Algorytmy i struktury danych

Zmienne
Proste typy danych
Strukturalne typy danych

Witold Marańda
maranda@dmcs.p.lodz.pl

1

Zmienne

Liczby (i struktury danych) występują w algorytmach i programach komputerowych w postaci zmiennych.

Zmienna jest obiektem, który posiada dwa atrybuty:
- nazwę, czyli symbol poprzez który odwołujemy się do zmiennej,
- wartość, czyli liczbę przypisaną do danego symbolu.

Podstawowe operacje wykonywane na zmiennych:

arytmetyczne :	a+4	b-a	a+d*5
przypisanie :	a←5	b←a	c←a+b (operator „ := ”)
porównanie :	a=5	b>a	c<=a+b

Symbole operatorów przypisania i porównania zależą od języka programowania, ale zawsze operator przypisania zmienia wartość zmiennej, a operatory porównania nie.

Wyrażenie zawierające operatory porównania można oceniać w sensie logiki matematycznej (prawda lub fałsz).

2

Typy danych

- Wszystkie obiekty w programach na których dokonuje się operacji, cechuje pewna własność (matematyczna) zwana typem. Każda stała, zmienna, wyrażenie lub funkcji jest określonego typu.
- Typ obiektu określa zbiór wartości, które mogą przyjmować stałe lub zmienne, albo które może przyjmować wyrażenie, lub które mogą być zwracane przez funkcję.
- Typy muszą być określone przez deklarację zawsze przed użyciem obiektu (typ danej można określić na podstawie ich postaci lub deklaracji).

3

Standardowe typy danych

Standardowe (wbudowane) typy danych są typami podstawowymi, zdefiniowanymi dla większości języków programowania. Typów tych nie potrzeba deklarować przed ich użyciem.

4

Liczby całkowite - *Integer*

- Obiekty typu całkowitego przyjmują wartości ze zbioru liczb całkowitych. Ze względu na ograniczoną reprezentację w pamięci komputera zakres przyjmowanych wartości zależy od implementacji.
- Wartości obiektów typu całkowitego są przechowywane w pamięci zawsze dokładnie (tj. bez zaokrągleń, czy przybliżeń). Również wyniki operacji na obiektach typu całkowitego są dokładne. Jedynym problemem jest możliwość przekroczenia zakresu.
- Najczęściej stosowane typy całkowite spotykane w różnych językach programowania i ich zakresy:

short	8-bitów	od –128 do +127	lub	od 0 do 255
integer	16-bitów	od -2^{15} do $2^{15}-1$	lub	od 0 do $2^{16}-1$
long	32-bity	od -2^{31} do $2^{31}-1$	lub	od 0 do $2^{32}-1$

5

Liczby całkowite - *Integer*

Typy całkowite w języku PASCAL:

- SHORTINT (-128...127) 1 bajt
- INTEGER (-32 768...32 767) 2 bajty
- LONGINT (-2 147 483 648...2 147 483 647) 4 bajty
- BYTE (0...255) 1 bajt
- WORD (0...65 535) 2 bajty

Przykład deklaracji zmiennych (PASCAL):

```
var
   A: byte;
   x,y,z: integer;
```

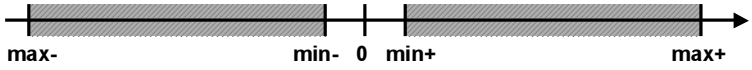
Przykład użycia zmiennych (PASCAL):

```
A:= 12;
x:= y+z;
```

6

Liczby rzeczywiste - *Real*

- Nie można zapisać w skończonej liczbie bitów wszystkich liczb rzeczywistych, nawet z określonego zakresu. Pomiędzy każdymi dwoma liczbami rzeczywistymi, zawsze będzie nieskończona ilość liczb, których nie będzie można zapisać w danej maszynie.
- Zbiór liczb typu rzeczywistego używane w komputerach opisuje się przez podanie największej i najmniejszej liczby (dodatniej i ujemnej) możliwej do zapisania w pamięci maszyny.



W reprezentacji komputerowej liczb rzeczywistych, pomiędzy **min+** a **max+** (oraz **max-** a **min-**) znajduje się jedynie ograniczona ilość liczb, przy czym odstępy miedzy nimi nie są jednakowe.

Możliwe jest ponadto zapisanie liczby 0.

Natomiast nie jest możliwy zapis liczb znajdujących się pomiędzy **0** a **min+** (oraz **min-** a **0**) oraz większych od **max+** (i mniejszych od **min-**).

