Krzysztof Regulski
AGH, WIMiIP

PASCAL – FUNKCJE PREDEFINIOWANE

- **Zadanie 1**: Przypominając sobie wiadomości z poprzedniej sekcji, napisz program, który pobiera od użytkownika jakąś liczbę, a następnie wyświetla na ekranie jej dwukrotność, poprzedzoną komunikatem:

```
„Mój program Cię zawstydził, podał dwukrotność Twojej liczby, jest nią: ”
```

1. Funkcja a procedura

- Funkcje w języku Pascal są to „podprogramy”, sekwencje instrukcji, które służą do obliczenia pewnej wartości na podstawie dostarczonych parametrów. Wynikiem funkcji jest (prawie) zawsze jakaś wartość. Istnieją funkcje wbudowane (gotowe funkcje, które można wywołać poprzez użycie ich nazwy), ale można także tworzyć własne funkcje.
- Procedury podobnie jak funkcje są sekwencją instrukcji, z tą różnicą, że ich działanie nie prowadzi do obliczenia wyniku, a do wykonania jakichś czynności.
- Przykład: funkcja może obliczać pierwiastek podanej liczby, procedura może np. czyścić ekran.

2. Jeszcze o typach danych.

- Na poszczególnych typach danych można wykonywać właściwe im operacje. Warto zapamiętać jakie są między nimi różnice.
- Na poprzednich zajęciach poznałeś specyficzne dla typu `integer` działania: `div` oraz `mod` (dzielenie całkowitoliczbowe oraz reszta z dzielenia)
- Dla typów rzeczywistych (`real` oraz `double`) charakterystyczne jest dzielenie (`/`)
- Dla typów znakowych (`string` i `char`):
- `length(S: String): Byte` – funkcja zwraca długość zmiennej typu `string`
- `str(Const N; Var S: String)` – procedura zamienia liczbę N na łańcuch znaków S
- `val(Const S: String; N; Var E: Integer)` – procedura zamienia łańcuch znaków S na liczbę N (jeśli nie było błędów: E=0)
- `UpCase(Ch: Char): Char` – funkcja zwraca kod ASCII po zmianie małego znaku na wielki

- `chr(B: Byte): Char` – funkcja zwraca kod ASCII dla wskazanego znaku B
- Wszystkie wyżej wymienione funkcje i procedury są „wbudowane w Pascalu” co znaczy, że są zapisane w module system, który nie wymaga deklaracji użycia.

3. Funkcja `Chr`

- Składnia:
```
znak:=Chr(liczba);
znak:=#liczba;
```

  Każdy litera, cyfra czy inny znak (np. ramki) posiadają swój własny numer (dokładniej jest to liczba od 0 do 255) i jest odczytywany z tablicy kodów `ASCII`, np. znakowi 'A' odpowiada liczba 65, znakowi 'a' liczba 97, a znakowi '2' liczba 50. Aby można było zamieniać liczbę na znak używa się właśnie funkcji `Chr`, która zwraca znak typu `CHAR` np.

```
Begin
 WriteLn(chr(70));
 WriteLn(chr(43));
 WriteLn(#43);
 WriteLn(chr(39));
End.
```

- Program wyświetli w kolejnych liniach znaki: 'F', '+', jeszcze jeden '+' (zamiast `chr()` możesz używać znaczka `#`) i w ostatniej linii pojawi się **nie dający się zwyczajnie wyświetlić** `znaku apostrofu`. Jeżeli wstawimy po prostu apostrof np. w procedurze `WriteLn` to procedura uzna go za znak zaczynający lub kończący zmienną typu `string`, nie da się więc tak zrobić:
```
WriteLn( 'Alfabet Morse'a' );
```
- Pascal widzi tylko zmienną 'Alfabet Morse' i nie wie co dalej zrobić z resztą (`a'`) a ponieważ zmienne `String` można łączyć ze zmiennymi `Char`, tak więc poprawna linijka wyglądała by tak:
```
WriteLn( 'Alfabet Morse'+chr(39)+'a' );
```
- lub tak:
```
WriteLn( 'Alfabet Morse'+#39+'a' );
```
- ewentualnie tak:
```
WriteLn( 'Alfabet Morse',#39,'a' );
```
- Niektóre kody są znakami sterującymi np. `chr(13)` odpowiada klawiszowi `ENTER`, `chr(27)` klawiszowi `ESC`, `chr(8)` klawiszowi `BACKSPACE`, a `chr(7)` odpowiada sygnałowi z głośniczka komputera, spróbuj coś takiego:
```
Begin
  WriteLn(chr(7));
End.
```

4. Funkcja `Ord`

- Składnia: `liczba:=Ord(znak);`
- Ta funkcja jest odwrotnością funkcji `Chr`, służy do odczytania kodu `ASCII` podanego znaku np.

```
ord('A')=65
ord('a')=97
ord('2')=50
```

5. Moduły (biblioteki) w języku `Pascal`.

- Wszystkie polecenia języka Pascal zostały uporządkowane w postaci grup zwanych modułami (lub bibliotekami, ang. units). Istnieje kilka standardowych modułów, które zostały dołączone do programu Pascal. Oprócz tego można tworzyć własne moduły. Wśród modułów standardowych wyróżniamy:
- `System` – zawiera procedury standardowe języka, do których dostęp nie wymaga specjalnych zabiegów.
- `Crt` –zawiera podstawowe procedury i funkcje odpowiadające za interakcję programu z użytkownikiem, czyli obsługę klawiatury i ekranu, oraz kilka innych przydatnych elementów. Najczęściej wykorzystywany.
- `Dos` – zawiera procedury współpracujące z systemem operacyjnym.
- `Graph` – zawiera procedury potrzebne do tworzenia grafiki.
- Pozostałe moduły standardowe noszą nazwy: `Turbo3, Graph3, Printer` i `Overlay.`
- Aby móc korzystać z procedur któregoś z modułów należy zadeklarować jego użycie (nie dotyczy to modułu system, który jest automatycznie deklarowany przez Pascala). Moduł deklaruje się poprzez użycie słowa `uses`. Poczym należy podać nazwę modułu lub modułów oddzielając je przecinkami.

```
program uzycie_klawiatury;
  uses crt;
var
  c: char;
begin
  c:=readkey;
  writeln('Wcisnąłeś klawisz ',c);
  readln;
end.
```

- W programie wykorzystujemy funkcję `readkey` z modułu `crt`, która zwraca znak odczytany z klawiatury (różni się od `readln` tym, że zwraca znak (`char`), a nie `string`, a także pozwala pobierać znaki bez ich wypisywania na ekranie)
- Zauważ specyficzny sposób wywołania funkcji, inny niż w przypadku `readln(string)`. Zmienna przyjmuje wartość znaku wprowadzonego z klawiatury.

6. Wybrane Procedury i Funkcje modułu CRT

- `Readkey` – funkcja pobiera znak z klawiatury.
- `ClrScr` – procedura czyści ekran i wyświetla kursor w lewym górnym rogu.
- `Delay` – procedura powoduje opóźnienie o podaną liczbę milisekund.
- `GotoXY` – procedura przesuwa kursor do podanej pozycji na ekranie. W nawiasie podajemy współrzędne ekranu (kolumna, wiersz) w jakich ma się znaleźć kursor. Standardowe okno ma wymiary 25x80.
- `KeyPressed` – funkcja sprawdza czy został naciśnięty jakiś klawisz.
- `TextBackground` – procedura ustawia kolor tła pod tekstem.
- `TextColor` – procedura ustawia kolor tekstu.
- Kolory podajemy w postaci angielskich nazw kolorów lub liczb.

Black	0	Czarny
Blue	1	Niebieski
Green	2	Zielony
Cyan	3	Morski
Red	4	Czerwony
Magenta	5	Fioletowy
Brown	6	Brązowy
LightGray	7	Jasnoszary
DarkGray	8	Ciemnoszary
LightBlue	9	Jasnoniebieski
LightGreen	10	Jasnozielony
LightCyan	11	Jasnomorski
LightRed	12	Jasnoczerwony
LightMagenta	13	Jasnofioletowy
Yellow	14	Żółty
White	15	Biały

```
program uzycie_crt;
  uses crt;
var
  c: char;
begin
  clrScr;
  gotoXY (30,15);
  TextColor (Green);
  TextBackground (LightGray);
  WriteLn ('To dopiero napis!');
  delay(2000);
end.
```

- **Zadanie 2:** Napisz program, który pobiera z klawiatury znak, a następnie drukuje na ekranie jego wartość w kodzie `ASCII`

- **Zadanie 3:** Napisz program, który pyta o imię użytkownika, a następnie podaje liczbę znaków w tym imieniu. Odpowiedź powinna być napisana w innym kolorze.
- Zauważ, że do kolejnych znaków w łańcuchu można się odwoływać podając w nawiasie indeks:

```
program operacja_na_stringu;
var
  s: string;
begin
  write ('podaj imie: ');
  readln (s);
  writeln(s);
  writeln(s[1]);  {tutaj odwołanie do pierwszego znaku}
  s[2]:='e';      {tutaj drugi znak zamieniamy na literę e}
  s:=s+'ek';      {tutaj następuje dodawanie ciągów znaków – konkatenacja}
  writeln(s);
  readln;
end.
```

- Zakładając, że użytkownik podał imię `Michał`, ostatecznie wynikiem będzie „`Mechałek`”

7. Wbudowane funkcje matematyczne

- `Sin(A: Real): Real` - sinus kąta A (w radianach)
- `Cos(A: Real): Real` - cosinus kąta A (w radianach)
- `ArcTan(X: Real): Real` - arcus tangens z X
- `Sqrt(X: Real): Real` - pierwiastek kwadratowy z X
- `Sqr(X: Real): Real` - kwadrat X
- `Ln(X: Real): Real` - logarytm naturalny z X
- `Exp(X: Real): Real` - wartość funkcji wykładniczej z X
- `Int(X: Real): Real` - część całkowita z X
- `Frac(X: Real): Real` - część ułamkowa z X
- `Abs (X: Real): Real` – wartość bezwzględna z X
- `Pi` - wartość liczby π

8. Operacje na zmiennych:

- `Inc(Var A [, B])` - zwiększa wartość zmiennej A o 1 lub o B
- `Dec(Var A [, B])` - zmniejsza wartość zmiennej A o 1 lub o B
- `Pred(X: TOrdinal): TOrdinal` - element poprzedzający `X` `(Pred(10) = 9)`
- `Succ(X: TOrdinal): TOrdinal` - element następny po `X` `(Succ('A') = 'B')`

9. **Zadanie 4.** (za 1 punkt)

- Napisz program, który pyta użytkownika o dwie liczby rzeczywiste, po czym podnosi pierwszą do potęgi drugiej.
- Pytania powinny się pojawiać mniej więcej na środku ekranu, jedno po drugim w tej samej linii (po udzieleniu odpowiedzi znika pytanie i pojawia się następne).
- Odpowiedź powinna pojawić się w tym samym miejscu co pytania.
- Liczby powinny być wyświetlane w polu o szerokości 5, bez miejsc po przecinku.
- Zdanie informujące o wyniku powinno być napisane na zielono, a sam wynik na czerwono, na żółtym tle.
- Wynik powinien być wyświetlony w polu o szerokości 10, bez miejsc po przecinku.
- Po udzieleniu odpowiedzi program powinien czekać na wciśnięcie dowolnego klawisza, po czym czyścić ekran i kończyć pracę.
- **Podpowiedź**: w Pascalu nie ma wbudowanej funkcji podnoszącej jedną liczbę do potęgi drugiej, należy zatem skorzystać z funkcji wykładniczej (`exp`) oraz logarytmu naturalnego (`ln`) korzystając z takiej zależności:

$$A^B = e^{\ln(A^B)} = e^{B*\ln A}$$

10. **Zadanie 5.** (za 1 punkt)

- Napisz program, który „zgaduje” myśli użytkownika.
- Powinien zapytać o znak z klawiatury, bez wyświetlania go na ekranie, a następnie odczekać 1500ms i napisać poniższe komunikaty, przy czym
- przerwa pomiędzy nimi 1500 ms,
- kolor wyświetlanej litery żółty, tekst zdania jasnoszary,
- wymieniana litera zamieniona na wielką literę:

```
Znakiem, który wybrałeś nie {i tutaj podać element poprzedzający (pred)).}
Nie jest nim nawet (i tutaj podać element następny (succ)).
```

- Przed ostatnim komunikatem program powinien wyczyścić ekran, odczekać 2000ms,
- Potem pada odpowiedź w kolorze czerwonym, a ostatnia, wynikowa litera powinna być wyświetlona na żółtym tle:

```
Znakiem tym jest li tylko: (tutaj odpowiedź)!!!
```