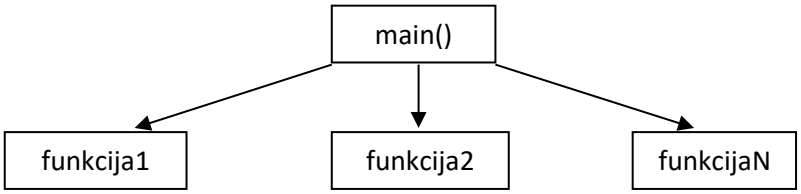
21. POJAM, DEFINICIJA I POZIVANJE FUNKCIJA

Svi programi napisani u C jeziku se sastoje od njamanje jedne funkcije koju definiše korisnik i koja se zove main(). Ova obavezna, osnovna i početna funkcija, da bi obavila svoj zadatak, rešila neki problem, uradila neki posao, uglavnom, poziva druge funkcije koje se mogu uključivati iz raznih biblioteka koje se isporučuju uz kompajler, editor, linker itd., ali mogu da budu i funkcije koje je napisao programer. **Funkcije su osnovni element za konstrukciju programa.** Osnovna ideja rada sa funkcijama u programiranju jeste da se posao izrade i pisanja programa podeli na skup manjih zadataka koji se rešavaju pojedinačno. **Određeni broj programerskih zadataka i poslova se, takođe, javljaju češće i ponavljaju se.** Ovi zadaci su kandidati da budu napisani kao funkcije, kao rešenja koja se mogu koristiti u svakom delu programa, tamo gde je to neophodno. Takođe, moguće je da se pojavi problem koji treba rešiti i koji neko drugi nije već rešio i ugradio u neku biblioteku funkcija, pa je potrebno napisati potpuno novu funkciju.



Funkcije se mogu koristiti da kontrolisano održavaju podatke i sakriju detalje načina obavljanja nekog posla, tj. zadatka, što je veoma važna osobina kvalitetnih programa.

Definicija funkcije: Funkcije su potprogrami koji na osnovu izvesnog broja argumenata daju i vraćaju u program, nakon izvršavanja, rezultat koji se naziva vrednost funkcije. Potprogrami su sastavni i osnovni elementi, tj. segmenti programskog koda koji se pozivaju radi obavljanja konkretno specifičnog zadatka.

Osobine funkcija:

- Vrednost funkcije može da se koristi ugrađivanjem poziva funkcije u izraze.
- Poziv funkcije se, u stvari, smatra operatorom kao i svi ostali operatori.
- Funkcije kontrolišu pristup do važnih podataka i čuvaju ih od neovlašćenih promena.
- Funkcija skriva detalje načina obavljanja određenog zadatka. Ovaj princip se zasniva na ideji da onoga ko poziva funkciju ne mora interesovati kako određena funkcija rešava odabrani problem koji je rezultat funkcionalne dekompozicije.

Funkcija se definiše naredbom za opis funkcije:

```
vraceni_tip naziv_funkcije(lista _argumenata/parametara)
{
 telo_ funkcije
}
```

Sve funkcije moraju da sadrže sledeća četiri elementa navedena u prethodnom opisu funkcije:

- **vraceni_tip** - predstavlja tip vrednosti koju funkcija vraća. To mogu biti svi standardni tipovi C-a ili izvedeni tipovi. Vrednost funkcije može da bude samo jedan podatak.
- **naziv_funkcije** - sastoji se od indentifikatora funkcije, koji je istovetan sa bilo kojom vrstom indentifikatora. Izbegavati nazive koji počinju sa **_** simbolom, zbog toga takva imena može da koristi program prevodilac.
- **lista_argumenata/parametara** - predstavljaju argumente funkcije pomoću kojih se vrši unošenje početnih vrednosti funkcije. Vrednost funkcije se izračunava na osnovu tih početnih podataka. Argumenti se u listi argumenata razdvajaju sa zarezom. Lista parametara

može biti prazna i tada se koristi službena reč **void** za funkciju koja ne vraća vrednost, tj. za proceduru.

- **telo_funkcije** - predstavlja sadržaj funkcije koja se definiše. Ona je po formi blok (nalazi se između vitičastih zagrada), što znači da može da sadrži deklarativne i izvršne naredbe. Ako funkcija kao rezultat svog rada vraća neku vrednost u okviru tela funkcije mora se nalaziti naredba **return**. Opšti oblik naredbe return je: return izraz; ili samo return;

Prva tri elementa čine zaglavlje funkcije koje sadrže informacije potrebne da bi ta funkcija mogla da se koristi i pozove. Telo funkcije sadrži naredbe koje obavljaju traženi zadatak.