7

Liczby rzeczywiste - *Real*

- Wyniki operacji na liczbach rzeczywistych nie muszą być dokładne, tj. wynik może być wyrażony jako przybliżenie lub zaokrąglenie. Z tego powodu, nie ma sensu operacja porównywanie dwóch liczb rzeczywistych operatorem '=', zamiast tego należy stosować relacje większości lub mniejszości.
- Liczby rzeczywiste zapisywane są w pamięci komputera w postaci wykładniczej, tj. za pomocą wykładnika i mantysy.
- Najczęściej stosowane typy rzeczywiste spotykane w różnych językach programowania i ich zakresy:

Typ	liczba bitów	min (-,+)	max (-,+)	cyfry znaczące
single (float)	32	1.2e-38	3.4e+38	6-9
double	64	2.2e-308	1.8e+308	15-17
extended double	80	3.4e-4932	1.2e+4932	18-21

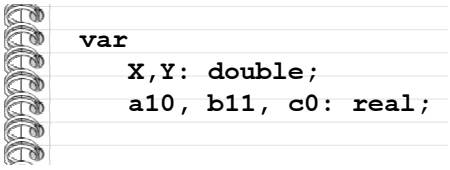
8

Liczby rzeczywiste - *Real*

Typy zmiennoprzecinkowe w języku PASCAL:

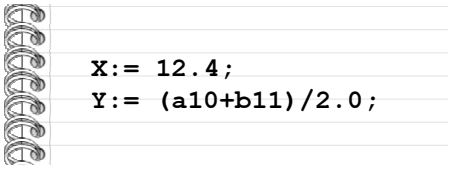
- SINGLE 4 bajty
- REAL 6 bajtów
- DOUBLE 8 bajtów
- EXTENDED 10 bajtów

Przykład deklaracji zmiennych (PASCAL):



```
var
   X,Y: double;
   a10, b11, c0: real;
```

Przykład użycia zmiennych (PASCAL):



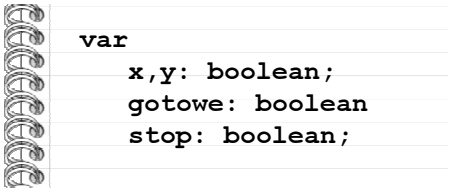
```
X:= 12.4;
Y:= (a10+b11)/2.0;
```

9

Wartości logiczne - *Boolean*

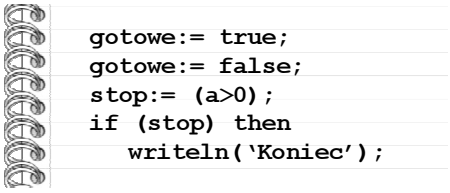
- Obiekty typu boolean mogą przyjmować tylko dwie wartości 1 (prawda-*true*) lub 0 (fałsz-*false*).
- Zmienne typu boolean stosowane są zwykle do sterowania przebiegiem programu, gdy podejmowane są decyzje wymagające odpowiedzi ‘prawda’ lub ‘fałsz’. Wszystkie instrukcje wyboru i cykli wymagają jako warunku wyrażenia typu boolean.

Przykład deklaracji zmiennych (PASCAL):



```
var
   x,y: boolean;
   gotowe: boolean
   stop: boolean;
```

Przykład użycia zmiennych (PASCAL):



```
gotowe:= true;
gotowe:= false;
stop:= (a>0);
if (stop) then
   writeln('Koniec');
```

10

Znaki drukowalne - *Char*

- Dane typu char reprezentują znaki drukarki lub terminala, które obejmują litery, cyfry, znaki przestankowe i inne. Najbardziej znany jest zbiór znaków, znany jako kod **ASCII** (*American Standard Code for Information Interchange*).
- Dane typu char służą najczęściej do komunikacji maszyny z użytkownikiem.
- Dane typu char tworzą zbiór spójny i uporządkowany. Na obiektach typu char dopuszczalne są pewne operacje arytmetyczne np. porównanie, natomiast inne nie mają sensu np. dodawanie.

