Uvod u programiranje

- predavanja -

listopad 2020.

Kontrola toka programa

- 1. dio -

Kontrola toka programa, kontrolne strukture

- jedan od najvažnijih aspekata programiranja jest mogućnost upravljanja redoslijedom izvršavanja naredbi (*flow control*): propisivanje koja naredba će se izvršiti u sljedećem koraku izvršavanja programa. U tu svrhu koriste se kontrolne programske strukture (*programming control structures*)
  - selekcija
    - jednostrana (*if ... then ...*)
    - dvostrana (*if ... then ... else ...*)
    - skretnica (*case ... then ..., case ... then ..., case ... then ..., ...*)
  - petlja
    - ponavljanje naredbi dok je zadovoljen uvjet (*condition-controlled*)
    - ponavljanje naredbi određeni broj puta (*count-controlled*)

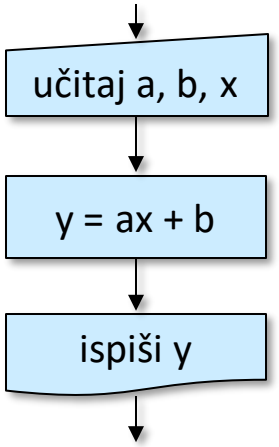
Niz ili sekvenca

- niz naredbi koje se izvršavaju točno jedna iza druge, redom kako su napisane

Pseudo-kod

```
...
učitaj(a, b, x)
y := ax + b
ispiši(y)
...
```

Dijagram toka



C program

```
...
scanf("%f %f %f", &a, &b, &x);
y = a * x + b;
printf("%f", y);
...
```

Selekcija

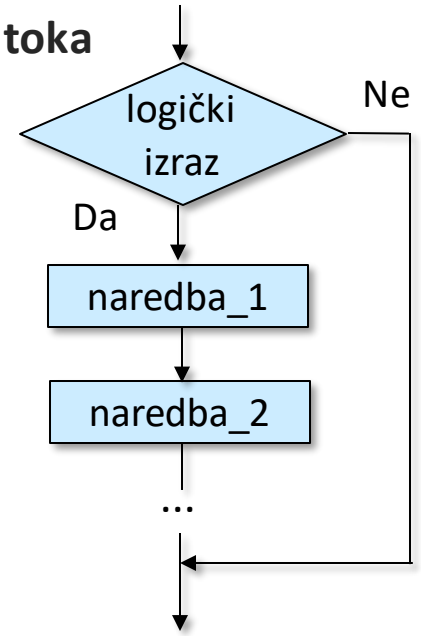
Jednostrana selekcija

Jednostrana selekcija - if

Pseudo-kod

```
...
ako je logički_izraz tada
   | naredba_1
   | naredba_2
   | ...
...
```

Dijagram toka



C program - sintaksa

```
if (logički_izraz)
   naredba;
```

jedna naredba!

C program - primjer

```
...
if (ocjena > 1)
   printf("Ispit je polozen!");
...
```

Što ako u slučaju, kada je ocjena pozitivna, treba izvršiti više od jedne naredbe?

Rješenje: koristiti složenu naredbu

Složena naredba (*compound statement*)

- **problem:** u naredbi if dopušteno je navođenje samo jedne naredbe koja će se izvršiti kada je rezultat logičkog izraza istina. Što učiniti ako je potrebno izvršiti više od jedne naredbe?
- **rješenje:** koristiti složenu naredbu (blok, blok naredbi)
  - jedna ili više naredbi omeđenih vitičastim zagradama
  - može se koristiti na mjestima gdje je prema sintaksi predviđena samo jedna naredba (u naredbama if, while, ...)
  - složena naredba se nikad ne terminira znakom ;

Sintaksa složene naredbe

```
{
   naredba_1;
   naredba_2;
   ...
}
```

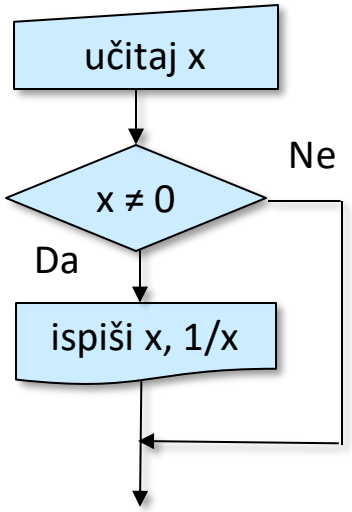
Primjer korištenja složene naredbe

```
...
if (ocjena > 1) {
   printf("Ispit je polozen.");
   brojPol = brojPol + 1;
}
...
```

