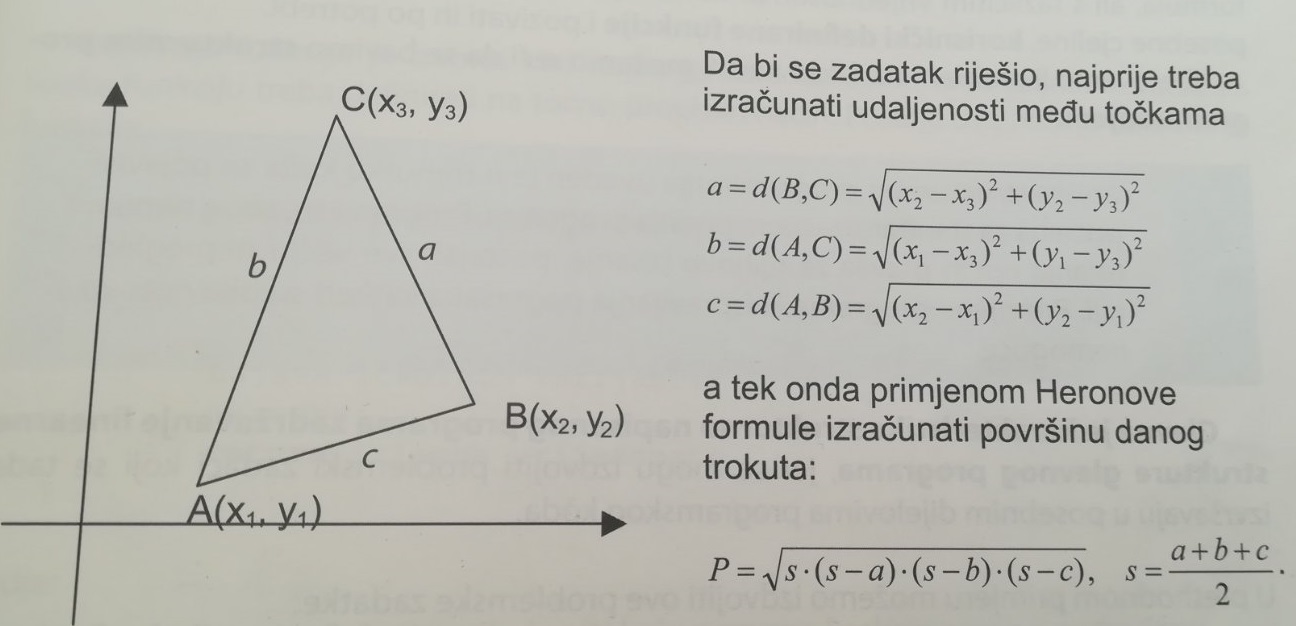
**Računalstvo 2 i 6h 23.03**

**Funkcije**

Već je rečeno da se svi programi u C-u sastoje od jedne ili više funkcija i da je jedina obvezatna funkcija **main()** (koja označava početak izvršavanja glavnog dijela programa). Prilikom programiranja često se neki dijelovi programa trebaju više puta izvoditi, ali s različitim vrijednostima varijabli.

Na primjer: Neka su zadane tri točke u pravokutnom koordinatnom sustavu. Potrebno je izračunati opseg i površinu trokuta koji nastaje spajanjem tih točaka.



Slika . Trokut u kordinatnom sustavu i formule

Rješenje:

Zadatak bi izgledao ovako:

#include<stdio.h>

#inluce<math.h>

void main()

{ float x1,x2,x3,y1,y2,y3,d1,d,3,d4,o,s,p;

printf(„Unesi koordinatne točke A:“);

scanf(„%f,%f“,&x1,&y1);

printf(„\nUnesi koordinatne točke B:“);

scanf(„%f,%f“,&x2,&y2);

printf(„\nUnesi koordinatne točke C:“);

scanf(„%f,%f“,&x3,&y3);

/\*računanje kordinate među točkama\*/

d1=sqrt(pow(x2-x1,2)+pow(y2-y1,2));

d2=sqrt(pow(x3-x2,2)+pow(y3-y1,2));

d3=sqrt(pow(x1-x3,2)+pow(y1-y3,2));

/\*primjena Henonove formule/

o=d1+d2+d3;

s=o/2;

p=sqrt(s\*(s-d1)\*(s-d2)\*(s-d3));

printf(„\nDuljina stranica trokuta su %.2f, %.2f, %.2f“,d1,d2,d3);

printf(„\nOpseg je %.2f\nPovršina je %.2f“,o,p);

}

Lako se može uočiti da se računanje udaljenosti među točkama zapravo rabi ista formula, ali s različitim vrijednostima. Takve dijelove programa praktično je izdvojiti u posebne cjeline, korisničko definiranje funkcije i pozivati po potrebi.

Rješavamo li problem na takav način, možemo reći da se bavimo strukturnim programiranjem.

Pojam strukturnog programiranja uveden je u trenutku kada se pojavila potreba za standardizacijim pisanja programa. Programi su, zbog nepostojanja općih pravila za njiovo pisanje, postoajali sve veći i nepregledniji. Promjene programa ili ispravljanje grešaka katkad su bile gotovo nemoguće

**Glavna je karakteristika strukturnog napisanog programa zadržavanje linearne strukture glavnog programa,** jer se mogu izdvojiti problemski zadatci koji se tada izvršavaju u posebnim dijelovima programskog koda.

