Tema 8

Dinamička alokacija memorije i jednostruko spregnuta lista

Dinamička alokacija memorije

- Koristi se u slučaju da potrebna količina memorije koju program koristi nije unapred poznata
- Upravljanje preko pokazivača
- Potrebno je zatražiti dodatnu memoriju od operativnog sistema i nakon korišćenja je osloboditi
    - Ako se memorija ne oslobodi, nastaje problem *Memory Leak*

Korišćene funkcije

Dinamičko zauzimanje memorije, bez inicijalizacije:

```
void *malloc(size_t size);
```

Dinamičko zauzimanje memorije, inicijalizacija na nulu:

```
void *calloc(int n, size_t size);
```

Prebacivanje dinamički zauzete memorije u veću/manju dinamički zauzetu memoriju:

```
void *realloc(void *ptr, size_t size);
```

Oslobađanje dinamički zauzete memorije:

```
void free(void *ptr);
```

Funkcije se nalaze u biblioteci `stdlib.h`

Primer 1

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

int main()
{
    int n;  // Dimenzija niza
    int *a;
    int i, max;

    do
    {
        printf("Unesi dimenziju niza: ");
        scanf("%d", &n);
    } while(n <= 0);

    // Sada kada znamo koliko je memorije potrebno
    // pozivamo funkciju malloc za dinamicku alokaciju
    // memorije
    a = (int *) malloc(n * sizeof(int));
```

```c
    if(a == NULL)
    {
        printf("Greska: Nema dovoljno memorije!\n");
        return 1;
    }

    // Nadalje a koristimo sa indeksnom notacijom
    for(i = 0;i < n;i++)
    {
        printf("a[%d] = ", i);
        scanf("%d", &a[i]);
    }

    // Nalazimo maksimum
    max = a[0];
    for(i = 1;i < n;i++)
    {
        if(a[i] > max)
        {
            max = a[i];
        }
    }
```

```
    printf("Najveci element je %d\n", max);

    // Obavezno oslobadjamo dinamicki alociranu memoriju
    free(a);

    return 0;
}
```

Zadatak 1

Modifikovati `Primer 1` tako da je moguće unositi proizvoljan broj prirodnih brojeva. Umesto unosa `n` na početku, postaviti ga na početnu vrednost 5. Svaki put kad se kapacitet niza popuni, proširiti `n` množenjem sa 2 i pomoću funkcije `realloc` napraviti, novi, veći niz te dimenzije. Unos podataka se završava kada korisnik unese `-1`.

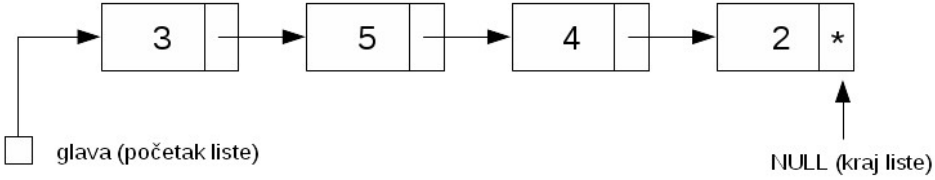
Primer rada programa:

```
Unositi prirodne brojeve. Uneti -1 za kraj unosa.
Broj: 3
Broj: 5
Broj: 7
Broj: 2
Broj: 6
Broj: 10
Broj: 35
Broj: 23
Broj: 11
Broj: -1

Najveci element je: 35
```

Jednostruko spregnuta lista

- Dinamička struktura podataka
- Element liste (čvor) se sastoji iz dva polja
  - Podatak (informacioni deo čvora liste, celobrojna vrednost, karakter, struktura itd.)
  - Pokazivač na sledeći element liste (sadrži adresu sledećeg elementa liste)
- Pristup elementima vrši se preko pokazivača na prvi element liste - glave liste



Operacije nad listom

1. Inicijalizacija liste
2. Unos novog elementa na kraj liste
3. Listanje liste (prikaz svih elemenata iz liste)
4. Traženje čvora sa zadatom vrednošću elementa u listi
5. Brisanje čvora sa zadatom vrednošću elementa iz liste
6. Brisanje cele liste (oslobađanje memorije)

Element (čvor) liste

```
typedef struct cvor_st
{
    int el;
    struct cvor_st *sledeci;
} CVOR;
```

- Podatak (informacioni deo čvora) je celobrojna vrednost `el`
- `sledeci` sadrži adresu sledećeg čvora
    - Ukoliko je vrednost `NULL`, u pitanju je poslednji čvor u listi

Inicijalizacija liste

```
void inicijalizacija(CVOR **pglava)
{
    *pglava = NULL;
}
```

- Glava liste prenosi se po referenci
  - Pošto će se adresa pokazivača `glava` iz main funkcije menjati u funkciji `inicijalizacija`, funkciji je potrebno proslediti adresu pokazivačke promenljive `glava`, otud dve zvezdice (**)
- Kreiranje prazne liste bez elemenata
  - Lista je prazna ukoliko je vrednost glave `NULL`
- glava liste dobija vrednost `NULL`

Pravljenje novog čvora

```
CVOR *napravi_cvor(int el)
{
    CVOR *novi = (CVOR *)malloc(sizeof(CVOR));

    if (novi == NULL)
    {
        printf("Nije moguce zauzeti memoriju!\n");
        exit(EXIT_FAILURE);
    }

    novi->el = el;
    novi->sledeci = NULL;

    return novi;
}
```

- Dinamička alokacija memorije za novi čvor pomoću funkcije `malloc`
- Popunjavanje informacionog dela čvora
- Postavljanje vrednosti `sledeci` na `NULL`

Unos novog elementa na kraj liste