Primjer

- Programski zadatak
  - u varijablu x s tipkovnice učitati realni broj. Ako je recipročna vrijednost za x definirana, na zaslon ispisati vrijednost varijable x i njezinu recipročnu vrijednost
- Pseudo-kod i dijagram toka

```
učitaj(x)
ako je x ≠ 0 tada
   ispiši(x, 1/x)
```



Primjer: C program

```c
#include <stdio.h>

int main(void) {
   float x;

   scanf("%f", &x);
   if (x != 0)
      printf("%f %f", x, 1/x);
   return 0;
}
```

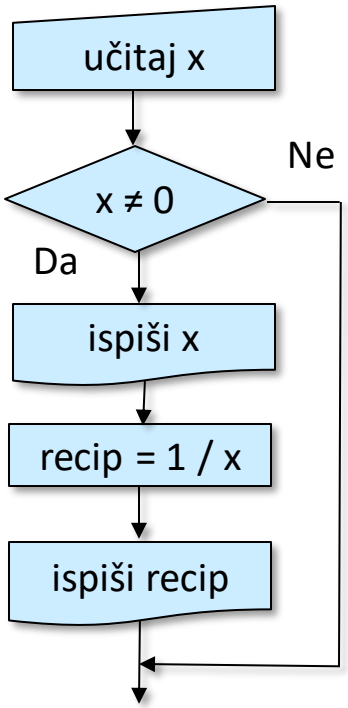
```c
#include <stdio.h>

int main(void) {
   float x;

   scanf("%f", &x);
   if (x != 0) {
      printf("%f %f", x, 1/x);
   }
   return 0;
}
```

- obje varijante su ispravne, međutim
  - iako se radi o samo jednoj naredbi koju treba obaviti kada je uvjet zadovoljen, **preporuča se** koristiti oblik složene naredbe, tj. naredbu omeđiti vitičastim zagradama
    - smanjuje se mogućnost logičke pogreške koja može nastati nepažljivim prepravljanjem programa, kao što je prikazano u sljedećem primjeru

Primjer

- Programski zadatak vrlo sličan prethodnom
  - u varijablu x s tipkovnice učitati realni broj. Ako je recipročna vrijednost za x definirana, na zaslon ispisati sadržaj varijable x, nakon toga izračunati i zatim ispisati recipročnu vrijednost
- Pseudo-kod i dijagram toka

```
učitaj(x)
ako je x ≠ 0 tada
  | ispiši(x)
  | recip := 1 / x
  | ispiši(recip)
```



- Dvije nove varijante programa pokušat će se napisati prepravljanjem dvaju programa iz prethodnog primjera

Primjer: C program

```c
#include <stdio.h>

int main(void) {
   float x, recip;

   scanf("%f", &x);

   if (x != 0)
      printf("%f", x);
      recip = 1 / x;
      printf(" %f", recip);
   return 0;
}
```

```c
#include <stdio.h>

int main(void) {
   float x, recip;

   scanf("%f", &x);

   if (x != 0) {
      printf("%f", x);
      recip = 1 / x;
      printf(" %f", recip);
   }
   return 0;
}
```

- program na lijevoj strani je neispravan! Prepravljajući lijevu varijantu programa iz prethodnog primjera programer je previdio da je trebalo dodati vitičaste zagrade i time napravio teško uočljivu logičku pogrešku

Primjeri

- U nekim slučajevima, radi kompaktnosti i preglednosti programa, prikladno je za jednu naredbu ne koristiti vitičaste zagrade . Npr.

```
// varijablu a postavi na njenu apsolutnu vrijednost
if (a < 0) a = -1 * a;
```

- Oprez!

