Veza izmedju pokazivaca i nizova

```
#include <stdio.h>
void print_array(int* pa, int n);
main() {
  int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  int num_of_elements = sizeof(a)/sizeof(int);
  int* pa;
  printf("Niz a : %p\n", a);
  printf("Adresa prvog elementa niza a (&a[0]) : %p\n", &a[0]);
  pa = a;
  printf("Pokazivac pa ukazuje na adresu : %p\n", pa);
  printf("a + 3 = %p\n", a + 3);    printf("&a[3] = %p\n", &a[3]);
  printf("pa[5] = %d\n", pa[5]);   printf("*(pa + 5) = %d\n", *(pa+5));
```

Veza izmedju pokazivaca i nizova

```
  printf("sizeof(a) = %d\n", sizeof(a));
  printf("sizeof(pa) = %d\n", sizeof(pa));
  print_array(a, num_of_elements);   print_array(pa, num_of_elements);
}

void print_array(int* pa, int n) {
  int i;
  for (i = 0; i<n; i++)  printf("%d ", pa[i]);
  putchar('\n');
}
```

NCP koji unosi niz proizvoljne dimenzije i nalazi najveci element.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int n, *a, i, max;
   printf("Unesi dimenziju niza : ");  scanf("%d", &n);
   a = (int*) malloc(n*sizeof(int));
   if (a == NULL)  {
      printf("Greska : Nema dovoljno memorije!\n");
      return 1;
   }
  for (i = 0; i<n; i++)   {
      printf("a[%d]=", i);    scanf("%d", &a[i]);
   }
   PRONALAZENJA MAKSIMUMA NIZA
   free(a);
   return 0;
}
```

NCP koji unosi niz proizvoljne dimenzije i nalazi najveci element – Ver.2

```
#include <stdio.h>
#include <stdlib.h>
int* CreateIntArray(int n) {
   return (int*) malloc(n*sizeof(int));
}
int main() {
   int n, *a, i, max;
   printf("Unesi dimenziju niza : ");  scanf("%d", &n);
   a = CreateIntArray(n);
   if (a == NULL)  {
      printf("Greska : Nema dovoljno memorije!\n");
      return 1;
   }
   /* Nadalje a koristimo kao obican niz */
  free(a);
   return 0;
}
```

Program demonstrira niz kome se velicina tokom rada povecava.

```
#include <stdio.h>
#include <stdlib.h>
#define KORAK 10
int main() {
  int* a = NULL;
  int duzina = 0, alocirano = 0;
  int n, i;
  do   {
    printf("Unesi ceo broj (-1 za kraj): ");    scanf("%d", &n);
    if (duzina == alocirano)    {
      alocirano = alocirano + KORAK;
      a = realloc(a, alocirano*sizeof(int));
    }
    a[duzina++] = n;
  } while (n != -1);
```

Program demonstrira niz kome se velicina tokom rada povecava.

```
  printf("Uneto je %d brojeva. Alocirano je ukupno %d bajtova\n",
    duzina, alocirano*sizeof(int));
  printf("Brojevi su : ");
  for (i = 0; i<duzina; i++)  printf("%d ", a[i]);
  free(a);
  return 0;
}
```

Program demonstrira funkciju calloc - funkcija inicijalizuje sadrzaj memorije na 0.

```
#include <stdio.h>
#include <stdlib.h>
#define BR_ELEM 10
main() {
  int *m, *c, i;
  m = malloc(BR_ELEM*sizeof(int));
  c = calloc(BR_ELEM, sizeof(int));
  for (i = 0; i<BR_ELEM; i++)
    printf("m[%d] = %d\n", i, m[i]);
  for (i = 0; i<BR_ELEM; i++)
    printf("c[%d] = %d\n", i, c[i]);
  free(m);
  free(c);
}
```

Vracanje niza iz funkcije.