Primer programa sa funkcijom: program sa funkcijom koja izračunava kvadratnu vrednost broja.

```
#include<stdio.h>

//deklaracija funkcije
float kvadrat(float x);

//osnovna funkcija main
int main()
{
    float m, n ;
    printf("\nUnesite broj za koji zelite izracunati kvadrat: \n");
    scanf("%f", &m);
    //poziv funkcije
    n=kvadrat(m);
    printf("\nKvadrat broja %f je %f",m,n);
}

//definicija funkcije
float kvadrat(float x)
{
    float p;
    p=x*x;
    return(p);
}
```

Rezultat izvršavanja:

Unesite broj za koji zelite izracunati kvadrat:
14

Kvadrat broja 14.000000 je 196.000000

22. FUNKCIJE I PARAMETRI

Funkcije su potprogrami koji na osnovu izvesnog broja argumenata daju i vraćaju u program, nakon izvršavanja, rezultat koji se naziva vrednost funkcije. Potprogrami su sastavni i osnovni elementi, tj. segmenti programskog koda koji se pozivaju radi obavljanja konkretno specifičnog zadatka.

Četiri osnovne osobine funkcija: Vrednost funkcije može da se koristi ugrađivanjem poziva funkcije u naredbe; poziv funkcije se smatra operatorom; funkcije kontrolišu pristup do važnih podataka i čuvaju ih od neovlašćenih promena; funkcija skriva detalje implementacije koda.

Parametri funkcije se navode na sledeći način:

```
funkcija(izraz1, izraz2, izraz3,..., izrazn)
```

- **funkcija** - označava funkciju čije se izvršavanje traži. Ona je u većini slučajeva **indentifikator funkcije** koja se poziva (ime, naziv), ali može da bude i adresni izraz čija je vrednost adresa željene funkcije.
- **izrazi** - predstavljaju stvarne **argumente funkcije** čije vrednosti služe za inicijalizaciju formalnih argumenata funkcije pre obrade tela funkcije. Stvarni **argumenti mogu biti: promenljive, konstante ili izrazi** koji moraju imati u potpunosti definisanu vrednost u momentu prosleđivanja (predaje) funkciji. Izrazi koji predstavljaju stvarne argumente funkcije izračunavaju se po proizvoljnom redosledu, neposredno pre pozivanja funkcije. Izraz za pozivanje funkcije može se koristiti kao operand u složenijem izrazu, tada se vrednost funkcije koristi u izračunavanju tog izraza. Kako je operator poziva funkcije () visokog prioriteta, što obezbeđuje da se izračunavanje vrednosti funkcije (pozivanje funkcije) izvrši pre bilo kog susednog operatora u izrazu. Ukoliko funkcija nema svoju vrednost (tip void) može da se koristi samo kao drugi operand operatora zarez ili kao drugi ili treći operand trinarnog operatora. **Funkcija koja nema svoju vrednost** najčešće se koristi kao operand izraza u prostoj naredbi fun(...) ime(lista stvarnih argumenata); Poziv funkcije je operator u C jeziku. Pojedini argumenti odvajaju se zarezom “,”. Ako funkcija nema argumenata, piše se samo **( )**.

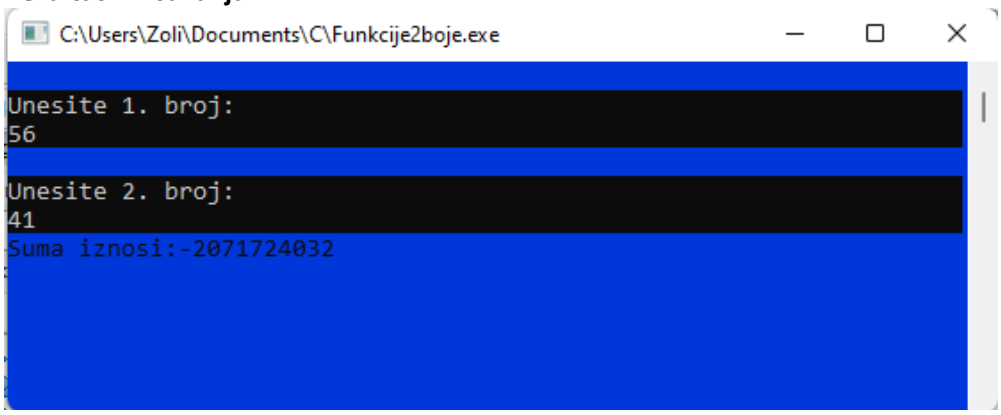
U C jeziku se kao **standardni metod predaje parametara koristi tehnika pod nazivom “poziv po vrednosti”**. To znači da se funkciji koja se poziva i koja prima parameter/argumente, kao stvarni argument daje kopije podataka koji se predaju, ne predaje se memorisjka lokacija promenljive koja se predaje, tako da pozvana funkcija ne može da utiče na vrednost promenljive.

Alternativna tehnika jeste “poziv po referenci”, koja prilikom poziva funkcije predaje memorijsku lokaciju promenljive koja je argument. Pozvana funkcija, u ovom slučaju, može da promeni vrednost promenljive koja je parameter. Ova tehnika u C-u preko korišćenja adrese lokacije, a to podrazumeva rad sa pokazivačima.