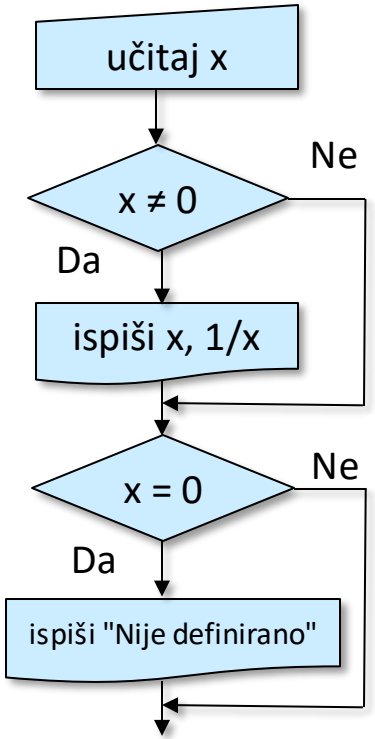
```
if (a < 0); a = -1 * a;
```

  - C prevodilac ovo smatra (sintaktički) ispravnim. "Ništa" terminirano znakom `;` naziva se nul-naredba (*null-statement*). Sasvim očekivano, nul-naredba ne obavlja nikakvu akciju
    - analizirati posljedice ove logičke greške

Primjer

- Programski zadatak
  - u varijablu x s tipkovnice učitati realni broj. Ako je recipročna vrijednost za x definirana, na zaslon ispisati vrijednost varijable x i njezinu recipročnu vrijednost, inače ispisati poruku "Nije definirano"
- **Loš** pseudo-kod i **loš** dijagram toka

```
učitaj(x)
ako je x ≠ 0 tada
   ispiši(x, 1/x)
ako je x = 0 tada
   ispiši("Nije definirano)
```



Selekcija

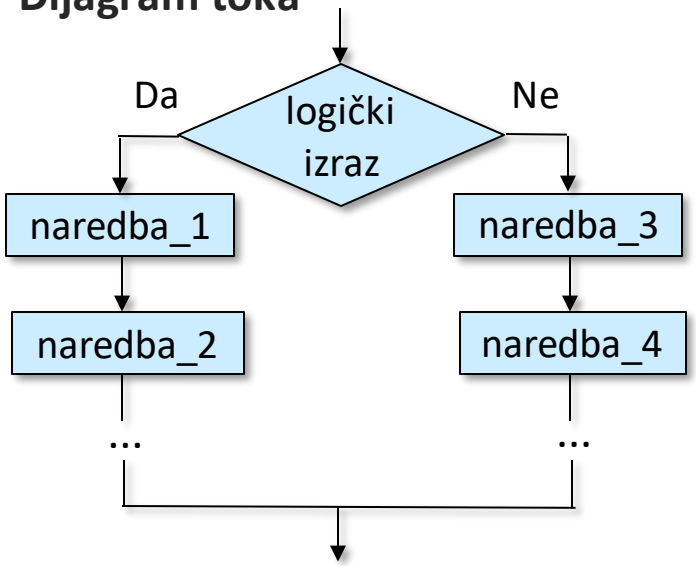
Dvostrana

Dvostrana selekcija - if - else

Pseudo-kod

```
...
ako je logički_izraz tada
   | naredba_1
   | naredba_2
   | ...
inače
   | naredba_3
   | naredba_4
   | ...
...
```

Dijagram toka



C program - sintaksa

```
if (logički_izraz)
   naredba_1;    jedna naredba!
else
   naredba_2;    jedna naredba!
```

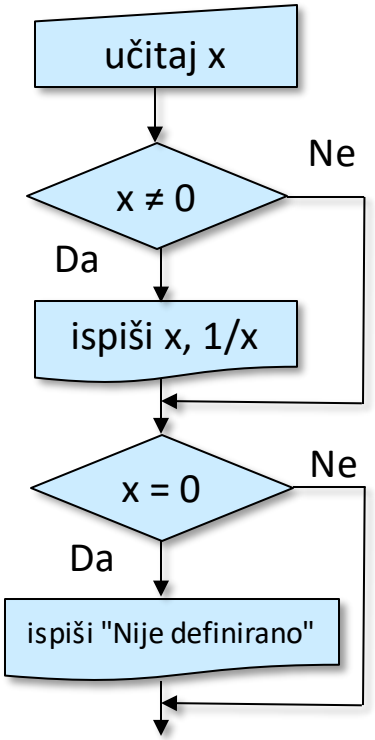
C program - primjer

```
if (ocjena > 1)
   printf("Ispit je polozen!");
else
   printf("Ispit nije polozen!");
```

