Programiranje II

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Sana Stojanović

29.05.08.

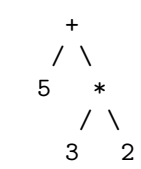
Sadržaj

1 Aritmetičko stablo

- Napisati program koji sa standardnog ulaza čita aritmetički izraz zapisan u prefiksnoj notaciji `operator izraz1 izraz2`, smešta ga u karaktersku nisku dužine do 20 karaktera i formira stablo u čijem se korenu nalazi zadati operator, u levom podstablu `izraz1` a u desnom `izraz2`. Pri tome se izraz zadaje ili kao `ceo broj` ili kao `operator izraz1 izraz2`. Napisati rekurzivnu funkciju koja od učitanog stringa formira binarno stablo.

  Prilikom zadavanja izraza očekujemo da su svi operandi razdvojeni jedan od drugog razmakom i da je izraz pravilno zadat.

  Na primer izraz zapisan u prefiksnoj notaciji: `+ 5 * 3 2` se može predstaviti stablom:



  Napisati i funkcije koje ovako zadato stablo ispisuju u prefiksnom i infiksnom poretku i funkciju koja računa vrednost izraza koji se nalazi u stablu.

- Funkcija `main` u kojoj unosimo sa standardnog ulaza izraz u prefiksnoj notaciji pomoću funkcije `fgets`.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>

main() {

      cvor * drvo = NULL;
      char s[20], *ps;
      int i=0;

      printf("unesite aritmeticki izraz u prefiksnoj notaciji, novi red za kraj\n");

      /* Funkcija fgets cita sa standardnog ulaza (stdin) liniju (karaktere
         do unetog novog reda) ili dok se ne popuni 20 karaktera */
      if (fgets(s, 20, stdin) == NULL)
      {
          fprintf(stderr, "greska");
```

```
            exit(1);
        }

        /* Kako s sada sadrzi i oznaku za novi red na kraju brisemo je */
        while(s[i]!='\n')
          i++;

        /* i na kraju dodajemo '\0' da bi pripremili string za prenosenje
           u funkciju */
        s[i] = s[i+1];

        /* Kako je s niz karaktera (a niz je konstantni pokazivac) on ne
           sme biti predat funkciji koja kreira stablo (posto funkcija
           menja pokazivac koji dobija kao prvi argument) pa uzimamo
           pomocni pokazivac da bismo mogli da menjamo njegovu vrednost */
        ps = s;

        /* Kreiramo stablo pozivom funkcije prevori_u_stablo */
        pretvori_u_stablo(&ps, &drvo);

        /* Ispisujemo ga u prefiksnom poredku */
        ispisi_drvo(drvo);
        printf("\n");

        /* A zatim i u infiksnom. */
        ispisi_drvo_infiksno(drvo);

        /* Ispisujemo vrednost pocetnog izraza */
        printf(" = %d\n", izracunaj_drvo(drvo));

        obrisi_drvo(drvo);
    }
```

- Opisati strukturu čvora stabla, kao i funkcije za kreiranje čvora i brisanje stabla.

```
/* Definisemo novi tip operacija */
typedef enum operacija {pl = '+',min = '-', pod = '/', put = '*'} operacija;

/* Struktura koja predstavlja jedan cvor */
typedef struct _cvor
{
  int ind;       /* indikator da li se u cvoru nalazi operator
```

```
                    (u tom slucaju ind ima vrednost 1)
                    ili broj (u tom slucaju ind ima vrednost 0) */

  int br;         /* polje u kome cuvamo ceo broj, popunjeno samo ako
                     je polje ind postavljeno na 0 */

  operacija op;  /* polje u kome cuvamo operator, popunjeno samo ako
                     je polje ind postavljeno na 1 */

  struct _cvor *l, *d;
} cvor;


/* Funkcija koja pravi jedan cvor pri cemu ostavlja neinicijalizovane
   vrednosti br i op */
cvor * napravi_cvor()
{
  cvor * novi = (cvor *)malloc(sizeof(cvor));
  if (novi == NULL)
  {
      fprintf(stderr, "greska prilikom alokacije memorije\n");
      exit(1);
  }

  novi->l = NULL;
  novi->d = NULL;

  return novi;
}

/* Rekurzivna funkcija koja uklanja drvo, obilazak mora biti
   postorder */
void obrisi_drvo(cvor *drvo)
{
  if (drvo!=NULL)
  {
      obrisi_drvo(drvo->l);
      obrisi_drvo(drvo->d);
      free(drvo);
  }
}
```

- Napisati funkciju koja od učitanog stringa pravi binarno stablo na prethodno opisan način.