```
#include <stdio.h>
int* KreirajNiz(int n) {
  return (int*)malloc(n*sizeof(int));
}
int* UnosNiza(int n) {
  int *x,i;
  x=KreirajNiz(n);
  for(i=0;i<n;i++) scanf("%d",&x[i]);
  return x;
}
main() {
  int i,n,*a;
  printf("Unesi dimenziju niza "); scanf("%d",&n);
  a=UnosNiza(n);
  for(i=0;i<n;i++) printf("%5d",a[i]);
  printf("\n");
  free(a);
}
```

NCP koji izracunava sumu kvadrata, kubova i dvostrukih vrednosti od 1 do n.

```
#include <stdio.h>
int kvadrat(int n) {
   return n*n;
}
int kub(int n){
   return n*n*n;
}
int parni_broj(int n){
   return 2*n;
}
int sumiraj(int (*f) (int), int n) {
   int i, suma=0;
   for (i=1; i<=n; i++)
        suma += (*f)(i);
    return suma;
}
```

NCP koji izracunava sumu kvadrata, kubova i dvostrukih vrednosti od 1 do n.

```
main()
{
   printf("Suma kvadrata brojeva od jedan do 3 je %d\n",
         sumiraj(&kvadrat,3));
   printf("Suma kubova brojeva od jedan do 3 je %d\n",
         sumiraj(&kub,3));
   printf("Suma prvih pet parnih brojeva je %d\n",
         sumiraj(&parni_broj,5));
}
```

NCP koji izracunava obim i površinu trogula i kvadrata u koordinatnoj ravni.

```
#include <stdio.h>
#include <math.h>
struct point {
   int x;
   int y;  };
float segment_length(struct point A, struct point B) {
   int dx = A.x - B.x;
   int dy = A.y - B.y;
   return sqrt(dx*dx + dy*dy);
}
float Heron(struct point A, struct point B, struct point C) {
   float a = segment_length(B, C);
   float b = segment_length(A, C);
   float c = segment_length(A, B);
   float s = (a+b+c)/2;
   return sqrt(s*(s-a)*(s-b)*(s-c));
}
```

NCP koji izracunava obim i površinu trogula i kvadrata u koordinatnoj ravni.

```
float circumference(struct point polygon[], int num) {
   int i;
   float o = 0.0;
   for (i = 0; i<num-1; i++)
      o += segment_length(polygon[i], polygon[i+1]);
   o += segment_length(polygon[num-1], polygon[0]);
   return o;
}

float area(struct point polygon[], int num) {
   float a = 0.0;
   int i;
   for (i = 1; i < num -1; i++)
      a += Heron(polygon[0], polygon[i], polygon[i+1]);
   return a;
}
```

NCP koji izracunava obim i površinu trogula i kvadrata u koordinatnoj ravni.

```
main() {
   struct point a, b = {1, 2}, triangle[3];
   struct point square[4] = {{0, 0}, {0, 1}, {1, 1}, {1, 0}};
   a.x = 0; a.y = 0;
   triangle[0].x = 0;   triangle[0].y = 0;
   triangle[1].x = 0;   triangle[1].y = 1;
   triangle[2].x = 1;   triangle[2].y = 0;

   printf("sizeof(struct point) = %d\n", sizeof(struct point));

   printf("x koordinata tacke a je %d\n", a.x);
   printf("y koordinata tacke a je %d\n", a.y);
   printf("x koordinata tacke b je %d\n", b.x);
   printf("y koordinata tacke b je %d\n", b.y);

   printf("Obim trougla je %f\n", circumference(triangle, 3));
   printf("Obim kvadrata je %f\n",  circumference(square, 4));
   printf("Pov. trougla: %f\n", Heron(triangle[0], triangle[1], triangle[2]));
   printf("Pov. kvadrata: %f\n", area(square, sizeof(square)/sizeof(struct point)));
}
```

Koriscenje `typedef`.

