P2: Osma nedelja
Stringovi i dinamička memorija
Uvod u funkcije

Zadatak 1

- **Sastaviti program na programskom jeziku C koji učitava decimalan ceo broj u obliku niza znakova i ispisuje njegovu binarnu predstavu. Pretpostaviti da se za interno predstavljanje celih brojeva koristi 16 bitova.**
- **Standardne bibliotečke funkcije za ispis, poput `printf`, ne podržavaju ispis bitova celog broja**
  - rešenje je u korišćenju bitskih operatora i odgovarajućih maski za ekstrakciju pojedinačnih bitova

Rad sa stringovima

- **Niz znakova na jeziku C koji na svom kraju ima specijalni karakter `'\0'` se naziva string**
  - karakter **`'\0'`** se naziva terminalni simbol i on označava kraj stringa
  - na taj način sve funkcije koje rade sa stringovima imaju mogu da odrede kraj stringa bez eksplicitnog podatka o njegovoj dužini
  - zadatak je programera da prilikom manipulacije stringovima obezbedi ispravno postavljanje završnog znaka
- **Za obradu strignova se koriste bibliotečke funkcije iz standardnog zaglavlja `<string.h>` kao što su:**
  - **`strlen(s)`** : string length – vraća dužinu stringa **`s`**
  - **`strcpy(t,s)`** : string copy – kopira string **`s`** u string **`t`** i prepisuje završni znak
  - **`strcat(t,s)`** : string concatenate – nadovezuje string **`s`** na kraj stringa **`t`**
- **Prilikom korišćenja ovih i sličnih funkcija programer je dužan da obezbedi dovoljno memorije za rezultujući string!**

Zadatak 1 - rešenje

- **Rešenje**
  - prilikom čitanja stringa se koristi konverzija **`%s`**
  - za konverziju stringa u ceo broj se koristi bibliotečka funkcija **`atoi()`** koja vraća vrednost celog broja ili 0 ukoliko konverzija nije uspela

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main() {
  char dec[10];
  short int bin, i;

  printf("Unesite decimalan broj: ");
  /* ucitava string sa standardnog ulaza (dec=&dec[0]) */
  scanf("%s",dec);

  if (strlen(dec)  &&  (bin=atoi(dec))) {
    printf("Binarni broj: ");
```

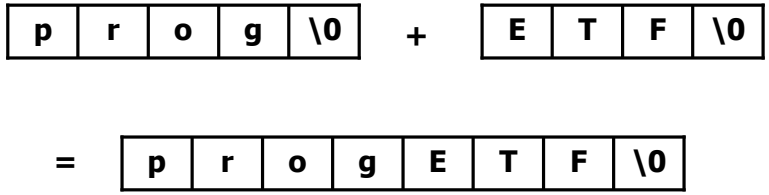
Zadatak 1

```
   i=-1;
   while (++i<16) {
      putchar((bin & 0x8000) ? '1' : '0');
      bin <<= 1 ;
      if (i%4 == 3) putchar(' ');
   }
  printf("\n");
  }
  else
      printf("Neispravan broj ili nula\n");
}
```

- **Ispis binarne predstave celog broja:**
  - u svakoj iteraciji petlje se ispiše jedan bit broja, počev od bita najveće težine
    `bit15 bit14 ... bit2 bit1 bit0`
  - koristi se maska 0x8000 za izdvajanje bita najveće težine, a zatim se broj pomera za jedno mesto ulevo

Zadatak 2 – konkatenacija (C70)

- **Napisati program na programskom jeziku C koji učitava dva znakovna niza, izvrši nadovezivanje drugog na prvi, okrene "naopako" dobijeni niz i ispiše ga na standardnom izlaznom uređaju.**
- **Konkatenacija:**



Zadatak 2 - rešenje

- **Rešenje**

