24. FORMIRANJE BIBLIOTEKE FUNKCIJA

Biblioteke funkcija se sastoje od skupa nezavisnih ili, najčešće, povezanih funkcija koje obavljaju neki zajednički zadatak. Standardne sistemske biblioteke se nalaze u ”/lib” direktorijumima i prema zadatim inicijalnim podešavanjima, kompajler i njegov linker (povezivač) moraju biti usmereni na biblioteke koje treba uključiti uvek, ali i koje treba pretraživati osim standardnih C biblioteka.

Uz programski jezik C se isporučuju standardne biblioteke rutina, koje sadrže procedure, funkcije i promenljive. Rutine iz biblioteka se uključuju u program ključnom reči:

```
#include <biblioteka.h>
```

Biblioteka stdio.h sadži rutine i promenljive vezane za ulazno-izlazne poslove.

Ostale biblioteke su:

- matematičke funkcije
- korisničke funkcije, konverzije, smeštanje podataka
- biblioteka za rad sa stringovima
- biblioteka za testiranje i konverziju karaktera
- biblioteka za rad sa datumom i vremenom

OS specifične:

- za rad sa konzolom
- za pristup rutinama BIOS-a
- za rad sa funkcijama DOS-a
- za rad sa funkcijama Windows-a

Stdio.h funkcije:

- fopen/fclose – otvaranje/zatvaranje datoteke
- fprintf/fscanf – štampanje/čitanje iz datoteke
- printf/scanf – štampanje/čitanje iz standardnog ulaza/izlaza (tastatura/ekran)
- putc/getc – štampanje/učitavanje jednog znaka iz standardnog ulaza/izlaza
- puts/gets – štampanje/učitavanje stringa iz standardnog ulaza/izlaza
- fread/fwrite – učitavanje/snimanje bloka bajtova iz datoteke

Math.h funkcije:

- sin, cos, tan
- asin, acos, atan
- sinh, cosh, tanh
- exp(x) … ex
- log(x) … ln(x)
- log10(x) … log(x)
- pow(x, y) … xy
- sqrt – kvadratni koren
- ceil – zaokružuje na gore
- floor – zaokružuje na dole

Stdlib.h funkcije:

- atof – konverzija stringa u float
- atoi – konverzija stringa u int
- atol – konverzija stringa u long
- rand, random – pseudoslučajni broj
- malloc/free – alokacija/oslobađanje memorije
- exit – završetak rada programa
- system – izvršava program
- abs – apsolutna vrednost
- qsort – QUICK SORT
- div – celobrojno deljenje

String.h funkcije:

- strcpy – kopira polazni string u odredišni
- strncpy – kopira polaznih n slova u odredišni string
- strcat – spaja drugi string na kraj prvog
- strncat – spaja najviše n slova drugog stringa na kraj prvog
- strcmp – poredi stringove
- strncmp – poredi najviše n slova prvog stringa sa drugim
- strchr – vraća pokazivač na prvo mesto zadatog slova u stringu
- strstr – vraća pokazivač na prvu pojavu zadatog stringa u prvom stringu
- strlen – vraća dužinu stringa

Ctype.h funkcije:

- isalpha – da li je slovo
- isdigit – da li je broj
- isalnum – da li je alfanumerik (slovo ili broj)
- iscntrl – da li je kontrolni znak
- islower/isupper – da li je malo/veliko slovo
- isxdigit – da li je heksadecimalna cifra

Time.h funkcije:

- time – vraća trenutno kalendarsko vreme
- difftime – vraća razliku dva vremena u sekundama
- strftime – pravi formatiranu tekstualnu reprezentaciju vremena i datuma

Primer korišćenja stdio.h i math.h biblioteka:

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

int main()
{
  double x = 1.5;
  double y = 2.0;
  double z;

  z = sin(x);
  printf("sin(%.2f) = %.2f\n", x, z);  //izlaz: sin(1.50) = 0.99
```

```
  z = cos(x);
  printf("cos(%.2f) = %.2f\n", x, z);  //izlaz: cos(1.50) = 0.03