```
#include <stdio.h>
#include <math.h>
typedef int ceo_broj ;
typedef struct point POINT;
struct point {
   int x;
   int y; };

main() {
   ceo_broj x = 3;
   POINT a;
   printf("x = %d\n", x);
   a.x = 1; a.y = 2;
   printf("sizeof(struct point) = %d\n", sizeof(POINT));
   printf("x koordinata tacke a je %d\n", a.x);
   printf("y koordinata tacke a je %d\n", a.y);
}
```

Šta je rezultat rada sledećeg programa?

```
#include <stdio.h>
union primer {
   int broj;
   char slovo;
   float broj_r;
};
main(){
   union primer p;
   printf("sizeof(int)=%d \t",sizeof(int));
   printf("sizeof(char)=%d \t",sizeof(char));
   printf("sizeof(float)=%d \n",sizeof(float));
   printf("sizeof(union primer)=%d\n",sizeof(union primer));
   p.broj=5;
   printf("p.broj=%d \n",p.broj);
   p.slovo='D';
   printf("p.slovo=%c\t p.broj=%d\n",p.slovo,p.broj);
   p.broj_r=1.23;
   printf("p.broj_r=%f\tp.slovo=%c\t p.broj=%d\n",p.broj_r,p.slovo,p.broj);
}
```

Primer definisanja unije:

```
#include <stdio.h>
#include <string.h>
struct radnik {
    char prezime[20];
    char ime[20];
    char plata_ili_nadnica;
    union {
        float plata;
        struct {
            int sati_rada;
            float satnica;
        } nadnica;
    } zarada;
} osoba[20];
```

Primer definisanja unije:

```
main(){
  strcpy(osoba[0].prezime,"Peric");
  printf("%s\t",osoba[0].prezime);
  osoba[0].zarada.plata=54358.5;
  printf("%.2f\n",osoba[0].zarada.plata);
  strcpy(osoba[1].prezime,"Lazic");
  printf("%s\t",osoba[1].prezime);
  osoba[1].zarada.nadnica.sati_rada=54;
  osoba[1].zarada.nadnica.satnica=850.2;
  printf("%.2f\n",osoba[1].zarada.nadnica.sati_rada*
                                    osoba[1].zarada.nadnica.satnica);
}
```

Pokazivači na strukture.

```
#include <stdio.h>
typedef struct point {
   int x, y;
} POINT;

void get_point_wrong(POINT p) {
   printf("x = ");
   scanf("%d", &p.x);
   printf("y = ");
   scanf("%d", &p.y);
}

void get_point(POINT* p) {
   printf("x = ");
   scanf("%d", &p->x);
   printf("y = ");
   scanf("%d", &p->y);
}
```

Pokazivači na strukture.

```
main() {
   POINT a = {0, 0};

   printf("get_point_wrong\n");
   get_point_wrong(a);
   printf("a: x = %d, y = %d\n", a.x, a.y);

   printf("get_point\n");
   get_point(&a);
   printf("a: x = %d, y = %d\n", a.x, a.y);
}
```

NCP koji ispisuje broj argumenata komandne linije i parametre komandne linije.

```
#include <stdio.h>
main(int argc, char *argv[] ) {
   int i;
   printf("Broj argumenata je %d\n", argc);
   for(i=0;i<argc;i++)
       printf( "argv[%d] = %s\n", i , argv[i] );
}
```

NCP koji na izracunava i na standardni izlaz ispisuje sumu brojeva koji se zadaju u komandnoj liniji. Celi brojevi se zadaju posle imena programa i opcije -d. Realni brojevi se zadaju posle imena programa i opcije -f.

NCP koji u datoteku se upisuje prvih 10 prirodnih brojeva, a zatim se iz iste datoteke citaju brojevi dok se ne stigne do kraja i ispisuju se na standardni izlaz.