```
/* Rekurzivna funkcija koja parsira string. Predajemo joj adresu
   stringa da bi znali dokle smo stigli u toku parsiranja */
void pretvori_u_stablo(char ** s, cvor ** pdrvo)
{
   int c, br;
   cvor *novi;

   /* Uzimamo sledeci karakter iz stringa. */
   c = *(*s);

   /* Ako smo stigli do kraja stringa izlazimo */
   if (c == '\0') return;

   /* U suprotnom popunjavamo kreiramo novi cvor */
   novi  = napravi_cvor();

   /* Ako nije u pitanju cifra znaci da smo ucitali operator... */
   if (!isdigit(c))
   {
      /* ...postavljamo indikator na 1 */
      novi->ind = 1;
      /* ...i operator na ucitan karakter */
      novi->op = c;

      /* Unosimo podatak u drvo */
      *pdrvo = novi;

      /* Prelazimo na sledeci relevantan podatak, preskacemo blanko*/
      *s = *s+2;

      /* I rekurzivno parsiramo string da bismo formirali levo pa
         zatim i desno podstablo */
      pretvori_u_stablo(s, &((*pdrvo)->l));
      pretvori_u_stablo(s, &((*pdrvo)->d));
   }

   /* A ako je u pitanju cifra... */
   else
   {
      /* ...ucitavamo i ostatak broja ako postoji*/
      br = 0;
      while(isdigit(c))
      {
         br = br*10+c-'0';   /* uracunavamo tekuci karakter u broj */
         (*s)++;             /* pomeramo se za jedno mesto u stringu */
```

```
            c = *(*s);          /* i citamo sledeci karakter */
        }

        /* postavljamo indikator na 0 */
        novi->ind = 0;
        /* i brojevnu vrednost na br */
        novi->br = br;

        /* Unosimo podatak u drvo */
        *pdrvo = novi;

        /* Prelazimo na sledeci relevantan podatak. Uzimamo u obzir da
           pokazivac vec pokazuje na blanko (sem ako nismo stigli do
           kraja stringa). */
        if (*(*s) != '\0')
            *s = *s+1;
    }
}
```

- Napisati funkciju koja ispisuje stablo u prefiksnom poretku.

```
void ispisi_drvo(cvor *drvo) {
    if (drvo!=NULL)
    {
        /* Prvo ispisujemo koren. Proveravamo vrednost indikatora */
        if (!drvo->ind)
            /* ako je indikator jednak 0 stampamo broj */
            printf("%d", drvo->br);
        else
            /* a inace stampamo karakter */
            printf("%c", drvo->op);

        /* ...a zatim i levo pa desno podstablo. */
        ispisi_drvo(drvo->l);
        ispisi_drvo(drvo->d);
     }
}
```

- Napisati funkciju koja ispisuje stablo u infiksnom poretku, sa sve zagradama.

```
void ispisi_drvo_infiksno(cvor *drvo) {
  if (drvo == NULL)
    return;

  if (!drvo->ind)
     /* Ako smo naisli na brojevnu vrednost stampamo je. */
     printf("%d", drvo->br);
  else
  {
     /* U suprotnom imamo pravo stablo pa ispisujemo prvo levu
        zagradu... */
     printf("(");

     /* pa levi izraz... */
     ispisi_drvo_infiksno(drvo->l);

     /* pa operator... */
     printf(" %c ", drvo->op);

     /* pa desni izraz... */
     ispisi_drvo_infiksno(drvo->d);

     /* i na kraju ispisujemo desnu zagradu */
     printf(")");
  }
}
```

- Napisati funkciju koja izračunava vrednost izraza koji se nalazi u stablu.

```
int izracunaj_drvo(cvor *drvo) {
  if (!drvo->ind)
     return drvo->br;
  else
     switch (drvo->op)
     {
        case '+': return izracunaj_drvo(drvo->l) + izracunaj_drvo(drvo->d);
        case '-': return izracunaj_drvo(drvo->l) - izracunaj_drvo(drvo->d);
        case '*': return izracunaj_drvo(drvo->l) * izracunaj_drvo(drvo->d);
        case '/': return izracunaj_drvo(drvo->l) / izracunaj_drvo(drvo->d);
     }
}
```