WIADOMOŚCI WSTĘPNE

Programowanie komputerów najogólniej mówiąc polega na zapisaniu pewniej listy poleceń do wykonania przez komputer w pewnym umownym języku. Taką listę poleceń nazywamy **programem**. Program jest to po prostu tekst zapisany w symbolice obowiązującej w danym **języku programowania**, który może być zrozumiały przez komputer.

Trzeba też podkreślić, że komputer bez programu sam nie może nic wykonać, musi być poinstruowany przez człowieka o tym, co ma robić.

Kolejne fazy procesu rozwiązywania problemów, za pomocą komputera to:

- sformułowanie problemu (specyfikacja),
- analiza problemu i znalezienie metody jego rozwiązania tzn. algorytmu,
- zakodowanie algorytmu (napisanie programu),
- uruchomienie i przetestowanie programu,
- konserwacja oprogramowania,

Najważniejszą fazą jest przeprowadzenie dokładnej analizy problemu, który należy rozwiązać. Analiza ta powinna doprowadzić do sformułowania metody rozwiązania problemu tzn. **algorytmu**.

Program w postaci ciągu instrukcji języka nazywa się **programem źródłowym**, natomiast program binarny uzyskany w wyniku kompilacji **programem wynikowym**. Tłumaczenie, czyli **kompilacja** ma na celu zastąpienia operacji elementarnych danego języka programowania operacjami elementarnymi komputera. Do tłumaczenia używane są specjalne programy nazywane **kompilatorami**. W naszym przypadku będzie wykorzystywany **kompilator języka Turbo Pascal.**

Kompilator może sam wykryć niektóre błędy popełnione przez programistę. Błędy te można porównać do błędów ortograficznych i składniowych języka naturalnego. Nazywa się je **błędami kompilacji**. Osobną kategorią błędów, które muszą być wykryte przez programistę, są **błędy logiczne**. Są to błędy, które sprawiają, że program działa nieprawidłowo np. podaje niepoprawne wyniki, mimo iż nie jest sygnalizowany żaden błąd. Błędy logiczne wynikają najczęściej z błędów popełnionych w fazie projektowania algorytmu. Są to błędy, które mogą być często bardzo trudne do wykrycia i poprawienia.

WPROWADZENIE DO JĘZYKA TURBO PASCAL

Po uruchomieniu programu (**Start -> Turbo Pascal 7.0**) należy ściągnąć dany programu z dysku do edytora (jeżli jest on już napisany). Do tego celu wykorzystuje się klawisz **F3**. Ukaże się wtedy okno, w którym można wybrać dowolny plik umieszczony w dowolnym katalogu na dysku. Po naciśnięciu klawisza Enter kursor przechodzi do ramki, w której umieszczone są wszystkie pliki z danego katalogu. Przy pomocy strzałek można wybrać dowolny plik, a po naciśnięciu klawisza Enter plik ten zostanie umieszczony w głównym oknie, w którym dokonuje się jego edycji. Zapamiętanie edytowanego programu pod aktualną nazwą odbywa się po naciśnięciu klawisza **F2**. Aby skompilować edytowany program należy nacisnąć jednocześnie dwa klawisze **Alt-F9**. Jeżeli występują błędy kompilacji pojawia się komunikat informujący o rodzaju błędu i kursor ustawia się w miejscu, w którym został stwierdzony błąd.

Uruchomienie skompilowanego programu następuje po naciśnięciu kombinacji klawiszy **Ctrl-F9** (również kompilacja, jeśli dokonano zmian w programie), natomiast aby obejrzeć wyprowadzone wyniki należy wcisnąć kombinację klawiszy **Alt-F5**. Wyjście z systemu Turbo Pascal następuje po wciśnięciu klawiszy **Alt-X**.