```
#include <stdio.h>
#include <string.h>
void main() {
  int n;
  char c, *d, *p, *prvi, *drugi;
  printf("Maksimalna duzina: ");
  scanf("%d\n", &n);
  p = calloc(2*n+1, sizeof(char)) ;
  d = calloc(n+1, sizeof(char));
  if ((NULL == d) || (NULL == p))
      printf("Nije moguca alokacija!\n");
  else  {
```

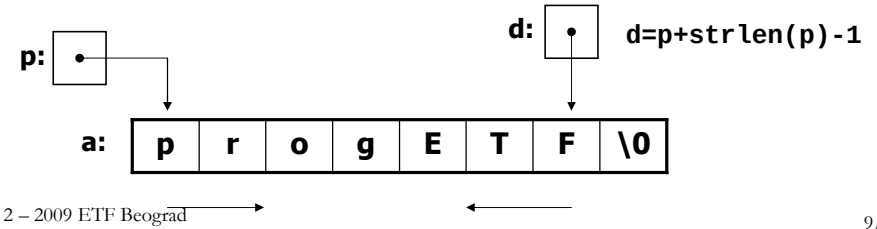

Zadatak 2

```
      prvi = p; drugi = d;
      while ((*p = getchar()) != '\n') p++;
      *p = '\0';
      while ((*d = getchar()) != '\n') d++;
      *d = '\0';
      p = prvi; d = drugi;
      /* konkatenacija */
      while (*p) p++;
      while (*p++=*d++);
      /* okretanje */
      p = prvi;
      for (d=p+(strlen(p)-1); p < d; p++, d--)
            c=*p, *p=*d, *d=c;
      printf("%s\n", prvi);

      free(p); free(d);
  }
}
```

Zadatak 2 – komentar

- **Za string `p` se alocira dvostruko više memorije zbog nadovezivanja stringova**
  - alocira se 2n+1 elemenata niza, jer je potrebno alocirati i jedan dodatni element za terminalni simbol
- **Učitavanje stringova se vrši znak po znak, korišćenjem bibliotečke funkcije `getchar()`, sve dok se ne učita znak za kraj reda `'\n'`**
- **Obrtanje dobijenog stringa se vrši poznatim algoritmom, ali se vodi računa da se ne pomeri završni znak**



Zadatak 2

- **Samu konkatenaciju (nadovezivanje) stringova je moguće izvršiti pomoću bibliotečke funkcije `strcat(t,s)` koja nadovezuje string `s` na string `t` uz ispravno postavljanje završnog znaka**
- **Deo koda:**

```
while (*p) p++;
while (*p++=*d++);
```

  - prva petlja pomera pokazivač p tako da ukazuje na mesto završnog znaka prvog stringa
  - druga petlja prepisuje sadržaj drugog stringa na kraj prvog, uz prepisivanje završnog znaka, kada se ciklus zaustavlja
- **bi trebalo zameniti sa pozivom:**

```
strcat(p,d);
```

- **Takođe, petlju `while (*p++=*d++)` u gornjem bloku je moguće zameniti pozivom funkcije `strcpy(t,s)` koja kopira sadržaj stringa na koji ukazuje pokazivač `s` na mesto u memoriji na koje ukazuje pokazivač `t`:**

```
strcpy(p,d);
```


Zadatak 3 (C57)

- **Da bi se učitani znakovni niz (string) precizno ispisao «unatrag» (samo znaci koji su unošeni sa standardnog ulaza), koja naredba se može staviti umesto `***`?**

```
scanf("%s", string1);
a = string1;
***
while (a>string1) putchar(*(--a));
```

- **Odgovori:**

A) `while (*a) a++;`

B) `for (; *a; a++);`

C) `while (*a++);`

Zadatak 3 - rešenje

- **Iz naredbe za ispis stringa:**

