PROGRAMIRANJE

PROGRAMSKI JEZIK C

Naredbe
C jezika

Naredbe C jezika

- Relativno mali broj naredbi (za razliku od operatora)
- Naredba izraza: neka je e izraz, tada je
  - e; naredba izraza
  - Ako se izraz izostavi, dobija se prazna naredba ;
- Blok naredba
  - Naredbe se grupišu i tretiraju kao jedna naredba
  - Nula ili više naredbi između {,} je blok naredba

```
{
        naredba
        naredba
        …
}
```

Deklaracije promenljivih se navode
na početku bloka, pre ostalih naredbi bloka

  - Blok naredba može biti ugnježdena (naredba blok naredbe je i sama blok naredba)

```
{
        Naredba
        …
        {
                …
        }
        …
        naredba
}
```

- Blok naredba se tretira kao jedna naredba
- Telo funkcije je blok naredba
- Blok naredba određuje oblast važenja pojedinih identifikatora

Upravljačke naredbe

- Redosled izvršavanja naredbi programa je sekvencijalan
  - Izvršavaju se po redosledu navođenja u programskom tekstu
- Ovo se može izmeniti upravljačkim naredbama
  - Naredbe selekcije (grananja)
  - Naredbe ciklusa (petlji, iteracija)
  - Naredbe skoka

Naredbe selekcije

- Uslovno izvršavanje neke naredbe
  - Redosled izvršavanja zavisi od nekog uslova
  - Uslov je logicki izraz
- Logički izraz
  - Izraz koji ima jednu od dve logičke vrednosti tačan odnosno netačan
  - Mogu se formirati
    - Uobičajenim binarnim relacionim operatorima <, >, <=, >=
    - Jednakosnim operatorima ==, !=
    - Logičkim operatorima !, &&, | |

Logički izrazi

- Relacioni operatori <, >, <=, >= za operande mogu imati aritmetičke izraze i kao rezultat daju 1(tačno) odnosno 0(netačno)
  - Nižeg su prioriteta od aritmetičkih

    i+j<k-1 ⟶ (i+j)<(k-1)
  - Levo su asocijativni

    i<j<k ⟶ (i<j)<k ⟵ Nema očekivanu interpretaciju $j \in (i,k)$
- Jednakosni operatori ==, != upoređuju vrednosti svojih peranada i kao vrednost daju 1(tačno) odnosno 0(netačno)
  - Nižeg su prioriteta od relacionih operatora i levo su asocijativni

    i<j==j<k ⟶ (i<j)==(j<k)
  - Trik

    (x>0)+(x>=0)-1 ⟷ $\mathrm{sgn}(x)$

Logički operatori

- Složeni logički izrazi se dobijaju primenom logičkih operatora
  - !: negacija (¬), unarni operator
  - &&: konjukcija (∧), binarni operator
  - ||: disjunkcija (∨), binarni operator
  - Operandi ovih operatora su logičke vrednosti 1(tačno) i 0(netačno) mada to i nije obavezno
  - U C jeziku svaka ne-nula vrednost se tretira kao tačan iskaz (1), a nula vrednost kao netačan iskaz (0)
  - Binarni operatori && i || su takozvani short-circuit operatori (kratko spojeni)
    - Prvo se izračunava vrednost levog operanda
    - ako se iz vrednosti levog operanda može izvesti vrednost celog izraza vrednost desnog operand se ne izračunava.

0 && a → 0

1 || a → 1

(i!=0) && (i/i>0) ← Bezbedno (nema deljanja sa 0)

if naredba

- Selektuje se jedna od dve alternative u zavisnosti od vrednosti logičkog izraza

```
if (izraz) naredba
```

  - Ako je vrednost izraza tačno (ne-nula) izvršiće se naredba u protivnom neće

```
if (line==MAX_LINES) line=0;
```

  - Pod kontrolom if naredbe je samo jedna naredba (prva koja sledi) koja može biti blok naredba (grupa naredbi koja se tretira kao jedna naredba)

```
if (line==MAX_LINES) {
    line=0;
    page++;
}
```

