PANEVROPSKI UNIVERZITET APEIRON  
FAKULTET INFORMACIONIH TEHNOLOGIJA  
SMJER POSLOVNA INFORMATIKA

**NIKOLINA BLAGOJEVIĆ  
JEZIK C - NAREDBE I KONTROLA TOKA  
SEMINARSKI RAD**

Mentor: Datum:  
Prof. dr Branimir Trenkić 30.1.2024, Banja Luka

SADRŽAJ

[1. Uvod 4](#_Toc189122860)

[2. Osnovne karakteristike jezika C 5](#_Toc189122861)

[3. Naredbe u jeziku C 6](#_Toc189122862)

[Izvršne naredbe 6](#_Toc189122863)

[Aritmetičke operacije 6](#_Toc189122864)

[Dodjela vrijednosti promjenjivim 7](#_Toc189122865)

[Pozivi funkcija i rad sa argumentima 7](#_Toc189122866)

[Uslovni izrazi i evaluacija logičkih operacija 8](#_Toc189122867)

[Deklarativne naredbe 9](#_Toc189122868)

[Cjelobrojni tipovi 9](#_Toc189122869)

[Brojevi sa pokretnim zarezom 10](#_Toc189122870)

[Znakovni tipovi 10](#_Toc189122871)

[Pokazivači 11](#_Toc189122872)

[Strukture i unije 11](#_Toc189122873)

[Enumeracije 12](#_Toc189122874)

[4. Kontrola toka u jeziku C 13](#_Toc189122875)

[Grananja 13](#_Toc189122876)

[If 13](#_Toc189122877)

[If-else 13](#_Toc189122878)

[If-else if-else 14](#_Toc189122879)

[Switch-case 14](#_Toc189122880)

[Petlje 15](#_Toc189122881)

[For petlja 15](#_Toc189122882)

[While petlja 16](#_Toc189122883)

[Do-While petlja 16](#_Toc189122884)

[Ugnježdene petlje 16](#_Toc189122885)

[Ugnježdena for petlja 17](#_Toc189122886)

[Ugnježdena while petlja 17](#_Toc189122887)

[Ugnježdena do-while petlja 18](#_Toc189122888)

[Skakanje 19](#_Toc189122889)

[Break 19](#_Toc189122890)

[Continue 19](#_Toc189122891)

[Goto 20](#_Toc189122892)

[Zaključak 21](#_Toc189122893)

[**Literatura** 22](#_Toc189122894)

# Uvod

Jezik C je jedan od najvažnijih i najuticajnijih programskih jezika u svijetu softverskog inženjeringa. Razvijen početkom 1970-ih godina, C je postao temelj za mnoge moderne programske jezike i alate koji se danas koriste u industriji. Njegova popularnost nastala je iz njegove efikasnosti, fleksibilnosti i blizine mašinskom jeziku, što ga čini pogodnim za razvoj operativnih sistema, sistemskog softvera, uređajskih drajvera, kompajlera i mnogih drugih aplikacija visokih performansi.

Osnovni cilj ovog rada je da pruži prijegled različitih tipova naredbi u jeziku C i njihov značaj u upravljanju tokom izvršenja programa. U radu će biti detaljno analizirane izvršne naredbe, deklarativne naredbe, kao i strukture kontrole toka.

Razumijevanje naredbi i kontrole toka u jeziku C od suštinskog je značaja za svakog programera, jer omogućava efikasno pisanje koda, optimizaciju performansi i rješavanje složenih programskih problema. Kroz rad ćemo analizirati kako ove naredbe funkcionišu, pružiti praktične primjere i uporediti ih sa sličnim mehanizmima u drugim programskim jezicima.

# Osnovne karakteristike jezika C

Jezik C je razvijen 1972. godine u Bell laboratoriji od strane Denisa Ričija i od tada se koristi kao jedan od najvažnijih jezika u svijetu programiranja. Njegove osnovne karakteristike uključuju:

**Strukturno programiranje** – C omogućava programerima da organizuju kod u funkcionalne module, što olakšava razvoj velikih softverskih sistema.

**Efikasno upravljanje memorijom** – Kroz pokazivače, dinamičku alokaciju i manipulaciju podacima, programeri imaju punu kontrolu nad memorijskim resursima.

**Blizina mašinskom jeziku** – C omogućava direktan pristup hardvijerskim resursima i operacijama niskog nivoa, što ga čini pogodnim za sistemsko programiranje.

**Prilagodljivost i prijenosivost** – Programi napisani u C jeziku mogu se lako prijebaciti na različite platforme uz minimalne izmjene.

**Jednostavna sintaksa** – C koristi sintaksu koja omogućava lako razumijevanje i učenje jezika.

**Podrška za razne paradigme programiranja** – Iako je prvenstveno proceduralni jezik, C omogućava korišćenje funkcionalnog i objektno orijentisanog pristupa.