```
while (a>string1) putchar(*(--a));
```

- **se jasno vidi da se pokazivač `a` prvo dekrementira, a zatim ispiše znak na koji on ukazuje**
- **Umesto `***` mora da stoji naredba koja će postaviti pointer `a` tako da pokazuje na `NULL` (`'\0'`) karakter pointera `string1`**
- **Odgovori pod A) i B) su korektni, jer predstavljaju ekvivalentne petlje**
- **Odgovor pod C) nije korektan, jer postavlja (zbog bočnog efekta) pointer `a` na znak iza `'\0'` karaktera**
- **Odgovor: V (A i B)**

Zadatak 4 – realokacija stringa (S1)

- **Napisati program na programskom jeziku C koji sa standardnog ulaza čita znak po znak do kraja linije. Znakovi se čitaju jedan po jedan. U početku se dinamički alocira niz od 10 elemenata. Nadalje, svaki put kada u nizu više nema mesta, niz se proširi za još 10 znakova. U slučaju da realokacija bude neuspešna, završiti čitanje i ispisati ono što može da stane u već alocirani prostor.**

Funkcija realloc()

- **Bibliotečka funkcija iz `<stdlib.h>`**

  `void* realloc(p, vel)`
- **se koristi za izmenu veličine dinamičkog niza (bloka memorije) koji je već alociran**
  - **`p`** je pokazivač na već dodeljenu memoriju koju želimo da realociramo
  - **`vel`** je nova veličina potrebnog prostora
- **U slučaju smanjivanja veličine bloka memorije, skraćivanje se vrši sa kraja, a sadržaj zadržanih bajtova se čuva**
- **U slučaju povećanja veličine bloka u memoriji, novi bajtovi se dodaju na kraj**
- **U opštem slučaju, adresa zauzetog bloka pre i posle realokacije ne mora biti ista!**
  - funkcija **`realloc()`** može nakon skraćivanja ili povećanja bloka da blok premesti na novo mesto u memoriji (zbog optimizacija utroška memorijskog prostora)
  - jedina validna adresa bloka nakon realokacije je ona koju preko svoje povratne vrednosti vrati funkcija **`realloc()`**
- **U slučaju neuspeha, funkcija `realloc()` neće pomerati, niti menjati postojeći blok u memoriji**

Zadatak 4 - rešenje

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

void main() {
 char *staro_p=NULL, *novo_p, c;
 int i = 0;
 while( (c=getchar()) != '\n') {
   if (i%10 == 0) {
    novo_p = realloc( staro_p, (i+10) * sizeof(char) );
    if (novo_p == NULL) {
      printf("Neuspesno realociranje\n");
      break;
    } else staro_p = novo_p;
   };
   staro_p[i++] = c;
 }
```


Zadatak 4

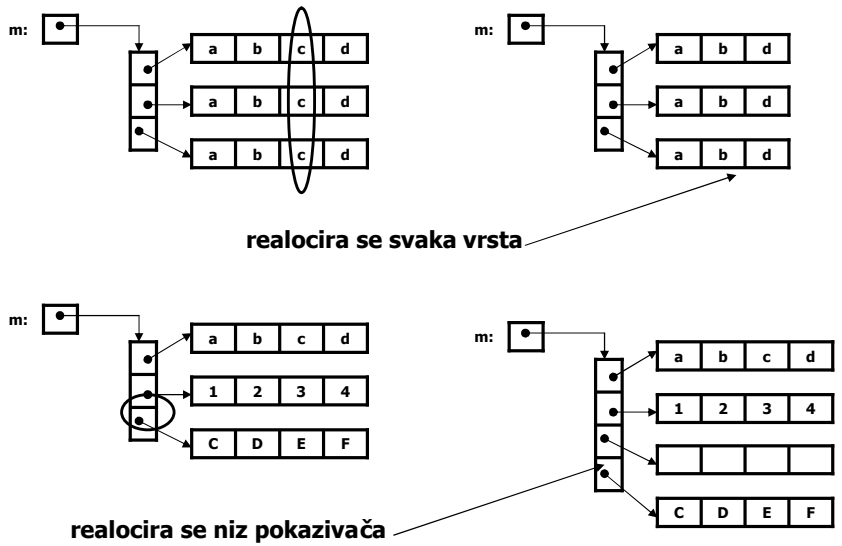
```
 /*ukoliko smo potrošili celokupan prostor, potrebno
   je da zauzmemo jos jedan bajt za završni znak*/
 novo_p = realloc( staro_p, (i+1) * sizeof(char) );
 if (novo_p == NULL) {
   printf("Neuspesno realociranje\n");
   i--;
 } else staro_p = novo_p;
 /*postavljamo završni znak*/
 staro_p[i] = '\0';
 printf("%s\n",staro_p);
 free(staro_p);
}
```

