

Dvočas X – Zadaci

Milan Banković
Matematički Fakultet

May 6, 2007

Sadržaj

1	Dinamički niz	2
1.1	Zadatak	3
1.1.1	array.h	3
1.1.2	test.c	5

1 Dinamički niz

Na ovom dvočasu ćemo demonstrirati rad sa funkcijama za dinamičku alokaciju memorije, na jednostavnom primeru implementacije dinamičkog niza (niza čija se dužina može menjati po potrebi). Naime, jedan od osnovnih nedostataka C nizova je to što njihova dužina mora biti poznata u vreme kompilacije, i ne može se dinamički menjati.

Rešenje se sastoji u tome da se prostor za skladištenje elemenata niza alokira dinamički, kao i da se po potrebi izvrši realokacija prostora. Ovo je naravno skupa operacija, i zato treba uvek alocirati više prostora nego što je neophodno, kako bi se realokacija redje izvršavala. U svakom trenutku ćemo opisati sa dve veličine: trenutnom dužinom niza – koja predstavlja broj elemenata koji su u upotrebi u datom trenutku, kao i trenutnom veličinom alociranog prostora (koja je veća ili jednaka od dužine niza) i koja predstavlja broj elemenata niza za koje je trenutno alociran prostor u memoriji. Kada se postavlja element na nekoj poziciji i na neku vrednost, tada se dešava sledeće:

- ako je i manje od trenutne dužine, tada se vrši ažuriranje elementa koji već postoji u nizu. Ovo je najjednostavniji slučaj.
- ako je i veće ili jednako od trenutne dužine, a manje od veličine alociranog prostora, tada se menja dužina niza, ali se ne alokira dodatni prostor jer za tim nema potrebe. Ovde se dobija na efikasnosti, jer se u velikom broju slučajeva ne alokira novi prostor.
- ako je i veće ili jednako od trenutno alociranog prostora, tada se mora izvršiti realokacija. Ovo je skupa operacija, pa zato tom prilikom alociramo više prostora nego što je potrebno kako bismo predupredili kasnije realokacije. Alociraćemo određeni procenat prostora više (npr. 25%).

Prilikom smanjivanja veličine niza (što je dozvoljeno) nećemo smanjivati alocirani prostor. Takođe, treba nakon korišćenja niza dealocirati kompletan prostor za skladištenje. Sve ovo je u najkraćim crtama uputstvo za zadatak koji sledi.

1.1 Zadatak

Dato je zaglavlje `array.h` koje predstavlja biblioteku funkcija za rad sa dinamičkim nizom, kao i fajl `test.c` koji sadrži `main()` funkciju koja testira funkcije ove biblioteke. Zadatak je implementirati funkcije biblioteke, tj. napisati fajl `array.c` koji sadrži definicije ovih funkcija. Detalji vezani za opis funkcija koje treba implementirati nalaze se u komentarima u datom zaglavlju. Nakon toga treba odvojeno prevesti biblioteku kao i test program, povezati sve u jedan izvršni fajl i pokrenuti ga.

1.1.1 array.h

```
#define MIN_SIZE 10
#define PERCENT 25

/* Struktura koja predstavlja dinamički niz */
typedef struct
{
    int length; /* trenutna dužina niza */
    int allocated; /* veličina alociranog prostora (>=length) */
    int *ptr; /* pokazivač na memoriju u kojoj se čuvaju elementi niza */
} Array;

/* pomoćna funkcija koja vrši realokaciju memorijskog prostora
   po potrebi. Ova funkcija ne može da smanji već alocirani
   prostor, već samo da ga poveća (na veličinu size). Funkcija
   vraća -1 u slučaju greske, a 0 ako je sve u redu. U slučaju
   greske niz a ostaje nepromenjen */
int reallocateArray (Array * a, int size);

/* funkcija vrši inicijalizaciju strukture Array.
   Mora se pozvati pre korišćenja strukture. Ona
   inicijalizuje dužinu niza na 0, i alocira prostor
   za MIN_SIZE elemenata. Vraća -1 u slučaju greske,
   a 0 ako je sve u redu. Ako je došlo do greske, niz
   se kasnije ne sme koristiti. */
int initializeArray (Array * a);

/* Funkcija vraća trenutnu dužinu niza */
int getLength (Array * a);
```

```

/* Funkcija vraca trenutnu velicinu alociranog prostora */
int getAllocated (Array * a);

/* Funkcija postavlja duzinu niza na length. U slucaju da je
nova duzina manja ili jednaka od dosadasnje duzine, elementi
na desnom kraju se gube (bivaju odseceni), ali se alocirani
prostor ne smanjuje (ne vrši se realokacija). U slucaju da je
nova duzina veca, tada se novi elementi na desnom kraju niza
inicijalizuju na nulu. Ukoliko je nova duzina veca i od trenutno
dostupnog alociranog prostora, vrši se realokacija uz proveru
pojave greske. Pri tom se uvek alokira PERCENT procenata vise
prostora nego sto je zahtevano novom duzinom, zbog efikasnosti
prilikom kasnijih dodavanja elemenata (da ne bismo realloc()
pozivali previse cesto). Funkcija vraca 0 ako je sve u redu,
a -1 u slucaju greske. U slucaju greske niz ostaje nepromenjen. */
int setLength (Array * a, int length);

/* postavlja element na poziciji i na vrednost value.
Funkcija vraca -1 ako je doslo do greske, 0 ako je
sve u redu. Ako je doslo do greske, niz ostaje
nepromenjen */
int setElement (Array * a, int i, int value);

/* funkcija vraca element na poziciji i. Ako je i vece
od getLength(a)-1, funkcija vraca 0 */
int getElement (Array * a, int i);

/* funkcija ubacuje na kraj niza novi element value.
Funkcija vraca -1 u slucaju greske, 0 ako je sve
u redu. U slucaju greske niz ostaje nepromenjen */
int pushBack (Array * a, int value);

/* Funkcija uklanja krajnji desni element niza i
vraca isti kao povratnu vrednost. Ovde ne moze
doci do greske zato sto se skracivanje niza uvek
uspesno obavlja (ne zahteva realokaciju, videti
funkciju setLength() ) */
int popBack (Array * a);

/* Funkcija oslobadja prostor koji je alocirani za elemente niza */
void deallocateArray (Array * a);

