Strukture podataka i algoritmi 1

I kolokvijum

22. 04. 2019.

Na **Desktop-u** u direktorijumu **Rad** kreirati direktorijum **ImePrezime_BrIndeksa** i unutar njega sačuvati programe koji sadrže rešenja datih zadataka. Rešenje 1. zadatka **mora** da se nalazi u fajlu **Zadatak1.c**, a rešenje 2. zadatka **mora** da se nalazi u fajlu **Zadatak2.c**. Samo navedeni fajlovi biće pregledani. Samo zadaci koje se uspešno kompajliraju će biti bodovani, a broj poena će biti određen procentom ispravnih izvršavanja za svaki test primer.

1. Napisati funkciju ***Bullet*** koja ispituje da li niz karakera, koji prihavata kao argument, predstavlja ispravnu liniju u nabrajanju. Ispravna linija u nabrajanju je ako
   - počinje brojem (bar jedna cifara),
   - potom se nalazi bar jedan od simbola slova, cifrara i SPACE karaker u proizvoljnom redosledu i
   - završava se jednim od znakova „.“, „?“, „!“, iza kojih su dozvoljeni samo SPACE karakteri, ali mogu biti izostavljeni.

   Koristeći funkciju ***Bullet*** napisti program koji najpre učitava broj linija teksta koje će biti učitane sa standardnog ulaza, a potom linije teksta čija dužina neće prelaziti 100 karaktera. Svaku liniju teksta koja predstavlja ispravnu liniju u najbrajanju ispisati na izlazu, a nakon završenog učitavanja ispisati broj neispravnih linija.

Primer.

Ulaz	Izlaz
5 1Prva linija. 2 linija Treca linija! 55? 41. C!	1Prva linija. 55? 41. C! 2

Bodovanja. Zadatak vredi 12 poena ukoliko je ispis ispravnih linija nakon završenog učitavanja i ukoliko se koristi dinamička alokacija memorija. Korišćenje samo statički definisanih nizova umanjuje vrednost zadatka za 25%. Ispisivanje ispravnih linija nakon svake učitane linije umesto na kraju, takođe, umanjuje vrednost zadataka za 20%.

2. Napisati rekurzivnu funkciju ***Parne*** koja određuje broj parnih cifara u nenegativnom ceolom broju koji prihvata kao argument funkcije. Napisati funkciju ***Ucitaj*** koja učitava niz celih brojeva. Napisati funkciju ***Ispisi*** koja na standardni izlaz ispisuje niz celih brojeva koje prihvata kao argumant. Napisati program koji koristeći navedene funkcije učitava dužinu niza A, potom niz celih brojeva A, zatim od njega formira nov niz B koji za

odgovarajuće elemente niza A sadrži broj parnih cifara broja ukoliko je vrednost pozitivna, odnosno vrednost -1 ukoliko je odgovarajući element negativan i ispisuje dobijeni niz. Program potom učitava koliko elemenata treba dodati na niz A, učitava te elemente, na niz B dodaje odgovarajuće vrednosti za unete brojeve i ispisuje čitav niz B.

Primer.

Ulaz	Izlaz
6 56 125 -3 3684 -56917 68742 3 -68 8654 36	1 1 -1 3 -1 4 1 1 -1 3 -1 4 -1 3 1

Bodovanja. Zadatak vredi 10 poena ukoliko su korektno definisane sve date funkcije i pri tome nisu korišćeni nizovi fiksne dužina, već je memorijski prostor alociran dinamički. Ukoliko je funkcija ***Parne*** definisana iterativno umesto rekurzivno, vrednost zadatka se umanjuje za 25%. Ukoliko su svi nizovi definisani statički vrednost zadatka se umanjuje za 25%. Ukoliko se učitavanje niza A realizuje u glavnom programu, a ne u funkciji, vrednost zadatka se umanjuje za 15%. Ukoliko se ne realizuje drugi deo zadatka, sa učitavanjem dodatnih elemenata niza A, vrednost zadatka se umanjuje za 10%.