U prethodom primjeru možemo izdvojiti ove problemske zadatke:

Računanje površine trokuta

Računanje udaljenosti između točaka

Unošenje koordinata točaka

Zadatak

Pod funkcijama se razumijevaju cjelovita skupina naredbi koja izvršavanjem ispunjava određene zahtjeve.

To znači da je funkcija u C-u isto što i funkcija, potprogram i procedura u ostalim višim programskim jezicima. One omogućavaju razgledanje problema na jednostavnije i manje cjeline, čime **pridonose boljoj preglednosti izvornog koda i jednostavnijem pronalaženju i otkrivanju pogrešaka.** Također, uporabom funkcija **dolazi do znatnijih ušteda memorijskog prostora.**

Funkcije se nemoraju rabiti samo da bi se izbjeglo višestruko pisanje određenih dijelova programa već su praktične i ako se izvršavaju samo jedanput. U tom slučaju olakšavaju razumijevanje programa, odnosno algoritma.

Jedanput napisanu funkciju moguće je rabiti u više različitih programa.

Da bi se napisala funcija, potrebno je znati kako je ispravno definirati i povezati.

Detaljno objašnjeno na stranici: <https://www.youtube.com/watch?v=-cucWUkd9Qc>

Lekcija je na 91.stranici udžbenika. Za sve ostale nejasnoće obratiti se na e-mail: [ivica\_tolo@net.hr](mailto:ivica_tolo@net.hr)

**Definiranje funkcije**

Definiranje funkcije je izvorni kod kojim je određen način rada pojedine funkcije. Svaku funkciju treba definirati na točno propisan način. Postoje dva načina definiranje funkcije.

Prvi način je noviji i preporučuje ga ANSI C:

**tip\_funkcije** ime\_funkcije(najava liste\_argumenata)

najava lokalnih varijabli;

naredbe;

return izlazna\_vrijednost;

gdje:

* **tip\_funkcije** određuje tip vrijednosti koju pozvana funcija vraća u nadređenu funkciju, a može biti bilo koja od osnovnih tipova podataka (int, char, float, double, void)
* **ime\_funkcije** je indefikator s pomoću kojeg se funkcija poziva
* **lista\_argumentata** je lista formalnih parametara s pripadajućim tipovima s pomoću kojih se niz podataka iz nadređene funkcije prenose u pozvanu funkciju.

Sljedeća funkcija ima takvu definiciju:

int pomnoži (int x, int y)

{

int rez; /\*najava lokalne varijable\*/

rez=x\*y; /\*naredbe u funkciji\*/

return rez; /\*izlazna vrijednost\*/

}

Drugi način razlikuje se od prvog po tome što se argumenti funkcije najavljuju izvan zagrade.

**tip\_funkcije** ime\_funkcije(lista\_argumenata)

najava liste\_argumenata;

{ najava lokalnih\_varijabli;

naredbe;

**return** izlazna\_vrijednost;

}

Prethodni primjer zapisan na ovaj način je:

int pomnoži (x, y)

int x,y; /\*najava liste argumenata\*/

{

int rez; /\*najava lokalne varijable\*/

rez=x\*y; /\*naredbe u funkciji\*/

return rez; /\*izlazna vrijednost\*/

}

Funkciju je dovoljno definirati jedanput. Nakon toga moguće ju je pozvati iz bilo kojeg dijela programa. Funkcija u C-u mogu **vratiti u glavni program jednu ili niti jednu vrijednost** (što je određeno tipom funkcije). Za to se rabi naredba **return** koja **prenosi izlaznu vrijednost u nadređeni program.** Funkcija se prestaje izvršavati nailaskom na prvu naredbu **return.**

Funkcija koja nevraća nikakvu vrijednost dodjeljuje se ti podataka **void.** Takve funkcije služe za obavljanje niza radnji, recimo za ispis, i ne sadržavaju naredbu **return.**

Na primjeru funkcija **proba** ne prenosi nikakvu vrijednost u glavni program, već samo ispisuje navedeni tekst.

void proba()

{

printf(„Ovo je funkcija“);

}

Ako u glavni program treba vratiti niz vrijednosti, onda se to radi s pomoću argumenata funkcije koji su najavljeni kao globalne vrijednosti. Sljedeća funkcija će u glavni program vratiti dvije vrijednosti.

#include <stfio.h>

int a,b; /\*najava globalnih varijabli\*/

/\*funkcija zamjena\*/

void zamjena ()

{

int t; /\*lokalna varijabla\*/

t=a; /\*zamjena korištenjem globalnih varijabli\*/

a=b;

b=t;

}

void main()

{

printf(„Unesite dva cijela broja:“);

scanf(„%d,%d“,&a,&b);

printf(„\nUpisali ste a=%d i b=%d“,a,b);

zamjena();

printf(“\Nakon funkcije a=%d i b=%d“,a,b),

}

Ako funkija vraća cjelobrojnu vrijednost, tip podataka se može izostaviti.

Na primjer, sljedeće su funkcije iste:

int proba1(int x)

{

return (x\*x);

}

proba1(int x)

{

return (x\*x);

}

Lekcija je na 93.stranici udžbenika. Za sve ostale nejasnoće obratiti se na e-mail: [ivica\_tolo@net.hr](mailto:ivica_tolo@net.hr)