ALGORYTM. SCHEMATY BLOKOWE. KONSTRUKCJE PROGRAMU, PODPROGRAMY, FUNKCJE

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).

Euklides - NWD

Słowo **algorytm** często kojarzone z **Euklidesem** (365 – 300 p.n.e.) i jego słynnym przepisem na **obliczanie największego wspólnego dzielnika 2 liczb a i b** (**NWD**)

Dane wejściowe **a** i **b**;
Dopóki *a>0* ***wykonuj****;*
podstaw za ***c*** *resztę z dzielenia a przez b;*
podstaw za ***b*** *liczbę a;*
podstaw za ***a*** *liczbę c;*
podstaw za ***wyn*** *liczbę bl*
rezultat *= wyn;*

Złożoność obliczeniowa

Złożoność obliczeniowa algorytmu: *ilość zasobów komputerowych potrzebnych do jego wykonania.*
W algorytmach uwzględnia się czas działania i ilość zajmowanej pamięci
czasowa T(n):
złożoność **pesymistyczna** – ilość zasobów potrzebna przy najgorszych danych wejściowych rozmiaru n,
złożoność **oczekiwana** algorytmu (średnia) dla typowych danych rozmiaru n
pamięciowa – potrzebny rozmiar pamięci (słowa maszyny)
logarytmiczna T=log(n)
liniowa T=n
Liniowo- logarytmiczna T=**n*log(n)**

Czas działania **n*log(n)** występuje np.
dla algorytmów typu:
zadanie rozmiaru n zostaje sprowadzone do 2 podzadań rozmiaru n/2
plus pewna ilość działań – liniowa względem n, do rozbicia i scalenia rozwiązań.
Przykładem jest **megasort**

Złożoność obliczeniowa
wielomianowa T=n^m , m=2,3...
wykładnicza T=**2^n**
(może też być n!)

Projektowanie algorytmów

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

Projektowanie algorytmów:

algorytmy zachłanne (nie analizujemy podproblemów dokładnie, tylko wybieramy najbardziej obiecującą w tym momencie drogę rozwiązania
algorytmy wg strategii "dziel i rządź" (dzielimy problem na kilka mniejszych, a te znowu dzielimy, aż ich rozwiązania staną się oczywiste)
algorytmy oparte na technice rekursji (rekurencji) (procedura lub funkcja wywołuje sama siebie, aż do uzyskania wyniku lub błędu)
algorytmy oparte na programowaniu dynamicznym (podproblemy)
algorytmy z powrotem
Heurystyka – na podstawie doświadczenia

Najważniejsze techniki implementacji algorytmów komputerowych

proceduralność – algorytm dzielimy na szereg podstawowych procedur, wiele algorytmów współdzieli wspólne biblioteki standardowych procedur, z których są one wywoływane w razie potrzeby,
praca sekwencyjna – wykonywanie kolejnych procedur algorytmu, według kolejności ich wywołań, na raz pracuje tylko jedna procedura,
praca wielowątkowa – procedury wykonywane są sekwencyjnie, lecz kolejność ich wykonania jest trudna do przewidzenia dla programisty
praca równoległa – wiele procedur wykonywanych jest w tym samym czasie, wymieniają się one danymi,
rekurencja – procedura lub funkcja wywołuje sama siebie, aż do uzyskania wyniku lub błędu,
obiektowość – procedury i dane łączymy w pewne klasy reprezentujące najważniejsze elementy algorytmu oraz stan wewnętrzny wykonującego je urządzenia.

Struktury danych

- **tablica**
- **lista**
- **kolejka**
- **stos**
- **zbiór**
- **graf**

Tablice w programowaniu
Arrays (macierz, tablica) jest to
podstawowa struktura danych składająca się z jednowymiarowej lub wielowymiarowej **tabeli**, którą

program traktuje jak jeden obiekt

Wszystkie obiekty muszą być tego samego rodzaju.

W tablicach mogą być np. liczby, napisy, znaki

Do każdej danej zapisanej w tablicy można się odwołać przez **nazwę tablicy** i **położenie** tej danej wewnątrz tablicy.

Lista

Kolejka

Kolejka (ang. **queue**) – liniowa struktura danych, w której nowe dane dopisywane są na końcu kolejki, a z początku kolejki pobierane są dane do dalszego przetwarzania
(bufor typu **FIFO**, First In, First Out; pierwszy na wejściu, pierwszy na wyjściu).