Klawisze skrótów

F3	- ściągnięcia programu
F2	- Zapamiętanie programu
Alt-F9	- Skompilowanie programu
Ctrl-F9	- Uruchomienie programu
Alt-F5	- Wyświetlenie ekranu użytkownika

W języku Pascal istnieją tak zwane **słowa kluczowe**. Są to wyróżnione słowa, które mają ściśle określone znaczenie. Zarówno słowa kluczowe jak i inne nazwy używane w programie mogą być zapisywane zarówno przy wykorzystaniu małych jak i dużych liter. W każdym miejscu programu mogą występować **komentarze** czyli dowolne opisy. Komentarz rozpoczyna się nawiasem klamrowym otwierającym i jest zakończony nawiasem klamrowym zamykającym i służy do omówienia funkcji danego programu czy podprogramu.

Przykład:

```
{to jest właśnie komentarz, można tu umieścić dowolny opis}
```

STRUKTURA PROGRAMU

Program w języku Pascal ma postać.

```
program nazwa;
     .
     . deklaracje zmiennych
     .
begin
.
. instrukcje (część wykonywalna programu)
.
end.
```

Nazwa jest dowolną nazwą używaną do zidentyfikowania programu. **Deklaracje** służą do określenia struktury danych programu czyli obiektów, na których działa program. **Instrukcje** natomiast opisują czynności, jakie powinien wykonać program w celu zrealizowania danego algorytmu. Instrukcje programu oddzielane są od siebie za pomocą średnika (przed instrukcją **end** średnik nie jest konieczny). Program powinien być zapisywany z wykorzystaniem wcięć tak, aby uwidocznić strukturę poszczególnych instrukcji.

TYPY DANYCH

Podstawowymi obiektami występującymi w programie są **stałe** i **zmienne**. Zmienne wykorzystywane w języku Pascal mogą być różnych typów. Typ zmiennej definiuje rodzaj możliwych wartości, które może przyjmować zmienna. Zmienne deklaruje się w części deklaracyjnej programu rozpoczynającej się od słowa kluczowego **var**. Podaje się listę zmiennych oddzielonych przecinkami i następnie po dwukropku nazwę typu zmiennych.

Np.

```
VAR lista_zmiennych: identyfikator_typu;
```

Stałe definiuje się w części deklaracyjnej programu rozpoczynającej się od słowa kluczowego **const**.

Np.

```
CONST nazwa_zmiennej=wartość;
```

- **Typ całkowity**. Nazwą typu jest słowo kluczowe **integer**. Zmienna zadeklarowana jako typ integer może przyjmować wartości całkowite z pewnego zakresu zależnego od typu komputera. **(-32768** do **32767**)

- **Typ rzeczywisty**. Nazwą typu jest słowo kluczowe **real**. Zmienna zadeklarowana jako typu redl może przyjmować wartości z podzbioru liczb rzeczywistych. **(-2.9e-39** do **1.7e38**) Możliwe postacie zapisu liczby rzeczywistej ilustrują poniższe przykłady:
  1. **15.67** – zapis zwykły
  2. **83.5E+3** – zapis wykładniczy

- **Typ znakowy**. Nazwą typu znakowego jest słowo kluczowe **char**. Zmienna zadeklarowana jako char może przechowywać pojedyncze znaki dostępne w danym systemie komputerowym (rozszerzony kod ASCII).

- **Typ łańcuchowy**. Nazwą typu łańcuchowego jest słowo kluczowe **string**. Zmienna zadeklarowana jako string może zawierać łańcuchy znaków o długości od 0 do długości podanej w deklaracji.

- **Typ logiczny**. Nazwą typu jest słowo kluczowe **boolean**. Zmienna typu boolean może przechowywać wartości logiczne prawdy – **true** lub fałszu – **false**.

WYRAŻENIA

Wyrażenia są zapisami operacji, które mogą być wykonywane na elementach języka. Wyrażenie składa się ze stałych, zmiennych i łączących je operatorów. Kolejność wykonywania działań określają nawiasy oraz priorytet operatorów. Obecnie wymienimy niektóre operatory dostępne w języku Turbo Pascal.