```
#include <stdio.h>
#include <stdlib.h>
main() {
   int i;
   FILE* f = fopen("podaci.txt", "w");
   if (f == NULL)   {
      printf("Greska prilikom otvaranja datoteke podaci.txt za pisanje\n");
      exit(1);
   }

   for (i = 0; i<10; i++)  fprintf(f, "%d\n", i);
   fclose(f);
```

NCP koji u datoteku se upisuje prvih 10 prirodnih brojeva, a zatim se iz iste datoteke citaju brojevi dok se ne stigne do kraja i ispisuju se na standardni izlaz.

```
  f = fopen("podaci.txt", "r");
  if (f == NULL)   {
    printf("Greska prilikom otvaranja datoteke podaci.txt za citanje\n");
    exit(1);   }
  while(1)   {
    int br;
    fscanf(f, "%d", &br);
    if (feof(f))  break;
    printf("Procitano : %d\n", br);   }
  fclose(f);
}
```

Funkciju **feof** ne bi trebalo pozivati pre pokusaja citanja. Sledeci kod moze dovesti do greske:

```
while (!feof(f))  scanf("%d",&br);
```

NCP koji u datoteku dodaje tekst.

```
#include <stdio.h>
main()  {
  FILE* datoteka;
  if ( (datoteka=fopen("dat.txt","a"))==NULL)   {
    fprintf(stderr,"Greska : nisam uspeo da otvorim dat.txt\n");
    return 1;
     }
  fprintf(datoteka,"Zdravo svima\n");
  fclose(datoteka);
}
```

NCP koji kopira datoteku cije se ime zadaje kao prvi argument komandne linije u datoteku cije se ime zadaje kao drugi argument komandne linije. Uz svaku liniju se zapisuje i njen broj.

```
#include <stdio.h>
#define MAX_LINE 256
int getline(char s[], int lim) {
   char* c = fgets(s, lim, stdin);
   return c==NULL ? 0 : strlen(s);
}

main(int argc, char* argv[]) {
   char line[MAX_LINE];
   FILE *in, *out;
   int line_num;
   if (argc != 3)    {
      fprintf(stderr,"Upotreba : %s ulazna_datoteka izlazna_datoteka\n",  argv[0]);
      return 1;
   }
```

NCP koji kopira datoteku cije se ime zadaje kao prvi argument komandne linije u datoteku cije se ime zadaje kao drugi argument komandne linije. Uz svaku liniju se zapisuje i njen broj.

```
  if ((in = fopen(argv[1],"r")) == NULL)   {
      fprintf(stderr, "Neuspesno otvaranje datoteke %s\n", argv[1]);
      return 1;
   }
   if ((out = fopen(argv[2],"w")) == NULL)   {
      fprintf(stderr, "Neuspesno otvaranje datoteke %s\n",argv[2]);
      return 1;
   }
   line_num = 1;
   while (fgets(line, MAX_LINE, in) != NULL)    {
      fprintf(out, "%-3d :\t", line_num++);
      fputs(line, out);
   }
   fclose(in);
   fclose(out);
}
```

Napisati rekurzivnu i iterativnu funkciju koja stepenuje realan broj na celobrojni izlozilac.

```
#include <stdio.h>
float power(float f, int k) {
  if (k == 0)  return 1;
  else  return f*power(f, k-1);
}

float power_iterative(float x, int k) {
  int i;
  float s = 1;
  for (i = 0; i<k; i++) s*=x;
  return s;
}

main() {
  printf("%f", power(2.5, 8));
}
```

```
#include <stdio.h>
float power(float f, int k) {
  return k==0 ? 1 : f*power(f, k-1);
}
```

Napisati rekurzivnu i iterativnu funkciju koja racuna faktorijel celog broja.