  z = tan(x);
  printf("tan(%.2f) = %.2f\n", x, z);  //izlaz: tan(1.50) = 32.33

  z = asin(x);
  printf("asin(%.2f) = %.2f\n", x, z); //izlaz: asin(1.50) = 1.04

  z = acos(x);
  printf("acos(%.2f) = %.2f\n", x, z); //izlaz: acos(1.50) = 0.96

  z = atan(x);
  printf("atan(%.2f) = %.2f\n", x, z); //izlaz: atan(1.50) = 0.98

  z = exp(y);
  printf("exp(%.2f) = %.2f\n", y, z);  //izlaz: exp(2.00) = 7.39

  z = log(y);
  printf("log(%.2f) = %.2f\n", y, z);  //izlaz: log(2.00) = 0.69

  z = sqrt(y);
  printf("sqrt(%.2f) = %.2f\n", y, z); //izlaz: sqrt(2.00) = 1.41

  return 0;
}
```

Kreiranje korisničkih biblioteka funkcija

Svaka biblioteka funkcija treba da obavlja specifični tip zadatka, pa treba da bude što je moguće više nezavisna u odnosu na druge fajlove. U datoteci sa ekstenzijom .h se nalaze implementacije i definicije funkcija, dok se u glavnom programu sa direktivom #include navodi naziv biblioteke, a zatim se vrši pozivanje funkcije(a) u main() funkciji programa.

Biblioteka funkcija "bibKalkulator.h":

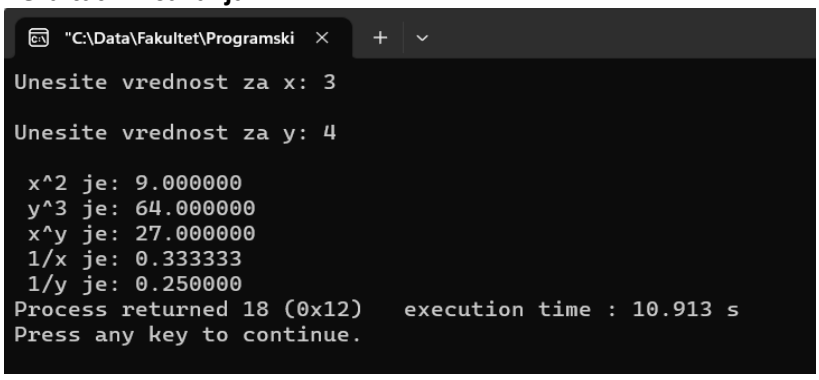
```
float kvadrat(float x)
{
      float rez;
      rez=x*x;
      return rez;
}
float kub(float y)
{
      float rez;
      rez=y*y*y;;
      return rez;
}
double stepen(float x, float y)
{
      double rez;
      int i;
      rez=1;
      for (i=1;i<=y;i++)
      {
        rez=rez*(double)x;
      }
```

```
	return rez;
}
float polinomial(float x)
{
	float rez;
	rez=1/x;
	return rez;
}
```

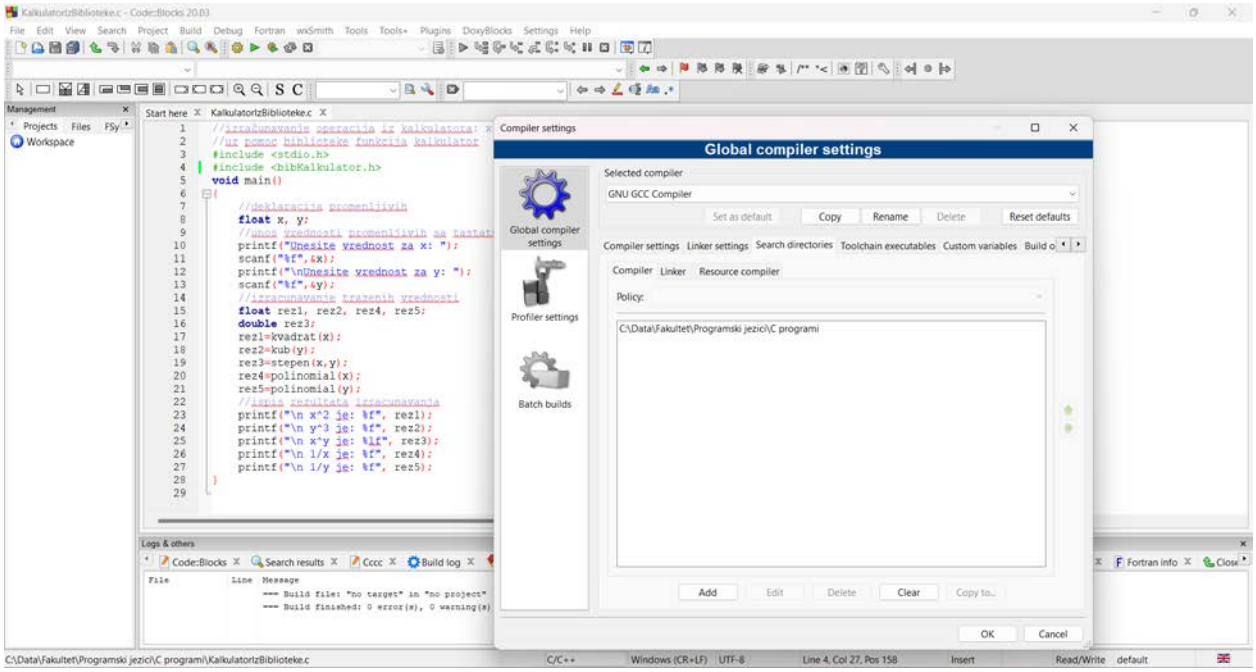
Glavni program:

