PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **31** (2003/2004)
Številka 6
Strani 334–337

Matjaž Zaveršnik:

UVOD V PROGRAMSKI JEZIK JAVA

Ključne besede: računalništvo, programiranje, programski jeziki, java, podatkovni tipi, kontrolne strukture, tabele, objekti, nizi.

Elektronska verzija: http://www.presek.si/31/1575-Zaversnik.pdf

UVOD V PROGRAMSKI JEZIK JAVA

V prejšnjem Preseku smo si ogledali, kako poteka delo s programskim jezikom java. Sestavili smo ogrodje preprostega programa v javi, pa tudi ogrodje programa, ki ga lahko vključimo na spletno stran (takemu programu pravimo *applet*). V tem članku se bomo lotili osnov programiranja v javi, predpostavili pa bomo, da bralec že pozna osnove programiranja v jeziku C ali C++, saj je med javo in tema dvema programskima jezikoma kar precej podobnosti.

Osnovni podatkovni tipi

Osnovni podatkovni tipi v javi so cela števila, realna števila in logične vrednosti. Nekateri med osnovne podatkovne tipe uvrščajo še znake, v resnici pa so to samo poseben primer celih števil.

Celoštevilski podatkovni tipi v javi imajo natanko določeno velikost. Tip `byte` zavzame 8 bitov, `short` 16 bitov, `int` 32 bitov in `long` 64 bitov. Spomnimo se, da standard za C ne predpisuje, kako velik mora biti posamezen tip, predpisuje samo njihov vrstni red (najmanjši je `short`, sledi `int`, temu `long`, v zadnji izdaji standarda pa se jim je pridružil še `long long`). V javi je torej veliko več reda. Medtem ko so v C-ju celoštevilski tipi lahko predznačeni ali nepredznačeni, java nepredznačenih celoštevilskih tipov ne pozna.

Tudi pri znakih (tip `char`) je pomembna razlika. V C-ju so znaki 8-bitni (lahko so predznačeni ali nepredznačeni), v javi pa so 16-bitni (vedno nepredznačeni). Seveda pa lahko z znaki računamo kot s celimi števili.

Podobno kot C in C++ tudi java pozna realna tipa `float` in `double`, ne pozna pa tipa `long double`.

Java pozna še logični tip `boolean`. Edini vrednosti tega tipa sta `true` in `false`. Programski jezik C nima logičnega tipa, v jeziku C++ pa obstaja tip `bool`.

Kontrolne strukture

Pogojne stavke in zanke se v javi uporablja na enak način kot v C-ju. Obstaja samo drobna razlika pri pisanju pogojev. V C-ju so pogoji lahko poljubni izrazi. Če je vrednost tega izraza enaka 0, pogoj ni izpolnjen, katerakoli druga vrednost pa pomeni, da je pogoj izpolnjen. V javi pa je pogoj lahko samo izraz, katerega vrednost je logičnega tipa.

Tabele

Tabela je v C-ju predstavljena s kazalcem na njen prvi element, v javi pa je tabela referenca na dinamično dodeljen pomnilnik, kjer je poleg elementov tabele zapisana še njena velikost. Podobno kot v C-ju so tudi v javi elementi tabele oštevilčeni od 0 naprej. Poglejmo si nekaj primerov definicij tabel celih števil v javi:

```
int[] t1 = {1, 5, 0, -3, 1, 0};
int[] t2 = new int[6];
int[] t3 = t1;
int[] t4 = null;
```

Tabelo `t1` smo ustvarili tako, da smo našteli vse njene elemente. To možnost poznamo že iz C-ja, samo oglate oklepaje moramo v C-ju pisati za imenom tabele. Tabelo `t2` smo ustvarili s pomočjo operatorja `new`, pri čemer smo določili samo njeno dolžino, vsi njeni elementi pa se nastavijo na vrednost 0, 0.0, `false` ali `null`, odvisno od tipa elementov (v zgornjem primeru dobimo cela števila 0). Dolžina tabele, ki jo zapišemo v oglatih oklepajih, je lahko poljuben izraz, katerega vrednost je nenegativno celo število. V C-ju bi nekaj podobnega lahko naredili na dva načina.

```
int t2[6];
int *t2 = malloc(6 * sizeof(int));
```

Prvi način bi uporabili, če je dolžina tabele znana že ob prevajanju programa, drugega pa, če lahko dolžino naračunamo ali omejimo šele med izvajanjem programa. Začetne vrednosti elementov teh dveh tabel niso določene (lahko so poljubne).

Tabela `t3` je enaka tabeli `t1`. To ni kopija tabele `t1`, pač pa sta `t1` in `t3` referenci na isto tabelo (različni imeni za isto tabelo). Pri deklaraciji tabele `t4` smo določili samo tip njenih elementov, tabela sama pa še ne obstaja.

V C-ju moramo vsako tabelo, ki jo ustvarimo s klicem funkcije `malloc`, sprostiti s klicem funkcije `free`, ko je ne potrebujemo več. V javi nam za to ni treba skrbeti. Za sproščanje tabel in objektov, ki jih ne potrebujemo več, poskrbi poseben mehanizem (angleško mu pravimo *garbage collector*).

