1. međuispit iz predmeta Algoritmi i strukture podataka

19. ožujka 2008.

Napomena za sve zadatke:
- Nije dopušteno korištenje goto naredbe, te statičkih i globalnih varijabli.

Zadatak 1. *(4 boda)*

Zadana je funkcija `Zbroj` koja vraća zbroj elemenata polja veličine `brojElemenata` čiji je prototip:

```
int Zbroj (int poljeZbroj[], int brojElemenata);
```

Napisati funkciju `IzborParnih` koja u zadanom polju cijelih brojeva `polje` pronalazi prvih 5 parnih brojeva i sprema ih u pomoćno polje `poljeParnih` veličine 5. Ako parnih brojeva ima manje od 5, ostatak polja `poljeParnih` popuniti nulama. Iz funkcije `IzborParnih` pozvati zadanu funkciju `Zbroj` koja će izračunati zbroj elemenata polja `poljeParnih` te ga vratiti preko argumenta `zbrojParnih`.
Prototip funkcije je:

```
void IzborParnih (int polje[], int brojElemenata, int *zbrojParnih);
```

Izgled glavnog programa (nije potrebno pisati):

```
int main (){
...
  IzborParnih (polje, brojElemenata, &zbrojParnih);
...
}
```

Nacrtati sistemski stog u trenutku neposredno prije završetka izvođenja funkcije `Zbroj` *(*prije poziva naredbe `return`).

Zadatak 2. *(4 boda)*

Napišite funkciju čiji je prototip:

```
int *vratiPolje (int *brelem);
```

koja će dinamički alocirati prostor za polje od 200 elemenata i napuniti ga slučajno odabranim cijelim brojevima iz intervala [-200,300]. Ako je broj elemenata s vrijednošću 0 manji od 10, polje treba proširiti za dodatnih 10 elemenata i njih napuniti nulama.
Funkcija vraća novonastalo polje, a preko argumenta `brelem` vraća broj elemenata polja. Treba napisati glavni program u kojem će se definirati potrebne varijable, pozvati funkcija `vratiPolje` i ispisati dobiveno polje.

Zadatak 3. *(3 boda)*

a) *(1 bod)* Odredite složenost **O** i **Ω** sljedećeg dijela kôda. Obrazložite svoje odgovore.

```
for (i = 0; i < n; i++) {
  if (p[i][i] == i){
    printf("%d", i);
     break;
  }
 }
```

b) *(2 boda)* Odredite složenost **O** masno otisnutog odsječka kôda (petlja *while*). Obrazložite svoj odgovor.

```
int b;
...
char *rezultat = ...
rezultat[0] = '\0';
while (n > 0) {
  if (n % 2 == 1)
    rezultat = strcat(rezultat, "1");
  else
    rezultat = strcat(rezultat, "0");
  n = n/2;
}
```

Zadatak 4. *(4 boda)*

Jedan zapis datoteke organizirane po načelu raspršenog adresiranja definiran je strukturom:

```
typedef struct{
      int sifra;
      char naziv[50+1];
      double cijena;
} zapis;
```

Zapis je prazan ako je na mjestu šifre vrijednost nula. Parametri za raspršeno adresiranje nalaze se u datoteci *parametri.h* i oni su:

- `BLOK` ..... veličina bloka na disku
- `MAXZAP`..... broj zapisa
- `C` ..... broj zapisa u jednom pretincu
- `M` ..... broj pretinaca

Napisati funkciju koja će vratiti broj punih pretinaca unutar datoteke. Funkcija treba imati prototip:

```
int broj_popunjenih (FILE *f);
```

ZADATAK 1

```
#include <stdio.h>

void IzborParnih (int polje[], int brojElemenata, int *zbrojParnih) {

	int i, j = 0;
	int poljeParnih [5] = {0};

	for (i = 0; i < brojElemenata; i++) {
		if (polje[i] % 2 == 0) {
			poljeParnih[j] = polje[i];
			j++;
		}
		if (j == 5) break;
	}
	*zbrojParnih = Zbroj (poljeParnih, 5);
}

int Zbroj (int poljeZbroj[], int brojElemenata) {

	int j, zbroj = 0;
	for (j = 0; j < brojElemenata; j++) {
		zbroj = zbroj + poljeZbroj [j];
	}
	return zbroj;
}
int main (void) {

int i, zbrojParnih;
int polje [10];

	srand ((unsigned) time (NULL));

	for (i = 0; i < 10; i++) {
		polje [i] = rand () % 50 + 1;
	}
	IzborParnih (polje, 10, &zbrojParnih);

	printf("%d Ovo je zbroj \n", zbrojParnih);


	return 0;
}
```

Sistemski stog:

zbroj
j
povratna adresa
poljeZbroj
brojElemenata
poljeParnih
j
i
povratna adresa
polje
brojElemenata
zbrojParnih

ZADATAK 2

```
int * vratiPolje(int * brojelem){
  int * polje, i, brojnula=0;
  *brojelem=200;
  polje=(int *)malloc(200*sizeof(int));
  for (i=0; i<200; i++){
    polje[i]=rand();
    if (polje[i]==0) brojnula++;
  }
  if (brojnula<10) {
    polje =(int*)realloc (polje, 210*sizeof(int));
    for (i=200; i<210; i++) polje[i]=0;
    if (brojnula<10) *brojelem=210;
  }
  return polje;
}


int main(){
  int *p, i, brelem;
  p=vratiPolje(&brelem);
  for (i=0; i<brelem; i++)
    printf("%d ", p[i]);
  free(p);
  return 0;
}
```

ZADATAK 3

a) O(n) Nagori slučaj nastupa kada je za svaki i p[i][i] != i. Tada se petlja (naredba if unutar nje) izvršava n puta.

$\Omega(1)$ Najbolji slučaj nastupa kada je p[0][0] == 0, tada iskačemo s break; iz for-petlje nakon izvođenja 2 naredbe.

b) Složenost: O(logn)

Broj n cijelo vrijeme dijelimo na pola. Zaustavljamo se kada je n postao 1 (tada je n/2 = 0).

Treba odrediti koliko koraka treba da djeleći n sa dva, dođemo do 0, odnosno 1?

Dakle u prvom koraku imamo broj n, u drugom n/2, pa n/4, ..., u k-tom koraku $n/2^k$. Tražimo k za koji je $n/2^k = 1$. Rješenje je $k = \log_2 n$.

Zadatak 4:

```
int broj_popunjenih (FILE *f) {

      zapis pretinac[C];
      int i, j, popunjen, br=0;
      for (i = 0; i < M; i++) {
            fseek (f, i*BLOK, SEEK_SET);
            fread (pretinac, sizeof (pretinac), 1, f);
            popunjen = 1;
            for (j = 0; j < C; j++) {
                  if (pretinac[j].sifra == 0) {
                        popunjen = 0;
                        break;
                  }
            }
            if (popunjen)
                  br++;
      }
      return br;
}
```