Vreme izrade zadataka 110 minuta.

Rešenja:

1. zadatak

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

/*
Funkcije iz ctype.h
isdigit(c) - vraca 1 ako je karakter c cifra, 0 ako nije
isalnum(c) - vraca 1 ako je karakter c cifra ili slovo, 0 ako nije
olaksavaju proveru karaktera, ne moraju da se koriste
*/

// vraca 1 ako je recenica dobra, 0 ako nije
int bullet(char *s)
{
	// ako ne pocinje cifrom, recenica nije dobra
	if (isdigit(s[0]) == 0)
		return 0;

	// trazimo poslednji karakter koji nije space
	int i = strlen(s) - 1;
	while(s[i] == ' ' && i > 0)
		i--;

	// ako taj karakter nije tacka, upitnik ili uzvicnik, recenica nije
dobra
	if (s[i] != '.' && s[i] != '?' && s[i] != '!')
		return 0;

	// proveravamo da li je bar jedan karakter izmedju cifra, slovo ili
space
	int j;
	for (j = 1; j < i; j++)
		if (isalnum(s[j]) || s[j] == ' ')
			return 1;

	return 0;
}

char* ucitaj()
{
	char str[200];
```

```
	gets(str); // ucitavanje linije

	// alociranje memorije potrebne za ucitani string
	// posto za svaki karakter treba 1 bajt, broj bajtova koji treba da
se alocira je duzina stringa
	char *s = (char*)malloc(strlen(str));
	strcpy(s, str); // kopiranje

	return s;
}

int main()
{
	int i, n;
	scanf("%d", &n); // ucitavanje broja linija
	getchar(); // kupi se enter, jer scanf nakon ucitavanja broja ostaje
u istoj liniji, i bez ovoga bi gets pokupimo samo enter

	char **recenice = (char**)malloc(n * sizeof(char*)); // alociranje
memorije
	for (i = 0; i < n; i++)
		recenice[i] = ucitaj(); // ucitavanje n linija

	int brojNeispravnih = 0;
	for (i = 0; i < n; i++)
	{
		if (bullet(recenice[i]) == 1)
			printf("%s\n", recenice[i]); // ako je dobra recenica,
prepisuje se
		else
			brojNeispravnih++; // u suprotnom, broji se broj
nepravilnih recenica
	}
	printf("%d\n", brojNeispravnih);
}
```

2. zadatak

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

int parne(int broj)
{
	if (broj < 0)
		return -1; // za negativne brojeve vraca -1
	else if (broj == 0)
		return 0; // kada se skinu sve cifre, rekurzija prestaje
```

```
	else if (broj % 2 == 0)
		return 1 + parne(broj / 10); // ako je cifra parna, broji se
	else
		return parne(broj / 10); // u suprotnom samo nastavlja dalje
da proverava
}

int* ucitaj(int n)
{
	int i;
	int *niz = (int*)malloc(n * sizeof(int));
	for (i = 0; i < n; i++)
		scanf("%d", &niz[i]);

	return niz;
}

void ispisi(int *niz, int n)
{
	int i;
	for (i = 0; i < n; i++)
		printf("%4d", niz[i]);
	printf("\n");
}

int main()
{
	int i, n;
	scanf("%d", &n);
	int *a = ucitaj(n);

	int *b = (int*)malloc(n * sizeof(int));
	for (i = 0; i < n; i++)
		b[i] = parne(a[i]);

	ispisi(b, n);

	int m;
	scanf("%d", &m);
	a = (int*)realloc(a, (n + m) * sizeof(int));
	for (i = n; i < n + m; i++)
		scanf("%d", &a[i]);

	b = (int*)realloc(b, (n + m) * sizeof(int));
	for (i = n; i < n + m; i++)
		b[i] = parne(a[i]);
```

```
        ispisi(b, n + m);
}
```