Osme vaje iz računalništva I

David Bokavšek

1. 12. 1998

Povzetek

Na osmih vajah so Alen, Aleksij in Vili dobili sledečo domačo nalogo: Alen je moral v prejšnjem zaporedju izpisati več zaporedij, ne da bi pri tem morali znova zagnati program. Aleksij je moral ustaviti ponavljanje števil v zaporedju. To je naredil na dva načina: Prvi način: z pomočjo *`for` zanke*. Drugi način: *z rekurzijo* – najprej izpišemo prvi člen zaporedja, nato pokličemo funkcijo, ki bo izpisala še ostale člene zaporedja.

1. Dopolnitev programa `zaporedje.c`

Kot je bilo povedano v povzetku, je moral Alen v prejšnjem zaporedju izpisati več zaporedij, ne da bi ponovno morali zagnati program. To smo pri programih `kvadrat.c` in `pravokotnik.c` naredili s pomočjo `while do` zanke. Sedaj si bomo pogledali program `pravokotnik.c` in v njem poiskali zanko `while do`. V tem primeru začnemo z `do`; najprej `preberi_velikost` nato `izrisi_pravokotnik`, ko pa pride do `while`, se to izvede pri pogoju, da je `sirina` > `najvecje sirine`. Kaj naredimo? Najprej pogledamo, kaj se dogaja:

Začetni člen se prebere, nato se prebere število členov in korak. Sedaj moramo zaporedje večkrat izpisati. Dodali bomo zanko od `zacetni_clen = ...` do `/* izpisi vsoto clenov */`. `do` zanka oklepa tisto kar hočemo ponavljati. `do` zanko dodamo v program `zaporedje.c` in sicer v 64 vrstico in v tretji do vključno peti stolpec. Sedaj se vrnemo v 60 vrstico in tretji stolpec in vpišemo `while (zacetni_clen > 0);`. Upoštevati moramo še ustavljalni pogoj, ker se takrat zanka ustavi. Ko je začetni člen 0 se program zaustavi, oziroma se nič ne naredi.

2. Zaustavitev ponavljanja števil v `zaporedju.c`

1. Način: z `for` zanko

2. Način izpisovanja zaporedja: Najprej se izpiše začetni člen zaporedja, nato pa še ostali. Preostali členi zaporedja, razen prvega, so neko novo zaporedje, ki se začne pri prvemu členu + korak. Ker je to novo zaporedje tudi zaporedje, ga lahko izpišemo z isto funkcijo. Naša funkcija izpiše prvi člen zaporedja in preostalo zaporedje pošlje v neko novo funkcijo in ga izpiše. Zaporedje zapišemo z funkcijo `izpisi_zaporedje`. Komentar v 58 vrstici in v osmem stolpcu izbrišemo.

   Ker se program ne zaključi, moramo iz glavnega programa število členov, ki naj se izpišejo, spraviti v funkcijo. To storimo tako, da v definicijo funkcije `void izpisi_zaporedje1(int clen, int korak)` dodamo še parameter `stevilo_clenov`, ki naj se izpišejo.

   `izpisi_zaporedje1` naredi: izpiše se `stevilo_clenov` zaporedja, ki ima začetni člen `clen`, korak med členi `korak`. Če je število členov večje, ki naj se ispišejo, večje od 0, zaporedje izpišemo, in če je enako 0 ga ne izpišemo. To preverimo z `if` stavkom.

3. Seštevanje členov zaporedja

3.1 Z rekurzijo

Uporabili bomo nove funkcije. Definirali bomo eno samo funkcijo, ki bo izračunala vsoto – vrnila celo število. Glava definicije funkcije bo tako `int sestej_zaporedje1`.

Kako bomo to naredili? Če znamo sešteti preostanek (člene od drugega naprej) zaporedja, znamo sešteti zaporedje. Preostanek zaporedja je neko novo zaporedje. Seštejemo ga z isto funkcijo a z novimi parametri.

Zaradi boljše preglednosti damo konec in začetek definicij funkcij še črto: `/* ------------------ */`.

V komentarjih smo napisali vse, *kar smo naredili.*

Ostale člene zaporedja seštejemo tako, da kličemo funkcijo

```
sestej_zaporedje1(zacetni clen + korak, korak, stevilo_clenov - 1)
```

ki stoji v 48 vrstici in enaindvajsetem stolpcu. Rezultat dobimo kot vrednost funkcije, moramo ga shraniti v spremenljivko: `int vsota;` v 47 vrstico in drugi stolpec. Vsoti prištejemo še prvi člen in dobimo vsoto zaporedja. To vsoto vrnemo kot rezultat.

Podobno kot v podprogramu, tudi v glavnem programu definiramo vsoto:

```
int main(void) {
  int vsota;
  ...
  vsota = sestej_zaporedje1( ... );
  ...
} /* main */
```

Spremenljivko `vsota` potrebujemo (kot v podprogramu), da v njo shranimo vrednost, ki jo vrne funkcija.

V primeru, da se program ne izvaja, uporabimo *debug.* Pogledamo začetni člen in parametre ter tako ugotovimo, zakaj ne dela, kakor bi moral.