11

Znaki drukowalne - *Char*

kod ASCII	opis
0 – 31	niewidoczne znaki sterujące
32	znak odstępu (spacja)
33 – 47	znaki interpunkcyjne i inne
48 – 57	cyfry
58 – 127	duże i małe litery i inne znaki

Przykład deklaracji zmiennych (PASCAL):

```
var
    p,k: char;
    znak: char;
```

Przykład użycia zmiennych (PASCAL):

```
p:= 'A';
p:= 'a';
znak:= #13;
```

12

Typy okrojone

- Typy okrojone są typami zbudowanymi na podstawie typu liczb całkowitych (integer) i znakowych (char). Wartości typów okrojonych należą do zdefiniowanego zakresu.
- Typy okrojone nie należą do typów wbudowanych, więc przed ich użyciem zmiennych tego typu danych, należy go zdefiniować.
- Stosowanie zmiennych typów okrojonych pozwala na lepsza kontrolę nad ich wartościami, gdyż każde przekroczenie zadeklarowanego zakresu powoduje zgłoszenie błędu przez kompilator języka.

13

Typy okrojone

- **Deklaracja:**

  type T = min..max

gdzie T jest nazwą typu, a min i max są dolnym i górnym zakresem dla wartości typu.

Przykład deklaracji typu (PASCAL):

```
type rok = 1900..1999
type dzientygodnia = 1..7
type litera = 'A'..'Z'
```

Przykład deklaracji zmiennych (PASCAL):

```
var
      x,y : rok;
      dd : dzientygodnia;
      znak : litera
```

14

Typy wyliczeniowe

- Typ wyliczeniowy definiuje się przez wyliczenie zbioru wszystkich jego wartości.
- Typy wyliczeniowe stosuje się dla ułatwienia programowania i poprawienia czytelności programu, gdyż operowanie nazwami symbolicznymi jest łatwiejsze od zapamiętywania liczb.

15

Typy wyliczeniowe

- **Deklaracja:**

  type T = (c1, c2, c3, …, cn)

  gdzie T jest nazwą typu, a c1, c2…cn są wartościami typu.

Przykład deklaracji typu (PASCAL):

```
type kolor = (czerwony, zolty, zielony)
type plec = (mezczyzna, kobieta)
type waluta = (frank, zloty, dolar, marka)
```

Przykład deklaracji zmiennych (PASCAL):

```
var
    osoba : plec;
    swiatlo : kolor
```

Przykład użycia zmiennych (PASCAL):

```
osoba:= kobieta;
swiatlo:= zielony;
swiatlo:= zolty;
```

16

Typy strukturalne

- Tablice
- Rekordy

17

Tablice - *Array*

- **Tablica** – struktura złożona z elementów tego samego typu, wskazywanych przez indeks lub zespół indeksów
- Tablice są szczególnie wygodne do reprezentowania danych tworzących regularne zespoły. Odpowiednikiem tablic w matematyce są np. wektory i macierze.

Deklaracja tablicy 1-wymiarowej (PASCAL):

type T = array[T_i] of T_0

gdzie T jest nazwą typu tablicowego, T_i jest typem indeksowym, a T_0 jest typem elementów tablicy.

Deklaracja tablicy n-wymiarowej (PASCAL):

type T = array[T_{i1},..,T_{i2},.., T_{in}] of T_0

gdzie T jest nazwą typu tablicowego, T_{i1}..T_{in} są typami indeksowymi, a T_0 jest typem elementów tablicy.

18

Tablice - *Array*

- Tablica nazywa się 1-wymiarową jeśli w deklaracji typu występuje tylko jeden typ indeksowy.
- W przypadku wystąpienia n-typów indeksowych, tablica jest n-wymiarową.
- Liczba elementów tablicy wynika z liczby elementów typu indeksowego: zwykle jest to typ okrojony lub wyliczeniowy np. 1..10, 0..99 itp.
- W przypadku tablic 2-wymiarowych pierwszy indeks jest numerem wiersza, drugi – numerem kolumny.

19

Tablice - *Array*

Jeśli zadeklarowany został typ tablicowy, możliwe jest deklarowanie i używanie w programie zmiennych tablicowych. Dostęp do danego elementu tablicy odbywa się przez podanie jego indeksów w nawiasach kwadratowych, po nazwie zmiennej tablicowej.

