6 ФУНКЦИИ

Пример за void функции и функции без параметри.

```
#include <stdio.h>

/* Deklaracija na funkcii */
void PrintMax(int broj);
void PrintPozdrav();

int main()
{

    PrintPozdrav();    /* Pecati Pozdrav */

    PrintMax(k);    /* Ja pecati maximalnata vrednost */

    return 0;
}

/* Definicija na funkciite */

void PrintMax(int broj)
{
    printf("Maksimalniot broj e %d\n",broj);
}

void PrintPozdrav()
{
    printf("Dobar Den. Kako se cuvstvuvate denes?\n");
}
```

Задачи:

1. Да се напише програма која ќе ги отпечати сите четирицифрени природни броеви кои се деливи со збирот на двата броја составен од првите две цифри и од последните две цифри на четирицифрениот број, и на крајот ќе отпечати колку вакви броеви се пронајдени. На пример: 3417 е делив со 34+17, 5265, 6578, ...

```
#include <stdio.h>
int zb2cif(int n);
```

```
int main()
{
       int br=0,i;
       for (i=1000; i<=9999; i++)
       {
          if (i%zb2cif(i)==0)
              {
                 printf("Brojot %d go zadovoluva uslovot\n",i);
                 br++;
              }
       }
      printf("Pronajdeni se %d broevi koi go zadovoluvaat uslovot\n",br);
      return 0;
}
int zb2cif(int n)
{
       int zbir;
       zbir=(n%100)+(n/100);
       return zbir;
}
```

2. Да се напише програма која за даден природен број ја пресметува разликата меѓу најблискиот поголем од него прост број и тој број.

```
#include <stdio.h>
int prost(int n);
int prostgore(int n);
int main()
{
       int broj,razlika;
       printf("Vnesi broj\n");
       scanf("%d",&broj);
       razlika=prostgore(broj)-broj;
       printf("Razlikata medu prostiot broj %d i %d e
%d\n",prostgore(broj),broj,razlika);
       return 0;
}

int prost(int n)
{
       int k;
       k=2;
       while (k*k<=n)
              {
                     if (n%k==0) return 0;
                     k++;
              }
       return 1;
}
```

```
int prostgore(int n)
{
       do
              n++;
       while (!(prost(n)));
       return n;
}
```

3. Да се напише програма што ќе ги отпечати сите прости броеви помали од 10000 чиј што збир на цифри е исто така прост број. На крајот да се отпечати колку вакви броеви биле пронајдени. На пример: 23, 179, 9613, ...

```
#include <stdio.h>
int eprost(int n);
int zbircif(int n);
int main ()
{
     int  br=0,i;
       for (i=2; i<=9999; i++)
       {
          if (eprost(i) && eprost(zbircif(i)))
              {
                 printf("Brojot %d go zadovoluva
uslovot\n",i);
                 br++;
              }
       }
      printf("Pronajdeni se %d broevi koi go zadovoluvaat
uslovot\n",br);
      return 0;
}
int eprost(int n)
{
     int i, prost;
     if (n<4) prost=1;
     else
          if (n%2)==0) prost =0;
          else
          {
               i=3; prost=1;
               while ((i*i<=n) && prost)
               {
                  if (n%i==0) prost=0;
                  i+=2;
               }
          }
     return prost;
```

```
}
int zbircif(int n)
{
     int zbir=0;
     while (n>0)
     {
          zbir+=(n%10);
          n/=10;
     }
     return zbir;
}
```

4. Да се напише програма што ќе ги отпечати сите парови прости броеви што се разликуваат меѓу себе за 2. На крај да се отпечати и нивниот број.

```
#include <stdio.h>
int eprost(int n);

int main ()
{
     int  br=0,i;
       for (i=1; i<=(1000-2); i++)
       {
          if (eprost(i) && eprost(i+2))
              {
                 printf("Prostire broevi %d I %d se
razlikuvaat za 2\n",I, (i+2));
                  br++;
              }
       }
      printf("Pronajdeni se vkupno %d parovi prosti broevi
koi go zadovoluvaat uslovot\n",br);
      return 0;
}
int eprost(int n)
{
     int i;
     if n<4 return 1;
     else
          if (n%2)==0) return 0;
          else
          {
               i=3;
               while (i*i<=n)
               {
                  if (n%i==0)
                          return 0;
                  i+=2;
```

```
                }
            }
      return 1;
}
```

5. Да се напише функција што прима два параметра *x* и *n* и враќа:

$$f(x)=\begin{cases} x+\dfrac{x^n}{n}-\dfrac{x^{n+2}}{n+2}, x\geq 0 \\ -\dfrac{x^{n-1}}{n-1}+\dfrac{x^{n+1}}{n+1}, x<0 \end{cases}$$

Потоа да се состави програма што ќе ја табелира оваа функција за прочитано *n* во интервал $x\in[-4, 4]$, со чекор 0.1.

```
#include <stdio.h>
//za vtorata verzija na stepen() so pow() treba
//#include <math.h>

double f(float i,int j);
float stepen(float i, int j);

int main ()
{    int n;
     float x;
     printf("Vnesi broj:\n");
     scanf(("%d", &n);
     if ((n>=-2) && (n<=1))
          printf("Neodredeno.\n");
     else {    x=-4.0;
               while (x<=4)
               {    printf("x=%3.1f,f(x)=%10.4f", x, f(x,n));
                    x+=0.1;   }
          }
     return 0;
}
double f(float i,int j)
{
     double vrednost;
     if (i>0)
          vrednost=i+stepen(i,j)-stepen(i,j+2);
     else
          vrednost=-stepen(i,j-1)+stepen(i,j+1);
     return vrednost;
}

float stepen(float i,int j)
{
```

```
     int k;
     double vrednost;
     if (i==0)
          vrednost=0.0;
     else
          {              // so koristewe na ciklus
               vrednost=1.0;
               for (k=1;k<=j;++k)
               vrednost*=i;
          }
     return vrednost;

}

//Vtora verzija za stepen() so koristenje na matem. funkcija
/* double pow(double x, double y) – ako x e negativno, y mora
da ima integer vrednost */

float stepen(float i,int j)
{
     return (pow(i,float(j));
}
```

Рекурзивни функции

1. Да се напише програма која за дадено N ќе го испише соодветниот Фибоначиев број. Фибоначиевите броеви се дефинирани на следниов начин:

$$a_1 = 1$$
$$a_2 = 1$$
$$a_n = a_{n-1} + a_{n-2}$$

Пример: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

```
#include <stdio.h>
int fib(int n);
int main ()
{
     int  broj;
     printf ("Vnesi broj: \n");
     scanf ("%d", &broj);
     printf ("Fibonacieviot broj e: %d\n", fib(broj));
     return 0;
}
int fib(int n)
{
     int rezultat;
     switch (n)
     {    case 1:
          case 2:
                    rezultat=1;
                    break;
          default:
                    rezultat=(fib(n-1)+fib(n-2));
                    break;
     }
     return rezultat;
}
```

2. Да се напише програма која соджи функција за пресметување на факториел од даден број.

```
#include <stdio.h>
int factorial(int n);
int main ()
{
     int  broj;
     printf ("Vnesi broj: \n");
     scanf ("%d", &broj);
     printf ("%d!=%d\n",broj, factorial(broj));
     return 0;
}
```

```
int factorial(int n)
{
     if (n==0)
          return 1;
     else
          return factorial(n-1)*n;
}
```

3. Да се напише рекурзивна функција која ќе пресметува најголем заеднички делител на два дадени броја.

```
#include <stdio.h>

int nzd(int i,int j);

int main ()
{
     int n,m;
     float x0=1.0, x1=2.0;
     printf("Vnesi dva broja:\n");
     scanf(("%d %d", &n, &m);
     printf("Najgolemiot zaednicki delitel na broevite %d i
     %d e %d.\n",n,m,nzd(n,m));
}

int nzd(int i, int j)
{
     int vrednost;
     if (i!=j)
          if (i<j)
               vrednost=nzd(j-i,i);
          else
               vrednost=nzd(i-j,j);
     return vrednost;
}
```

4. Да се напише програма што ќе ја испишува вредноста на произволен член на низата дефинирана со:

$$x_1 = 1$$
$$x_2 = 2$$
$$x_n = \frac{n-1}{n} x_{n-1} + \frac{1}{n} x_{n-2}$$

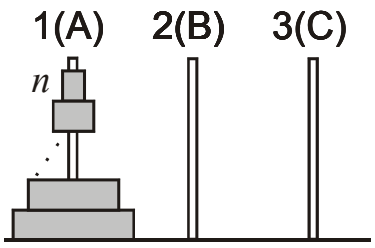
```
#include <stdio.h>

float xnn(float x0,float x1,int k);
```

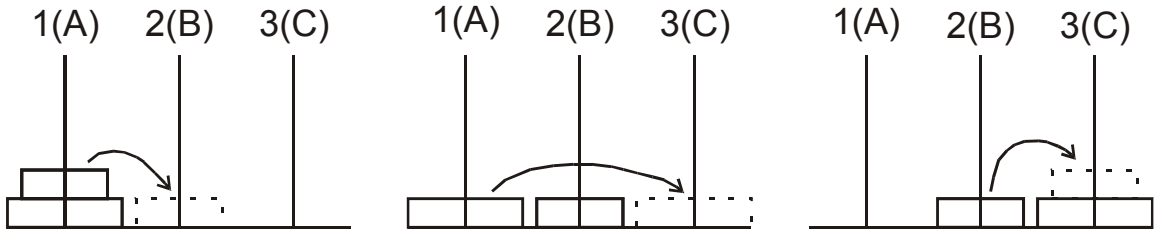
```
int main ()
{
     int n;
     float x0=1.0, x1=2.0;
     printf("Vnesi broj:\n");
     scanf(("%d", &n);
     printf("N-tiot clen na nizata e: %f.\n",xnn(x0,x1,n));
     return 0;
}
float xnn(float x0,float x1,int k)
{
     float clen;
     if (k==0)
          clen=x0;
     else
          if (k==1) clen=x1;
          else
               clen=xnn(x0,x1,k-1)*(k-1)/k+xnn(x-,x1,k-2)*1/n;
     return clen;
}
```

5. Да се напише програма која ќе го реши проблемот на Ханојските кули.

Легендата вели: Во големиот храм Brahma во Benares, под сводот кој го покрива центарот на светот постои месингана чинија во која има три вертикално поставени дијамантски прачки. На едната од нив има 64 диска од чисто злато со различен пречник секој со отвор во средината поставени на дијамантска прачка во вид на кула, со најголемиот диск на дното и најмалиот на врвот. Свештениците ги пренесуваат дисковите еден по еден помеѓу дијамантските прачки, според безусловниот закон на Brahma: Никогаш не смее да се постави поголем врз помал диск. (Секогаш се зема дискот од врвот и се преместува на некоја од останатите две прачки.) На почетокот на светот сите 64 диска ја формираат кулата на Брама на една од прачките. Кога и последниот диск ќе биде пренесен, формирајќи ја повторно кулата на Брама но сега на друга прачка, ќе дојде крајот на светот и сé ќе се претвори во прав. Модерната верзија на проблемот на Ханојските кули ја поставил Edouard Lucas, француски математичар околу 1883 година.



За 2 диска:



$$(1\text{-}2)\ A \rightarrow C \Rightarrow (1)\ A \rightarrow B,\ (2)\ A \rightarrow C,\ (1)\ B \rightarrow C$$

C A

$$(1\text{-}N)\ A \rightarrow C \Rightarrow (2\text{-}N)\ A \rightarrow B,\ (1)\ A \rightarrow C,\ (2\text{-}N)\ B \rightarrow C$$

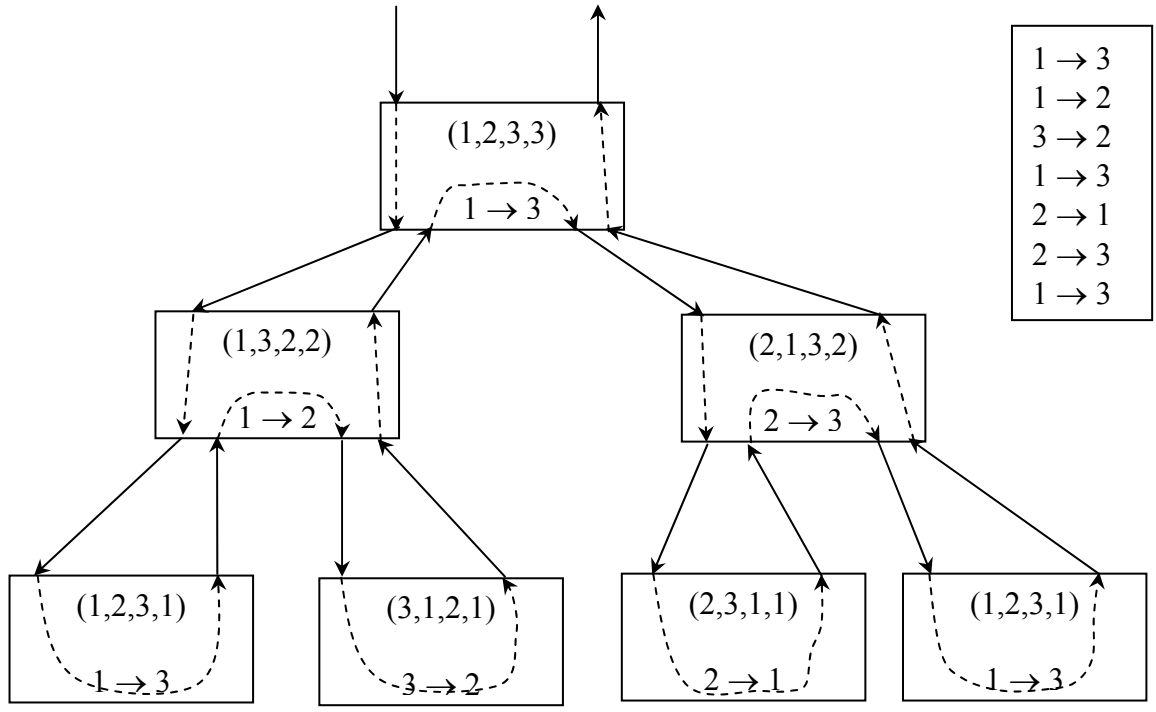
B A

$$(3\text{-}N)\ A \rightarrow C,\ (2)\ A \rightarrow B,\ (3\text{-}N)\ C \rightarrow B$$

```
#include <stdio.h>
void hanoi (char from, char to, char other, int n);

int main()
{
     int br_diskovi;
     printf("Vnesi go  brojot na diskovi\n", br_diskovi);
     scanf("%d",&br_diskovi);
     hanoi('A','B','C', br_diskovi)

}
void hanoi (char from, char to, char other, int n)
{
     if (n>0)
          {
               hanoi(from,other,to,n-1);
               printf("%c → %c\n", from, other);
               hanoi(to,from,other,n-1);
          }
     else
               printf("%c → %c\n", from, other);
}
```



——→ повик на процедура или излез од процедура

- - - - - - - - - - → тек на извршување во рамките на еден повик

(2,3,1,1) аргументи со кои е повикана процедурата (call stack) - магацин на повици

1 → 3 печатење во рамките на еден повик