**Osnova za druge jezike** – C je direktno ili indirektno inspirisao razvoj jezika kao što su C++, Java, Python i Go.

Ove karakteristike čine C jednim od najmoćnijih i najkorisnijih programskih jezika. U nastavku ćemo detaljno analizirati različite vrste naredbi i način na koji se koriste za efikasno programiranje.

# Naredbe u jeziku C

Naredbe u jeziku C mogu se klasifikovati u nekoliko glavnih kategorija:

## Izvršne naredbe

Izvršne naredbe prijedstavljaju osnovu svakog programa u C jeziku. One omogućavaju obavljanje različitih operacija nad podacima i kontrolišu ponašanje programa. Ove naredbe uključuju:

### Aritmetičke operacije

Aritmetičke operacije su osnovne matematičke operacije koje se mogu izvoditi nad numeričkim podacima u jeziku C. Ove operacije uključuju:

**Sabiranje (+)** – Zbrajanje dvije vrijednosti.

**Oduzimanje (-)** – Oduzimanje jedne vrijednosti od druge.

**Množenje (\*)** – Množenje dvije vrijednosti.

**Dijeljenje (/)** – Dijeljenje jedne vrijednosti drugom.

**Moduo (%)** – Računanje ostatka pri cjelobrojnom dijeljenju.

Primjer:

int a = 10, b = 3;

int zbir = a + b;

int razlika = a - b;

int proizvod = a \* b;

int kolicnik = a / b;

int ostatak = a % b;

printf("Zbir: %d, Razlika: %d, Proizvod: %d, Kolicnik: %d, Ostatak: %d", zbir, razlika, proizvod, kolicnik, ostatak);

### Dodjela vrijednosti promjenjivim

Dodjela vrijednosti promenljivama u jeziku C se vrši pomoću operatora =. Prilikom dodjele moguće je koristiti i kombinovane operatore, poput **+=, -=, \*=, /=** i **%=.**

Primjer:

int x;

x = 5; // Dodjela vrijednosti 5 promjenjivoj x

x += 3; // x sada ima vrijednost 8 (x = x + 3)

### Pozivi funkcija i rad sa argumentima

Funkcije su osnovni blokovi koda koji omogućavaju modularnost i ponovnu upotrebu. Svaka funkcija može primati argumente i vraćati vrijednost.

Primjer funkcije:

int saberi(int a, int b) {

return a + b;

}

int main() {

int rezultat = saberi(10, 5);

printf("Rezultat sabiranja: %d", rezultat);

return 0;

}

### Uslovni izrazi i evaluacija logičkih operacija

Uslovni izrazi omogućavaju donošenje odluka u programu. U jeziku C koriste se **if, else if** i **else** za grananje toka izvršenja.

Primjer:

int broj = 10;

if (broj > 0) {

printf("Broj je pozitivan");

} else if (broj < 0) {

printf("Broj je negativan");

} else {

printf("Broj je nula");

}

Logičke operacije (&&, ||, !) omogućavaju kombinovanje više uslova.

Primjer:

U ovom Primjeru dodjeljujemo vrijednost 5 promjenjivoj **x** i koristimo funkciju **printf** da ispišemo njen sadržaj. Izvršne naredbe omogućavaju manipulaciju podacima i interakciju programa sa korisnikom.

## Deklarativne naredbe

Deklarativne naredbe koriste se za definisanje promjenjivih, konstanti i funkcija prije njihove upotrebe. Svaka promenljiva mora biti deklarisana prije nego što se koristi u programu, jer to omogućava kompilatoru da rezerviše odgovarajući memorijski prostor i spriječi greške pri izvršenju programa. Deklarativne naredbe takođe omogućavaju definisanje korisničkih tipova podataka, struktura i pokazivača, čime se poboljšava čitljivost i organizacija koda.

U jeziku C, deklaracije mogu uključivati različite tipove podataka:

### Cjelobrojni tipovi

Cjelobrojni tipovi koriste se za skladištenje cijelih brojeva, pri čemu različiti tipovi omogućavaju različit opseg vrijednosti:

**int** – Standardni cjelobrojni tip podrazumijevanog opsega.

**short** – Kratki cjelobrojni tip sa manjim opsegom vrijednosti.

**long** – Produženi cjelobrojni tip sa većim opsegom vrijednosti.

**long** **long** – Tip sa još većim opsegom, često korišćen za rad sa velikim brojevima.

Primjer:

int godine = 25;

short brojDana = 365;

long populacija = 7800000000;

long long atomskiBroj = 9223372036854775807;

### Brojevi sa pokretnim zarezom

Ovi tipovi omogućavaju skladištenje decimalnih vrijednosti:

**float** – Tip sa jednostrukom preciznošću.

**double** – Tip sa dvostrukom preciznošću.

**long double** – Tip sa još većom preciznošću.

Primjer:

float temperatura = 36.5;

double pi = 3.14159265359;

long double prijeciznijiPi = 3.14159265358979323846;

### Znakovni tipovi

Koriste se za čuvanje pojedinačnih karaktera:

**char** – Standardni znakovni tip koji skladišti jedan karakter.

Primjer:

char inicijal = 'A';

### Pokazivači

Pokazivači su promjenjive koje čuvaju memorijske adrese drugih promjenjivih:

**int\*** – Pokazivač na cjelobrojni tip.

**char\*** – Pokazivač na karakter.

**void\*** – Generički pokazivač koji može pokazivati na bilo koji tip podataka.

Primjer:

int broj = 10;

int\* ptr = &broj;

printf("Vrijednost broj: %d, Memorijska adresa: %p", \*ptr, ptr);

### Strukture i unije

Strukture omogućavaju skladištenje različitih tipova podataka u jednom objektu:

struct Student {

char ime[50];

int godina;

float prosjek;

};

Unije koriste zajednički memorijski prostor za više promjenjivih:

union Podaci {

int broj;

float decimala;

};

### Enumeracije

Enumeracije omogućavaju definisanje skupa imenovanih konstanti:

**enum Dani {PONEDELJAK, UTORAK, SRIJEDA, CETVRTAK, PETAK};**

**Dani danas = UTORAK;**

Pored jednostavnih tipova, moguće je deklarisati i konstante koristeći **const** i **#define** direktive:

const int MAX\_VRIJEDNOST = 100;

#define PI 3.14159

Pravilna deklaracija i organizacija promjenjivih omogućava efikasno korišćenje memorije, olakšava razumijevanje koda i povećava bezbijednost programa. Koristeći pravilne deklarativne naredbe, programeri mogu osigurati stabilan i pouzdan rad aplikacija.

# Kontrola toka u jeziku C

Kontrola toka omogućava programima da donose odluke i ponavljaju operacije na osnovu određenih uslova. Ove strukture uključuju grananja, petlje, ugnježdenje petlje I skakanje.

## Grananja

Grananja omogućavaju selektivno izvršavanje koda u zavisnosti od postavljenih uslova. Najčešće korišćene naredba za grananje su **if**, **if-else** i **switch-case**:

### If

If naredba omogućava izvršavanje određenog bloka koda samo ako je zadovoljen definisani uslov. Ako uslov nije ispunjen, taj blok koda se preskače.  
Primjer:

if (a > b) {

printf("A je veće od B");

} else {

printf("B je veće ili jednako A");

}

### If-else

Kombinacija else sa if omogućava da se izvrši jedan blok ako je uslov tačan, a drugi blok ako uslov nije tačan.  
Primjer:  
if (broj > 0) {

printf("Broj je pozitivan.\n");

} else {

printf("Broj nije pozitivan.\n");

### If-else if-else

Koristi se kada imamo više mogućih uslova. Kao što je predstavljeno u primjeru:  
if (broj > 0) {

printf("Broj je pozitivan.\n");

} else if (broj < 0) {

printf("Broj je negativan.\n");

} else {

printf("Broj je nula.\n");

}

### Switch-case

Dok je switch-case struktura koja omogućava efikasno donošenje odluka kada postoji više mogućih vrijednosti.  
Primjer:   
switch (broj) {

case 1:

printf("Izabrali ste opciju 1");

break;

case 2:

printf("Izabrali ste opciju 2");

break;

default:

printf("Nepoznata opcija");

}

## Petlje

Petlje su ključni mehanizmi u programiranju koji omogućavaju ponavljanje određenog bloka koda dok je zadovoljen određeni uslov. U jeziku C postoje tri glavne vrste petlji: **for, while** i **do-while** petlja. Svaka od ovih petlji ima specifične primjene i koristi se u zavisnosti od situacije u programu.

### For petlja

For petlja se koristi kada je poznat tačan broj iteracija unaprijed.   
  
Primjer:  
  
for (int i = 0; i < 10; i++) {

printf("Iteracija: %d\n", i);

}

U ovom primjeru, promjenjiva i počinje od 0, povećava se za 1 nakon svake iteracije i petlja se izvršava dok i ne postane 10.

### While petlja

While petlja se koristi kada nije poznat tačan broj iteracija unaprijed, već se izvršava sve dok je određeni uslov tačan.  
Primjer:  
int i = 0;

while (i < 10) {

printf("Broj: %d\n", i);

i++;

}

Ova petlja se izvršava sve dok je i manje od 10. Svaka iteracija povećava i za 1.