```
#include <stdio.h>
long factorial(int n) {
  if (n == 1)  return 1;
  else
    return n*factorial(n-1);
}

long factorial_iterative(int n) {
  long f = 1;
  int i;
  for (i = 1; i<=n; i++)
    f *= i;
  return f;
}

main() {
  printf("5! = %d\n", factorial(5));
}
```

```
#include <stdio.h>
long factorial(int n) {
  return n == 1 ? 1 : n*factorial(n-1);
}
```

Napisati rekurzivnu i iterativnu funkciju koja racuna Fibonacijeve brojeve: f(0) = 1, f(1) = 1, f(n) = f(n-1) + f(n-2)

```
#include <stdio.h>
#include <stdlib.h>
int Fib(int n) {
   printf("Racunam Fib(%d)\n",n);
   return (n == 0 || n == 1) ? 1 : Fib(n-1) + Fib(n-2);
}
int Fib_array(int n) {
  int *fib;
  int i, result;
  if ((fib = malloc(n*sizeof(int))) == NULL)  return -1;
  fib[0] = 1;
  fib[1] = 1;
  for (i = 2; i<=n; i++)  fib[i] = fib[i-2] + fib[i-1];
  result = fib[n];
  free(fib);
  return result;
}
```

Napisati rekurzivnu i iterativnu funkciju koja racuna Fibonacijeve brojeve: f(0) = 1, f(1) = 1, f(n) = f(n-1) + f(n-2)

```
int Fib_iterative(int n) {
   int pp = 1, p = 1;
   int i;
   for (i = 0; i <= n-2; i++)  {
      int tmp = pp;
      pp = p;
      p = p + tmp;
   }
   return p;
}

main() {
   printf("Fib(5) = %d\n", Fib(5));
   printf("Fib(5) = %d\n", Fib_array(5));
   printf("Fib(5) = %d\n", Fib_iterative(5));

}
```

Napisati rekurzivnu i iterativnu funkciju sabira n brojeva.

```
#include <stdio.h>
int array_sum(int a[], int n) {
   return n == 0 ? 0 : array_sum(a, n-1)+a[n-1];
}
```

```
#include <stdio.h>
int array_sum_iterative(int a[], int n) {
   int sum = 0;
   int i;
   for (i = 0; i<n;  i++) sum += a[i];
   return sum;
}
```

```
main() {
   int a[] = {1, 2, 3, 4, 5, 6, 7};
   printf("sum = %d\n",  array_sum(a, sizeof(a)/sizeof(int)));
   printf("sum = %d\n",  array_sum_iterative(a, sizeof(a)/sizeof(int)));
}
```

Napisati rekurzivnu funkciju koja racuna maksimum niza brojeva.

```
#include <stdio.h>
int max_rek1(int a[], int n, int i) {
   int rezultat;
   printf("Pozvana funkcija max za i = %d, n = %d\n", i, n);
   if (i == n-1) {
        printf("Jednoclan niz, max = %d\n", a[i]);
        rezultat = a[i];
   } else {
      int max_ostatka = max_rek1(a, n, i+1);
      printf("a[i] = %d, ", a[i]);
      printf("max_ostatka = %d\n", max_ostatka);
      rezultat =  a[i] > max_ostatka ? a[i] : max_ostatka;
   }
   printf("zavrsen poziv za i = %d, n = %d\n", i, n);
   printf("rezultat je: %d\n", rezultat);
   return rezultat;
}
```

Napisati rekurzivnu funkciju koja racuna maksimum niza brojeva.

```
int max_rek2(int a[], int n) {
  if (n == 0) return a[n];
  else {
    int max_pocetka = max_rek2(a, n-1);
    return a[n] > max_pocetka ? a[n] : max_pocetka;
  }
}

main() {
  int a[] = {2, 8, 3, 7, 9, 6, 4, 5, 1, 2};
  int n = sizeof(a)/sizeof(int);
  printf("%d\n", max_rek1(a, n, 0));
}
```

NCP koji ucitava rec sa standdardnog ulaza koja ima ne vise od 100 karaktera i ispisuje je na standardni izlaz. Rec je ma koja niska karaktera koja ne sadrzi beline (blanko, tab, prelaz u novi red,...). Ucitanu rec cuvati u stringu (nisci).