Stos

Przeciwieństwem kolejki jest **stos**, bufor typu **LIFO** (ang. Last In, First Out; ostatni na wejściu, pierwszy na wyjściu), w którym jako pierwsze obsługiwane są dane wprowadzone jako ostatnie.

Graf

Graf to – w uproszczeniu – zbiór wierzchołków, które mogą być połączone krawędziami, w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków.

Grafy to podstawowy obiekt rozważań **teorii grafów.**

Za pierwszego teoretyka i badacza grafów uważa się] Leonarda Eulera, który rozstrzygnął zagadnienie mostów królewieckich.

Sposoby zapisu algorytmów:

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

Opis słowny

Opis słowny jest to **zapis kolejnych kroków algorytmu w języku naturalnym** np. polskim wyszczególniamy, **jakie czynności należy wykonać**
zaznaczamy, kiedy **powinien nastąpić koniec algorytmu**

Przykład opisu słownego

Dany jest:
punkt płaszczyzny **P(xp,yp)**
środek **S(a,b)** **i** długość promienia **r** okręgu o równaniu **(x-a)2+(y-b)2=r2.**
Określić położenie punktu P względem tego okręgu.

Oznaczenia, rozwiązanie

- Oznaczamy **d** - odległość punktu od okręgu.
- Jeżeli **d<r**, to punkt leży wewnątrz okręgu,
- jeżeli **d=r**, to punkt leży na okręgu
- a gdy **d>r**, to punkt leży na zewnątrz okręgu
- Opis słowny algorytmu

- **Pobierz** wartości **xp ,yp, a, b, r.**
- **Oblicz** odległość punktu według wzoru
- **Sprawdź**, czy **d<r** , jeśli TAK,
  to „punkt wewnętrzny" i przejdź do KONIEC.
- **Sprawdź**, czy **d=r**, jeśli TAK, to
  „punkt leży na okręgu" i przejdź do KONIEC
- W innym przypadku (d>r)
  „punkt leży na zewnątrz okręgu" (d>r)
- KONIEC

Pseudokod

Opis słowny algorytmu w języku częściowo sformalizowanym, podobnym np. do Pascala

Elementy:

pseudosłowa kluczowe → odpowiadają **słowom kluczowym** składającym się na instrukcje sterujące w językach proceduralnych, np.:

jeśli – if, wtedy – then, powtórz – repeat

dopóki – while, **czytaj** – read, **pisz** - write

wyrażenia arytmetyczne → pozwalają na reprezentację obliczeń arytmetycznych

Pseudokod - elementy

1. **Początek** i **koniec**
2. **Zmienne**: całkowite, rzeczywiste, …
3. **Instrukcja warunkowa**: jeśli – if … then,
if … then … else
4. **Skok** do instrukcji: goto
5. **Etykiety**
6. **Czytaj** i **pisz**: read, write
7. **Pętle**: powtarzaj - repeat, dopóki - while, dla - for

Schemat blokowy

Reprezentacja graficzna algorytmu sekwencyjnego

- **Elementy schematu blokowego**:
- **elipsa** (lub zaokrąglony prostokąt) → początek/koniec algorytmu
- **prostokąt** → operacja
- **romb** → decyzja/warunek
- **równoległobok** → wczytanie/wyprowadzenie danych
- **strzałki** → kolejność sterowania/przepływu
- **etykiety** → oznaczenia, komentarz

*W języku angielskim schemat blokowy nazywamy **flowchart**.*

*Przykład: **Obliczenie n!** - **pseudokod***

- Policzenie silni z zadanej liczby naturalnej
- **Pseudokod**
- **wczytaj** n
- **jeżeli** n<0 idź do 1

- licznik=0, silnia=1
- **jeżeli** licznik != n **wtedy** powiększ licznik o 1, silnia=silnia*licznik, idź do 4
- **wypisz** silnia

INSTRUKCJE

Instrukcja warunkowa wyboru **if**: **jeśli W to I**

Instrukcja wyboru **if W then I1 else I2** - **alternatywa**:
jeśli W to I1 w przeciwnym przypadku I2;

Instrukcja **wyboru** – **case**
przypadek W spośród (I1, I2, …In)

Iteracje - **powtarzaj I aż do W**;

Instrukcja iteracji **while** : **dopóki W wykonuj I;**
Instrukcja iteracji **for** - **dla N:=W1 do W2 wykonuj I;**

Przykład schematu blokowego: obliczenie iloczynu n liczb
Dany jest zbiór podanych **n** liczb:
X1, X2, ..., Xn.
Należy obliczyć **iloczyn**
y=P Xi = X1*X2*...*Xn.