Operator	Znaczenie
*	Mnożenie
/	Dzielenie
+	Dodawanie
-	Odejmowanie
div	Dzielenie całkowite
mod	Reszta z dzielenia

INSTRUKCJE

- **Instrukcja przypisania**. Służy do przypisania zmiennym pewnej wartości. Jest ona postaci:

```
zmienna := wyrażenie
```

wykonanie tej instrukcji powoduje przypisanie (podstawienie) wartości wyrażenia zapisanego po prawej stronie symbolu **:=** zmiennej stojącej po lewej stronie.
Przykład:

```
a := 10;
b := 15;
c := a + b {zmienna c przyjmuje wartość 25}
```

- **Instrukcja wejścia/wyjścia**. Są to podstawowe instrukcje umożliwiające komunikację programu z użytkownikiem.
  - ***wczytywanie danych***. Wykorzystuje się tu instrukcję **read** oraz **readln**. Instrukcja read ma postać.

```
read(lista_argumentów)
```

gdzie **lista_argumentów** zawiera ciąg zmiennych oddzielonych przecinkami. Wykonanie instrukcji read spowoduje wprowadzenie ciągu danych, których wartości zostaną przypisane zmiennym wyszczególnionym na **liście_argumentów**. Postać instrukcji **readln** jak i jej działanie jest takie samo jak instrukcji **read** z tym, że po wprowadzeniu danych następuje przesunięcie wskaźnika w pliku wejściowym do następnego wiersza (and. ***L**ine **N**ew*). Instrukcję Readln bez parametrów można wykorzystać do zatrzymania programu do chwili naciśnięcia <ENTER>.

  - ***wyprowadzanie wyników/komunikatów***. Wykorzystuje się w tym przypadku instrukcję **write** oraz **writeln**. Postać instrukcji writeln jest następująca:

```
writeln(lista_argumentów)
```

gdzie **lista_argumentów** jest ciągiem stałych, zmiennych lub wyrażeń oddzielonych przecinkami. Wykonanie instrukcji writeln spowoduje wprowadzenie ciągu wartości. Po wprowadzeniu wszystkich wartości elementów umieszczonych na **liście_argumentów** następuje przejście do nowej linii. Postać instrukcji write i jej działanie jest takie samo jak instrukcji writeln z tym, że przy wprowadzeniu danych nie następuje przejście do nowej linii. Kolejne wykonania instrukcji write powodują zatem wprowadzenie wartości w tej samej linii.

Przykład:

```
Program dialog;
var z1, z2, z3: string[15];
begin
     Write(‘ Podaj swoje imie: ‘);
     Redln(z1);
     Writeln(‘Czesc ‘,z1. Ja nazywam się IBM PC. ‘);
     Write(‘Ile masz lat? ‘);
     Readln(z2);
     Writeln(‘Do jakiej chodzisz szkoły?’ );
     Readln(z3);
     Write(‘Dziękuję za informację. ‘);
     Writeln(‘Zobacz, ze wszystko zapamiętałem. ‘);
     Writeln(‘****** ‘,z1, ‘ ma ‘, z2, ‘ lat ‘, ‘*****’);
     Writeln(‘***** Uczy się w ‘, z3, ‘*****);
end.
```

- **Instrukcja warunkowa IF** (w języku umownym **Jeśli**)
  - Instrukcja warunkowa prosta

```
if warunek then instrukcja
```

  - Instrukcja warunkowa z alternatywą

```
if warunek then instrukcja_1 else instrukcja_2
```

gdzie **warunek** jest wyrażeniem logicznym przyjmującym dwie wartości: prawdy i fałszu. Instrukcję **if** wykonuje się następująco. Jeśli warunek jest spełniony, to jest wykonywana **instrukcja_l**, a w przeciwnym przypadku ***instrukcja_2***. ***Instrukcja_2*** może nie występować i wtedy **instrukcja_l** jest albo wykonywana albo nie w zależności od wartości, jaką przyjmuje *warunek*.