Zadatak 5 – realokacija matrice (S2)

- **Napisati program na programskom jeziku C koji učita broj vrsta i broj kolona i potom učita matricu tih dimenzija. Program potom učita broj kolone koju treba izbaciti i izbacuje tu kolonu iz matrice. Zatim čita redni broj vrste na koji umeće vrstu koju će učitati sa standardnog ulaza.**
- **Izbacivanje kolone iz matrice podrazumeva:**
  - pomeranje elemenata u svim vrstama matrice za jedno mesto, od odgovarajuće pozicije kolone koja se izbacuje
  - realociranje svih vrsta i smanjivanje veličine za jedan
- **Ubacivanje vrste u matricu podrazumeva:**
  - realociranje niza pokazivača na vrste i povećanje za jedan
  - pomeranje pokazivača u nizu od odgovarajućeg mesta za ubacivanje nove vrste
  - alociranje prostora za novu vrstu

Zadatak 5 - rešenje

- **Primer matrice 3x4 – izbacivanje kolone i ubacivanje vrste**



Zadatak 5

- **Zbog čitljivosti programa, u ovom rešenju su izostavljene provere da li je alokacija (realokacija) uspešno izvršena**
  - pri pisanju bilo kakvog programa ove provere NE SMEJU biti izostavljene
  - izostavljanjem tih provera ponašanje programa u slučaju nedostatka prostora nije predvidivo i može dovesti do katastrofalnih posledica

```
#include<stdlib.h>
#include<stdio.h>
void main(){
 unsigned int **m, **novo_m;
 int M, N, i, j, k;

 scanf("%d %d", &M, &N);
 /*potrebno je napisati i provere o uspesnosti alokacije*/
 m = malloc(M * sizeof(unsigned int*));
 for (j = 0; j < M; j++){
   m[j] = malloc( N * sizeof(unsigned int) );
   for (i = 0; i < N; i++) scanf("%u", &m[j][i] );
 }
```


Zadatak 5

```
 printf("Unesite broj kolone koju izbacujete: ");
 scanf("%d", &k);

if ( k >= 0 && k < N) {
   for (j = 0; j < M; j++){
    for (i = k+1; i < N; i++) m[j][i-1] = m[j][i];
    /*neophodna provera uspesnosti realokacije*/
    m[j] = realloc( m[j], (N-1) * sizeof(unsigned int) );
   }
   N--;
 }

 for (j = 0; j < M; j++){
   for (i = 0; i < N; i++) printf("%d ",m[j][i]);
   printf("\n");
 }
 printf("\n\n");
```

Zadatak 5

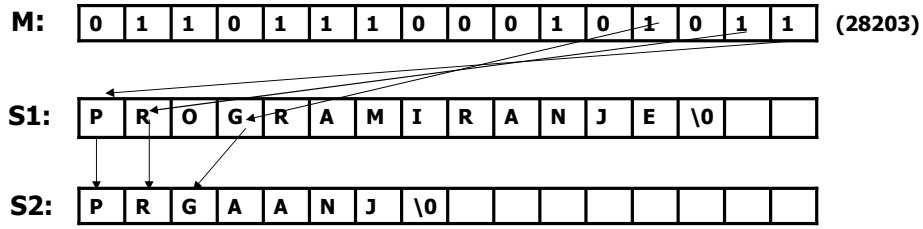
```
 printf("Unesite redni broj na koji umecete vrstu: ");
 scanf("%d", &k);

 if ( k >= 0 && k < M) {
   /*neophodna provera uspesnosti realokacije*/
   m = realloc( m, (M+1) * sizeof(unsigned int*) );
   M++;
   for( j = M-1; j > k; j--) m[j] = m[j-1];
   m[k] = malloc( N * sizeof(unsigned int) );
   for( i = 0; i < N; i++) scanf("%u", &m[k][i]);
 }

 for (j = 0; j < M; j++){
   for (i = 0; i < N; i++) printf("%u ",m[j][i]);
   free(m[j]);
   printf("\n");
 }
 free(m);
}
```