  - Skraćenje

```
If(e==0) → if(!e)
If(e!=0) → if(e)
```

Česta greška

```
if (i=0) umesto if (i==0)
If (i<j<k)  umesto if (i<j && j<k)
```

else klauzula

- if naredba može sadržati i else klauzulu

```
if (izraz) naredba
else naredba
```

  - Ako je vrednost izraza 0(netačno) izvršiće se naredba koja sledi iza službene (ključne) reči else

```
if (i>j)  max=i;
else max=j;
```

```
if (i>j) {
    max=i;
} else {
    max=j;
}
```

- Slično kao i sa zagradama u izrazima, dodavanje vitičastih zagrada (i kada to nije neophodno)
  - doprinosi čitljivosti programa
  - takođe se izbegava mogućnost greške pri kasnijem dodavanju naredbi u nekoj od sekcija (if odnosno else)

Kaskadno if

```
if (n<0)
    printf("Manje od nule\n");
else
        if (n==0)
            printf("Jednako nuli\n");
        else
            printf("Vece od nule\n");
```



```
if (n<0)
    printf("Manje od nule\n");
else if (n==0)
    printf("Jednako nuli\n");
else
    printf("Vece od nule\n");
```

- Na ovaj način dobijamo kaskadnu if naredbu koja je čitljivija i lakša za pisanje nego niz uvučenih if odnosno else sekcija

```
if (izraz) naredba
else if (izraz) naredba
....
else if (izraz) naredba
else naredba
```

  - Poslednja else sekcija nije obavezna (kao kod jednostavne if naredbe)

Viseće else

- U slucaju ugnježdenih if naredbi

```
if (y!=0)
    if (x!=0)
        result=x/y;
else
    printf("y je jednako nuli\n");
```

Kojoj if sekciji odgovara (viseće) else?

```
if (y!=0)
    if (x!=0)
        result=x/y;
    else
        printf("y je jednako nuli\n");
```

else sekcija se vezuje za poslednje if koje nije upareno (bez obziru na uvlačenje)

```
if (y!=0) {
    if (x!=0)
        result=x/y;
} else
    printf("y je jednako nuli\n");
```

Ovo podrazumevano uparivanje se može izbeći blok naredbom

Uslovni izraz

- U C jeziku postoji ternarni (zahteva tri operanda) uslovni operator ?:

```
izraz1 ? izraz2 : izraz3
```

  - vrednost odgovarajućeg izraza je neka od ponuđene dve vrednosti i zavisi od vrednosti nekog izraza
  - Prvo se izračunava vrednost prvog izraza izraz1 ako je tačan (ne-nula vrednost) izračunava se vrednost drugog izraza izraz2 i to je vrednost uslovnog izraza, u suprotnom izračunava se vrednost trećeg izraza izraz3 i to je vrednost uslovnog izraza

- Obično su komplikovani za razumevanje, ali u nekim situacijama su pogodni

```
if (i>j)
    printf("%d\n", i);
else
    printf("%d\n", j);
```

```
printf("%d\n", i>j?i:j);
```

- izraz2 i izraz3 mogu biti celobrojni, razlomljenii ili njihova kombinacija i u tom slučaju je vrednost uslovnog izraza razlomljena

```
int i=1;
float j=2.f;
(i>0)?i:j
```

vrednost je 1.f

Logički tip podataka

- C jezik nema ugrađen logički tip podataka
  - To se može obezbediti na sledeći način:

```
#define TRUE 1
#define FALSE 0
…
int flag;
…
flag=FALSE;
…
flag =TRUE;
…
if(flag==TRUE)
…
if(flag==FALSE)
```

```
#define BOOL int
…..
BOOL flag;
```

ili samo if(flag)

ili samo if(!flag)

switch naredba

- Često se javlja sledeća situacija. U zavisnosti od vrednosti izraza (koja ne mora biti logička) treba izvršiti određenu akciju

```
if (ocena==5) printf("Odlican\n");
else if (ocena==4) printf("Vrlo dobar\n");
else if (ocena==3) printf("Dobar\n");
else if (ocena==2) printf("Dovoljan\n");
else if (ocena==1) printf("Nedovoljan\n");
else printf("Pogresna ocena\n");
```

```
switch(ocena) {
case 5: printf("Odlican\n");
        break;
case 4: printf("Vrlo dobar\n");
        break;
case 3: printf("Dobar\n");
        break;
case 2: printf("Dovoljan\n");
        break;
case 1: printf("Nedovoljan\n");
        break;
default: printf("Pogresna ocena\n");
        break;
}
```

switch naredba

- Struktura switch naredbe je sledeća

```
switch(izraz) {
case konstantni_izraz: naredbe
…
case konstantni_izraz: naredbe
default: naredbe
}
```