```
//izračunavanje operacija iz kalkulatora: x^2, y^3, x^y, 1/x i 1/y
//uz pomoc biblioteke funkcija kalkulator koja se nalazi u istom folderu
#include <stdio.h>
#include "bibKalkulator.h"
void main()
{
    //deklaracija promenljivih
    float x, y;
    //unos vrednosti promenljivih sa tastature
    printf("Unesite vrednost za x: ");
    scanf("%f",&x);
    printf("\nUnesite vrednost za y: ");
    scanf("%f",&y);
    //izracunavanje trazenih vrednosti
    float rez1, rez2, rez4, rez5;
    double rez3;
    rez1=kvadrat(x);
    rez2=kub(y);
    rez3=stepen(x,y);
    rez4=polinomial(x);
    rez5=polinomial(y);
    //ispis rezultata izracunavanja
    printf("\n x^2 je: %f", rez1);
    printf("\n y^3 je: %f", rez2);
    printf("\n x^y je: %lf", rez3);
    printf("\n 1/x je: %f", rez4);
    printf("\n 1/y je: %f", rez5);
}
```

Rezultat izvršavanja:



Povezivanje CodeBlocks razvojnog okruženja sa bibliotekom funkcija: u meniju Settings – Compiler – Search directories, upisati ili izabrati (Browse) putanju foldera u kojoj se nalazi datoteka sa funkcijama:



U glavnom program se #include direktiva piše na sledeći način:

```
#include <stdio.h>
#include <bibKalkulator.h>
void main()
{
    //deklaracija promenljivih
    float x, y;
    //unos vrednosti promenljivih sa tastature
    printf("Unesite vrednost za x: ");
    scanf("%f",&x);
    printf("\nUnesite vrednost za y: ");
    scanf("%f",&y);
    //izracunavanje trazenih vrednosti
    float rez1, rez2, rez4, rez5;
    double rez3;
    rez1=kvadrat(x);
    rez2=kub(y);
    rez3=stepen(x,y);
    rez4=polinomial(x);
    rez5=polinomial(y);
    //ispis rezultata izracunavanja
    printf("\n x^2 je: %f", rez1);
    printf("\n y^3 je: %f", rez2);
    printf("\n x^y je: %lf", rez3);
    printf("\n 1/x je: %f", rez4);
    printf("\n 1/y je: %f", rez5);
}
```

25. VIŠEDIMENZIONALNI NIZOVI

Niz je izvedeni tip podatka koji predstavlja strukturu u kojoj su svi elementi istog tipa. Tip niza proizilazi iz tipa iz tipa njegovih elemenata. Indeks niza je promenljiva ili izraz koji se nalazi između zagrada [ ]. Početni element niza ima indeks 0 (nula). S obzirom da niz u sebi sadrži više elemenata, da bi se mogao koristiti niz se mora prvo **deklarisati**, a zatim se može (ne mora nužno) izvršiti **inicijalizacija niza**, koja podrazumeva da svi elementi nakon toga imaju neku početnu vrednost.

Definisanje i inicijalizacija dvodimenzionalnog niza - dvodimenzionalni niz koji se još naziva i matrica je niz čiji su elementi jednodimenzionalni nizovi i deklariše se na sledeći način:

```
tip naziv[veličina1][veličina2];
```

Definicija matrice: Dvodimenzionalni niz ili matrica sastoji se od vrsta, pri čemu je svaka vrrsta jedan vektor (jednodimenzionalni niz), tj matrica je niz vektora.

Primer definisanja i inicijalizacije dvodimenzionalnog niza:
Dvodimenzionalni niz se može predstaviti i kao tabela koja ima m vrsta i n kolona.

Matrica a[3][4], vrednosti se tabelarno mogu predstaviti na sledeći način:

	Kolona 0	Kolona 1	Kolona 2	Kolona 3
Vrsta 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Vrsta 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Vrsta 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Elementi dvodimenzionalnog niza identifikuju sa sa **a[i][j]**, gde su **i** i **j** pozitivni celi brojevi i predstavljaju indekse nizova. Dvodimenzionalni nizovi se mogu inicijalizovati na sličan način kao jednodimenzionalni, dodelom vrednosti svim elementima niza na sledeći način:

```
int a[3][4] =
{
    {0, 1, 2, 3} ,    /*  inicijalizacija vrste sa indeksom 0 */
    {4, 5, 6, 7} ,    /*  inicijalizacija vrste sa indeksom 1 */
    {8, 9, 10, 11}   /*  inicijalizacija vrste sa indeksom 2 */
};
```

Definisanje i inicijalizacija višedimenzionalnog niza - Višedimenzionalni nizovi su nizovi čiji su elementi takođe nizovi. Po analogiji sa matricom, trodimenzionalni niz je niz matrica, itd. Deklaracija višedimenzionalnog niza: Deklaracija višedimanzionalnog niza je slična sa deklaracijom vektora, s tom razlikom što višedimenzionalni niz ima više dimenzija. Deklariše se na sledeći način:

```
tip naziv[veličina1][veličina2]…[veličinaN];
```

Svaka zagrada predstavlja novu dimenziju niza, a podatak u zagradi predstavlja maksimalni broj elemenata koji poseduje data dimenzija. Elementi predstavljaju nizove manjih dimenzija.

Primer definisanja višedimenzionalnih nizova:

```
int matrica[3][4], trodniz[3][6][4];
```

Svi elementi višedimenzionalnih niza u prethodnom primeru su celi brojevi. Matrica je dvodimenzionalni niz i prva dimenzija ima maksimalno 3 vektora pri čemu se svaki vektor sastoji od maksimalno 4 elementa. Trodimenzionalni niz u prethodnom primeru ima tri dimenzije pri čemu je prva dimenzija sa maksimalno 3 matrice, a svaka matrica se sastoji od maksimalno 6 vektora, pri

čemu svaki vektor ima 4 elementa koji su celi brojevi. Višedimenzionalne nizove treba koristiti kada je potrebno opisati složeni podatak koji je opisan sa više podataka, pri čemu i indeksi koji opisuju višedimenzionalni niz predstavljaju podatak neophodan za opis elementa.

Pristup odgovarajućem podatku matrice može se vršiti pomoću indeksa ili pokazivača. Kod matrice imamo dva indeksa (prvi je indeks vrste, a drugi je indeks kolone), a kod trodimenzionalnog tri indeksa. Indeksi moraju biti prirodni brojevi ili promenljive koje su celobrojnog tipa. Kao i kod vektora u indeksu može postojati i celobrojni izraz.

Primer:
unsigned s, a[3][4]={{3, 5, 0, 1},{9, 8, 7, 1},{4, 2, 3, 5}};

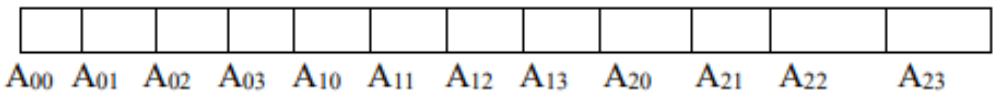
Pristup pomoću operatora indeksiranja:

Prvi način:
s=a[2][1]; //nakon naredbe dodele promenljiva s ima vrednost 2.

Drugi način:
int i=1, j=0;
s=a[i][j]; //nakon naredbe dodele vrednost promenljive s je 9.

Smeštanje višedimenzionalnih nizova u memoriju - Operativna ili RAM memorija je po prirodi linearno organizovana (predstavlja jednodimenzionalni niz), sa rastućim adresama, vektor se u memoriji smešta na prirodan način, gde su logički i fizički redosled elemenata u memoriji isti. Prilikom smeštanja matrice u memoriji mora se izvršiti njena linearizacija. Linearizacija se u programskom jeziku C vrši po vrstama. Od početne adrese matrice prvo se smešta prva vrsta, pa druga vrsta, i tako redom do poslednje vrste. Vrsta se tretira kao zapis koji ima onoliko elemenata koliko ima kolona.

Primer: int a[3][4];



Prolasci kroz višedimenzionalni niz - Kroz matricu se može izvršiti jedna od sledećih vrsta prolazaka:

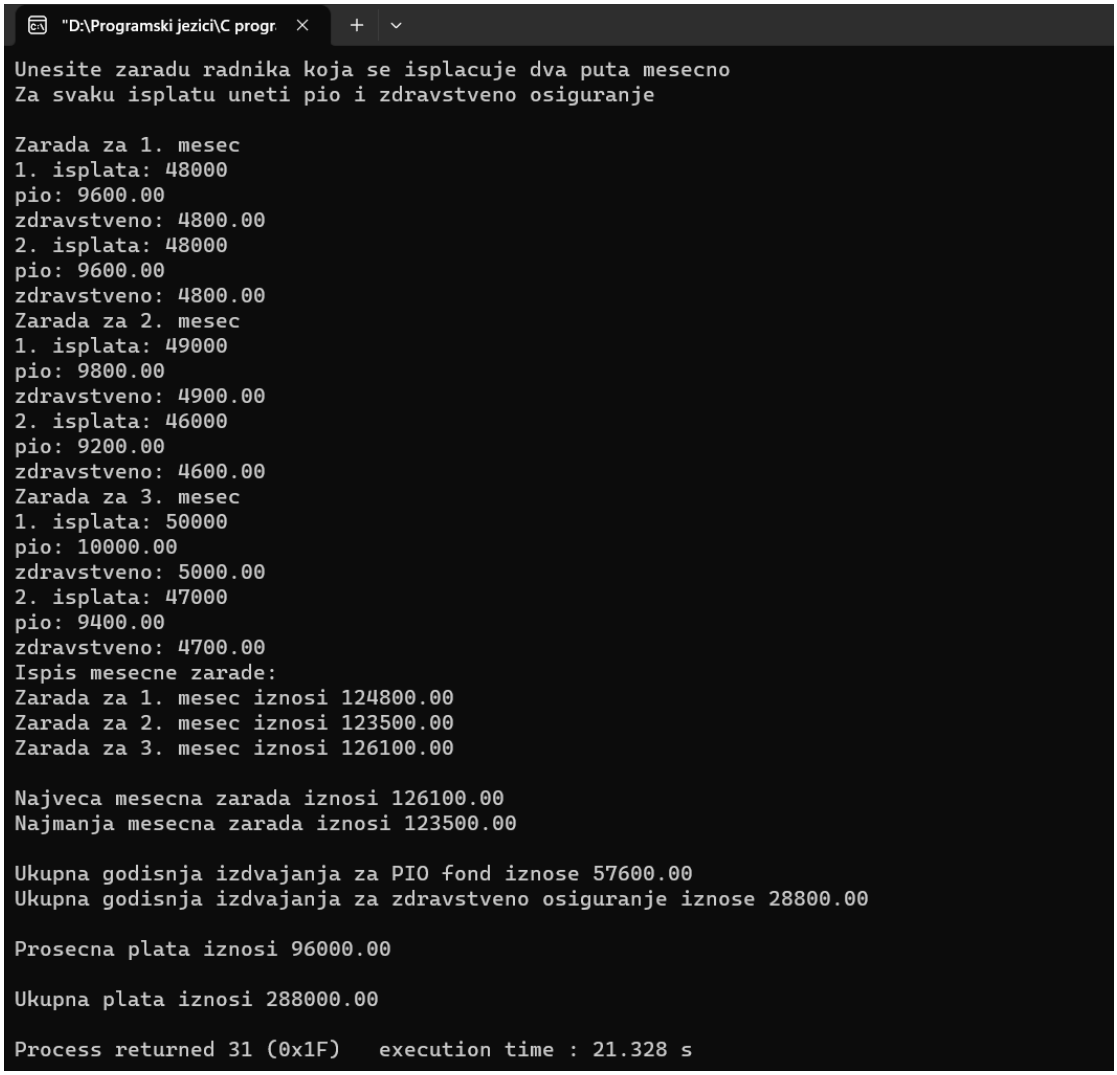
- Prolazak kroz niz vrsta po vrsta;
- Prolazak kroz niz kolona po kolona;
- Prolazak kroz niz preko dijagonala (paralelnih glavnoj dijagonali pri čemu glavnu dijagonalu čine elementi koji imaju istu vrednost indeksa vrste i kolone);
- Spiralni prolasci kroz niz.

Primer rada sa višedimenzionalnim nizom: Plata radnika se isplaćuje dva puta mesečno u određenom iznosu (u dinarima). Smestiti isplate u višedimenzionalni niz, vrednosti učitati sa tastature, izračunati i ispisati mesečne zarade, pio doprinos (24%) i zdravstveno osiguranje (10%), prosek plata za celu godinu, najmanju i najveću mesečnu zaradu.