Zadatak 6 (C60)

- **Napisati program na programskom jeziku C koji učitava jedan znakovni niz (string) S1 i jedan ceo broj M, a zatim formira novi znakovni niz S2 samo od onih znakova na čijim su pozicijama odgovarajući bitovi broja M jednaki 1. Smatrati da se ceo broj predstavlja u 16-bitnoj lokaciji, a da znakovni niz može imati najviše 16 znakova. Na kraju, program ispisuje novi znakovni niz S2.**



Zadatak 6 - rešenje

```
#include <stdio.h>

void main() {
short int M, maska;
char S1[17], *s1, S2[17], *s2;

   printf("Unesi string S1 i broj M:\n");
   scanf("%s%d", S1, &M);
   s1 = S1;
   s2 = S2;
   maska = 1;
   while (maska && *s1)
   {
       if (M & maska) {*s2 = *s1; s2++; };
       s1++; maska<<=1;
   }
   *s2 = '\0';
   printf("Novi string S2: %s\n", S2);
}
```


Zadatak 6 - komentar

- **Promenjiva `maska` služi da ispitamo gde se nalaze jedinice u broju `M` koji zadaje korisnik**
- **Broj bitova broja `M` (16) odgovara broju znakova stringa `S1`**
  - na osnovu pozicije tih jedinica određuju se znakovi koji će biti u stringu **S2**
- **Svaki od stringova `S1` i `S2` zauzima ukupno 17 bajtova u memoriji – 16 za same znakove i 1 za završni nulti znak**
- **Uslov (`maska && *s1`) je neophodan da bi se dobio ispravan niz `s2`**

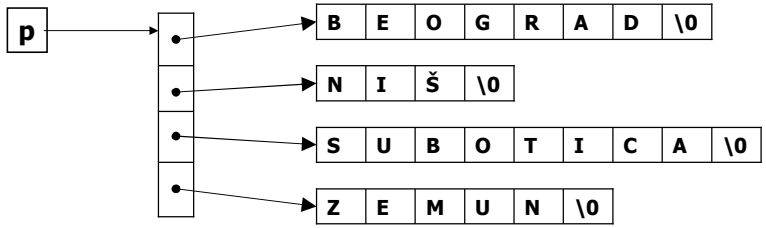
Zadatak 7 – nizovi stringova (C80b)

- **Sastaviti program na jeziku C za učitavanje imena gradova uz njihovo uređivanje po abecednom redosledu (leksikografskom poretku) i ispisivanje rezultata. U svakom redu se učitava po jedno ime sve dok se ne učita prazan red.**