  - Pri čemu je:
    - izraz: celobrojni izraz
    - konstantni_izraz: je celobrojni izraz koji ne sadrži promenljive ili pozive funkcija npr. 5+10 jeste konstantni izraz dok 2+n nije (sem ako n nije simbolička konstanta)
    - naredbe: proizvoljan broj naredbi, koje ne moraju biti u bloku (vitičaste zagrade nisu neophodne). Obično je poslednja naredba u nizu, naredba break kojom se prekida izvršavanje switch naredbe.

switch naredba

- Izračunava se vrednost izraza izraz, a zatim se upoređuje sa konstantnim izrazima u case sekcijama
  - Ako je izračunata vrednost jednaka nekoj od ponuđenih izvršavaju se naredbe koje slede odgovarajuću case sekciju kao i sve naredbe koje slede posle date grupe naredbi
  - Ako izračunata vrednost nije jednaka nijednoj od ponuđenih izvršava se grupa naredbi koja odgovara default sekciji.
- default sekcija nije neophodna i u tom slučaju ako vrednost izraza nije jednak nijednoj od ponuđenih prekida se izvršavanje switch naredbe
- Redosled case izraza nije bitan i pri tome nisu dozvoljeni duplikati
  - Samo jedan konstantni izraz moze slediti iza ključne reči case
  - Sa druge strane više case izraza moze prethoditi (odgovarati) jednoj grupi naredbi. Na ovaj način se moze realizovati case sekcija sa grupom vrednosti

```
switch(ocena) {
   case 5:
   case 4:
   case 3:
   case 2: printf("Polozio\n");
           break;
   case 1: printf("Nije polozio\n");
           break;
   default: printf("Pogresna ocena\n");
}
```

Poslednja sekcija u switch naredbi ne mora sadržati break naredbu (efekat je isti).

break naredba

- break naredba između ostalog prekida izvršavanje switch naredbe
  - Prelazi se na izršavanje naeredbe koja sledi posle naredbe switch
- break naredba je naredba skoka
- Postoje situacije kada se izostavlja iz odgovarajuće sekcije switch naredbe

```
switch(ocena) {
    case 5:
    case 4:
    case 3:
    case 2: broj_polozili++;
            /* izostavlja se break naredba idi dalje*/
    case 1: ukupno_ocena++;
            break;
    default: printf("Pogresna ocena\n");
             break;
}
```

Naredbe ciklusa

- Ovim naredbama je moguće ponavljati izvršavanje neke druge naredbe (telo ciklusa)
- Svaki ciklus ima takozvani kontrolni izraz
  - Kada se izvrši telo ciklusa izračunava se vrednost izraza
    - ako je njegova logička vrednost tačno (ne-nula vednost) nastavlja se izvršavanje tela ciklusa
    - ako je njegova logička vrednost tačno (vednost nula) prekida se izvršavanje naredbe ciklusa i prelazi se na naredbu koja sledi
- U C jeziku postoje tri naredbe ciklusa
  - while
  - do
  - for

while naredba

- while(izraz) naredba
  - Izraz unutar zagrada je kontrolni izraz, a naredba koja sledi je telo ciklusa.
  - Izračunava se vrednost izraza izraz, ako je tačan (ne-nula vrednost) izvršava se telo ciklusa
  - Proces se dalje nastavlja na isti način, sve dok izračunata vrednost ne bude netačna (nula vrednost)

```
while (1) naredba
```

