Enajste vaje iz računalništva

Mirnesa Klisura

5.1.1999

Povzetek

Na današnjih vajah smo si ogledali kriterij za ocenjevanje domačih nalog, pojasnili prenos parametrov po vrednosti in po referenci, naučili smo se izvajanja programa po korakih z uporabo funkcije `printf`, nazadnje pa smo se še nekaj naučili o matrikah in začeli z izdelavo programa `matrika.c`.

1. Ocenjevanje domačih nalog

Pri ocenjevanju domačih nalog se upoštevajo 3 bistvene stvari, in sicer pravilnost, dokumentacija in izvedba programa.

pravilnost: Sem spadajo opis delovanja funkcij, funkcije, podprogrami, ...

dokumentacija: To so pogodbe; komentarji; kakšna imena uporabljamo za spremenljivke, funkcije ...

izvedba programa: Tu je mišljeno to, če program lahko prevedemo in če dela.

Podrobnejši kriterij za ocenjevanje domačih nalog pa si lahko ogledate na spletni strani (kjer so tudi zapiski z vaj):

`http://valjhun.fmf.uni-lj.si/~brodnik/Andrej/RacunalnistvoI`

Čisto na koncu strani kjer piše *Ostalo* kliknete na povezavo *WWW sestavek za šolsko leto 1997/98*, nato pa pod *Obveznosti študentov* kliknete na povezavo *domače naloge*.

2. Prenos parametrov po vrednosti in referenci

Prenos parametrov po vrednosti pomeni, da se vrednost prekopira in nato lahko z njo poljubno počnemo kar hočemo (npr.: beremo, brišemo, popravljamo, ...), zunaj funkcije pa se te spremembe ne poznajo.

Pri prenosu parametrov po referenci pa se vsaka sprememba, ki jo naredimo v notranjosti funkcije pozna tudi zunaj (izven te funkcije) – prenesemo le referenco na spremenljivko.

Primer za klic *po vrednosti*:

```
... Funkcija(int spremenljivka)
{
  spremenljivka = 10;
} /* Funkcija */
```

in za klic *po referenci*:

```
... Funkcija(int *spremenljivka)
{
 *spremenljivka = 10;
} /* Funkcija */
```

Razlika med prenosom parametrov po vrednosti in prenosom parametrov po referenci je v tem, da pri prenosu po referenci imamo zvezdico pred parametrom pri vrednosti pa ne.

Ko uporabljamo funkcijo, ki uporablja prenos po vrednosti, spremembe vrednosti spremenljivk znotraj te funkcije, ki smo jih podali kot parametre funkcije, **niso** vidne navzven – proti klicočemu programu.

Ko pa uporabljamo funkcijo, ki uporablja prenos po referenci, lahko znotraj te funkcije spreminjamo vrednosti spremenljivk, ki smo jih podali kot parametre funkcij in te spremembe **so** vidne v klicočem programu.

Ko prenašamo nize, tabele, matrike je vedno uporabljen prenos po referenci, vendar zvezdice v tem primeru **ne** pišemo. Ko prenašamo realna števila, cela števila, znake, logične spremenljivke itd. po referenci, pa zvezdica **je**.

Omenili smo še en pojem, in sicer **kazalec**, ki ga bomo podrobneje spoznali na predavanjih. Kazalec je v bistvu referenca na nekaj.

3. Testno izpisovanje – namesto *debug*

Na prejšnjih vajah smo se pogovarjali o semaforju. Začeli smo delati program, ki naj bi obračal tablice na semaforju. David in Voranc pa sta za domačo nalogo morala ta program dopolniti.

Pri Vorancu je nastopil problem, da ko smo mi vpisali znak, ki je trenutno na semaforju in znak, ki naj bi bil na semaforju, ter bi se kolešček moral zavrteti od znaka, ki se trenutno nahaja pa do znaka, ki naj bi se nahajal in to izpisati (recimo, kolešček bi se moral zavrteti od F do S.), se je kolešček vrtel kar v nedogled in se ni hotel ustaviti.

Da bi ugotovili v čem je problem, smo hoteli posamezne vrednosti spremenljivk pogledati z *debug*-om. Ker pa *debug* ni delal, smo si pomagali tako, da smo izpisovali na zaslon posamezne vrednosti kolesčkov od tistega znaka, ki se trenutno nahaja pa vse do tistega znaka, ki naj bi se nahajal. To pa smo naredili uporabljajoč funkcijo `printf`:

```
  printf("i = %d, kolescek[i] = %c, znak2 = %c\n",
         i, kolescek[i], znak2);
```

Napaka je bila v tem, da smo mi vpisovali male črke, katerih sploh ni na koleščku, namesto velikih.

4. Matrike

Naredili bomo program, ki prebere s tipkovnice kvadratno matriko celih števil in jo tudi izpiše, potem pa še izračuna determinanto te matrike.

Naloge smo se lotili tako, da smo najprej v glavnem programu napisali kaj bomo naredili:

```
int main(void)
{
  /* preberi matriko */

  /* izpisi matriko */
} /* main */
```

Nato pa smo naredili funkcijo, ki nam bo prebrala matriko `preberi_matriko` in jo uporabili v glavnem programu.

Da smo prebrali dimenzijo matrike in jo zapisali v spremenljivko `dimenzija` smo za to uporabili funkcijo

```
dimenzija = preberi_vrednost("dimenzija matrike");
```

katera ima kot parameter besedilo (niz), ki se uporabi kot pojasnilo uporabniku kaj beremo. V našem primeru zahteva od uporabnika, da vpiše dimenzijo matrike in potem ta funkcja vrne prebrano vrednost, ki se potem zapiše v spremenljivko `dimenzija`.

```
int preberi_matriko(int matrika[max_vel][max_vel])
{
  int dimenzija;

              /* preberi dimenzijo matrike */
  dimenzija = preberi_vrednost("dimenzija matrike");
              /* preberi elemente matrike */
  return dimenzija;
} /* preberi_matriko */
```

Za branje vrstic matrike smo uporabili `for` zanko (pri čemer gre števec i od začetne vrednosti 0 pa do dimenzije matrike), ki deluje tako, da vsakič ko se števec poveča za ena preberemo eno vrstico:

```
              /* preberi i - to vrstico matrike */
  for(i = 0; i < dimenzija; i++)
  {
    preberi_vrstico(i, matrika[i])
  }
```

Funkcija `preberi_matriko` na koncu vrne velikost dimenzije, (ki smo jo prebrali z uporabo funkcije `preberi_vrednost`, ki se nato uporabi v glavnem delu programa zato, da vemo kolikšna je dimenzija matrike, ki jo bomo nato izpisali – zopet v glavnem programu.

```
int main(void)
{
  int matrika[max_vel][max_vel];
  int dimenzija;

  /* preberi matriko */
  dimenzija = preberi_matriko(matrika)
} /* main */
```