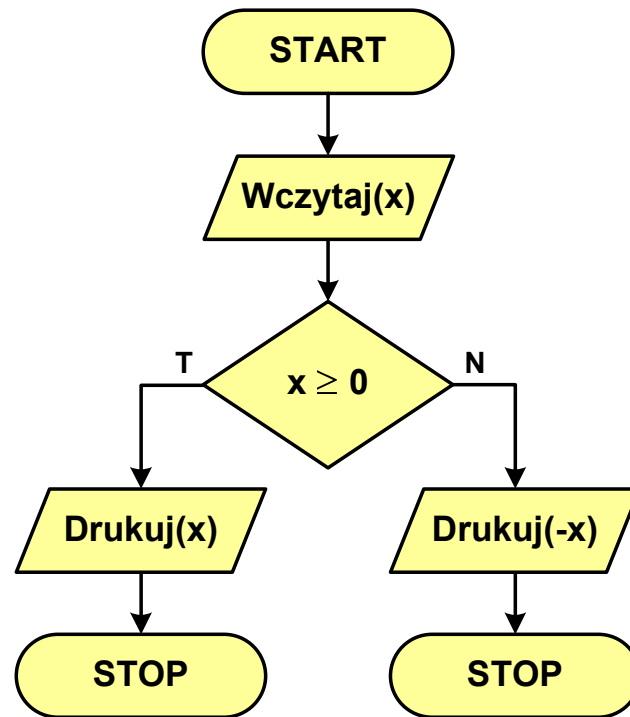
    while (a>0 && b>0)
        if (a>b)
            a = a % b;
        else
            b = b % a;

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

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

Wartość bezwzględna liczby - schemat blokowy

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



Silnia - schemat blokowy

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$$

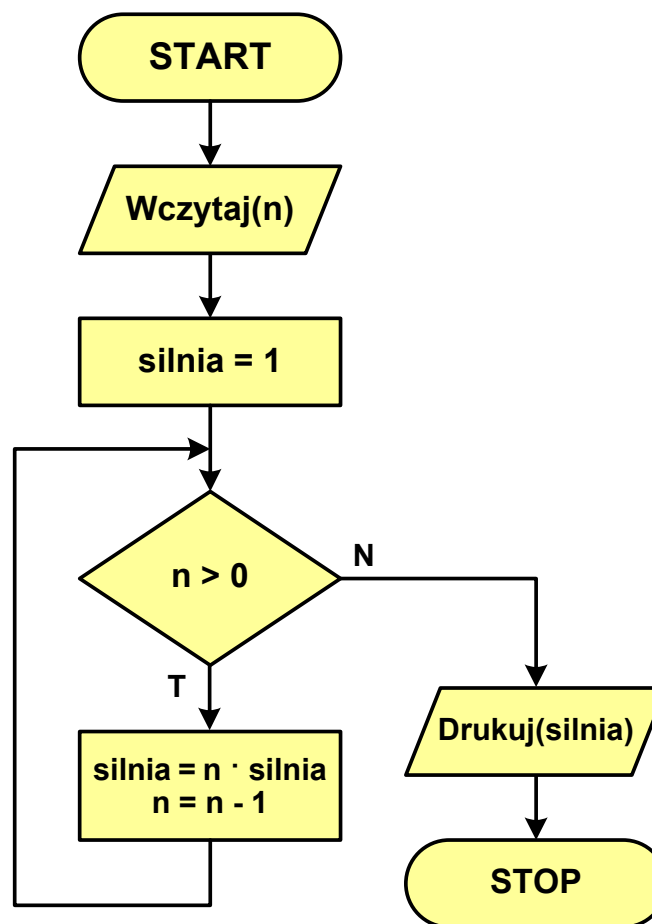
$$0! = 1$$

$$1! = 1$$

$$2! = 1 \cdot 2$$

$$3! = 1 \cdot 2 \cdot 3$$

...



Koniec wykładu nr 12

Dziękuję za uwagę!