Datum: 2.12.2015.

FUNKCIJE

UVOD

Do sada smo razmatrali programe, koji su se sastojali iz samo jedne programske jedinice odnosno iz glavne funkcije, koja je označena sa **main**(). Ovaj postupak je pogodan za rješavanje jednostavnijih zadataka. Po pravilu se, kod rješavanja zadataka, nastoji cijeli problem rastaviti na više manjih problema, koji su lakši za riješiti. Rezultat ove podjele u jednostavnije korake je nastanak programa sa modularnom strukturom, pri čemu je u svakom modulu implementirano rješenje jednoga problema ili dijela problema.

Potprogrami u programskom jeziku C++ su definirani kao funkcije. **Oni su dio hijerarhijski ustanovljene strukture na čijem početku stoji glavni program.**

Primjena funkcija ima mnogo prednosti:

Složeniji programi se mogu modularno graditi. Svaki modul obuhvata samo dio problema, koji je lakše riješiti od cijelog problema. Na ovaj način se cijeli zadatak pojednostavljuje. U programiranju, primjenom funkcija, povećava se preglednost programa. Svaka funkcija se može posebno kreirati i testirati. Ovakav pristup olakšava kasniju potragu za greškama, jer je mala programska jedinica uvijek preglednija od kompleksnog programa.

Dijelovi programa sa instrukcijama, koji se u programu susreću na različitim mjestima i sa različitim vrijednostima, mogu se definirati kao funkcija koja se poziva gdje je to potrebno. Funkcije, koje rješavaju određene tipove problema, mogu se koristiti i kod drugih programa (reusing).

Dakle, svaki zadatak lakše ćemo riješiti ako ga podijelimo na manje zadatke (podzadatke). Te "podzadatke" u jeziku C++ obavljaju funkcije.

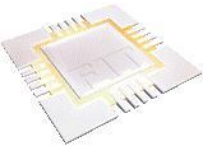
Zašto upotrebljavamo funkcije?

- Raščlanjujemo problem na manje dijelove.
- Razlučujemo dijelove koji imaju zajednička svojstva, tj čine cjelinu.
- Upotrebljavamo funkcije iz svojih ili drugih biblioteka funkcija i iz drugih programa. Stvaramo vlastite biblioteke funkcija!
- Mogućnost poziv istoga programskog modula sa drugim podacima.

Matematički pojam funkcije

"Ja ću vam dati jedan broj. Vi trebate tom broju dodati 2 i produkt pomnožiti sa danim brojem" . Slične instrukcije možete dati i kompjuteru izražene na nekom programskom jeziku.

Tada očekujete da broj *x* date, da mašina izračuna *x + 2*, zatim *(x + 2) * x.* Za dati ulaz *x* mašina će proizvesti jedinstven izlaz *(broj (x + 2) * x)*.



U matematici, proces koji, za dati ulaz (određenog tipa), proizvodi jedinstven izlaz naziva se funkcija.

Efekt procesa obrade na računaru može se promatrati kao evaluacija funkcija. Zato su funkcije najvažnija matematička tema u oblasti kompjuterske obrade.

Funkcija F iz skupa X u skup Y prima ulazne vrijednosti iz skupa D, gdje svaki element iz skupa D pripada skupu X a nije nužno da skup D obuhvata cijeli skup X. Za svaki element (x) iz D, funkcija F proizvodi jedinstven odgovarajući izlaz $F(x)$. Ako x nije iz D, $F(x)$ nije definirana. Za funkciju F skup X je izvorni skup a skup Y ciljni skup.Skup D je domena funkcije F. Pišemo $F : X -> Y$

STANDARDNE FUNKCIJE

U programski jezik C++ je ugrađeno mnoštvo funkcija (smješteno u standardne biblioteke funkcija).

Svaki C ++program, koji koristi standardnu biblioteku mora, na početku, sadržavati:

```
#include <iostream>
```

Ovo je preprocesorska naredba za ulaz/izlaz standard. Kao aktualni parametri (sadržaj unutar zagrada) kod standardnih funkcija mogu se navoditi samo izrazi i varijable. To znači da poziv funkcije slijedi kao dio aritmetičkog izraza. Matematičke funkcije se definiraju u okviru biblioteke cmath, i prije njihove upotrebe mora se cmath (datoteka zaglavlja) "uključit" (include) u naš program.

```
#include <cmath>
```

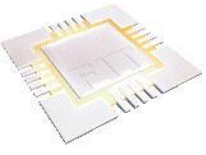
Slijedi kratak pregled nekoliko standardnih funkcija sa kratkim objašnjenjima:

- abs(x)aritmetička funkcija, koja izračunava apsolutni iznos nekog parametra x(int). Rezultat ima isti tip kao i parametar.

- atan(x) aritmetička funkcija, koja izračunava arkus tangens nekog parametra x(double). Rezultat ima tip double. Rezultat je dat u stupnjevima.