Primer programa sa funkcijom sa argumentima i bez argumenata: program sa funkcijom koja izračunava sumu dva broja i bez argumenata za promenu boja.

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

void promeniboje();     //deklaracija funkcije bez parametara
int suma(int a, int b); //deklaracija funkcije sa parametrima

int main()
{
   int a, b;
   printf("\nUnesite 1. broj: \n");
   scanf("%f", &a);
   printf("\nUnesite 2. broj: \n");
   scanf("%f", &b);
   promeniboje();      //poziv funkcije
   int c=suma(a,b);    //poziv funkcije
   printf("Suma iznosi:%d\n",c);
   getch();
}

int suma(int a, int b) //telo funkcije sa parametrima
{
   int c;
   c=a+b;
   return c;
}

void promeniboje() //telo funkcije bez parametara
{
   SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_RED);
   SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
BACKGROUND_BLUE);
   //dodatne boje: INTENSITY & GREEN
}
```

Rezultat izvršavanja:



23. REKURZIVNE FUNKCIJE

Algoritamske strukture u opštem slučaju mogu biti linearne, razgranate, iterativne ili rekurzivne. Programski jezik C podržava rad sa ova četiri tipa algoritma gde se mogu koristiti obične i rekurzivne funkcije. Osnovna ideja rekurzije uključuje dva aspekta:

- Da se poznati problem redukuje i rasloži na manje i jednostavnije.
- Da se u rešavanju problema svakim rekurzivnim korakom približavamo željenom rešenju izmenom parametara prvobitnog problema.

Definicija rekurzivne funkcije:
Rekurzivne funkcije su funkcije koje neposredno ili posredno (preko drugih funkcija) pozivaju same sebe.

Karakteristike rekurzije - Korišćenje rekurzije pruža prednost u nekim situacijama tako što pojednostavljuje logiku rešenja i zato što je potrebno pisati manje programskog koda. Rekurzija se primenjuje kod rada sa hijerarhijski organizovanim sistemima fajlova, u algoritmima za sortiranje, ali ima i negativne osobine. Rekurzivna rešenja su po pravilu manje efikasna od iterativnih rešenja istih problema. Dešava se da rekurzivna rešenja traju neprihvatljivo dugo ili se traži dosta operativne memorije za čuvanje međustanja (vrednosti struktura podataka) između rekurzija. Teže se otkrivaju greške nego kod iterativnih, razgranatih i linearnih algoritama. Kod rekurzije se mora voditi računa da se obezbedi uslov za završetak pozivanja funkcije, u protivnom će se desiti beskonačna rekurzija. Ovaj uslov ima naziv “sigurnosni pojas”. Rekurzivna rešenja mogu da se napišu i iterativno, ali u većini slučajeva zahteva se dosta vise koda.

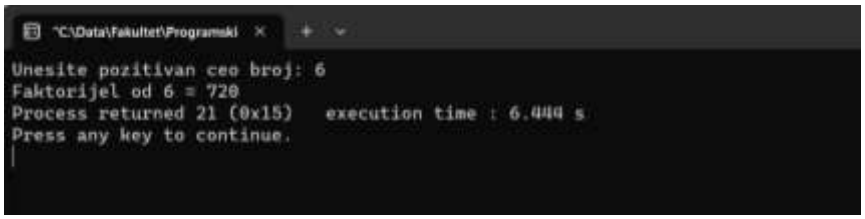
Još neki problemi čije rešavanje jeste pogodno za rekurzivne algoritme: statistika, matematički problemi, kao što su izračunavanje faktorijela, Fibonačijevi nizovi itd.

Primer: Faktorijel pozitivnog broja n je dat kao:
faktorijel od n (n!) = 1 * 2 * 3 * 4 *... * n

Faktorijel negativnih brojeva ne postoji, dok je faktorijel od 0 jednak 1.

```
#include<stdio.h>

//deklaracija rekuzivne funkcije
long int faktorijel(int n);

void main()
{
    int n;
    printf("Unesite pozitivan ceo broj: ");
    scanf("%d",&n);
    //poziv funkcije faktorijel
    printf("Faktorijel od %d = %ld", n, faktorijel(n));
}

//implementacija rekuzivne funkcije
long int faktorijel(int n)
{
    if (n>=1)
        return n*faktorijel(n-1); //poziv rekurzivne funkcije
    else
        return 1;
}
```

Rezultat izvršavanja:



Kada korisnik unese npr. 6, prvo se funkcija fakotrijel(n) poziva iz main() funkcije sa prosleđenim n=6 kao argumentom. Zatim se n-1=5 prosleđuje u poziv faktorijel(n) iz iste funkcije (rekurzivni poziv). U svakom rekurzivnom pozivu, vrednost argumenta n smanjuje se za 1. Kada je vrednost n manja od 1, tj. 0 (nula), nema rekurzivnog poziva i faktorijel se konačno vraća funkciji main().