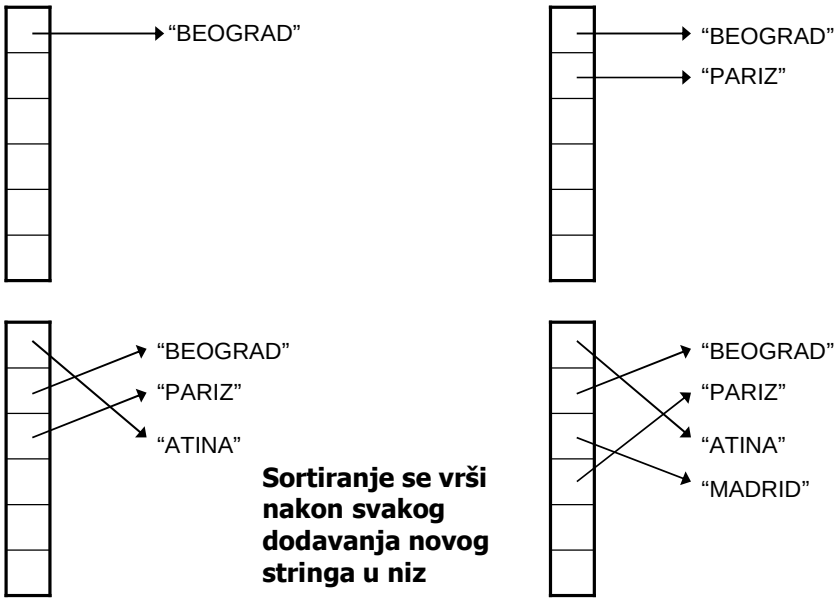
- **Rešenje zadatka prikazuje rad sa nizom pokazivača na znakovne nizove (stringove)**
- **Sortiranje se vrši nakon svakog dodavanja novog stringa u niz**
- **Umesto da se vrši prepisivanje stringova (strcpy), vrši se izmena vrednosti pokazivača**

Nizovi stringova

- **Po sličnom principu kao kod formiranja dinamičke matrice, može se formirati i dinamički niz stringova**
  - alocira se niz pokazivača koji će pokazivati na pojedinačne stringove (kao za vrste matrice)
  - za svaki string se alocira onoliko prostora koliko je potrebno za smeštanje svih znakova i završnog znaka
  - za razliku od matrica gde je svaka vrsta imala isti broj elemenata, sada se alocira samo onoliko prostora koliko je potrebno za smeštanje konkretnog stringa



Zadatak 7



Sortiranje se vrši nakon svakog dodavanja novog stringa u niz

Zadatak 7 - rešenje

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

#define MAX_GRAD 100
#define MAX_DUZ 30

void main() {
   char *adrese[MAX_GRAD], *adresa;
   int znak, br_grad = 0, duz, i;

   do {
        adresa = calloc(MAX_DUZ, sizeof(char));
        for (duz = 0; duz < MAX_DUZ - 1; duz++)
               if ((znak = getchar()) != '\n')
                      *(adresa + duz) = znak;
               else
                      break;
        *(adresa + duz) = '\0';
```

Zadatak 7

```
      if (!duz) {
            free(adresa); break;
      }
      adresa = realloc(adresa, duz + 1);
      for (i = br_grad - 1; i >= 0; i--)
            if (strcmp(adrese[i], adresa) > 0)
                  adrese[i + 1] = adrese[i];
            else
                  break;
      adrese[i+1] = adresa;
  } while (++br_grad < MAX_GRAD);

  for (i = 0; i < br_grad; i++)
      printf("%s\n", adrese[i]);
}
```


Funkcija strcmp()

- **Upoređuje dva stringa na koje ukazuju pokazivači `t` i `s` po leksikografskom poretku**