Beskonacna petlja, sem ako telo ciklusa ne sadrži naredbu skoka (break, …) ili poziv funkcije koja prekida izvršavanje programa

```
#include <stdio.h>
main() {
    int i, n;
    scanf("%d", &n);
    i=1;
    while(i<=n) {
        printf("%10d %10d\n", i, i*i);
        i++;
    }
}
```

```
#include <stdio.h>
main() {
        int n, suma=0;
        printf("Unesite ceo broj (0 kraj unosa): ");
        scanf("%d", &n);
        while(n!=0) {/* ili while (n) */
                suma+=n;
                scanf("%d", &n);
        }
}
```

do naredba

- do naredba ima ima isti efekat kao i while naredba samo što se kontrolni izraz testira posle izvrsavanja tela ciklusa.

  do naredba while(izraz);

- Izvršava se telo ciklusa nakon toga se izračunava kontrolni izraz
  - ako je izračunata vrednost tačan (ne-nula vrednost) postupak se ponavlja
  - ako je izračunata vrednost tačan (vrednost nula) prekida se izvršavanje do naredbe.

```
#include <stdio.h>
main() {
   int broj_cifara=0, n;
   scanf("%d", &n);
   do {
      n/=10;
      broj_cifara++;
   } while(n>0);
   printf("Broj cifara broja %d je %d\n", n, broj_cifara);
}
```

```
while(n>0) {
   n/=10;
   broj_cifara++;
}
```

U slučaju da n ima vrednost 0
broj_cifara ← 0?

for naredba

- Posebno pogodna za formiranje brojačkih ciklusa, ali se moze koristiti i za formiranje drugih tipova ciklusa

```
for(izraz1; izraz2; izraz3) naredba
```

  - izraz1: izraz koji se izračunava samo jednom (obicno inicijalizacija brojačke promenljive)
  - izraz2: kontrolni izraz
    - izračunava se pre izršavanja tela ciklusa
    - ponavlja se izvršavanje tela ciklusa sve dok je njegova vrednost tačno (ne-nula vrednost)
  - izraz3: izraz koji se izvršava nakon svakog izvršavanja tela ciklusa (obično inkrementacija (dekrementacija) brojačke promenljive)

```
for(i=0; i<n; i++) naredba
for(i=n; i>0; i--) naredba
```

```
izraz1;
 while(izraz2) {
        naredba
        izraz3;
}
```

Može se opisati while naredbom (ne uvek)

for naredba

- Neki od izraza: izraz1, izraz2, izraz3 se mogu izostaviti

- inicijalizacija odnosno izraz izraz1 se izračunava pre for naredbe

```
i=10;
for(; i>0; i--)
…
```

- Izraz izraz3 (koji obezbeđuje odgovarajuću promenu brojačke promenljive) se ugrađuje u telu ciklusa

```
for(i=10; i>0;)
    printf("i=%d\n", i--);
```

- Ako se izostave izraz1 i izraz3 dobićemo while naredbu

```
for(;izraz2;) naredba  ⟶  while(izraz2) naredba
```

- Ako se izostavi izraz2, podrazumeva se da je uvek tačan
  - for petlja se nikad ne završava (beskonačna petlja), sem ako se to ne izvede na neki drugi način unutar tela ciklusa

```
for(; ;) naredba  ⟵  Beskonačna petlja
```

for naredba

- U standardu C99 brojačka promenljiva se moze deklarisati u okviru for petlje

```
for(int i=0; i<n; i++)
```

  - Tako uvedena promenljiva je dostupna samo u okviru tela ciklusa
  - Ako je potrebno da se promenljiva koristi i van tela ciklusa onda se to mora učiniti pre for naredbe (na početku odgovarajućeg bloka)
  - Slično moguće je u okviru for petlje deklarisati više promenljivih pod uslovom da su istog tipa

```
for(int i=0, j=n-1; i<n; i++, j--)
```

Operator ,

- Nekada je npr. u okviru izraza za inicijalizaciju u for naredbi neophodno inicijalizovati dve (ili više) brojačke promenljive. To se može izvesti operatorom ,

  izraz1, izraz2

  - Vrednost odgovarajućeg izraza se izračunava na sledeći nacin
    - Prvo se izračunava izraz izraz1 i njegova vrednost se odbacuje
    - Zatim se izračunava vrednost izraza izraz2 i to je konačna vrednost izraza
    - Ako izraz izraz1 nema bočni efekat onda on i nema mnogo smisla
  - , je binarni operator najnižeg prioriteta

```
int i, a=1, b=2;
i=(a, b);
i=a,b;
```

ovde je , separator

i će dobiti vrednost od b, vrednost od a se odbacuje

i ce dobiti vrednost od a (isto kao (i=a), b)