Algorytm

- **1) Język naturalny:**
- **Przyjmij wartość y równą 1 i skocz do B**
- **Przyjmij wartość i równą 1 i przejdź do C**
- **Przyjmij y równe iloczynowi y*Xi i skocz do B**
- **Jeżeli i=n to Koniec.**
  W przeciwnym razie przyjmij i równe i+1 i skocz do C

2) **Pseudokod** - **obliczenie iloczynu n liczb**
y :=1
i := 1
y := y*Xi
Jeżeli i=n to Koniec.
W przeciwnym razie i := i+1
i skocz do 3

Konstrukcje programu spotykane w schematach blokowych:

sekwencja instrukcji – blok
(begin… end - Pascal; { } - C, END - Basic)
Test (if, if … else),
instrukcja **wyboru** (**case, switch**),
instrukcje **iteracji**
(**while, repeat, do while, for**),
instrukcje **we/wy (read, write)**,
Procedury (procedure),
funkcje (**function**)

Sekwencja instrukcji, operacje testu
Instrukcja wyboru: **case** of , switch case , select case
Instrukcja **while**

Podprogram (funkcja lub procedura)

- **Podprogram** to wydzielona część programu wykonująca jakieś operacje.
- Podprogramy stosuje się, aby uprościć program główny i zwiększyć czytelność kodu.
- W pewnych językach programowania dzieli się podprogramy na **funkcje** i **procedury**:
- Funkcja - function
- **Funkcja** ma wykonywać obliczenia i zwracać jakąś wartość, nie powinna natomiast mieć żadnego innego wpływu na działanie programu (np. funkcja obliczająca pierwiastek kwadratowy)

Procedura

Procedura nie zwraca jednej wartości jak funkcja, zamiast tego wykonuje pewne działania (np. procedura czyszcząca ekran)
Przez zwracanie wartości należy rozumieć możliwość użycia wywołania funkcji wewnątrz wyrażenia.
Procedury często też zwracają wartości, ale poprzez odpowiednie parametry.
Podział na funkcje i procedury
Podział na funkcje i procedury występuje w językach takich jak Pascal i Ada.
W języku Pascal istnieją 2 rodzaje podprogramów: **procedury** i **funkcje**.
W pozostałych językach (m. in. w C i C++)
nie ma już takiego rozróżnienia
i funkcją jest każdy podprogram, niezależnie od tego czy zwraca jakieś wartości i czy ma wpływ na program.

Struktura programów w językach Basic, Pascal, C:

Basic

```
' Komentarz' Początek programu głównego - brak wyróżnienia
' Deklaracje zmiennych, funkcji, podprogramow
' DECLARE function funkcja1 ()
' DECLARE SUB podprogram1()
I1 : I2   ‘ Instrukcje   I1 i I2 – może być kilka w linii
In        ‘ Instrukcja In
END       ' Nie musi być jeśli po nim nie m a definicji procedur lub funkcji
```

Pascal

```
{Pascal:}
Program Nazwa;
{Komentarz}
{Deklaracje}
Begin
{Instrukcje}
I1; I2;
In;
End.
```

Język C /C++

```
// C
// Komentarz
// Deklaracje, definicje funkcji
void funkcja()
{
};
main()
{
// Instrukcje
// Sekwencja instrukcji - poczatek
{I1; I2; // Instrukcje I1; I2
In;    // Instrukcja In
} // Sekwencja instrukcji - koniec
} // koniec programu main
```

Przykłady algorytmów

- Sprawdzanie parzystości liczby całkowitej
- Określenie znaku liczby
- Obliczeni silni
- Obliczenie pierwiastka metoda iteracyjną
- Algorytm Euklidesa znajdowania największego wspólnego dzielnika
- Obliczenie kata kierunkowego (azymutu) ze współrzędnych