Przykład deklaracji typu (PASCAL):

```
type wektor = array[1..100] of integer;
type macierz = array[1..10,1..10] of real;
type waluta = (frank, złoty, dolar, marka);
type konto = array[waluta] of real;
```

Przykład deklaracji zmiennych (PASCAL):

```
var
    A,B,C : macierz;
    T : wektor;
    PKO : konto;
```

Przykład dostępu do elem. składowych:

```
A[1,1]:=0.0;
B[3,3]:=C[2,7];
T[79]:=100;
PKO[dolar]:=100000;
```

20

Tablice - przykład

```
type Wiersz
     = array[1..5] of real;

var
     x : Wiersz;
```

```
type Macierz
     = array[1..5,1..3] of real;

var
    wy : Macierz;
```

Wiersz x

x[1]	0.5
x[2]	0.25
x[3]	0.125
x[4]	0.0625
x[5]	0.03125

Macierz wy

[1,1] 12	[1,2] 31	[1,3] 22
[2,1] 34	[2,2] 34	[2,3] 43
[3,1] 8	[3,2] 54	[3,3] 11
[4,1] 23	[4,2] 87	[4,3] 12
[5,1] 1	[5,2] 22	[5,3] 29

21

Tablice - przykład

Przykład użycia zmiennych tablicowych: znajdowanie największego elementu w tablicy

```
const
    w = 100;
    k = 100;
type macierz = array[1..w,1..k] of real;
var
    i,j : integer;
    max : real;
    X : macierz;
begin
{zakładamy, że wszystkie elementy macierzy X mają nadane
wartości}
    max:=X[1,1];
    for j:=1 to k do
       for i:=1 to w do
          if (X[i,j]>max) then max:= X[i,j];
    writeln('Największa liczba w tablicy wynosi: ',max);
end
```

22

Rekordy - *Record*

- **Rekord** - struktura złożoną z elementów niekoniecznie tego samego typu.
  - Rekordy są szczególnie użyteczne przy posługiwaniu się kompletami danych różnego typu.
  - Rekordy znajdują zastosowanie zwłaszcza przy konstrukcji baz danych.

Na przykład, komplet danych o studencie mogą stanowić: nazwisko, nr. indeksu, kierunek, rok studiów.

Dane takie, choć różnego typu (np. tekstowe i liczbowe) stanowią logiczną całość i wygodnie jest nimi manipulować jako jednym obiektem.

23

Rekordy - Record

Deklaracja:

```
type T =  record
             p1 : T1;
             p2 : T2;
             ...
             pn : Tn;
          end;
```

gdzie T jest nazwą typu rekordowego, p_1.. p_n są polami rekordu typu T_1.. T_n.

- Typ rekordowy definiuje strukturę, w skład której wchodzą elementy o podanych identyfikatorach 'p_i', zwanych polami rekordu i zadeklarowanym typie T_i.
- Jeśli zadeklarowany został typ rekordowy, możliwe jest deklarowanie i używanie w programie zmiennych rekordowych. Dostęp do danego elementu rekordu, zwanego polem, odbywa się przez podanie jego nazwy oddzielonej kropką od nazwy zmiennej rekordowej.

24

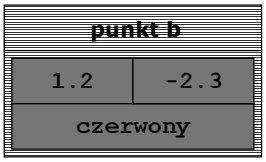
Przykład deklaracji typu strukturalnego

Przykład deklaracji typu (PASCAL):

```
type wektor = array[1..100] of integer;
type data =    record
                 dzien : 1..31;
                 miesiac : 1..12;
                 rok : integer;
               end;
type punkt =   record
                 x : real;
                 y : real;
                 kolor : (czerwony,zielony,niebieski);
               end;
type student = record
                 nazwisko : array[1..20] of char;
                 indeks : integer;
                 kierunek : array[1..10] of char;
                 rok : integer;
               end;
```

25

Przykład deklaracji typu strukturalnego




```
type wektor = array[1..100] of integer;
type data =    record
                 dzien : 1..31;
                 miesiac : 1..12;
                 rok : integer;
               end;
type punkt =   record
                 x : real;
                 y : real;
                 kolor : (czerwony,zielony,niebieski);
               end;
type student = record
                 nazwisko : array[1..20] of char;
                 indeks : integer;
                 kierunek : array[1..10] of char;
                 rok : integer;
               end;
```

Przykład deklaracji zmiennych (PASCAL):

```
var
  urodziny : data;
  a,b,c : punkt;
  obecny: student;
  grupa : array[1..30] of student;
```

Przykład dostępu do pól rekordów (PASCAL):

```
urodziny.dzien:=13;  urodziny.miesiac:=10;
urodziny.rok:=1984;
b.x:=1.2;   b.y:=-2.3;   b.kolor:=czerwony;
obecny.nazwisko:='Nowak';   obecny.indeks:=120034;
grupa[5].nazwisko:='Kowalski';
grupa[5].indeks:=34534;
```