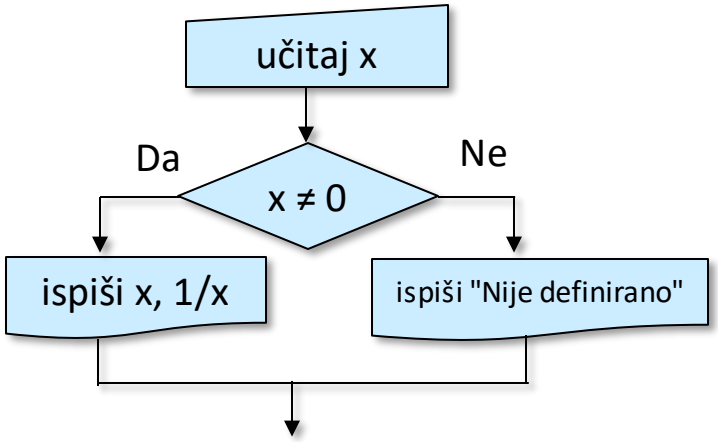
Primjer

- Ispravak lošeg rješenja za izračunavanje recipročne vrijednosti

Loše



Dobro




```
if (x != 0)
   printf("%f %f", x, 1/x);
else
   printf("Nije definirano");
```

Primjer

- Programski zadatak
  - Na zaslon ispisati poruku `Upisite cijeli broj >`
  - Učitati cijeli broj s tipkovnice
  - Izračunati apsolutnu vrijednost učitanog broja te na zaslon ispisati učitani broj i njegovu apsolutnu vrijednost
  - Primjer izvršavanja programa

```
Upisite cijeli broj > -47↵
Apsolutna vrijednost od -47 je 47↵
```

Rješenje (loše!)

```
#include <stdio.h>

int main(void) {
   int broj, aps;

   printf("Upisite cijeli broj > ");
   scanf("%d", &broj);

   if (broj < 0) {
      aps = -1 * broj;                 // aps = -broj;
      printf("Apsolutna vrijednost od %d je %d\n", broj, aps);
   } else {
      aps = broj;
      printf("Apsolutna vrijednost od %d je %d\n", broj, aps);
   }
   return 0;
}
```

Rješenje (dobro!)

```
#include <stdio.h>

int main(void) {
   int broj, aps;

   printf("Upisite cijeli broj > ");
   scanf("%d", &broj);

   if (broj < 0) {
      aps = -1 * broj;              // aps = -broj;
   } else {
      aps = broj;
   }

   printf("Apsolutna vrijednost od %d je %d\n", broj, aps);
   return 0;
}
```

Rješenje (također dobro!)

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
   int broj;

   printf("Upisite cijeli broj > ");
   scanf("%d", &broj);
   printf("Apsolutna vrijednost od %d je %d\n", broj, abs(broj));
   return 0;
}
```

Primjer

- Programski zadatak
  - Na zaslon ispisati poruku `Upisite cijeli broj >`
  - Učitati cijeli broj s tipkovnice i ovisno o učitanoj vrijednosti ispisati jednu, obje ili niti jednu od sljedećih poruka:
    - Broj je pozitivan
    - Broj je paran
- Primjeri izvršavanja programa

