Функции

1. Што се функции?

Од задача 2 кај switch се гледа делумно потребата за дефинирање на делови од програмата како посебни целини наречени функции, кои можат да се повикаат повеќе пати во текот на програмата со различни параметри.
Така, на пример, можевме делот со switch да го издвоиме во посебна функција која ќе ни ја печати текстуално дадената цифра и потоа два пати да ја повикаме функцијата – еднаш за цифрата mala еднаш за golema.
Оваа функција нема потреба да ни враќа никаков резултат. Сепак, во C, функциите вообичаено враќаат некаков резултат, па и ние ќе ја направиме функцијата поопшта со тоа што ќе ставиме да враќа 1 ако отпечатила нешто, или 0, во спротивно.

```
#include <stdio.h>
int cifra_vo_tekst(int cifra){

	int rez=1;

	switch (cifra) {
		case 0:
			printf("nula ");
			break;
		case 1:
			printf("eden ");
			break;
		case 2:
			printf("dva ");
			break;
		case 3:
			printf("tri ");
			break;
		case 4:
			printf("cetiri ");
			break;
		case 5:
			printf("pet ");
			break;
		case 6:
			printf("sest ");
			break;
		case 7:
			printf("sedum ");
			break;
		case 8:
			printf("osum ");
			break;
		case 9:
			printf("devet ");
			break;
		default:
			rez=0; /* znaci ne otpechatila nishto */
			break;
	}
	return rez; /* vrakja rezultat tamu od kade shto
			e povikana f-ijata */
}
```

```
int main(){
	int broj,mala,golema;
	int rez1,rez2;
	printf("Vnesete dvocifren broj:");
	scanf("%d", &broj);
	mala = broj % 10;
	golema = broj/10;

	rez1=cifra_vo_tekst(golema);
	rez2=cifra_vo_tekst(mala);

	if (!(rez1 && rez2)){
		printf("Greshka: trebashe da vnesete ");
		printf("pozitiven dvocifren broj!");
	}
	return 0;
}
```

Може пред main да се дефинира само заглавјето на функцијата, а потоа било каде после main да се дефинира и телото на функцијата:

```
int cifra_vo_tekst(int cifra); /* se najavuva samo zaglavjeto */

int main()
{
...
}

...

int cifra_vo_tekst(int cifra)
{
...		/* definicija na samata f-ija */
}

...
/* drugi funkcii */
```

Дефиницијата на една функција е од обликот:

```
tip_na_rezultatot ime_na_funkcijata(deklaracija_na_parametri, ako gi ima)
{
deklaracii na promenlivi, konstanti…
naredbi
}
```

Типичен пример за функција е функцијата степен (која не е стандардна функција во C).

```
#include <stdio.h>
int stepen(int osnova, int n); /* najavuvanje na funkcijata */

int main(){
      int i;
      for (i = 0; i < 10; ++i)
      printf("%d %d %d\n", i, stepen(2,i), stepen(-3,i));
      return 0;
}

/* stepen: podigni ja osnovata na n-ti stepen; n >= 0 */
int stepen(int osnova, int n){
      int i, st;
      st = 1;
      for (i = 1; i <= n; ++i)
      st = st * osnova;
      return st;
}
```

Дури и main е функција! Ако не се дефинира поинаку, сите функции се смета дека враќаат резултат од типот int.

Во C нема вгнездени функции една во друга како што има во Pascal.

Во примерот се гледа дека функцијата stepen е повикана двапати со различни параметри (2 и -3). Веднаш се гледа предноста во дефинирањето на функциите: поедноставни програми за читање и најчесто пократки.
Понекогаш дел од програмата се прави функција дури и само еднаш да се повикува – заради полесна читливост. Ако е добро напишана функцијата, може да се употребува и без да се знае како точно работи – доволно е да се знае кои параметри ги прима на влез и што враќа на излез од функцијата.
Еве како би изгледала задачата која за даден природен број m ќе ги испише сите броеви помеѓу 1 и 1000 за кои збирот на цифрите изнесува m.

```
#include<stdio.h>
int zbir_na_cifri (int broj)
{
      int suma, cifra;
      suma = 0;
      while (broj> 0) {
            cifra = broj % 10;

            suma += cifra;
            broj /= 10;
      }
      return suma;
}
```

