Wstęp do programowania

Różne różności

Typy danych

Typ danych określa dwie rzeczy:

- Jak wartości danego typu są określane w pamięci
- Jakie operacje są dozwolone na obiektach danego typu

Rodzaje typów

- Proste (liczby, znaki, wartości logiczne, typy porządkowe, wskaźniki) – nie mają wewnętrznej struktury.

- Złożone (tablice (napisy!), rekordy, zbiory, pliki, ...)

Typy proste

- Wszystkie typy proste mają określone relacje

- <
- <=
- =
- >
- >=
- <>

Wartości typów prostych są rozumiane przez instrukcje czytania/pisania

Typy porządkowe

Wyliczamy wszystkie interesujące nas wartości definiując je jednocześnie.

- Przykład:

```
type odpowiedzi = (BardzoZle, Zle,
  Nijakie, Dobre, BardzoDobre, NieWiem)
```

Wszystkie typy porządkowe mają zdefiniowane operacje

- Ord – numer w typie liczony od 0 (np. Ord(Nijakie)=2)
- Pred, Succ – poprzednik, następnik,np. Pred(BardzoDobrze)=Dobrze
- Low, High – pierwsza i ostatnia z wartości (Low=BardzoZle, High=NieWiem)

Przykłady

- Typ wartości logicznych. Zawiera tylko 2 wartości: `false,true`.
- Operatory:

- `not` (negacja; jednoargumentowa)
- `or` (alternatywa)
- `and` (koniunkcja)
- `xor` (alternatywa wyłączająca: albo-albo)

Typ znakowy (char)

- Znaki dostępne w komputerze

Uwaga znaki ’a’ i ’A’ są inne, choć w identyfikatorach wielkość znaków nie gra roli.

Funkcja chr dla danej liczby z przedziału 0..255 daje znak o tym numerze.

Np. Chr(65)=’A’, tak jak Ord(’A’)=65.

Funkcje Chr i Ord są wzajemnie do siebie odwrotne.

Liczby całkowite

Shortint	-128 .. 127	1 bajt
Smallint	-32768 .. 32767	2 bajty
Longint	-2147483648..2147483647	4 bajty
Byte	0..255	1 bajt
Word	0..65535	2 bajty
Integer	-2147483648..2147483647	4 bajty
Cardinal	0.. 2147483647	4 bajty

Typy okrojone

- Zamiast podawać cały typ ograniczamy się do jego kawałka, np.
- 'a'..'z'
- 0..9
- '0'..'9'
- 0..1
- BardzoZle..Nijakie

Typy rzeczywiste

To też typy proste, ale nie porządkowe (nie ma w nich operacji następnika)

- Real48 2.9e-39 .. 1.7e38; 11-12 cyfr dokładnych; 6 bajtów
- Single 1.5e-45..3.4e38 7-8 cyfr dokładnych; 4 bajty
- Double 5.0e-324 ..1.7e308; 15-16 cyfr dokładnych; 8 bajtów
- Real = Double
- Extended 3.6e-4951..1.1e4932; 19-20 cyfr dokładnych; 10 bajtów
- Comp -2^63+1..2^63-1; 19-20 cyfr dokładnych; 8 bajtów
- Currency -2^59..2^59; 19-20 cyfr dokładnych; 8 bajtów

Funkcje rzeczywiste

- abs, sqr, sqrt, sin, cos, ln, exp
- round zaokrągla do najbliższej liczby całkowitej
- trunc obcina do najbliższej mniejszej liczby całkowitej
- Int obcina do najbliższej liczby rzeczywistej mającej wartość całkowitą
- FloatToStr przekształca liczbę rzeczywistą na napis
- StrToFloat przekształca napis będący poprawną stałą rzeczywistą na jej wartość

Tablice

- Komputerowe reprezentacje ciągów wartości

```
Var     a: array[1..100] of Real;
b: array[-1..1] of String;
c: array[char] of Integer;
```

Odwoływanie się do konkretnych elementów tablicy przez podanie indeksu w nawiasach, np

```
a[1]:=-50.25e3;
b[-1] := ’Ala ma kota’;
c[’;’]:=0;
```

Inne typowe funkcje dobrze ufundowane

- Dla pętli for i:=1 to n f(i,n)=n-i
- Dla algorytmów Euklidesa f(m,n)=m+n
- Dla algorytmów przechodzenia grafu: f(G)=liczba nieodwiedzonych węzłów