```

1.1.2 test.c

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

#include "array.h"

/* test program */
int
main (int argc, char **argv)
{

    Array a;
    int i, v;

    printf ("\nTestiranje funkcije initializeArray()\n");
    printf ("Inicijalizujemo niz a (duzina 0, alocirano 10)\n");
    if (initializeArray (&a) < 0)
    {
        fprintf (stderr, "%s: initializeArray() error!\n", argv[0]);
        exit (EXIT_FAILURE);
    }

/* ispis niza */
    printf ("-----\n");
    printf ("Velicina: %d\n", getLength (&a));
    printf ("Alocirano: %d\n", getAllocated (&a));
    printf ("Elementi niza:\n");
    for (i = 0; i < getLength (&a); i++)
        printf ("%d ", getElement (&a, i));
    printf ("\n");
    printf ("-----\n");

    printf ("\nTestiranje funkcije setElement()\n");

    printf("Unesite element koji zelite da dodate na kraj niza,
           ili <CTRL-D> za prekid:\n");
```

```

/* unosimo elemente niza */
while (scanf ("%d", &v) > 0)
{
    if (setElement (&a, getLength(&a), v) < 0) /* moze i pushBack(&a,v) */
    {
        fprintf (stderr, "%s: setElement() error!\n", argv[0]);
        exit (EXIT_FAILURE);
    }

    /* ispis niza */
    printf ("-----\n");
    printf ("Velicina: %d\n", getLength (&a));
    printf ("Alocirano: %d\n", getAllocated (&a));
    printf ("Elementi niza:\n");
    for (i = 0; i < getLength (&a); i++)
        printf ("%d ", getElement (&a, i));
    printf ("\n");
    printf ("-----\n");
    printf("Unesite element koji zelite da dodate na kraj niza,
           ili <CTRL-D> za prekid:\n");
}

/* dodajemo cetiri nule na kraj niza */
printf ("\nTestiranje funkcije setLength()\n");
printf
("Dodajemo jos cetiri nule na kraj niza
 (pritisnuti bilo koji karakter)\n");
while (isspace (getchar ()));

if (setLength (&a, getLength (&a) + 4) < 0)
{
    fprintf (stderr, "%s: setLength() error!\n", argv[0]);
    exit (EXIT_FAILURE);
}

/* ispis niza */
printf("-----\n");
printf ("Velicina: %d\n", getLength (&a));
printf ("Alocirano: %d\n", getAllocated (&a));

```

```

printf ("Elementi niza:\n");
for (i = 0; i < getLength (&a); i++)
    printf ("%d ", getElement (&a, i));
printf ("\n");
printf ("-----\n");

printf ("\nTestiranje funkcije pushBack()\n");
printf ("Uneti jos jedan element, ili <CTRL-D> ako ne zelite:\n");
/* dodajemo jos jedan element */
if (scanf ("%d", &v) > 0)
    if (pushBack (&a, v) < 0)
    {
        fprintf (stderr, "%s: pushBack() error!\n", argv[0]);
        exit (EXIT_FAILURE);
    }

/* ispis niza */
printf ("-----\n");
printf ("Velicina: %d\n", getLength (&a));
printf ("Alocirano: %d\n", getAllocated (&a));
printf ("Elementi niza:\n");
for (i = 0; i < getLength (&a); i++)
    printf ("%d ", getElement (&a, i));
printf ("\n");
printf ("-----\n");

/* skidamo elemente i ispisujemo ih u obrnutom poretku */
printf ("\nTestiranje funkcije popBack()\n");
printf ("Ispis elemenata niza u obrnutom poretku:\n");
while (getLength (&a))
    printf ("%d ", popBack (&a));

printf ("\n");

printf("\nNakon uzastopnog skidanja elemenata sa desnog kraja,
      \nniz bi sada trebalo da bude prazan!\n");
printf
    ("Primetimo da je alocirani prostor ostao isti,
     tj.\nne smanjuje se smanjenjem dimenzije niza!\n");

/* ispis niza */

```

```

printf ("-----\n");
printf ("Velicina: %d\n", getLength (&a));
printf ("Alocirano: %d\n", getAllocated (&a));
printf ("Elementi niza:\n");
for (i = 0; i < getLength (&a); i++)
    printf ("%d ", getElement (&a, i));
printf ("\n");
printf ("-----\n");

/* oslobadjamo niz */
deallocateArray (&a);

return EXIT_SUCCESS;
}

```