```
void dodaj_na_kraj(CVOR **pglava, CVOR *novi)
{
    if (*pglava == NULL)
    {
        *pglava = novi;
    }
    else
    {
        CVOR *tekuci = *pglava;
        while (tekuci->sledeci != NULL)
        {
            tekuci = tekuci->sledeci;
        }
        tekuci->sledeci = novi;
    }
}
```

Unos novog elementa na kraj liste

- glava liste se prenosi po referenci (**), zbog slučaja kad je lista prazna
- Pomoću `while` petlje ide se od glave liste do zadnjeg čvora
- Za vrednost `sledeci` trenutno aktuelnog zadnjeg čvora postaviti adresu novog čvora umesto `NULL`

Listanje (ispis) liste

```
void ispisi_listu(CVOR *glava)
{
    CVOR *tekuci = glava;

    printf("[");
    while (tekuci != NULL)
    {
        printf("%d", tekuci->el);
        if (tekuci->sledeci != NULL)
        {
            printf(", ");
        }

        tekuci = tekuci->sledeci;
    }
    printf("]\n");
}
```

- Pomoću `while` petlje prolazi se kroz listu od prvog do zadnjeg elementa i ispisuje se njihova vrednost

Traženje čvora

```
CVOR *nadji_cvor(CVOR *glava, int el)
{
    CVOR *tekuci = glava, *nadjen = NULL;

    while (tekuci != NULL)
    {
        if (tekuci->el == el)
        {
            nadjen = tekuci;
            break;
        }
        tekuci = tekuci->sledeci;
    }
    return nadjen;
}
```

Traženje čvora

- Pomoću `while` petlje prolazi se kroz listu i traži se čvor koji sadrži prosleđenu vrednost
- Ukoliko je pronađen element, traženje će stati (naredba `break`) i funkcija će vratiti pokazivač na nađeni element
- Funkcija vraća `NULL` pokazivač ako nema traženog elementa

Brisanje liste

```
void obrisi_listu(CVOR **pglava)
{
    CVOR *tmp;

    while (*pglava != NULL)
    {
        tmp = *pglava;

        *pglava = (*pglava)->sledeci;
        tmp->sledeci = NULL;
        free(tmp);
    }
}
```

Brisanje liste

- Glava liste prenosi se po referenci (biće menjana)
- Pokazivač ka trenutnoj glavi liste čuva se u pokazivaču `tmp`
- Glava liste postaje sledeći element u listi
- Čvor čija adresa je u `tmp` raskida vezu sa ostatkom liste (sledeći se postavlja na `NULL`) i oslobađa se njegova dinamički alocirana memorija pomoću funkcije `free`

Test program

```
int main()
{
    CVOR *glava;

    inicijalizacija(&glava);

    dodaj_na_kraj(&glava, napravi_cvor(1));
    dodaj_na_kraj(&glava, napravi_cvor(2));
    dodaj_na_kraj(&glava, napravi_cvor(3));
    dodaj_na_kraj(&glava, napravi_cvor(4));

    ispisi_listu(glava);
```

```
    obrisi_listu(&glava);

    ispisi_listu(glava);

    return EXIT_SUCCESS;
}
```

Provera postojanja curenja memorije

- Uz pomoć programa Valgrind
- Pokretanje: `valgrind <ime-programa>`
  - Primer: `valgrind ./a.out`

Testirati prethodni primer jednostruko spregnute liste pomoću programa `Valgrind`. Porediti ispise programa pre i posle zakomentarisanog poziva funkcije `obrisi_listu`.

- Napomena: ukoliko nemate `Valgrind` na vašem sistemu, instalira se na sledeći način: `sudo apt-get -y install valgrind`

Zadatak 1

U jednostruko spregnutoj listi nalaze se ocene koje je student dobio tokom godine. Izračunati prosek i rezultat ispisati na standardnom izlazu (ekranu).

```
int main()
{
    char broj_indeksa[] = "ee300-2020";
    char ime_studenta[] = "Jovan Jovanovic";
    CVOR *glava;

    inicijalizacija(&glava);

    dodaj_na_kraj(&glava, napravi_cvor(8));
    dodaj_na_kraj(&glava, napravi_cvor(7));
    dodaj_na_kraj(&glava, napravi_cvor(9));
    dodaj_na_kraj(&glava, napravi_cvor(10));
    dodaj_na_kraj(&glava, napravi_cvor(6));
    dodaj_na_kraj(&glava, napravi_cvor(8));
    dodaj_na_kraj(&glava, napravi_cvor(9));
    dodaj_na_kraj(&glava, napravi_cvor(8));
```

```
    printf("Student %s, sa brojem indeksa %s, ima prosek %.2lf\n",
           ime_studenta, broj_indeksa, prosek_studenta(glava));

    obrisi_listu(&glava);

    return EXIT_SUCCESS;
}
```

Očekivani ispis programa:

```
Student Jovan Jovanovic, sa brojem indeksa ee300-2020, ima prosek 8.12
```

Zadatak 2

Napisati jednostruko spregnutu listu, koja sadrži stringove maksimalne dužine 30 karaktera. Popuniti listu sa određenim stringovima (može biti zadato unapred, bez potrebe za unosom), zatim putem tastature uneti karakter. Izračunati srednju vrednost udela zadatog karaktera na nivou cele liste. Ispisati rezultat zaokružen na dve decimale.

Primer:

Zadato slovo a i reči u listi:

Pera -> 0.25

Mika -> 0.25

Laza -> 0.5

Srednja vrednost udela: 0.33 (zaokruženo na dve decimale)