```
#include <stdio.h>
#include <ctype.h>
void get_word(char s[]) {
  int c, i = 0;
  while (!isspace(c=getchar())) s[i++] = c;
  s[i] = '\0';       /*OBAVEZAN kraj stringa u C-u */
}
main() {
  char s[100];
  get_word(s);
  printf("%s\n", s);
}
```

Napisati funkcije za rad sa niskama karaktera.

Sadrzaj datoteke mstring.h

```
int string_length(char s[]) ;
void string_copy(char dest[], char src[]) ;
void string_concatenate(char s[], char t[])
int string_compare(char s[], char t[])
int string_char(char s[], char c)
int string_last_char(char s[], char c)
int string_string(char str[], char sub[])
```

Napisati funkcije za rad sa niskama karaktera.

Sadrzaj datoteke mstring.c

```
#include "mstring.h"
int string_length(char s[]) {
   int i;
   for (i = 0; s[i]; i++) ;
   return i;
}
void string_copy(char dest[], char src[]) {
   int i;
    for (i = 0; (dest[i]=src[i]) != '\0'; i++) ;
}
void string_concatenate(char s[], char t[]) {
    int i, j;
   for (i = 0; s[i]; i++);
   for (j = 0; s[i] = t[j]; j++, i++) ;
}
int string_compare(char s[], char t[]) {
    int i;
   for (i = 0; s[i]==t[i]; i++)
         if (s[i] == '\0')  return 0;
   return s[i] - t[i];
}
```

Napisati funkcije za rad sa niskama karaktera.

Sadrzaj datoteke mstring.c

```
int string_char(char s[], char c) {
   int i;
   for (i = 0; s[i]; i++)     if (s[i] == c)  return i;
   return -1;
}
int string_last_char(char s[], char c) {
   int i;
   for (i = 0; s[i]; i++) ;
  for (i--; i>=0; i--)  if (s[i] == c)  return i;
  return -1;
}
int string_string(char str[], char sub[]) {
    int i, j;
    for (i = 0; str[i]; i++)
       for (j = 0; str[i+j] == sub[j]; j++)
         if (sub[j+1]=='\0') return i;
   return -1;
}
```

Napisati funkcije za rad sa niskama karaktera.

Sadrzaj datoteke stringovi.c

```
#include <stdio.h>
main() {
   char s[100];
   char t[] = "Zdravo";
   char u[] = " svima";
   string_copy(s, t);                                  strcpy(s,t);
   printf("%s\n", s);
   string_concatenate(s, u);                           strcat(s,u);
   printf("%s\n", s);
   printf("%d\n",string_char("racunari", 'n'));
   printf("%d\n",string_last_char("racunari", 'a'));
   printf("%d\n",string_string("racunari", "rac"));
   printf("%d\n",string_string("racunari", "ari"));
   printf("%d\n",string_string("racunari", "cun"));
   printf("%d\n",string_string("racunari", "cna"));
}
```

gcc -o prog stringovi.c mstring.c

Koriscenje eksternih promenljivih

Program1.c

```
…
int brojac;
extern double rezultat;
…
main() {
  …
  funk_1()
}
```

Program2.c

```
…
extern int brojac;
double rezultat;
…
funk_1(){
…
}
```

gcc -o prog Program1.c Program2.c

NCP koji čita tekst sa standardnog ulaza i svaku linju teksta ispisuje na standardni izlaz šifrirane po shemi koja utiče samo na slova.

SHEMA

A	B	...	Y	Z	a	b	...	y	z
c	d	...	a	b	D	E	...	B	C

Broj linija teksta nije unapred poznat, linije su limitirane dužine. Može se pretpostaviti da mašinski set znakova odgovara ASCII kôdu.

PRIMER

ULAZ	IZLAZ
1. Baba 23/05	1. dDED 23/05
32. Zaza 34/04	32. bDCD 34/04