Krzysztof Regulski
AGH, WIMiIP

PASCAL – WPROWADZENIE

1. Uruchom kompilator

- Stwórz nowy plik, a w nim napisz:

```
program hello;
begin
writeln('Witaj Pascalu!');
end.
```

2. Struktura programu w Pascalu.

- W strukturze programu w Pascalu wyróżniamy trzy elementy:
- Nagłówek programu – `program hello;`
- Blok deklaracji – pomiędzy nagłówkiem a słowem kluczowym `begin` – zawiera deklaracje stałych, zmiennych, typów i funkcji
- Blok instrukcji (operacyjny) – `Begin ... End`
- Kropka kończąca program

3. Komentarze

- Ważnym elementem każdego programu są komentarze. Nie wpływają one na działanie programu, ale ich stosowanie już nawet w niewielkich projektach jest niezbędne. Zwiększają one czytelność programu i informują „o co chodzi" w kodzie. Komentarzem jest fragment programu objęty parą nawiasów `{ }` lub parą dwuznaków `(* *)`. Komentarz jest ignorowany przez kompilator, który traktuje go jak odstęp. Poniższy przykład ilustruje użycie komentarzy, które szczegółowo wyjaśniają strukturę programu.

```
(*-------------------------------------------------------)
   Szkielet struktury programu
(-------------------------------------------------------*)
Program moj_naglowek1;
{
   Część deklaracyjna - miejsce na następujące
   deklaracje i definicje:
     label     - etykiet,
     const     - stałych,
     type      - typów,
     var       - zmiennych,
     function  - funkcji oraz
     procedure - procedur.
}
Begin
  { Część operacyjna - miejsce na instrukcje }
End.
```

- Czy powyższy kod jest poprawnym programem? Jak to sprawdzić?
- Zwróć uwagę na wcięcia w kodzie, odtąd **zawsze je stosuj**!

4. Instrukcje wyjścia

- Instrukcja `write()` służy do „wyprowadzania napisów” na „standardowe wyjście programu” czyli do wypisywania na ekranie.
- Instrukcja `writeln()` również wypisuje na ekranie, ale na końcu napisu przechodzi do nowej linii (dodaje znak końca linii).
- Napisz program:

```
program write1;
begin
   write('Twoje Imię i Nazwisko. ');
   write('Twoja Grupa, Numer Indeksu. ');
end.
```

- Zmień poprzedni program na:

```
program writeln1;
begin
   writeln('Twoje Imię i Nazwisko. ');
   writeln('Twoja Grupa, Numer Indeksu. ');
end.
```

- Czym się różnią?
- Napisz program:

```
program oblicz1;
begin
   writeln(10/3);
end.
```

- Napisz program i przeanalizuj jego działanie:

```
program stale_i_zmienne1;
Const                          {początek bloku deklaracji stałych}
 jakas_stala=999;              {jakas_stala – identyfikator stałej}
Var                            {początek bloku deklaracji zmiennych}
 jakas_zmienna:integer;        {integer – typ zmiennej}
Begin
  jakas_zmienna:=448;          {instrukcja przypisania}
  Write(‘Liczba 123: ‘);
  Writeln(123);
  Writeln(‘Wynik z 4*8 to: ‘,4*8);
  Writeln();
  Write(jakas_stala, ‘ ‘);
  Write(jakas_zmienna);
  Writeln(jakas_zmienna+12);
  Writeln(3/4);
End.
```

- Napisz program:

```
program formatowanie1;
begin
   writeln(10/3 :1:2);
{w nawiasie pojawił się zapis :1:2 oznaczający formatowanie ‘szerokości
pola’ do 1, oraz dwóch miejsc po kropce}
end.
```

- Napisz program:

```
Program formatowanie2;
Begin
   writeln(' ' :2000);                  {ta linia czyści ekran}
   writeln(10/3 :20:2);                 {piszemy pierwszą linię obliczeń}
   writeln('+',1000/3 :19:2);           {druga linia obliczeń}
   writeln('------------' :20);         {kreska pod obliczeniami}
   writeln('SUMA: ',10/3+1000/3 :14:2); {linia z wynikiem}
End.
```

5. Stałe

- Stałe określają pewne z góry zadeklarowane wartości, którymi mogą być ciągi znaków (tzw. literały łańcuchowe np. `'Pascal'`, czy liczbowe np. `1234`). Stałe sprawiają, że program staje się czytelniejszy. Można używać ich wielokrotnie, a ewentualna zmiana stałej wymaga poprawienia kodu tylko w jednym miejscu. Stała symboliczna jest więc niczym innym, jak identyfikatorem przypisanym do pewnej wartości na podobnej zasadzie jak litera π przypisana jest do wartości 3,141592... Stałe symboliczne definiuje się w części opisowej programu po słowie kluczowym `const`.

```
Program const_demo;
Const
  marg = '           ';
  linia = '+----------------------+';
  tytul = '| Program DEMO stałych  |';
BEGIN
  writeln(' ' :2000);        {ta linia czyści
ekran}
  writeln(marg, linia);
  writeln(marg, tytul);
  writeln(marg, linia);
END.
```

- Zauważ, że do regulacji odsunięcia ramki od prawej krawędzi zdefiniowano i użyto stałej o nazwie `marg`, która zawiera same spacje. Teraz, żeby zmienić odsunięcie wystarczy przedefiniować stałą `marg` i przekompilować program. Gdyby nie użyto tu stałej, to taka korekcja programu wymagałaby dokonania zmian wszędzie tam, gdzie użyta jest ta stała czyli aż w trzech miejscach. Zatem zastosowanie stałej sprawiło, że program można szybciej zmodyfikować unikając przy tym błędów.

- Ważna uwaga stałe są tylko innymi nazwami literałów lub wyrażeń stałych i nie można im w treści programu nadawać nowych wartości. Zatem umieszczenie gdzieś w programie instrukcji np

  ```
  tytul:=' Nowy tytuł '
  ```

  będzie błędem.
- Rada: stosuj stałe. Zawsze, gdy musisz użyć jakiegoś literału (liczbowego czy tekstowego) rozważ użycie w tym miejscu stałej.
- **Zadanie 1**: Napisz program wypisujący na ekranie:

  ```
  [twoje login] niech żyje!!!
  [twoje login] – Hip hip hurra!!!
  [twoje login] – Hip hip hurra!!!
  [twoje login] – Hip hip hurra!!!
  ```

  Zamień teraz swój login na imię i nazwisko.

  Jeśli skończyłeś, zawołaj mnie i poproś o punkcik.

6. Zmienne

- Zmienna to takie pudełko w pamięci komputera, w którym można przechowywać pewne wartości podczas wykonywania programu. Aby można było skorzystać ze zmiennej, należy ją zadeklarować w bloku deklaracyjnym programu:

```
program p1;
var
a: integer;       {integer to typ zmiennej, liczba całkowita}
b, tmp: integer;
begin
end.
```

- Do zmiennych możemy ‘wrzucać’ tylko wartości zgodne z zadeklarowanym typem danych:
  - **TYPY CAŁKOWITE** - są to wartości liczbowe, które mieszczą się w podanych zakresach, im większy zakres to automatycznie zwiększa się zapotrzebowanie liczby na pamięć, podaję więc również ile bajtów dany typ liczby zajmuje w pamięci:

    ```
    SHORTINT (-128..127)                1 bajt
    INTEGER  (-32768..32767}            2 bajty
    LONGINT  {-2147483648..2147483647}  4 bajty
    BYTE     {0..255}                   1 bajt
    WORD     {0..65535}                 2 bajty
    ```
  - w działaniach na liczbach całkowitych nie można używać operatora dzielenia "/". W zastępstwie używa się operatora "`div`" oraz "`mod`". Ten drugi jest szczególnie przydatny przy sprawdzaniu podzielności jednej liczby przez drugą (jeśli `a mod b=0` to `b` jest dzielnikiem liczby `a`).
  - **TYPY RZECZYWISTE** (czyli te, które mogą być ułamkami) - uniwersalnym i w zupełności wystarczającym do większości zadań

jest typ `REAL`. Na zmiennych rzeczywistych można wykonywać dzielenie operatorem "`/`", natomiast operatorami "`div`" i "`mod`" - nie można.

- **TYP LOGICZNY -** `BOOLEAN` - typ ten może przyjmować jedynie dwie wartości: `TRUE` (prawda) lub `FALSE` (fałsz) dzięki temu zajmuje on jedynie 1/8 bajta
- **TYP ZNAKOWY -** `CHAR` - typ ten przyjmuje dowolny pojedynczy znak o kodach `ASCII (0..255)` np. znak "`A`" czy "`!`"
- **TYP ŁAŃCUCHOWY -** `STRING` - jest to ciąg o długości 0-255 znaków, przykładowym łańcuchem jest: `'To jest tekst`', zwróć uwagę na użyte apostrofy. Jeżeli jakiekolwiek znaki umieścisz pomiędzy apostrofami będzie to uznawane wtedy za tekst nawet jeżeli znajdować się tam będą w środku liczby.

- Użycie zmiennej: ze zmienną można zrobić dwie rzeczy: zapisać coś do niej lub odczytać. Do nadawania zmiennym wartości służy Instrukcja przypisania . W naszym programie wygląda ona tak

```
program p1;
var
a, b, tmp: integer;
begin
a:= 198;
b:= 12;
end.
```

- Zapamiętaj, każdy wpis do zmiennej **kasuje jej poprzednią zawartość**. Aby zamienić zawartość zmiennych `a` i `b` między sobą potrzebna zatem będzie trzecia zmienna, w której na chwilę przechowamy jedną wartość. (Wyobraź sobie, że trzymasz w każdej ręce tom encyklopedii, jeśli chcesz je zamienić ze sobą, musisz odłożyć na chwilę jedną z nich na stół, dopiero potem przełożyć tom z ręki do ręki, a na końcu podnieść pozostały – ten stół to nasza trzecia zmienna `tmp`).
- Zamiana wartości zmiennych:

```
program p1;
var
a, b, tmp: integer;
begin
a:= 198;
b:= 12;
writeln('Zmienna a= ',a,'; Zmienna b= ',b,'; tmp= ',tmp);
{zauważ, że wywołując zmienną nie stosujesz apostrofów}
tmp:=a;
a:=b;
b:=tmp;
writeln('Zmienna a= ',a,'; Zmienna b= ',b,'; tmp= ',tmp);
end.
```

7. Instrukcja `readln` – wprowadzanie wartości z klawiatury

- Aby program wykorzystał zmienną, której wartość pochodzi od użytkownika, należy zastosować instrukcję `readln`, która podobnie jak `read`, pozwala wprowadzić do zmiennej wartość podawaną z klawiatury w czasie działania programu, z tym, że po przyjęciu wartości program przechodzi do następnej linii.
- Jeżeli natomiast wywołamy tą procedurę bez żadnych parametrów to program czeka w tym momencie na wciśniecie klawisza `Enter`. W przypadku wprowadzania zmiennych trzeba uważać aby wprowadzana wartość była zgodna z zadeklarowanym typem danych.
- Wypróbuj program:

```
Program przywitanie_policz;
var
   imie: string;
   c:real;
Begin
   write( 'Podaj swoje imię : ' );
   readln( imie );
   write( 'Podaj liczbę, jaką chcesz podnieść do kwadratu : ' );
   readln( c );
   writeln('Witaj ', imie ,'! Liczba ‘,c,’ podniesiona do kwadratu wynosi:
‘,c*c);
   readln;
End.
```

- **Zadanie 2:** Zmodyfikuj program tak, aby
  - na początku program czyścił ekran,
  - liczby wyświetlały się bez miejsc po przecinku,
  - program dodatkowo obliczał dwukrotność podanej liczby, a wynik podawał z odpowiednim komunikatem w następnej linii,
  - na końcu działania program podawał komunikat:
    `Wduś Enter aby zakończyć.`
  - Następnie zawołaj mnie i zaprezentuj działanie programu, jeśli chciałbyś otrzymać punkt.

8. Zadania (za każde można otrzymać 1 punkt)

- **Zadanie 3:** Napisz program, który prosi użytkownika o podanie Imienia, Nazwiska i adresu, a następnie drukuje **2** wizytówki w ramce, jedna pod drugą mniej więcej na środku ekranu (środek wysokości i szerokości ekranu) i czeka na wciśniecie dowolnego klawisza. Wynik powinien wyglądać mniej więcej tak:

```
+------------------------+
  Imie Nazwisko
  ul. Nazwa Ulicy 58/4C
+------------------------+
+------------------------+
  Imie Nazwisko
  ul. Nazwa Ulicy 58/4C
+------------------------+
```

- **Zadanie 4:** Napisz program, który wydrukuje na ekranie wynik działania: $\frac{1}{2*3}$ oraz iloczyn sum 2+3 i 4+5
- **Zadanie 5:** Napisz program wypisujący wynik zwykłego i całkowitoliczbowego dzielenia podanej przez użytkownika liczby przez liczbę `3`, oraz resztę z tego dzielenia (`div` i `mod`). Instrukcje wypisujące wyniki powinny wyprowadzać również informacje o działaniach.