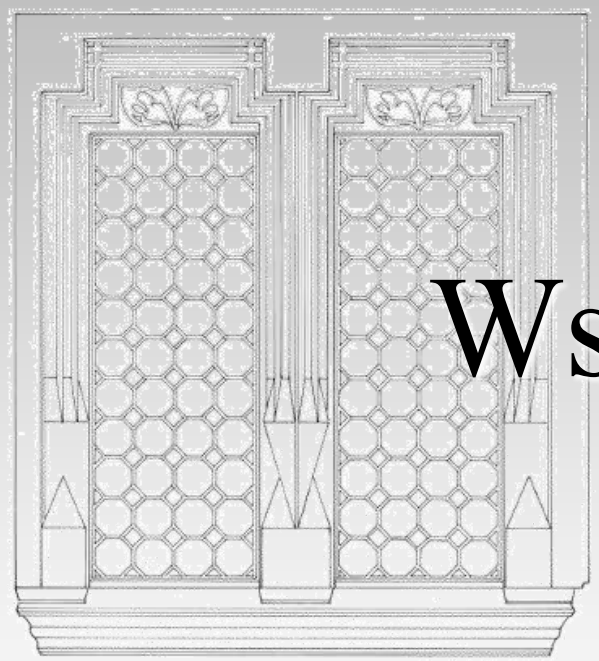
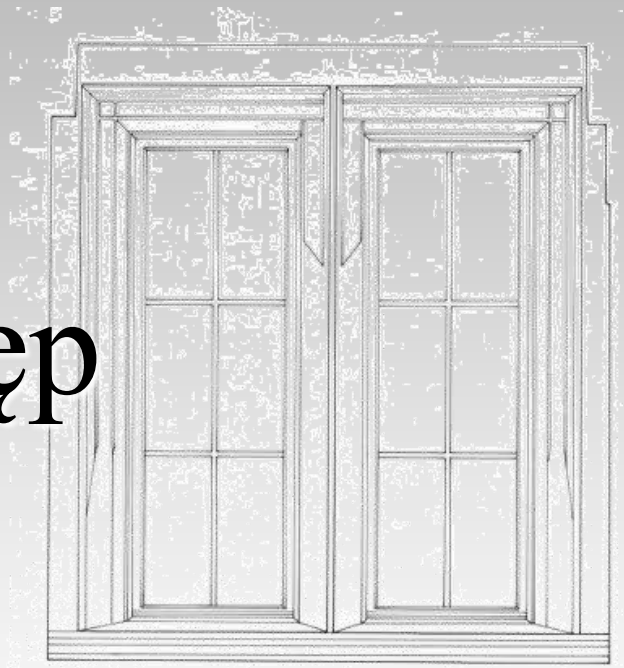


Wstęp



INFORMATYKA jest nauką

o przetwarzaniu informacji (danych).

KOMPUTER jest elektronicznym urządzeniem do przetwarzania danych.

Dane wejściowe (input data) → KOMPUTER → Dane wyjściowe (output data)

W tym sensie informatyka jest również nauką o komputerach. Jednak głównym przedmiotem informatyki jako nauki teoretycznej są *algorytmy*.

ALGORYTM to przepis na rozwiązanie problemu w skończonej liczbie kroków.

Algorytm musi być sformułowany w *języku* zrozumiałym dla podmiotu, który go wykonuje (który wg algorytmu postępuje).

PROGRAM to algorytm zapisany w języku zrozumiałym dla komputera (języku wewnętrznym komputera).

KOMPUTER wykonuje PROGRAMY

Jednostkową realizacją algorytmu jest *proces*.