Пример како може и без посебна променлива cifra:

```
int zbir_na_cifri (int broj)
{
      int suma=0;

      while (broj> 0) {
            suma += broj % 10;
            broj /= 10;
      }
      return suma;
}
```

```
int main ()
{
	int  m, i;
	printf("Vnesete go brojot m zasporedba: ");
	scanf("%d", &m);
	if (m > 27)
		printf("Ne postoibroj od 1 do 1000 so suma %d.\n", m);
	else {
		printf("Broevi cij zbir na cifri e %d se: \n", m);
		i = 1;
		while (i<=1000) {
			if (zbir_na_cifri(i) == m)
				printf("%d\t", i);
			i++;
		}
	}
	return 0;
}
```

Може да забележите дека се елиминира потребата од многу променливи во главната програма – повеќето од нив се префрлаат во функцијата. Но треба да се запомни дека во C параметрите се пренесуваат според вредноста и не постои можност да се менува содржината на една променлива од главната програма во рамки на функцијата, ако таа се пренесе како параметар во функцијата. Името на променливите нема значење при повикот. Во примерот, во функцијата променливата се вика broj, а во програмата се повикува со i. Дури и исто да се викаа, не се менува содржината на i во главната програма. C-компајлерот создава нова променлива со важност само во рамки на функцијата и тоа додека таа се извршува. По завршувањето на функцијата вредноста на таа променлива се губи, а исто и на сите други променливи декларирани во рамки на функцијата – наречени локални. Наспорти нив, глобални променливи се дефинираат надвор од main и тие важат во сите функции вклучувајќи ја и main.

Пример за void функции без параметар:

```
#include <stdio.h>

/* Deklaracijanafunkcii */
void PrintMax(intbroj);
void PrintPozdrav();

int main()
{
	PrintPozdrav();	/* PecatiPozdrav */
	PrintMax(k);	/* Japecatimaximalnatavrednost */
	return 0;
}
/* Definicijanafunkciite */
void PrintMax(intbroj)
{
	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>
#include <math.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 ");
      printf("broj %d i %d e %d\n", prostgore(broj), broj, razlika);
      return 0;
}

int prost(int n)
{
      int k;
      for(k=2; k<sqrt(n); k++)
      {
            if(n%k == 0)
                  return 0; //slozhen e
      }
      return 1; /*ako vo prethodniot opseg ne se najde nitu eden delitel,
brojot e prost */
}
```

```
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 k,brDeliteli=0;
	for(k=2; k<sqrt(n); k++){
		if(n%k == 0)
			brDeliteli++:
	}
	if(brDeliteli!=0)
		return 0;
	else
		return 1;
}

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 до 1000 што се разликуваат меѓусебе за 2. На крај да се отпечати и колку парови се пронајдени.

```
#include<stdio.h>

int eprost(int n) {
	int k,brDeliteli=0;
	for(k=2; k<sqrt(n); k++){
		if(n%k == 0)
			brDeliteli++;
	}
	if(brDeliteli!=0)
		return 0;
	else
		return 1;
}

int main (){
	int br=0,i;
	for (i=1; i<=(1000-2); i++)
	{
		if (eprost(i) && eprost(i+2)){
			printf("Prostite broevi ");
			printf("%d I %d se razlikuvaatza 2\n", i, (i+2));
			br++;
		}
	}
	printf("Pronajdeni se vkupno ");
	printf("%d paroviprostibroevi koi go zadovoluvaat uslovot\n", br);
	return 0;
}
```

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>
#include <math.h>

double f(float i,int j);
float stepen(float i, int j);
float stepen1(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)/j-stepen(i,j+2)/(j+2);
	else
		vrednost=-stepen(i,j-1)/(j-1)+stepen(i,j+1)/(j+1);
	return vrednost;
}


float stepen(float i,int j)
{
	int k;
	double vrednost;
	if (i==0)
		vrednost=0.0;
	else
	{
		vrednost=1.0;
		for (k=1;k<=j; ++k)
			vrednost*=i;
	}
	return vrednost;

}

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

float stepen1(float i,int j)
{
	return pow(i,(float)j);
}
```