- cos(x) aritmetička funkcija, koja izračunava (realnu) vrijednost kosinusne funkcije nekog parametra x(double). Parametar x se je u stupnjevima.

- exp(x) aritmetička funkcija, koja izračunava vrijednost eksponencijalne funkcije. Parametar x je tipa double. Rezultat je tip double.



- fab(s) aritmetička funkcija, koja izračunava apsolutni iznos nekog parametra x(double). Rezultat ima isti tip kao i parametar.

- log(x) aritmetička funkcija, koja izračunava prirodni logaritam sa bazom e. Parametar (double) x ima vrijednost nula ako je nula ili negativan.

- log10(x) aritmetička funkcija, koja izračunava logaritam po bazi 10. Rezultat ima tip double.

- pow(x,y) aritmetička funkcija, izračunava x na y. Parametri x i y moraju biti tipa double. Rezultat je jednak nuli, ako je y negativan ili ako su x i y jednaki nuli.

- sin(x) aritmetička funkcija, koja izračunava (realnu) vrijednost sinusne funkcije nekog parametra x(double). Parametar x je u stepenima.

- sqrt(x)aritmetička funkcija, koja izračunava (realnu) vrijednost kvadratnog korjena nekog parametra x(double). Rezultat ima vrijednost nula ako je x negativan.

Svaki C++ program mora imati funkciju main.

Izvršavanje programa uvijek počinje funkcijom main.

Bilo koja druga funkcija je potprogram i mora biti pozvana.



OSNOVNI ELEMENTI FUNKCIJE I UPOTREBA FUNKCIJE

Program izrađen u programskom jeziku C++ sastoji se, uglavnom, od funkcija koje mogu biti:

- Funkcije, koje rezultate daju kroz listu parametara. Definiraju se i deklariraju kao **tip *void***.
- Funkcije, koje rezultat daju preko svoga imena, imaju tip "zadane vrijednosti". Primjer za ovaj tip funkcije je standardna trigonometrijska funkcija sin(x). Ova funkcija daje vrijednost *sinus x* preko svoga imena, stoji na desnoj strani znaka dodjeljivanja i odgovara funkciji u strogom smislu te riječi.
  Primjer:

y = sin(x);

U programskom jeziku C++ postoji razlika između deklariranja i definiranja funkcija:

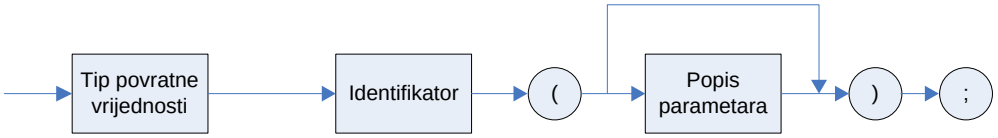
- **Deklariranjem funkcije** saopćava se ostalom dijelu programa da u tom programu postoji funkcija sa navedenim imenom.
- **Definiranjem funkcije** se utvrđuje, šta funkcija radi.

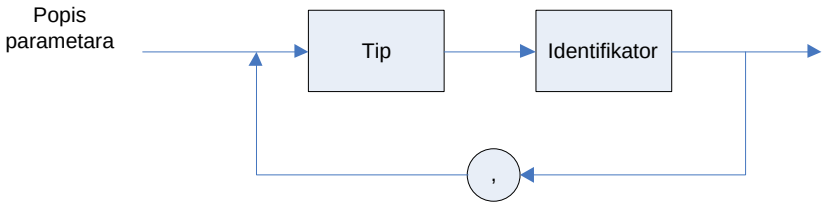


DEKLARACIJA (PROTOTIP) FUNKCIJE

Sve funkcije, koje se u sklopu jednog programa pozivaju, moraju biti deklarirane .

Osnovna forma deklaracije:





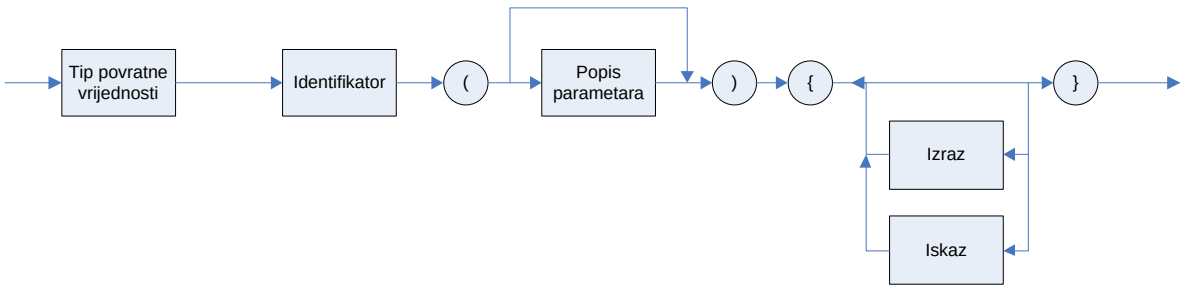
- npr. na početku glavnog programa ("glavna funkcija") - main( ).

Prototip izgleda kao zaglavlje funkcije ali mora završiti sa ; (semicolon) i njena lista parametara mora sadržavati tip podatka za svaki parametar.

```
int Kub( int ); //prototip
```

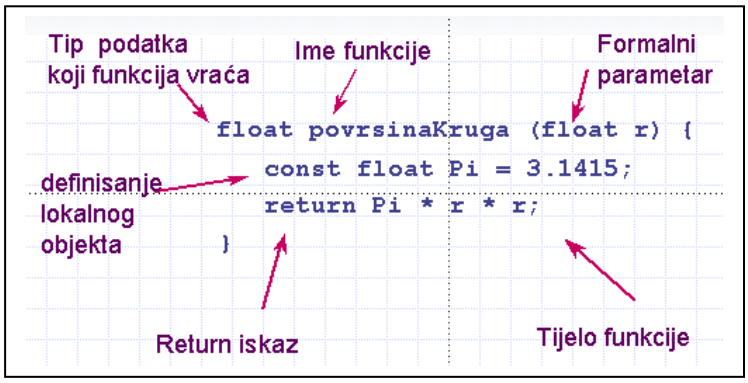
DEFINICIJA FUNKCIJA

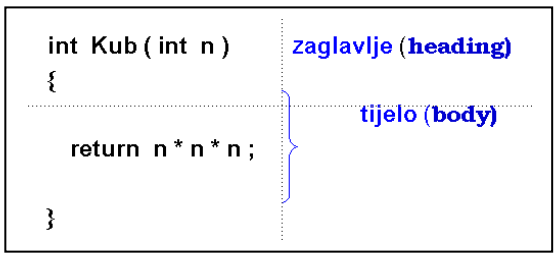
Osnovna forma definicije:




```
Tip vrijednosti koji funkcija vraća   ImeFunkcije (Formalni parametri)
{
 iskaz;
 izraz;                                (TipArg1 ImeArg1, TipArg2 ImeArg2,..)
}
```

Na primjer:





Svaka funkcija ima ime kojim se poziva i kroz svoje ime je "definirana". Tip funkcije može biti ili void ili tip "povratne vrijednosti" o čemu će biti riječi kasnije. Navedeni parametri u zagradama se koriste kod poziva funkcije. Funkcije u programiranju međusobno razmjenjuju podatke kroz listu parametara. Potrebno je razumjeti jasnu razliku između *formalnih i aktualnih* parametara.

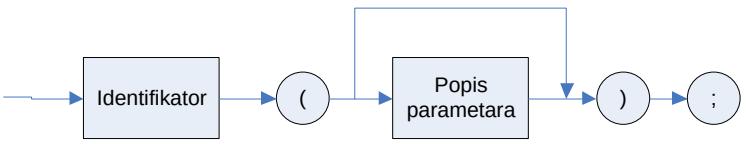
Aktualni parametri su oni parametri, koji pri pozivu funkcije stoje u listi parametara i njihova aktualna vrijednost je poznata.

Formalni parametri stoje u zaglavlju (dio definicije funkcije) pozvane funkcije i , prije poziva ove funkcije, nije im pridijeljena vrijednost. To znači da oni samo formalno postoje i da dobivaju vrijednost kad se ova funkcija pozove iz programa (iz glavne funkcije main ili iz neke druge funkcije).

Funkcija može biti definirana i unutar glavne (main()) funkcije i tada se naziva inline funkcija.

POZIV FUNKCIJE

Funkcija se u programskom bloku glavne funkcije ili neke druge funkcije koristi tako da se poziva imenom (identifikatorom funkcije) i „slanjem" aktualnih vrijednosti parametara ili memorijskih adresa parametara. Nakon izvršavanja funkcije (na mjestu gdje je funkcija definirana), programski blok koji je pozvao funkciju prima povratnu vrijednost funkcije i koristi je u daljem procesu.





Prije poziva funkcije u programu, kompajler mora obraditi ili deklaraciju (prototip) funkcije ili njenu definiciju (zaglavlje i tijelo funkcije) !



Za pozvanu funkciju:

određuje se privremeni memorijski prostor
- za vrijednosti njenih parametara i bilo koju lokalnu varijablu,
- za ime funkcije ako vrijednost koju funkcija vraća nije tipa void.

Zatim kontrola toka izvršavanja programa prelazi na prvi iskaz u tijelu funkcije. Iskazi u tijelu funkcije se izvršavaju sve dok se ne desi nešto od:

return iskaz (sa ili bez vraćene vrijednosti),

ili

zatvorena vitičasta zagrada tijela funkcije.

Tada se kontrola izvršavanja programa prenosi ponovo u programski blok - na mjesto odakle je funkcija pozvana.

U zavisnosti od vrste prenošenja aktualnih parametara na formalne, razlikuju se

dvije vrste poziva funkcije:

1. **call by value (poziv po vrijednosti)**
2. **call by reference (poziv po referenci)**