```
int strcmp(t,s);
```

- **Rezultat poređenja je sledeći:**
  - 0 – stringovi su identični
  - >0 – string **`t`** je leksikografski posle
  - <0 – string **`t`** je leksikografski pre
- **Ovaj zadatak je detaljno obrađen u knjizi profesora Krausa, te se savetuje studentima da detaljno prouče primere iz knjige**

Zadatak 8 - strlen(), strcpy(), strcat()

- **Napisati potprograme strlen(), strcpy() i strcat() koji imaju isto ponašanje kao odgovarajuće funkcije iz standardnog zaglavlja `<string.h>`**
- **Podsetimo se kako se ponašaju ove funkcije:**
- **strlen() : string length – vraća dužinu stringa**
- **strcpy() : string copy – kopira stringove**
- **strcat() : string concatenate – nadovezuje string na kraj drugog stringa**

Funkcije na jeziku C (1)

- **Funkcije (potprogrami) se koriste da bi se određeni kompleksan problem rastavio na manje celine koje vrše određenu obradu (funkciju) i daju konkretne rezultate te obrade**
- **Deklaracija (prototip, zaglavlje) funkcije:**
  `[povratni_tip] ime (tip arg1, tip arg2,…, tip argN);`
  - **`povratni_tip`** može biti bilo koji skalarni tip jezika C
  - ukoliko se ne navede, pretpostavlja se **`int`**
  - ukoliko funkcija ne vraća vrednost, koristi se ključna reč **`void`**
  - ime predstavlja identifikator funkcije
  - svaka funkcija može imati proizvoljan broj formalnih argumenata
  - formalni argumenti se navode tako što im se zada tip i ime, a ukoliko ih ima više odvajaju se zarezom
- **Argumenti se u funkciju na programskom jeziku C prenose po vrednosti!**
  - prilikom poziva funkcije, vrednosti argumenata se kopiraju na stek, i sve što se radi nad stvarnim argumentima se radi nad njihovim kopijama

Funkcije na jeziku C (2)

- **Telo funkcije se piše nakon navođenja zaglavlja funkcije unutar para vitičastih zagrada `{` i `}`**
- **U svakom trenutku se telo funkcije može napustiti pozivom naredbe `return`**
  - ukoliko funkcija vraća vrednost, nakon **`return`** se navodi vrednost koja se vraća

```
return 0;
```

  - unutar funkcije može postojati više **`return`** naredbi
- **Formalni argumenti funkcije i loklane promenljive su vidljive samo unutar funkcije**
- **Za komunikaciju sa okruženjem funkcije mogu koristiti argumente, povratnu vrednost i globalne promenljive**

Funkcije na jeziku C (3)

- **Globalne promenljive se definišu van tela funkcije**
  - važe od mesta definisanja do kraja datoteke
  - treba ih koristiti sa OPREZOM, jer funkcije treba da budu i jasne i dobro izolovane celine sa jasnom funkcionalnošću i upotrebom
- **Ukoliko želimo da promenimo neki od argumenata funkcije unutar njenog tela, onda takav parametar treba da prenesemo putem njegove adrese odnosno preko pokazivača**
- **Primer funkcije koja sabira dva broja:**

```
int saberi (int a, int b) {
      return a+b;
}
```

- **Poziv:**

```
saberi(2,5);
```


Zadatak 8 – strlen()

strlen() prva varijanta

```
int strlen(char *s){
int i;
 for(i=0; s[i] != '\0'; i++);
 return i;
}
```

strlen() druga varijanta

```
int strlen(char *s) {
int i;
  for(i=0; *s != '\0'; i++,s++);
  return i;
}
```

strlen() treća varijanta

```
int strlen(char *s) {
char *s2 = s;
   while(*s2 != '\0') s2++;
   return s2-s;
}
```

strlen() četvrta varijanta

```
int strlen(char *s){
   char *s2 = s;
   while(*s2++);
   return s2-s-1;
}
```

Oduzimanje pokazivača je dozvoljeno. Sabiranje i ostale aritmetičke operacije nisu!

Pitanje:
Šta se dešava ako se prosledi NULL pokazivač (s == NULL)?

Zadatak 8 – strcpy()

strcpy() prva varijanta

```
char *strcpy(char *dst, const char *src){
   int i = 0;
   while( src[i] != '\0' ) {
       dst[i] = src[i];
       i++;
   }
   dst[i] = '\0';
   return dst;
}
```

Vrednost 0, koja označava kraj stringa mora eksplicitno da se doda.

Zadatak 8 – strcpy()

strcpy() druga varijanta

```
char *strcpy(char *dst, const char *src) {
   char *d = dst;
   while( *src != '\0' )
        *d++ = *src++;
   *d = '\0';
   return dst;
}
```

strcpy() treća varijanta