26

Rekordy - przykład

Przykład użycia zmiennych rekordowych: znajdowanie studenta o największym numerze indeksu.

```
const
     n = 250;
type student = record
                     nazwisko : array[1..20] of char;
                     indeks : integer;
                     kierunek : array[1..10] of char;
                     rok : integer;
                end;
type zaoczne = array[1..n] of student;
var
     i : integer;
     max, imax : integer;
     WOiZ : zaoczne;
begin
{zakładamy, że wszystkie elementy macierzy WOiZ mają nadane wartości}
     max:= WOiZ[1].indeks;
     imax :=1
     for i:=2 to n do
         if (WOiZ[i].indeks > max) then
         begin
            max:= WOiZ[i].indeks;
            imax:=i;
         end;
     writeln('Najwiekszy numer indeksu, który wynosi: ',max);
     writeln('posiada student o nazwisku: ',WOiZ[imax].nazwisko);
end;
```

27

Pliki - *File*

- Plik jest nieograniczoną strukturą danych tego samego typu, w której dostęp do tylko elementów odbywa się sekwencyjnie.
- Sekwencyjny dostęp oznacza, że w danej chwili czasowej, możliwy jest dostęp tylko do jednego, bieżącego, elementu pliku, wskazywanego przez aktualną pozycję mechanizmu dostępu.
- W systemach komputerowych, pliki służą do przechowywania danych o dowolnych rozmiarach w pamięci zewnętrznej komputera, np. na dyskach, taśmach. Zaletą plików jest ich jest nieograniczona pojemność (zależna tylko od pojemnością dysków), ale wadą jest długi czas dostępu do elementów wynikający z natury sekwencyjnego dostępu oraz położenia w pamięci zewnętrznej komputera.

28

Pliki - *File*

Deklaracja:

$$\text{type T = file of } T_0$$

gdzie T jest nazwą typu plikowego, a T_0 jest typem elementów składowych pliku.

- Jeśli zadeklarowany został typ plikowy, możliwe jest deklarowanie i używanie w programie zmiennych plikowych. Dostęp do bieżącego elementu pliku odbywa się za pomocą funkcji ‘read’, ‘write’ oraz pomocniczych ‘assign’, ‘reset’, ‘rewrite’ i ‘eof’.
- Plik którego elementami są obiekty typu ‘char’, nazywa się plikiem tekstowym.

29

Pliki

Przykład deklaracji typu (PASCAL):

```
type list = file of char;
type dane = file of integer;
type temperatura = file of real;
```

Przykład deklaracji typu (PASCAL):

```
var
   wyniki_obliczeń : dane;
   konin : temperatura;
   totolotek : file of array[1..49] of integer;
   pismo : file of char;
```

30

Pliki - podstawowe operacje

- write(x,e)
  Procedura pisania do pliku, ‘x’ jest zmienną plikową, ‘e’ jest zmienną, której wartość zostanie dopisana do pliku. Zmienna ‘e’ musi być typu zgodnego z typem elementów pliku.
  Każda operacja ‘write’ powoduje automatyczne przesunięcie mechanizmu dostępu na następny element pliku.
- read(x,e)
  Procedura czytania z pliku, ‘x’ jest zmienną plikową, ‘e’ jest zmienną, której zostanie przypisana wartość bieżącego elementu z pliku. Zmienna ‘e’ musi być typu zgodnego z typem elementów pliku.
  Każda operacja ‘read’ powoduje automatyczne przesunięcie mechanizmu dostępu na następny element pliku.

31

Pliki - podstawowe operacje

- reset(x)
  Procedura ustawienia mechanizmu dostępu na pierwszy element pliku (przygotowanie pliku do czytania).
- rewrite(x)
  Procedura utworzenia pliku pustego i ustawienie mechanizmu dostępu na jego początek (przygotowanie pustego pliku do pisania).
- eof(x)
  Funkcja zwracająca informację o położeniu mechanizmu dostępu dla pliku ‘x’. Funkcja zwraca wartość ‘true’ gdy nie istnieje element następny za wskazywanym (osiągnięto koniec pliku), a ‘false’ w przeciwnym wypadku.

32

Pliki - przykład

Przykład użycia zmiennych plikowych: obliczanie liczby znaków w dokumencie.

```
var
    dokument : file of char;
    licznik : integer;
    znak : char;
begin
    {zakładamy że plik 'opis.txt' istnieje na dysku}
    licznik:=0;
    assign(dokument,'opis.txt');
    reset(dokument);
    while (not eof(dokument)) do
    begin
       read(dokument,znak);
       licznik:=licznik+1;
    end;
    close(dokument);
    writeln('Liczba znaków w pliku wynosi ',licznik);
end;
```

33