```
#include <stdio.h>
#define mesec 3
#define isplata 2
void main()
{
int i, j, k;
```

```
float zarada[mesec][isplata][3];
float ukupnazarada[mesec];
float zaradamesec, ukupnaplata=0, pioukupno=0, zdravstvenoukupno=0;
//unos zarade dva puta mesecno + pio doprinos + zdravstveno osiguranje
printf("Unesite zaradu radnika koja se isplacuje dva puta mesecno\n");
printf("Za svaku isplatu uneti pio i zdravstveno osiguranje\n");
for(i = 0; i < mesec; i++)
{
   printf("\nZarada za %d. mesec", i + 1);
   for(j = 0; j < isplata; j++)
   {
        k=0;
        printf("\n%d. isplata: ", j + 1);
        scanf("%f", &zarada[i][j][k]);
        zarada[i][j][k+1]=zarada[i][j][k]*0.2;
            printf("pio: %.2f", zarada[i][j][k+1]);
        zarada[i][j][k+2]=zarada[i][j][k]*0.1;
        printf("\nzdravstveno: %.2f", zarada[i][j][k+2]);
        pioukupno=pioukupno+zarada[i][j][k+1];
        zdravstvenoukupno=zdravstvenoukupno+zarada[i][j][k+2];
   }
}
//ispis mesecne zarade
printf("\nIspis mesecne zarade:\n");
for(i = 0; i < mesec; i++)
{
      //ponistavanje sume mesecne zarade
      zaradamesec=0;
      for(j = 0; j < isplata; j++)
      {
        for(k = 0; k < 3; k++)
        {
            zaradamesec=zaradamesec+zarada[i][j][k];
            if (k==0) ukupnaplata=ukupnaplata+zarada[i][j][0];
            ukupnazarada[i]=zaradamesec;
        }
      }
      printf("Zarada za %d. mesec iznosi %.2f\n", i + 1, ukupnazarada[i]);
}
//izracunavanje maks i min, suma pio i zdravstveno
float maksz=ukupnazarada[0];
float minz=ukupnazarada[0];
float prosek;
prosek=ukupnaplata/mesec;
for(i = 0; i < mesec; i++)
{
      //izracunavanje min i max
            if (maksz<ukupnazarada[i]){maksz=ukupnazarada[i];}
            if (minz>=ukupnazarada[i]){minz=ukupnazarada[i];}
}
//ispis max i min zarade mesecno, pio i zdravstveno
printf("\nNajveca mesecna zarada iznosi %.2f\n", maksz);
printf("Najmanja mesecna zarada iznosi %.2f\n", minz);
printf("\nUkupna godisnja izdvajanja za PIO fond iznose %.2f\n",
pioukupno);
printf("Ukupna godisnja izdvajanja za zdravstveno osiguranje iznose
%.2f\n", zdravstvenoukupno);
printf("\nProsecna plata iznosi %.2f\n", prosek);
printf("\nUkupna plata iznosi %.2f\n", ukupnaplata);
}
```

Rezultat izvršavanja (za mesec=3, isplata=2):



26. STRINGOVI

Po definiciji string je niz karaktera, tj. znakova. Završava se znakom '\0'. Prazan string sadrži samo znak '\0'. Pošto je string niz, to znači da se kroz string prolazi kao i kroz jednodimenzionalni niz sa indeksom. Indeks prvog znaka stringa je 0. Za string ne moramo znati koliko ima znakova pošto je poslednji znak stringa '\0'. Uslov izlaska iz ciklusa, kada se prolazi kroz string, je sve dok je znak stringa različit od znaka '\0'.

Deklaracija stringa:

```
char imestringa[maks];
```

Prilikom **inicijalizacije stringa** prvo se navodi rezervisana reč **char** koja označava da elementi niza predstavljaju znakove. Zagrade **[]** označavaju da je u pitanju niz. Konstantan izraz **maks** mora imati uvek za 1 veću vrednost u odnosu na klasičan niz, jer string mora sadržati znak '\0'.

Primer1: JMBG (jedinstveni matični broj građana) koji uvek ima 13 brojeva:

```
char jmbg[14];
```

Primer2: `char rec[]=”Programiranje”;`

P	r	o	g	r	a	m	i	r	a	n	j	e	\0				

String se može inicijalizovati prilikom deklaracije stringa ili u toku izvršavanja programa. Inicijalizacijom stringa se vrši tako što se prvom elementu stringa dodeli znak '\0 ' koji označava kraj stringa.

Inicijalizacija stringa prilikom deklaracije:

```
char imestringa[0]= ””;
```

Inicijalizacija stringa u toku izvršavanja programa:

```
imestringa[0]='\0';
```

Unos stringa se najčešće vrši bez indeksa, kao što je to slučaj kod niza, a uz pomoć funkcija scanf ili gets.

```
scanf("%s”,imestringa);
```

ili:

```
gets(imestringa);
```

Format %s označava konverziju u string. Funkcija gets() omogućava unos stringa sve dok se ne pritisne ENTER i time dodaje znak '\0'. Ako funkcija naiđe na grešku prilikom unosa vraća simboličku konstantu NULL.

Ispis stringa se najčešće vrši bez korišćenja indeksa (za razliku od niza) a uz pomoć funkcija printf ili puts.

```
printf(”%s”,imestringa);
```

ili

```
puts(imestringa);
```

Funkcija puts izvršava ispis znakova sve dok se ne dođe do znaka '\0' te prelazi u novi red.

Sve osobine koje važe za niz važe i za string. Kao i kod nizova string se ne može preneti kao argument funkcije, već se po referenci prosleđuje adresa početka stringa. Za rad sa stringovima u funkcijama, osim adrese početka stringa, ništa više nije potrebno, jer se string završava sa znakom '\0'.

Funkcije za rad sa stringovima i znacima

Zaglavlje <ctype.h> deklariše funkcije za proveravanje znakova, tj. određuje u koju kategoriju spada neki znak (malo slovo, veliko slovo, broj, itd).

1. Funkcija ***insalnum*** vraća vrednost različitu od nule ako je navedeni znak slovo ili cifra:
```
int insalnum(int character);
```

2. Funkcija ***isalpha*** vraća vrednost različitu od nule ako je navedeni znak slovo:
```
int isalpha(int character);
```

3. Funkcija *isdigit* vraća vrednost različitu od nule ako je navedeni znak cifra:
```
int isdigit(int character);
```

4. Funkcija ***isgraph*** vraća vrednost različitu od nule ako se navedeni znak može prikazati na ekranu (opseg kodova od 33 do 126) izuzimajući znak za razmak:
```
int isgraph(int character);
```

5. Funkcija ***islower*** vraća vrednost različitu od nule ako je navedeni znak malo slovo:
```
int islower(int caracter);
```

6. Funkcija ***isupper*** vraća vrednost različitu od nule ako je navedeni znak veliko slovo:
```
int isupper(int character);
```

7. Funkcija ***isspace*** vraća vrednost različitu od nule ako je navedeni znak nevidljiv na ekranu (tabulator, vertikalni tabulator,…):
```
int isspace(int character);
```

8. Funkcija ***isprint*** vraća vrednost različitu od nule ako se navedeni znak može prikazati na ekranu:
```
int isprint(int character);
```

9. Funkcija ***toupper*** konvertuje malo slovo u veliko. Ako je navedeno slovo veliko funkcija ga ostavlja nepromenjenim:
```
int toupper(int character);
```

10. Funkcija ***tolower*** konvertuje veliko slovo u malo. Ako je navedeno slovo malo, funkcija ga ostavlja nepromenjenim:
```
int tolower(int character);
```

pri čemu je: `character` – je znak koji se proverava.

Funkcije za rad sa stringovima - U zaglavlju <string.h> definisane su sledeće funkcije za rad sa stringovima:

1. Funkcija ***strcpy*** kopira string ***t*** u string ***s*** znak po znak. Prethodni sadržaj stringa ***s*** se gubi. Funkcija vraća pokazivač na string ***s***:
```
char *strcpy(char *s, char *t);
```

2. Funkcija ***strncpy*** kopira prvih ***n*** znakova iz stringa ***t*** u string ***s***. Funkcija vraća pokazivač na string ***s***:
```
char *strncpy(char *s, char *t, int n);
```

3. Funkcija ***strcat*** dodaje string ***t*** na kraj stringa ***s*** i vraća pokazivač na string ***s***:
```
char *strcat(char *s, char *t);
```

4. Funkcija ***strncat*** dopisuje najviše ***n*** znakova iz stringa ***t*** na kraj stringa ***s***:
```
char *strncat(char *s, char *t, int n);
```

5. Funkcija ***strcmp*** poredi dva stringa znak po znak. Ako je znak u prvom stringu ***s*** veći od odgovarajućeg znaka u drugom stringu ***t***, funkcija vraća negativnu vrednost. Ako su stringovi indentični funkcija vraća 0.

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

6. Funkcija ***strncmp*** poredi prvih ***n*** bajtova dva niza znakova. Funkcija je identična funkcija ***strcmp***, stim što poredi samo prvih ***n*** karaktera.