Operator ,

- Operator , se koristi u situacijama kada se zahteva jedan izraz, a neophodno je ugraditi više izraza

```
/* tabela kvadrata  (n+1)*(n+1)=n*n+2n+1*/
for(i=1, kvad=1, nep=1; i<=n; i++) {
    printf("%10d %10d\n", i, kvad)
    nep+=2, kvad+=nep;
}
```

ovo može u izraz3 (zajedno sa i++)

- Dalje može se koristiti u okviru kontrolnog izraza u if, while, do, for

```
if(y=f(x), y>x) ...
```

- Ili da bi se izbegao blok

```
if(x == 1) {
    y = 2;
    z = 3;
}
```

```
if(x == 1) y = 2, z = 3;
```

Izlaz iz ciklusa

- Izlaz iz ciklusa može biti
  - na početku (for, while) ili
  - na kraju (do)
- Nekad je neophodno obezbediti izlaz iz ciklusa i unutar samog ciklusa
- break naredba ovo omogućava ovo u bilo kom tipu ciklusa (uključujući i switch naredbu)

break naredba

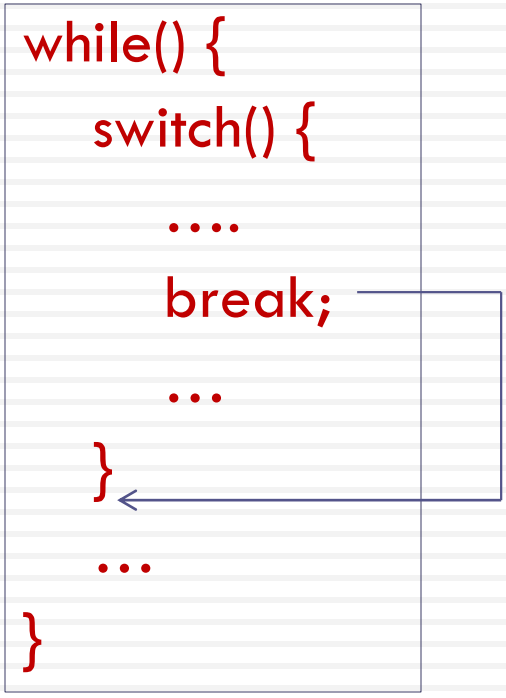
- break naredba
  - prekida izvršavanje switch naredbe
  - prekida izršavanje naredbe ciklusa (izlaz iz ciklusa) while, do, for

```
for(d=2; d<n; d++)
    if(n%d==0) break;
if(d<n)
    printf("%d je deljiv sa %d\n", n, d);
else
    printf("%d je prost broj\n", n);
```

```
for(; ;) {
    printf("Unesite broj (0 za kraj): ");
    scanf("%d", &n);
    if(!n) break;
    …
}
```

break naredba

- U slučaju ugnježdenih naredbi ciklusa, switch naredbi
  - break naredba prekida izvršavanje poslednje započete naredbe ciklusa odnosno switch naredbe



continue naredba

- break naredba izvršavanje programa prenosi u tačku posle samog kraja odgovarajuće naredbe ciklusa odnosno switch naredbe
- continue naredba prenosi izvršavanje u tačku pre samog kraja tela ciklusa
  - može se koristiti samo unutar naredbi ciklusa (while, do, for)

```
/* suma 10 ne-nula brojeva */
int n=0, suma=0;
while(n<10) {
    scanf("%d", &i);
    if(!i) continue;
    sum+=i;
    n++;
}
```

continue naredba

```
for(izraz1; izraz2; izraz3) naredba
```



```
izraz1;
 while(izraz2) {
            naredba
            izraz3;
}
```

- gornja realizacije for naredbe preko while naredbe nije adekvatna ako u telu for naredbe postoji naredba continue

```
for(izraz1; izraz2; izraz3) {
    …
    continue;
    …
}
```

U for naredbi se izraz3 uvek izračunava

```
izraz1;
while(izraz2) {
    …
    continue;
    …
    izraz3;
}
```

U realizaciji preko while naredbe izraz3 se neće izračunavati ako se izvrši naredba continue

goto naredba

- break naredba prenosi izvršavanje programa u tačku posle samog kraja tela switch, while, do odnosno for naredbe
- continue naredba prenosi izvršavanje programa u tačku pre samog kraja tela while, do odnosno for ciklusa
- goto naredba prenosi izvršavanje na proizvoljnu naredbu (u okviru iste funkcije) koja ima labelu (obeležje)
- Labela naredbe je proizvoljni identifikator koji prethodi samoj naredbi

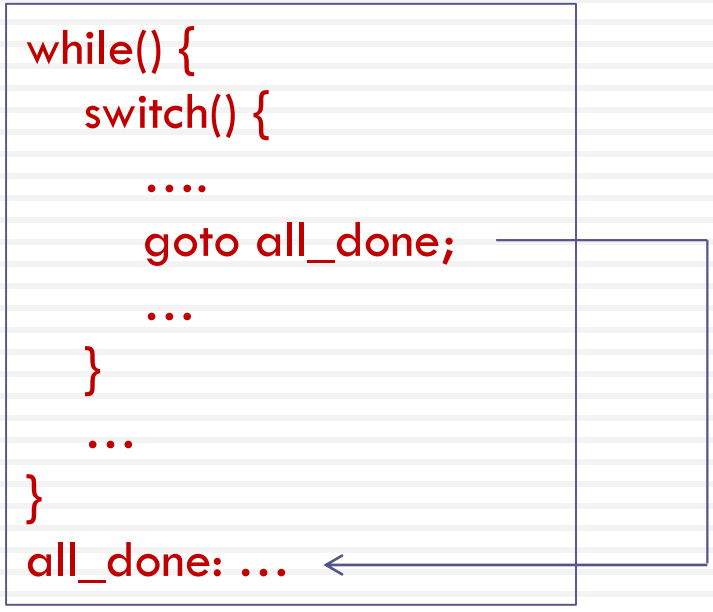
identifikator: naredba

obeležje (labela) naredbe

```
for(d=2; d<n; d++)
    if(n%d==0) goto done;
done:
if(d<n)
    printf("%d je deljiv sa %d\n", n, d);
else
    printf("%d je prost broj\n", n);
```

goto naredba

- U slučaju ugnježdenih naredbi ciklusa, switch naredbi
  - break naredba prekida izvršavanje poslednjeg započetog ciklusa, odnosno switch naredbe
  - goto naredbom se mogu prekinuti izvršavanja svih ugnježdenih naredbi ciklusa, switch naredbi



Prazna naredba

- Sastoji samo od znaka ;
- Ne proizvodi nikakvu akciju
- Prazna naredba se obično koristi u situacijama kada sintaksa jezika zahteva na nekom mestu naredbu ali u datoj situaciji nije potrebno izvršiti nikakvu akciju.
  - To je neophodno kod naredbe ciklusa koja nema telo ciklusa

```
for(d=2; d<n; d++)
    if(n%d==0) break;
```



```
for(d=2; d<n && n%d; d++) ;
```

  - mada se to može izvesti naredbom continue ili praznim blokom

```
for(d=2; d<n && n%d; d++)  continue;
```

```
for(d=2; d<n && n%d; d++)  {}
```

- Slično prazna naredba je neophodna ako se želi preneti izvršavanje programa na sam kraj bloka

```
{
    …
    goto end_statement;
    …
    end_statement: ;
}
```

Prioritet operatora

Operator Type	Operator	Associativity
Primary Expression Operators	() [] *expr*++ *expr*--	left-to-right
Unary Operators	+ - ! ++*expr* --*expr* (*typecast*) sizeof	right-to-left
Binary Operators	* / %	left-to-right
	+ -	
	< > <= >=	
	== !=	
	&&	
	||	
Ternary Operator	?:	right-to-left
Assignment Operators	= += -= *= /= %=	right-to-left
Comma	,	left-to-right