Stałe

- Stałe pomagają nazwać pewne wartości

Przykład

```
const Maxp = 460;
      dobrze = 1;
      MaxReal = 1e24;
      pusty = [];

  var Poslowie:array[1..Maxp] of string;
```

Rekordy

- Rekordy, to kolekcje danych różnych typów
- Type pracownik = record

```
                id        : Integer;
Imie,Nazwisko     : String[15];
RokZatrudnienia   : 1960..2215;
case  plec : (mezczyzna,kobieta) of
  mezczyzna : (broda: Boolean; NrKsWoj:String[11]);
  kobieta   : (NazwiskoPanienskie:String[15]);
end
```

Warianty w rekordach

```
type zespolona=record
case Boolean of
false: (r,fi:Real);
true: (Re,Im:Real)
end;
```

- Ten typ zajmie dokładnie tyle pamięci, co dwie liczby rzeczywiste.
- Typ Boolean nie jest odrębnym polem. rekordu

Warianty w rekordach

```
type zespolona1=record
case kartez:Boolean of
false: (r,fi:Real);
true: (Re,Im:Real)
end;
```

- Ten typ zajmie dokładnie tyle pamięci, co dwie liczby rzeczywiste plus jeden bajt na wartość logiczną.
- kartez jest odrębnym polem. rekordu
  - var z:zespolona1;:
  - z.r:=1.0; z.fi:=pi/2; z.kartez:=false;...

Warianty w rekordach

```
Function Tryg_To_Kart(var z:zespolona1);
{zamienia postać trygonometryczną na kartezjańską}
var pom:Real;
begin
 if not z.kartez then
  begin
   pom:=z.r*cos(z.fi);
  z.Im:=z.r*sin(z.fi);
  z.Re:=pom;
  z.kartez:=true
end;
```

Zbiory

```
type znaki = set of char;
var z:znaki;
begin
   z:=[];
   z:=z+['a','c'..'e'];
   Read(ch);
   if ch in z then...
end;
```

Instrukcja wyboru

- <instrukcja wyboru> ::= case <wyrażenie> of <lista wyborów> end
- <lista wyborów> ::= {<wybór> : <instrukcja>;}
- <wybór> ::= <stała> | <stała>,<wybór> | <zakres> | <zakres>,<wybór>
- <zakres> ::= <stała>..<stała>

Przykład

```
Case n of
 0, 1 : ;
 2..5,7..9,11,13,16..17,19: JedenDzielnikPierwszy;
 6,10,12,14,15,18 :DwaDzielnikiPierwsze;
 else SpozaZakresu;
end
```

Operacje na zbiorach

- +,*,- (suma, przecięcie, różnica)
- X in Z (należenie)
- Uwaga: zbiory mogą zawierać stosunkowo niewielką liczbę elemenntów (zazwyczaj rzędu kilkuset.
- Są reprezentowane przez bitmapy.
- Raczej nie przyspieszają kodu
- Są wygodne do definiowania stałych zbiporowych, np
  - If n in [2,3,5,7,11,13,17] then ...

Pętla repeat

- <instrukcja>::=<pętla repeat>
- <pętla repeat>::= repeat <ciąg instrukcji> until <wyrażenie logiczne>
- Ciąg instrukcji nie wymaga begin..end

Semantyka pętli repeat

`Repeat P1;P2;...;Pn until B` daje ten sam efekt, co

```
begin
P1;P2;...;Pn;
while not B do
 begin
    P1;P2;...;Pn;
 end;
end
```

Semantyka pętli repeat

- Petla repeat wykona się zawsze co najmniej raz
- Jest wygodna, gdy wiemy, że tak własnie ma być, np:

```
repeat
   Write('Podaj ile masz lat:');
   Read(n);
   if (n<MinLat) or (n>MaxLat) then
     Write('Wiek spoza zakresu ')
until (n>=MinLat) and (n<=MaxLat)
```

Pętla for

- <pętla for>::=

for <zmienna> := <wyrażenie> <jak> <wyrażenie> do
    <instrukcja>

<jak> ::= to | downto

Semantyka pętli for

`For i:=E1 to E2 do P` ma znaczenie takie, jak

```
begin
   i:=E1;
   while i<=E2 do
    begin
      P;
      i:=i+1
    end
i:=?
end;
```

Semantyka pętli for

- Zastrzeżenia dotyczące pętli for
  - Zmienna sterująca (i) musi być typu wyliczeniowego lub okrojonego
  - Nie wolno wewnątrz pętli zmieniać ani wartości zmiennej sterującej, ani zakresu E2.