Algorytm i program komputerowy

Aby powstał program komputerowy niezbędnym jest przejście drogi zacząwszy od zadania, algorytmu rozwiązania i tworzenia programu.

Algorytm

Algorytm to sposób (schemat) postępowania prowadzący do rozwiązania problemu z określonej klasy w skończonej liczbie kroków. Obejmuje on skończony ciąg operacji na obiektach (np. liczbach, relacjach między liczbami typu a>b itp) ze ściśle ustalonym porządkiem ich wykonywania, dający możliwość realizacji zadania z określonej klasy.

Program komputerowy

To zespół poleń zapisywanych w postaci instrukcji, określający dokładnie przebieg wykonywanych w komputerze operacji arytmetycznych i logicznych.

Dalej podano podstawowe pojęcia dotyczące procesu tworzenia programu.

Algorytm w sensie informatycznym algorytm wykorzystuje określone dane o zdefiniowanych strukturach (*np. liczby całkowite, rzeczywiste, zespolone, tablice jedno i wielowymiarowe, rekordy itp*) i daje określone wyniki.

Algorytm powinien posiadać cechy:

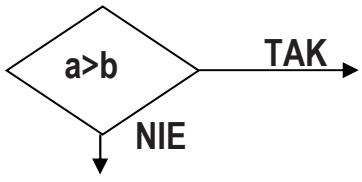
- ***Poprawności*** *– dla każdego zestawu poprawnych danych wyniki powinny być poprawne;*
- ***Skończoności****- dla każdego zestawu poprawnych danych algorytm powinien dawać poprawne wyniki po skończonej liczbie kroków,*
- ***Określoności*** *– z algorytmu musi wynikać jednoznacznie jaka ma być następna operacja,*
- ***Efektywności****- algorytm powinien realizować zadanie przy najmniejszej liczbie kroków,*
- ***Uniwersalności****- powinien rozwiązywać nie tylko specyficzne zadanie ale całą klasę zadań.*

Algorytm można zapisać w postaci:

- ***Opisu słownego,***
- ***Listy kroków,***
- ***Schematu blokowego*** *(flow diagram, flow chart), (często stosowana)*
- ***Pseudokodu***

Opis słowny algorytmu jest ogólną ale mało precyzyjną formą prezentacji . Niesie ona ryzyko niejednoznaczności przy złożonych algorytmach. Stosuje się ją dla zasugerowania rozwiązania. Popularnym sposobem jest lista kroków postępowania. Kroki stanowią opis operacji i są zazwyczaj mieszaniną sformułowań matematycznych i języka naturalnego. Kolejność kroków jest zgodna z opisem. Wyjątkiem jest operacja przejścia do wskazanego kroku lub też zakończenia algorytmu.

Schemat blokowy (sieć działań) składa się z symboli graficznych w których opisywane są operacje algorytmu. np. zmiany kolejności kroków



Stosuje się różne kształty symboli dla rozróżnienia operacji. Następstwo operacji określają strzałki. Zapewnia dobrą komunikacją między „zleceniodawcą" a programistą ale trudno jest przedstawić złożone problemy. Dlatego stosuje się różne poziomy szczegółowości (od ogółu do szczegółu- schemat zstępujący).
Pseudokod jest wygodną formą prezentacji algorytmów. Bazuje na opisie w języku zbliżonym do bardzo ogólnej formy języka programowania (np. typu Pascal). Występują tu bardzo uproszczone formy opisu operacji wprowadzania i wyprowadzania danych. W opisach algorytmów przy wykorzystaniu pseudokodu stosuje się typowe nazwy instrukcji występujące w niektórych językach wysokopoziomowych jak np:

- **If (warunek) then** ***instrukcja1***
  else ***instrukcja2***
- **for** *zmienna:=war_poczatkowa* **step** *war_kroku* **until** *war_koncowa* **do**
  instrukcja lub blok instrukcji,

W powyższym zapisie pierwsza konstrukcja zmienia sterowanie kolejności wykonywania instrukcji (kroków). Drugi z zapisów określa tzw. pętlę, która pozwala na wielokrotne powtarzanie wykonania określonych instrukcji. Pętla for jest stosowana jeśli dokładnie wiadomo ile razy ma nastąpić powtarzanie. Jeśli liczba powtórzeń nie jest znana to stosowane są inne konstrukcje pętli np. pętla typu:

while (***warunek***) **do**
instrukcja lub blok instrukcji

. Wówczas badany jest pewien warunek logiczny i tylko w przypadku jego prawdziwości następuje kolejne wykonanie pętli.

Wśród algorytmów wyróżnia się:
Innym podział wyróżnia algorytmy:

- ***liniowe (sekwencyjne)*** *– poszczególne kroki wykonywane są jeden po drugim,*
- ***rozgałęzione*** *- w wyniku sprawdzenia warunku logicznego (może być prawdziwym lub fałszywym) mogą być dalej wykonywane różne kroki algorytmu (np. wybór dalszej drogi postępowania np. przy obliczaniu pierwiastków równania kwadratowego),*
- ***iteracyjne*** *– w których pewien zespół kroków jest wielokrotnie powtarzany (pętla), aż do momentu spełnienia określonego warunku,*
- ***rekurencyjne*** *– to algorytmy które wywołują same siebie dla zmieniających się danych,*
- *proces wykonywania zadania wywołuje same siebie,*
- ***mieszane*** *– łączące cechy różnych algorytmów.*

Algorytm powinien posiadać **specyfikację** gdzie określamy **dane** z jakich algorytm korzysta oraz **wyniki** które powinien dawać, a także niezbędne **zmienne** pomocnicze – jeśli są one wymagane.

W algorytmach mogą również występować **wyrażenia** zbudowane ze zmiennych, stałych, operatorów (np. +, - , *, /) i funkcji matematycznych. Wyrażenia mogą też mieć postać **wyrażeń warunkowych**. Wówczas do ich budowy wykorzystywane są operatory relacyjne (np. =, >, <, <=, >=, <>).

W algorytmach występują często tzw. **operacje** przypisania zapisywane w różny sposób (np. x:=x +5, x=x+5), określające, że obliczona wartość wyrażenia z prawej strony zostanie podstawiana pod zmienną ze strony lewej.

Idea formułowania prostego algorytmu

Zadanie

Sformułuj algorytm obliczający pole prostokąta. Zapisz algorytm w postaci listy kroków, schematu blokowego i pseudokodu.

Specyfikacja algorytmu:

Dane wejściowe:

a- pierwszy bok, liczba rzeczywista dodatnia- w cm,

b- drugi bok, liczba rzeczywista dodatnia- w cm.

Wynik: p –pole prostokąta (w cm^2)

a) Lista kroków:

K1: Wczytaj wartość pierwszego boku i podstaw pod zmienną a,

K2: Wczytaj wartość drugiego boku i podstaw pod zmienną b,

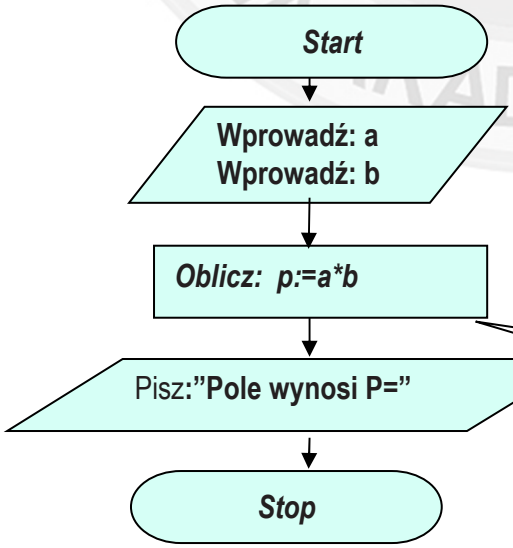
K3: Oblicz pole p=a*b i podstaw pod zmienną p

K4: Wyświetl wynik p

b) Pseudokod:

```
Algorytm pole                          {nagłówek}
zmienne
      a,b,p: real                      {deklaracja zmiennych}
begin                                  {początek właściwego programu}
   czytaj (a)                          {wczytanie danych}
   czytaj(b)
   p:=a*b                              {obliczenia}
   wyświetl(„Pole wynosi  P=”,p)       {wyświetl wynik}
end                                    {koniec właściwego programu}
```

c) Schemat blokowy:

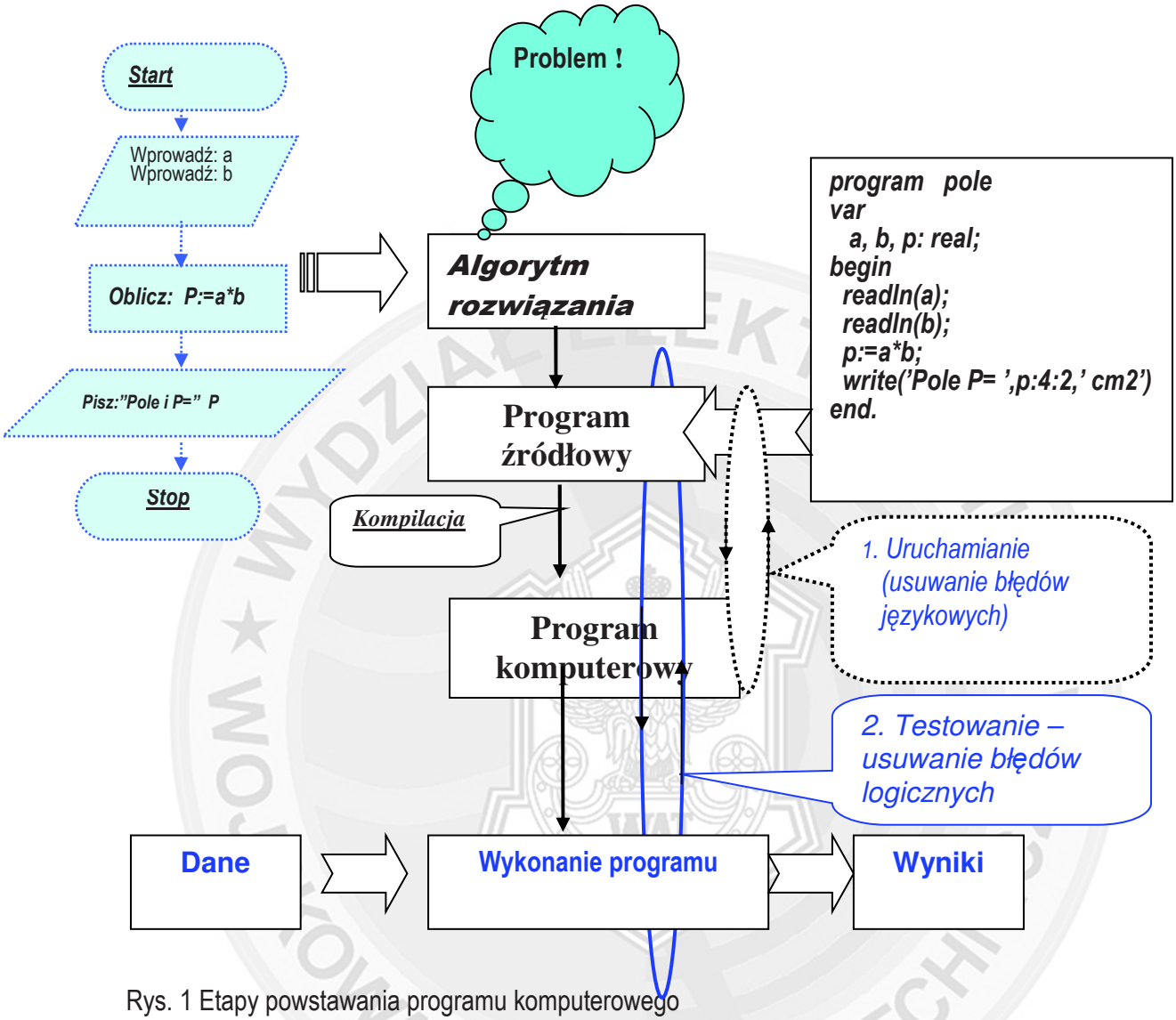


d) Zapis w języku programowania (np. Pascal)

```
program   pole
var
   a, b, p: real;
begin
  readln(a);
  readln(b);
  p:=a*b;
  write('Pole P= ',p:4:2,'cm2')
end.
```

W Wordzie elementy graficzne występują w zakładce : Wstawianie) → Kształty: Schematy blokowe, Strzałki blokowe

Rozwiązanie danego problemu przy zastosowaniu przygotowanego ***programu komputerowego*** wymaga realizacji wielu etapów projektowych i uruchomieniowych. Problem ten przedstawiono na poniższym rysunku, zakładając, że dotyczy on programu obliczającego pole prostokąta.



Rys. 1 Etapy powstawania programu komputerowego

Na rysunku pokazano drogę od problemu, który wymaga rozwiązania do programu komputerowego rozwiązującego problem, przy założeniu, że potrafimy poprawnie sformułować algorytm. Pętle sugerują, że niektóre etapy będą powtarzane z powodu różnego rodzaju błędów, których trudno uniknąć przy bardziej złożonych problemach. Pierwsza grupa błędów wynika zwykle z błędnych zapisów algorytmu w języku programowania (błędy składni języka). Druga grupa dotyczy błędów wykonania wynikających z niepoprawnego algorytmu lub niewłaściwego kodowania w danym języku. Aby **skompilować** program (*tzn. przetłumaczyć go na ciąg rozkazów w zapisie zero-jedynkowym zrozumiałym dla komputera*) to musi on być całkowicie zgodny ze standardem języka (tu Pascala). Do tworzonego programu dołączane **(linkowanie**) są automatycznie różnego rodzaju programy istniejące już w bibliotekach danego języka (np. funkcje obsługujące klawiaturę, obliczające pierwiastek z zadanej liczby itp.) Ale uzyskanie programu poprawnego językowo nie musi oznaczać, że poprawnie rozwiązuje problem. Stąd może ponownie wynikać konieczność zmiany jego wersji źródłowej – co pokazuje większa pętla. Czasami również będzie konieczność jej rozszerzenia obejmującej poprawienie algorytmu.

Algorytmy iteracyjne

W sytuacjach praktycznych często występuje konieczność powtarzania pewnych obliczeń. W algorytmach i programowaniu służą do tego pętle. Dla każdej pętli muszą być określone:

- warunki początkowe (rozpoczęcia pętli);
- operacje wykonywane w pętli;
- warunek zakończenia pętli

Np. w przypadku sumowania n liczb kolejną liczbę dodajemy do dotychczasowego wyniku dodawania. Wielokrotne dodawanie zastępujemy wówczas jedną, powtarzaną n krotnie operacją dodawania. A pierwszy istniejący wynik musi mieć wartość 0 nadaną mu przed zainicjowaniem pętli !. Warunek zakończenia pętli jest istotny bo złe jego określenie może spowodować pętlę nieskończoną. Warunek zakończenia pętli może występować:

- na początku pętli- wówczas decyduje czy instrukcje pętli będą wykonane czy pominięte.
- na końcu pętli - wówczas zapewnione jest przynajmniej jednokrotne wykonanie instrukcji.

Pętle nieskończone są czasami stosowane świadomie i odgrywają istotną rolę w programowaniu. Są stosowane wtedy gdy nie można przewidzieć ile razy pętla ma być powtórzona.
Fizyczną interpretację można podać obserwując kasjerkę w sklepie, która nie wie ile jest artykułów w koszyku. Przed przystąpieniem do obsługi musi ona wyzerować rachunek (nadanie wartości początkowej). Następnie pobiera z koszyka kolejny towar, odczytuje jego cenę i dodaje do dotychczasowej sumy. Tę czynność powtarza dotąd dopóki koszyk nie jest pusty. Tu stwierdzenie że koszyk jest pusty jest warunkiem logicznym. Jeśli warunek jest prawdą to następuje zakończenie pętli.