### Do-While petlja

Do-While petlja je slična while petlji, ali se razlikuje po tome što se blok koda uvijek izvršava barem jednom, bez obzira na uslov.  
Primjer:  
int i = 0;

do {

printf("Broj: %d\n", i);

i++;

} while (i < 10);

U ovom primjeru, petlja će se sigurno izvršiti barem jednom, čak i ako je početni uslov netačan.

## Ugnježdene petlje

Ugnježdene petlje su petlje koje se nalaze unutar drugih petlji. Ove petlje se često koriste za rad sa višedimenzionalnim strukturama podataka, kao što su matrice.

### Ugnježdena for petlja

Ovo je specifična vrsta ugnježdene petlje u kojoj su dva ili više "for" izraza povezani. Vanjska "for" petlja pokreće unutrašnju "for" petlju svaki put kada napravi jedan ciklus. Svaka petlja ima svoju kontrolu i uvjete izlaza. U primjeru možemo vidjeti kod koji prolazi kroz sve moguće kombinacije vrijednosti i i j u opsegu od 0 do 2:

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

printf("i = %d, j = %d\n", i, j);

}

}

### Ugnježdena while petlja

Slično kao ugnježdena "for" petlja, ugnježdena "while" petlja znači da unutar jednog "while" izraza postoji još jedan "while" izraz. Vanjska petlja kontroliše unutrašnju, a unutrašnja petlja se izvodi dok njen uslov bude ispunjen.  
Primjer:  
int i = 0;

while (i < 3) {

int j = 0;

while (j < 3) {

printf("i = %d, j = %d\n", i, j);

j++;

}

i++;

}

### Ugnježdena do-while petlja

Ova petlja je specifična po tome što uvijek izvršava svoj blok barem jednom, čak i ako uslov za nastavak nije odmah zadovoljen. U ugnježdenom "do-while" scenariju, unutrašnje "do-while" petlja će se izvoditi dok njen uslov bude ispunjen, a vanjska petlja nastavlja upravljati njezinim ponovnim pokretanjem.  
Primjer:  
int i = 0;

do {

int j = 0;

do {

printf("i = %d, j = %d\n", i, j);

j++;

} while (j < 3);

i++;

} while (i < 3);

## Skakanje

Skakanje predstavlja kontrolnu strukturu koja se koriste za upravljanje tokom izvršavanja petlji i drugih struktura. Neke od ovih naredbi su **break**, **continue** I  **goto.**

### Break

Naredba break izlazi iz trenutne petlje ili switch izraza.  
Primjer:  
for (int i = 0; i < 10; i++) {

if (i == 5) {

break;

}

printf("%d ", i);

}

### Continue

Naredba continue preskače preostali dio koda u trenutnoj iteraciji petlje i prelazi na sledeću iteraciju.  
Primjer:  
for (int i = 0; i < 10; i++) {

if (i == 5) {

continue;

}

printf("%d ", i);

}

### Goto

Naredba goto koristi se za skakanje na određeni dio koda, ali se ne prijeporučuje jer može otežati praćenje toka programa.  
Primjer:  
int i = 0;

start:

printf("%d ", i);

i++;

if (i < 5) goto start;

# Zaključak

Jezik C je jedan od najvažnijih i najkorišćenijih programskih jezika zbog svoje jednostavnosti, efikasnosti i mogućnosti rada sa sistemskim resursima. Njegove glavne karakteristike uključuju rad sa različitim tipovima podataka (cijeli brojevi, brojevi sa pokretnim zarezom, znakovi), pokazivače, strukture, unije i enumeracije. Posebno je moćan zbog sposobnosti upravljanja memorijom na niskom nivou, što ga čini idealnim za razvoj brzih i efikasnih softverskih sistema.

Osnovne naredbe u C jeziku uključuju aritmetičke operacije, dodelu vrijednosti promjenljivama i pozive funkcija. Uslovni izrazi (if, else if, switch-case) omogućavaju donošenje odluka u programu, dok petlje (for, while, do-while) omogućavaju ponavljanje određenih operacija. Pravilno korišćenje ovih elemenata ključno je za pisanje efikasnih programa.

Kontrola toka programa određuje redosled izvršavanja naredbi. Grananja i petlje omogućavaju fleksibilnost i bolje upravljanje kodom. Dodatne naredbe poput break i continue pomažu u upravljanju izvršavanjem petlji, dok se goto rijetko preporučuje jer može otežati čitljivost koda.

Jezik C pruža velike mogućnosti, ali zahtijeva pažljivo planiranje i dobro razumijevanje kontrole toka i upravljanja podacima. Programeri koji ovladaju ovim konceptima mogu pisati optimizovane i moćne programe, koristeći puni potencijal ovog jezika.

**Literatura**

*Kernighan, B. W., & Ritchie, D. M. (1988). The C Programming Language. Prijentice Hall.*

*Harbison, S. P., & Steele, G. L. (1995). C: A Reference Manual. Prijentice Hall.*