Przykład:

```
if a>b then writeln('a jest większe od b')
     else writeln('a nie jest większe od b");
if a<>0 then
     Begin
          x:= -b/a
          writeln (x:6:2)
     end;
```

- **Instrukcja REPEAT** realizuje zdanie **powtarzaj**. Instrukcja ta ma postać:

```
repeat
     instrukcja_l;
     {...}
     instrukcja_n;
until warunek;
```

Działanie instrukcji **repeat** jest następujące. Wykonywane są wszystkie instrukcje umieszczone pomiędzy słowem kluczowym **repeat**, a słowem kluczowym **until**. Wykonywanie tych instrukcji kończy się, jeżeli wartość warunku warunek jest **true**, a w przeciwnym przypadku powtarzane jest ponownie.

Przykład:

```
Program petla_1;
var ile:integer;
begin
ile:=1
repeat
     writeln('uczę się Pascala');
     ile:=ile+1
until ile>10;
```

- **Instrukcja WHILE** realizuje zdanie **dopóki**. Instrukcja ta ma postać:

```
while warunek do
     instrukcja;
```

Instrukcja **while** powoduje wykonywanie **instrukcji** tak długo, jak długo spełniony jest **warunek**. Instrukcja może być instrukcją prostą lub złożoną.

Przykład:

```
Program petla_2;
var ile:integer;
begin
ile:=1
while ile<=10 do
     begin
     writeln('uczę się Pascala');
     ile:=ile+1
     end;
end.
```

- **Instrukcja FOR.** Realizuje ona zdanie **dla**. Instrukcja ta jest bardzo wygodna, gdy z góry można określić liczbę niezbędnych powtórzeń obliczeń. Postać tej instrukcji jest następująca:

```
for zmienna := wartość_pocz to wartość_końcowa do instrukcja
```
lub
```
for zmienna := wartość_pocz downto wartość_końcowa do instrukcja
```

Zmienna nazywana jest zmienną sterującą instrukcji **for**. Musi ona być typu całkowitego, znakowego lub logicznego. **Wartość_pocz i wartość_końcowa** są wyrażeniami takiego samego typu jak typ zmiennej sterującej. Wykonanie instrukcji **for** przebiega następująco. Instrukcja wyszczególniona po słowie kluczowym do wykonywana jest tyle razy ile wartości znajduje się w zakresie:

```
Wartość_pocz .. wartość_końcowa
```

Jeżeli w instrukcji **for** użyto słowa kluczowego to, to zmienna sterująca za każdym razem przyjmuje kolejną wartość leżącą w przedziale między **wartość_pocz** a **wartość_końcowa**. Natomiast gdy w instrukcji for użyto słowa kluczowego **downto**, to zmienna sterująca przyjmuje kolejną wartość mniejszą od danej z przedziału **wartość_pocz, wartość_końcowa**.

- **Instrukcja CASE**. Instrukcja ta realizuje zdanie wybierz. Instrukcja ta ma postać.

```
case wyrażenie of
    wartość_1: instrukcja_1;
    { ... }
    wartość_n: instrukcja_n ;
    else instrukcja_awaryjna
end;
```

Wykonanie tej instrukcji przebiega następująco. Obliczana jest wartość **wyrażenia**, a następnie wykonywana jest ta **instrukcja**, przed którą stoi stała równa wartości wyrażenia. Jeżeli stała ta nie występuje, to wykonywana jest **instrukcja_awaryjna.**

Przykład:

```
program test;
var
    ocena: integer; { ocena ucznia }
begin
    writeln('Podaj ocenę: ');
    read(ocena);
    case ocena of
        6: writeln('celujący');
        5: writeln('bardzo dobry');
        4: writeln('dobry');
        3: writeln('dostateczny');
        2: writeln('dopuszczający');
        1: writeln('niedostateczny');
        else writeln('Podaj liczbę z zakresu 1-6')
    end
end.
```