```
int *strncmp(char *s, char *t, int n);
```

7. Funkcija ***strchr*** traži prvo pojavljivanje navedenog ASCII znaka u stringu ***s***, te ako uspešno obavi zadatak funkcija vraća pokazivač na prvo pojavljivanje navedenog znaka ili vrednost **NULL** u suprotnom.

```
char *strchr(char *s, int character);
```

8. Funkcija ***strrchr*** vraća pokazivač na prvu pojavu određenog znaka s desne strane stringa ***s***. Ako funkcija uspešno obavi zadatak biće vraćen pokazivač na prvu poziciju unutar stringa, ili vrednost **NULL** u suprotnom.

```
char *strrchr(char *s, int character);
```

9. Funkcija ***strstr*** vraća pokazivač na prvu pojavu podstringa ***t*** unutar stringa ***s***. Ako uspešno obavi zadatak funkcija vraća pokazivač na prvu pojavu podstringa, ili vrednost **NULL** u suprotnom.

```
char *strstr(char *s, char *t);
```

10. Funkcija ***strlen*** vraća dužinu stringa ***s*** u bajtovima, tj broj znakova stringa bez znaka '\0'.

```
int strlen(char *s);
```

11. Funkcija ***strtok*** vraća pokazivač na lokaciju tokena u stringu. U stringu **s** se traže tokeni. Ako je string **NULL**, funkcija strtok koristi završnu lokaciju prethodnog stringa, a **token** predstavlja string koji sadrži tokene koje treba naći. Ako uspešno obavi zadatak, funkcija vraća pokazivač na poziciju tokena. Ako ni jedan token nije pronađen funkcija strtok vraća vrednost **NULL**.

```
char *strtok(char *s, char *tokeni);
```

Funkcije za numeričke konverzije stringova

1. Funkcija atoi konvertuje string **s** u celobrojnu vrednost. Funkcija vrši konvertovanje sve dok se ne naiđe na nevažeći karakter unutar stringa ili znaka '\0'. Funkcija ne pruža informaciju da li postoje nevažeći znakovi unutar stringa.

```
int atoi(char *s);
```

2. Funkcija atof konvertuje string s u decimalnu vrednost. Funkcija vrši konvertovanje sve dok se ne naiđe na nevažeći karakter unutar stringa ili znaka ‘\0’. Funkcija ne pruža informaciju da li postoje nevažeći znakovi unutar stringa.

```
double atof(char *s);
```

Primer programa koji vrši spajanje stringova unetih sa tastature:

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

void main()
{
    char stringA[30],stringB[30];
    int duzA, duzB;
    printf("Spajanje stringova \n");
    printf("Unesite prvi string: ");
    gets(stringA);
    printf("Unesite drugi string: ");
    gets(stringB);
    duzA=strlen(stringA);
    duzB=strlen(stringB);
    char *stringF=strcat(stringA,stringB);
    printf("StringA je duzine %d.\t StringB je duzine %d.",duzA,duzB);
    printf("\nSpojeni string: %s",stringF);
    printf("\n");
    puts(stringF);
    puts(stringB);
}
```

Rezultat izvršavanja: