**31.-32. Radna nedelja (21. 04. 2020. - 30.04. 2020.)**

Predmet : Računari i programiranje

Odeljenje: III2

Nastavna jedinica : Prenos parametara po reference, Prototip funkcije

Nastavnik : Ivan Mladenović

Odgovore na pitanja i rešenja zadatka **poslati do 30.04.2020 god.** na e-mail

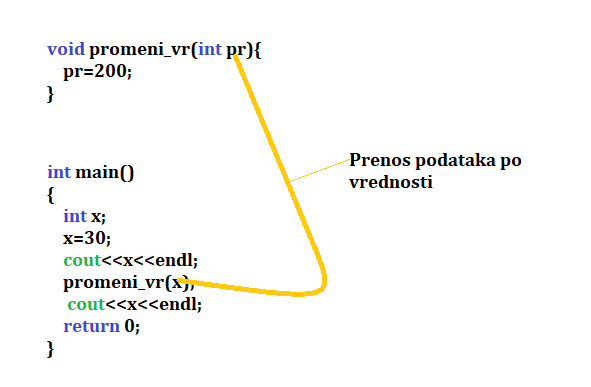
*ivan.mladenovic9901@gmail.com*

***Prosleđivanje parametra po vrednosti i po referenci***

U prethodnom primeru prikazano je prosleđivanje parametra po vrednosti. Pošto parametri u drugoj funkciji predstavljaju novu memoriju koja samo iz memorije parametara glavne funkcije dobijaju kopije vrednosti, svaka eventualna promena vrednosti unutar druge funkcije neće se odraziti na podatke definisane u prvoj.  
​Ovo možemo ilustrovati kroz sledeći primer:

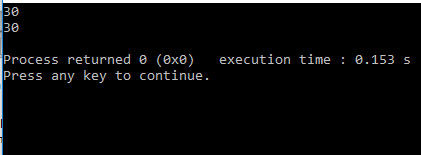
**Primer 2: Zamena vrednosti podatku**

Zadati celobrojnom podatku vrednost 20, a zatim napraviti metodu koja ovu vrednost menja na 100.  
Napravimo funkciju koja će promeniti vrednost poslatom podatku:

[](https://izprogramiranja.weebly.com/uploads/6/1/4/6/61467803/promenapovrednosti_orig.png)

Slika 6: Primer "zamena vrednosti podatku"- po vrednosti

Kad pokrenemo ovaj program na izlazu dobijamo

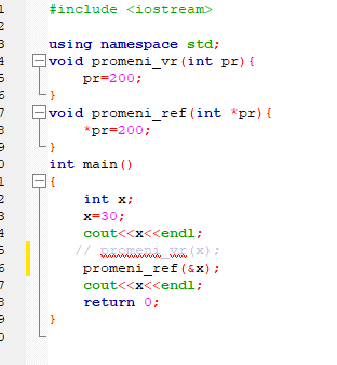
[](https://izprogramiranja.weebly.com/uploads/6/1/4/6/61467803/zamena-vrednosti-podatku-primer-po-vrednosti_orig.png)

Slika 7: Primer "Zamena vrednosti podatku" - izlaz

Vidimo da u glavnoj funkciji ova vrednost nije promenjena i ako na prvi pogled izgleda da je u kodu sve ispravno.  
Ova vrednost se jeste promenila unutar funkcije promeni\_vr, ali se ova promena ne odražava na podatke u glavnoj funkciji.  
Da bi ovo bilo ispravno, parametri se moraju preneti po referenci.

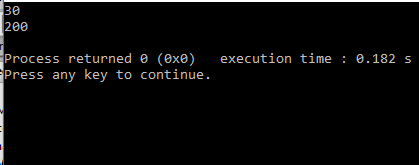
***Prenos parametara funkcije po referenci***

Za razliku od prenosa po **vrednosti**, kod prenošenja po **referenci**, ne pravi se nova memorija za parametre u drugoj funkciji "promeni\_ref", nego su ti parametri zapravo reference(drugi naziv) za istu memoriju koju zauzimaju podaci parametri u glavnoj funkciji. To će prouzrokovati da se **promene vrednosti parametara u drugoj funkciji odrazi i na prvu**.  
  
Izmenimo prethodni primer zamenom funkcije koja sada prosleđuje parametre po referenci. Parametri sada zapravo nisu podaci nego pokazivači na te podatke..  
Prethodni primer sada izgleda kao na slici 8:

[](https://izprogramiranja.weebly.com/uploads/6/1/4/6/61467803/zamena-vrednosti-podatku-primer-po-referenci_orig.png)

Slika 8: Primer "zamena vrednosti podatku" - po referenci

Posle pokretanja posle izmene na izlazu dobijamo

[](https://izprogramiranja.weebly.com/uploads/6/1/4/6/61467803/zamena-vrednosti-podatku-primer-po-referenci-izlaz_orig.png)

Slika 9: Primer "zamena vrednosti podatku" - po referenci-izlaz

**Прототипови функција**

Дефиниција функције

тип\_вредности име\_функције (низ\_параметара) тело\_функције

наводи све особине функције: тип вредности функције, број и типове параметара и опис шта и како ради функција. За корисника функције није битно како функција ради, већ само на који начин је треба користити. Другим речима, потребно је само знати ког су типа параметри и ког је типа резултат.

Прототипови функција наводе само те, споља видљиве, особине функција. Општи облик прототипа функције је:

тип\_вредности име\_функције (низ\_параметара);

Ако функција нема ниједан параметар, у прототипу функције треба уместо низа параметара ставити службену реч void.

С обзиром да прототип функције наводи све информације неопходне за исправно позивање функције, пре првог позивања функције довољно је навести само њен прототип. Потпуна дефиниција може да се наведе и касније.

/\* program koji izracunava zbir prvih n prirodnih brojeva \*/

#include<stdio.h>

int zbir(int n); /\* prototip funkcije \*/

main() /\* definicija glavnog programa \*/

{

int k;

cout<<("k?");

cin>>k;

cout<<("zbir=",zbir(k));

}

int zbir(int n) /\* definicija funkcije \*/

{

int s=0,i;

for(i=1;i<=n;i++)s+=i;

return s;

}

***Deklaracija funkcija***

Pod deklaracijom podatka ili funkcije u programiranju naziva se određivanje identifikatora i opisivanje osobina podataka ili funkcije, bez dodeljivanja memorijskog prostora za smeštanje podataka ili funkcije. U slučaju funkcije određuje se tip vrednosti funkcije  i broj i tipovi argumenata. U opštem slučaju deklaracija izgleda:  
  
**oznaka\_tipa naziv\_funkcije(niz\_argumenata);**  
  
Deklaracija funkcije još se naziva i prototip funkcije, kao i potpis funkcije.  
Ako se funkcija nalazi u fajlu iznad main funkcije, kao u prethodnim primerima onda nije potrebno posebno navesti deklaraciju. U suprotnom deklaracije je potrebno napisati iznad main funkcije. Npr. prototip funkcije max bi bio:  
  
**int max(int a,int b);**   ili samo  
  
**int max(int ,int);**Prototip funkcije promeni\_ref izgleda:  
  
**void promeni\_ref(int \* );**

**Main funkcija**   
  
O main funkciji je već bilo reči. Ona je glavna funkcija u kojoj se nalaze sve naredbe. Ima dosta karakteristika sličnih korisničkim funkcijama. Ima telo i vraća vrednost. Pošto vraća broj i ne prima argumente, onde se može pisati:

**int** main(void)

ali se ne mora. Napominjem da main ipak može da prima argumente, ali to je domen naprednog C++-a. Ime joj se ne može menjati, kao ugrađenim funkcijama.

Pitanja :

1. Objasniti razliku između postupka prosleđivanja parametara po vrednosti I prosleđivnja po reference?
2. Šta predstavlja prototip funkcije ( kako se definiše)?
3. Napisati primer definicije neke funkcije i njenog prototipa ?
4. Koja je to main() funkcija?

Zadatak :

1. Koristeći funkciju napisati program koji izračunava proizvod prvih n brojeva (n unosite preko tastature).

Prilikom pozivanja funkcije koristiti prosleđivanje parametara po referenci.

1. Koristeći funkciju napisati program koji izračunava aritmetičku sredinu brojeva od a do b. (a i b unosite preko tastature).

Prilikom pozivanja funkcije koristiti prosleđivanje parametara po referenci.