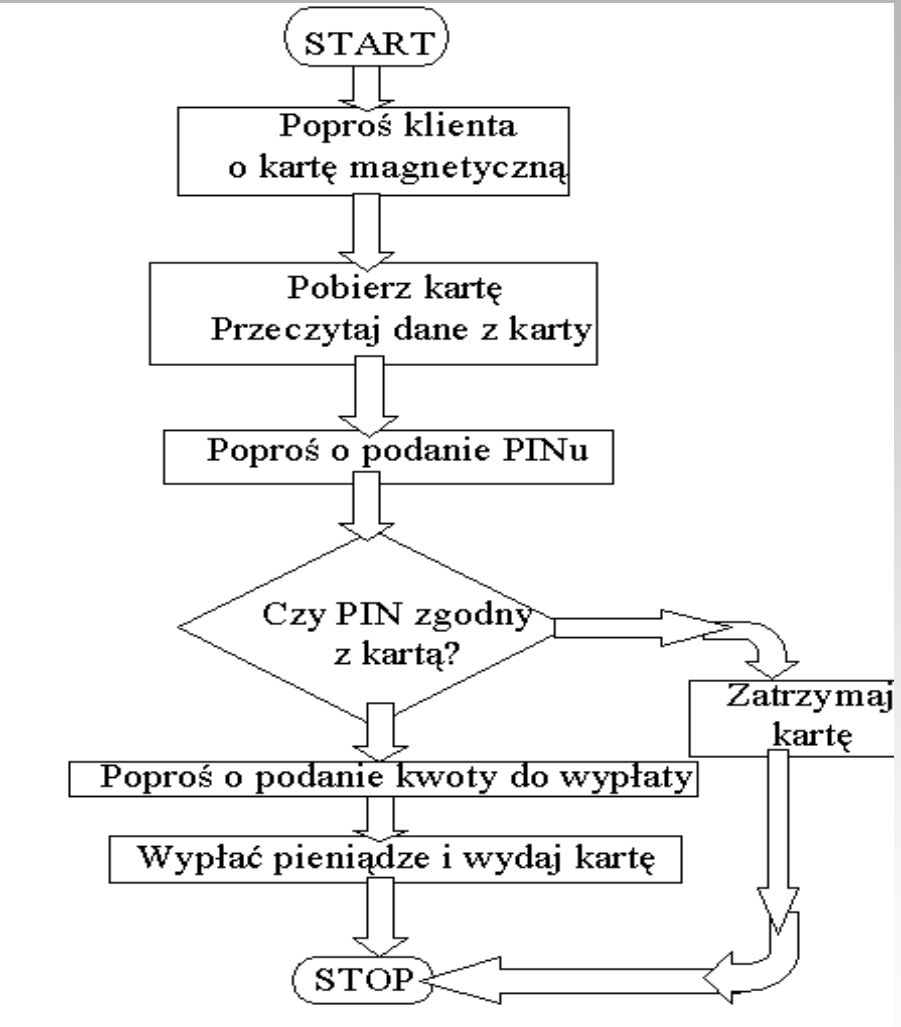
Oprócz algorytmów *sekwencyjnych* istnieją algorytmy *równoległe (współbieżne)*.

Realizacją algorytmów współbieżnych są *procesy biegnące równocześnie*, które mogą na siebie *współoddziaływać*.

Komputer jest *uniwersalnym* urządzeniem do wykonywania programów.

Uniwersalność komputera polega na możliwości wykonywania dowolnych programów zapisanych w jego języku wewnętrznym.

Algorytm pracy bankomatu



Programiści piszą programy w tzw. *wersji źródłowej* (source code), następnie poddają ją kompilacji używając programów użytkowych zwanych *kompilatorami*.

W wyniku kompilacji powstaje tzw. *wersja wykonawcza* programu, która może być uruchomiona na konkretnym komputerze pod kontrolą określonego systemu operacyjnego.

Zmiany w programie mogą być wykonywane jedynie w wersji źródłowej.

Użytkownicy otrzymują wersję wykonawczą.

Programy przeznaczone na dany typ komputera i pracujące pod danym systemem operacyjnym na ogół nie mogą być uruchomione na komputerze innego typu z innym systemem operacyjnym.

Obecnie trwają intensywne prace nad narzędziami do tworzenia *uniwersalnego oprogramowania*, które mogłoby być uruchamiane na dowolnej platformie sprzętowej i pod dowolnym systemem operacyjnym.

Generacje języków programowania

I. Kod maszynowy (język wewnętrzny).
II. Język asemblera (najprostszy język symboliczny).
III. Języki proceduralne (obiektowe) (Pascal, C, C++, Basic).
IV. Języki 4GL (Fourth Generation Languages) (Oracle, Informix, Progress, Magic). Silne wspomaganie programowe procesu tworzenia aplikacji , „ukrycie” części funkcji programu przed programistą; mała elastyczność języka, szybkie tworzenie i modyfikowanie aplikacji (RAD–Rapid Application Development)

PRZYKŁAD ALGORYTMU. Przepis kulinarny.

Wejście: 22 dag twardej czekolady półsłodkiej, 2 łyżki wody, 1/4 filiżanki cukru pudru, 6 jajek, ...
Wyjście: bity krem czekoladowy (kuchnia francuska)

Algorytm: Włóż czekoladę z dwiema łyżkami wody do garnka o podwójnym dnie. Kiedy czekolada się rozpuści domieszaj cukier puder; dodaj po trochu masło. Odstaw. Ubijaj żółtka około 5 minut, aż staną się gęste i nabiorą koloru cytrynowego. Delikatnie dołóż czekoladę. Ponownie lekko podgrzej, aby rozpuścić czekoladę, jeśli to będzie konieczne. Domieszaj rum i wanilię. Ubijaj białka aż do spienienia. Ubijając dodaj dwie łyżki cukru i ubijaj dalej, aż utworzą się sztywne pagórki. Delikatnie połącz białka z masą czekoladowo-żółtkową. Wylej do oddzielnych naczyń, które będą podane na stół. Ochładzaj przez co najmniej 4 godziny. Wedle życzenia, podawaj z bitą śmietaną. Wyjdzie z tego 6 do 8 porcji.

Powyższy algorytm może być również zapisany jako algorytm równoległy.

Podstawowe koncepcje algorytmiczne w przepisie kulinarnym:

- sekwencje czynności

- warunkowe wykonanie

- powtarzanie, aż zajdzie warunek

- zestawy czynności zdefiniowane wcześniej

PRZYKŁAD ALGORYTMU. Obliczanie największego wspólnego dzielnika.

Wejście: a,b - liczby naturalne > 1

Wyjście: c = NWD(a,b)

1. Wypisz czynniki pierwsze liczby a, powstaje lista A={a_1,a_2,...,a_n} (mogą wystąpić powtórzenia).
2. Wypisz czynniki pierwsze liczby b, powstaje lista B={b_1,b_2,...,b_m} (mogą wystąpić powtórzenia).
3. Utwórz C jako listę wspólnych elementów list A i B (też z ewentualnymi powtórzeniami).
4. Oblicz c jako iloczyn wszystkich elementów z C (jeśli C jest pusta to c=1).
5. Wypisz c i zatrzymaj się.

../Przykłady/NWD.pascal

Powyższy algorytm posługuje się pojęciami, które wymagają dookreślenia:

-jak znajdować czynniki pierwsze?

- jak obliczać część wspólną list ?

PRZYKŁAD ALGORYTMU: Sumowanie listy płac

Wejście: kartoteka pracowników składająca się z rekordów o strukturze: nazwisko, płaca, inne
Wyjście: suma zarobków wszystkich osób

1. Zanotuj “na boku” liczbę 0.
2. Przejrzyj kolejno rekordy kartoteki pracowników dodając zarobki każdej osoby do liczby “na boku”.
3. Kiedy osiągniesz koniec listy przedstaw wartość liczby “na boku” jako wynik.

Ważna cecha tego algorytmu: Wielkość programu taka sama dla dowolnie długich list płac.

../Przykłady/SumRecord.pascal

Podstawowe koncepcje algorytmiczne

Stała to: *liczba, znak, ciąg znaków*

W skompilowanym programie zajmuje określone miejsce w pamięci komputera (komórka lub ciąg sąsiednich komórek). Zawartość tego miejsca nie jest zmieniana w trakcie algorytmu.

w programie źródłowym zapisywana :

- jawnie, np. liczba -138.24, znak ‘A’, ciąg znaków ‘ala ma kota’, albo
-przez przypisanie nazwy (identyfikatora) określonej wartości, i posługiwanie się dalej tą nazwą

Podstawowe koncepcje algorytmiczne

Definicja stałej:

```
const  pi = 3.1415;          // na ogół na początku programu
                                  // to jest komentarz
```

. . . .

Instrukcja (gdzieś dalej w algorytmie):

```
 pole_kola := pi * r * r ;          // to instrukcja podstawienia
```

Podstawowe koncepcje algorytmiczne

Zmienna

Po kompilacji programu jest to komórka pamięci komputera (lub ciąg sąsiednich komórek), której zawartość może być zmieniana w algorytmie.

Zmienna ma nazwę (identyfikator, kojarzony przez kompilator z adresem zmiennej w pamięci operacyjnej).

Zawartość komórki, to wartość zmiennej.

Podstawowe koncepcje algorytmiczne

Typ zmiennej (stałej), to zbiór wartości, które może przyjmować zmienna (lub przyjęła stała)

Typ określa sposób interpretowania zawartości komórki.

Przykłady typów:
- liczbowy (podtypy: całkowity (integer), rzeczywisty (real), i in.)
- tekstowy
-logiczny (boolean – ma dwie możliwe wartości:
PRAWDA - *true* oraz FAŁSZ - *false*)

W opisach algorytmów, jeśli nie będzie niejednoznaczności, deklaracje zmiennych typów liczbowych lub tekstowych będziemy pomijać.

Podstawowe koncepcje algorytmiczne

W językach programowania deklaracje typu zmiennych są na ogół konieczne (i pożyteczne).

Deklaracja zmiennych:

var temp : integer;

lub

var x, y : integer;

Podstawowe koncepcje algorytmiczne

Instrukcja przypisania (podstawienia) ma postać ogólną:

zmienna := wyrażenie

W jej wyniku zmienna przyjmuje wartość równą wartości wyrażenia.

Innymi słowy, dotychczasowa zawartość komórki zostaje zastąpiona wartością wyliczoną jako wynik wyrażenia.
Wyrażeniem jest np. wyrażenie arytmetyczne, zawierające stałe i zmienne lub np. stała tekstowa. Każdy język programowania określa precyzyjnie postać wyrażenia.

Podstawowe koncepcje algorytmiczne

Następujące po sobie instrukcje algorytmu są oddzielane średnikami (;).

Podstawowe koncepcje algorytmiczne

PRZYKŁADY

```
beta  :=  88;
gamma := 2 * pi * r;
i := i + 1;              // zmienna może wystąpić po obu stronach !
                         // poprzednia wartość zmiennej  i  wynosiła 12,
                        // po tej instrukcji wartość i wynosi 13
delta := b * b – 4 * a * c;
x1 := 2 * (x1 – 1)  + x2;
```

Przed podstawieniem: x1 = -3, x2 = 6

Po podstawieniu: x1 = -2, x2 = 6

```
linia := ‘Komunikat nr 1’; // podstawienie ciągu znakowego
```

Podstawowe koncepcje algorytmiczne

Jak zamienić wartości dwóch zmiennych x, y miedzy sobą ?

Należy użyć trzeciej zmiennej, pomocniczej, tego samego typu:

```
temp := x;
x := y;
y := temp;
```

Podstawowe koncepcje algorytmiczne

Kolejność wykonywania instrukcji (przepływu sterowania) w algorytmie

Bezpośrednie następstwo

Kolejne instrukcje algorytmu wykonywane są jedna po drugiej

```
a := 5;
b := 25;
b := b / a;
a :=  b / a;
a := b / a;
a := a – 1;
```

wynik: a = 4, b = 5

Podstawowe koncepcje algorytmiczne

Instrukcja warunkowa (wybór warunkowy) **„if … then ...”**
(„jeśli … to ...”)

if warunek **then** instrukcja

warunek - wyrażenie, którego wartość jest logiczna (*true* lub *false*)

instrukcja - dowolna

Znaczenie instrukcji **„if ... then ...”** :

jeżeli warunek jest w danej chwili spełniony, to wykonywana jest instrukcja po słowie **then**, w przeciwnym przypadku jest ona pomijana.

Podstawowe koncepcje algorytmiczne

PRZYKŁADY

```
a := 12;
if  a > 10  then  a := a / 2;
b := a * a;
```

wynik: b = 36

```
a := 9;
if   a > 10  then  a := a / 2;
b := a * a;
```

wynik: b = 81

Podstawowe koncepcje algorytmiczne

Postać “if ... then ... else”
(“jeśli … to … w przeciwnym przypadku ...”) **instrukcji warunkowej**

if warunek **then** instrukcja-1 **else** instrukcja-2

Instrukcja-1 oraz instrukcja-2 wzajemnie się wykluczają.
Zawsze dokładnie jedna z nich jest wykonywana. To, która jest wykonywana, zależy od spełnienia warunku.

Podstawowe koncepcje algorytmiczne

PRZYKŁAD

if x > 0 **then** wynik := x **else** wynik := -x
// wynikiem jest wartość bezwzględna zmiennej x

Podstawowe koncepcje algorytmiczne

Zagnieżdżanie instrukcji

Jeśli istnieje potrzeba warunkowego wykonania więcej niż jednej instrukcji, to ciąg instrukcji ujęty w słowa **begin** oraz **end** traktowany jest jako jedna instrukcja, i jest nazywany *blokiem instrukcji* lub *instrukcją złożoną*.

Podstawowe koncepcje algorytmiczne

PRZYKŁADY

if r1 + r2 = d **then begin** n := n+1; tekst := ‘styczne’ **end**

```
if  zysk > dolny-limit  then
   begin
      wczasy := ‘adriatyk’;
      na-meble := 4000
   end
else
   begin
         wczasy := ‘pod gruszą’;
         na-meble := 2000
   end;
```

Podstawowe koncepcje algorytmiczne

Instrukcje **if** mogą być zagnieżdżone:

```
if  x > 0  then   s := 1  else
        if  x < 0   then  s := -1  else  s := 0;
```

(trzy wykluczające się wzajemnie instrukcje podstawienia)