```
char *strcpy(char *dst, const char *src)
{
char *d = dst;
   while( *d++ = *src++ );
    return dst;
}
```

Rezultat izraza
`*d++ = *src++`
je dodeljena vrednost. Kada je ta vrednost 0, ciklus se prekida.

Zadatak 8 – strcat()

- **strcat nadovezuje string src na kraj stringa dst**
  - prvi karakter stringa src se prepisuje preko simbola za kraj stringa dst
  - primetiti - strcat se delimično realizuje kao strcpy

strcat() prva varijanta

```
char *strcat(char *dst, const char *src) {
int i, j;
   for(i=0; dst[i] != '\0'; i++);
   for(j=0 ; src[j] != '\0'; i++, j++)
        dst[i] = src[j];
   dst[i] = '\0';
   return dst;
}
```


Zadatak 8 – strcat()

strcat() druga varijanta

```
char *strcat(char *dst, const char *src) {
   char *d = dst;
   while( *d != '\0')   d++;
   while( *src != '\0' )
        *d++ = *src++;
   *d = '\0';
   return dst;
}
```

strcat() treća varijanta

```
char *strcat(char *dst, const char *src) {
char *d = dst;
   while( *d != '\0') d++;
   while( *d++ = *src++ );
   return dst;
}
```


Zadatak 9

- **Šta ispisuje sledeći program?**

```
#include <stdio.h>
void fja(char *s[4])  {
  (*s)++;
}
void main ()  {
char *s[] = {"abc", "def", "ghi", "jkl"};
  fja(s);
  printf("%s",*s);
}
```

- **Odgovori:**
  - **A) bc**
  - **B) abc**
  - **C) def**

Zadatak 10

- **Šta ispisuje sledeći program?**

```
#include <stdio.h>
#include <stdlib.h>
void main() {
  int i, *a[5], **b;
  for (i=0; i<5; i++)  {
   a[i]=(int*) malloc(sizeof(int));
   *a[i]=i%2;
  }
  for (b=a+--i; i>0; i--) {**(b-i)=**b+i%3;}
  for (i=0; i<5; i++) printf("%d",*b[-i]), free(b[-i]);
}
```

- **Odgovori:**
  - **A) 10201**
  - **B) 01234**
  - **C) 01201**

Zadatak 11

- **Navesti tip identifikatora `x` u sledećim definicijama:**

  `char **x;`
  pokazivač na pokazivač na `char`

  `int (*x)[10];`
  pokazivač na niz od 10 `int`

  `int *x[10];`
  niz od 10 pokazivača na `int`

  `int *x();`
  funkcija koja vraća pokazivač na `int`

  `int (*x)();`
  pokazivač na funkciju koja vraća `int`

  `char (*(*x())[])();`
  funkcija koja vraća pokazivač na niz pokazivača na funkciju koja vraća `char`

  `char (*(*x[3])())[5];`
  niz od tri pokazivača na funkciju koja vraća pokazivač na niz od 5 `char`

- **U C-u nije dozvoljeno praviti niz funkcija, pa sledeća definicija nije valjana:**

```
char *((*x)[](char **));
```


Čitanje deklaracija na C-u

- **Deklaracije se čitaju počev od identifikatora ka spoljašnjosti**
  - poštuje se prioritet zadat zagradama
- **Prvo se čitaju modifikatori desno od identifikatora, i to redom s leva udesno**
- **Zatim se čitaju modifikatori levo od identifikatora, i to redom sdesna ulevo**
- **Postupak se ponavlja dok se ne obrade sve obuhvatajuće zagrade, ako postoje**
- **Na kraju se čita tip podatka koji stoji na početku deklaracije**

Zadatak 12 – čitanje deklaracija

- **Neka je tip identifikatora `x` deklarisan na sledeći način:**

  x *je pokazivač na niz pokazivača na*

  funkcije koje vraćaju pokazivač na double

- **Kako bi bila napisana ta deklaracija na programskom jeziku C?**

- **Odgovori:**
  - **A) double *(*(*(* x) ()) []) ();**
  - **B) double *(*(*x)[])();**
  - **C) double *(*(*x)())[];**