V C-ju moramo biti še posebej pazljivi, če tabelo prenašamo kot parameter pri klicu funkcije. V funkciji namreč ne moremo več izračunati njene dolžine, zato moramo dolžino v večini primerov prenašati kot dodaten parameter. V javi takih težav nimamo. Za vsako tabelo (če tabela obstaja) lahko ugotovimo njeno dolžino. Dolžino tabele `tab` nam vrne izraz `tab.length`.

Objekti

Objekt je posebna podatkovna struktura, v kateri lahko poleg različnih vrst podatkov (komponente) hranimo tudi metode za delo s temi podatki. V programskem jeziku C objektov nimamo, imamo pa strukture, v katerih lahko hranimo samo komponente, brez metod. Opis objektov iste vrste imenujemo *razred*. Pri programiranju opišemo, kakšne komponente in metode bo vseboval posamezen razred, potem pa z operatorjem `new` ustvarimo nov objekt tega razreda. Objektov nam ni treba sproščati, ko jih ne potrebujemo več. Java vsebuje obsežne knjižnice, v katerih je že definiranih preko 2500 razredov, ki jih lahko uporabljamo v svojih programih.

Nizi

V C-ju je niz tabela znakov, pri čemer je zadnji znak v nizu vedno znak s kodo 0. Za delo z nizi imamo na razpolago več funkcij iz knjižnice `<string.h>`. V javi pa je niz objekt razreda `String`. Podrobnosti o načinu predstavitve niza nam ni treba poznati, saj imamo za vse, kar potrebujemo, na razpolago več metod. Najpomembnejši sta `length` in `charAt`, spisek vseh ostalih pa dobite v dokumentaciji razreda `String`.

```
String niz = "abcdefgh";       // definicija niza
int dolzina = niz.length();    // dolžina niza
char znak = niz.charAt(2);     // znak na mestu z indeksom 2
```

V javi lahko nize tudi seštevamo. Vsota dveh nizov je stik teh nizov, torej nov niz, ki ga dobimo tako, da na konec prvega niza dodamo drugi niz. Vsota `"abc" + "xy"` je tako `"abcxy"`. Pravzaprav lahko nizu prištejemo karkoli, čeprav je učinek včasih težko vnaprej napovedati. Sumand, ki ni niz, bo program predelal v niz in naredil običajen stik nizov. Tako je `"abc" + 5` enako `"abc5"` in `"abc" + 2.5` enako `"abc2.5"`.

Branje in izpis podatkov

Za izpis podatkov na standardni izhod ima java metodi `print` in `println` objekta `System.out`. Obe metodi izpišeta dani podatek, edina razlika je ta, da metoda `println` na koncu naredi še skok v novo vrsto. Spodnja stavka izpišeta račun in rezultat, vse skupaj v eni sami vrstici.

```
System.out.print("1 + 2 = ");
System.out.println(1 + 2);
```

Enak izpis lahko dosežemo tudi samo z enim stavkom, če uporabimo stik niza `"1 + 2 = "` in vsote `1 + 2`. Pri tem moramo vsoto zapisati

v oklepajih, sicer bo program naredil dva stika (najprej bo dodal enko, potem pa še dvojko).

```
System.out.println("1 + 2 = " + (1 + 2));
```

Branje s standardnega vhoda je bolj zapleteno. Java sicer pozna metodo `read` objekta `System.in`, ki pa ni nič kaj uporabna, saj lahko z njo beremo samo enega ali več znakov. Zato običajno ustvarimo drugačen objekt, ki nam omogoča branje celih vrstic.

```
BufferedReader vhod =
   new BufferedReader(new InputStreamReader(System.in));
```

Uporabljena razreda `BufferedReader` in `InputStreamReader` pripadata paketu `java.io`, zato ga moramo na vrhu datoteke vključiti z ukazom `import`. Ker lahko pri branju podatkov pride do najrazličnejših napak, s katerimi se zaenkrat ne želimo ukvarjati, pa moramo pri definiciji metode `main` na koncu vrstice (za parametri) napisati še `throws IOException`.

Z metodo `readLine` objekta `vhod` lahko zdaj beremo cele vrstice s standardnega vhoda. Tako vpisane podatke preberemo v obliki niza. Če je podatek v resnici število, ga moramo ustrezno predelati. Celo število dobimo iz niza s pomočjo metode `Integer.parseInt`, realno število pa s pomočjo metode `Double.parseDouble`.

```
String niz = vhod.readLine();
int celo = Integer.parseInt(vhod.readLine());
double realno = Double.parseDouble(vhod.readLine());
```

V zgornjem primeru preberemo tri vrstice. Prvo shranimo v niz, drugo predelamo v celo število, tretjo pa v realno število. Če števila ne vpišemo v pravilni obliki, lahko pri pretvorbi pride do napake, ki prekine izvajanje programa.

Zaključek

Videli smo, da obstajajo določene razlike med jezikoma java in C/C++, vendar niso tako strašne, da se ne bi mogli navaditi nanje. Pravzaprav je razlik še več. Tu smo našteli samo tiste najbolj pomembne, s katerimi se nov programer v javi sreča takoj na začetku.

Da bi se privadili na razlike med jezikoma, je potrebno samo nekaj vaje in sedenja za računalnikom. Pobrskajte torej po starih številkah Preseka, poiščite kakšen program, napisan v C-ju, in ga poskusite prepisati v javo. V eni od prihodnjih številk bomo kakšen primer naredili tudi pri Preseku.

Matjaž Zaveršnik