```
Upisite cijeli broj > 12↵
Broj je pozitivan↵
Broj je paran↵
```

```
Upisite cijeli broj > 13↵
Broj je pozitivan↵
```

```
Upisite cijeli broj > -12↵
Broj je paran↵
```

```
Upisite cijeli broj > 0↵
Broj je paran↵
```

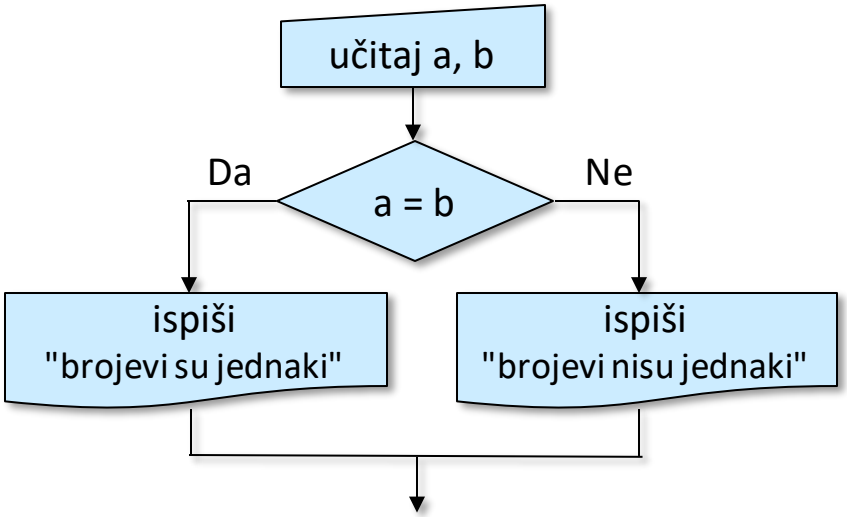
```
Upisite cijeli broj > -47↵
```

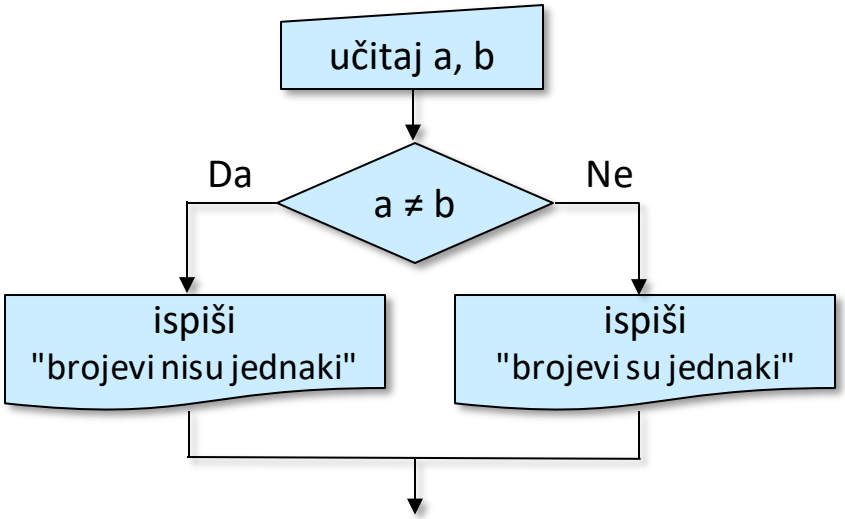
Rješenje

```
#include <stdio.h>

int main(void) {
   int broj;

   printf("Upisite cijeli broj > ");
   scanf("%d", &broj);

   if (broj > 0) {
      printf("Broj je pozitivan\n");
   }

   if (broj % 2 == 0) {
      printf("Broj je paran\n");
   }
   return 0;
}
```

Primjer

- Programski zadatak
  - S tipkovnice učitati dva cijela broja. Ovisno o učitanim vrijednostima, na zaslon ispisati ili poruku "brojevi su jednaki" ili poruku "brojevi nisu jednaki"





  - uočiti: rješenja su jednako vrijedna
  - za vježbu: napisati C programe za oba dijagrama

Primjer

- Programski zadatak
  - s tipkovnice učitati tri različita cijela broja (nije potrebno kontrolirati jesu li brojevi ispravno upisani)
  - na zaslon ispisati najveću učitanu vrijednost
  - primjer izvršavanja programa

```
Upisite tri razlicita cijela broja > 1 17 -2↵
Najveci broj je 17↵
```

Rješenje (1. varijanta)

```
#include <stdio.h>

int main(void) {
   int x, y, z, rez;
   printf("Upisite tri razlicita cijela broja > ");
   scanf("%d %d %d", &x, &y, &z);
   if (x > y) {
      if (x > z) {
         rez = x;
      } else {
         rez = z;
      }
   } else {
      if (y > z) {
         rez = y;
      } else {
         rez = z;
      }
   }
   printf("Najveci broj je %d\n", rez);
   return 0;
}
```

Rješenje (2. varijanta)

```
#include <stdio.h>

int main(void) {
   int x, y, z, rez;
   printf("Upisite tri razlicita cijela broja > ");
   scanf("%d %d %d", &x, &y, &z);
   rez = x;
   if (y > rez)
      rez = y;
   if (z > rez)
      rez = z;
   printf("Najveci broj je %d\n", rez);
   return 0;
}
```

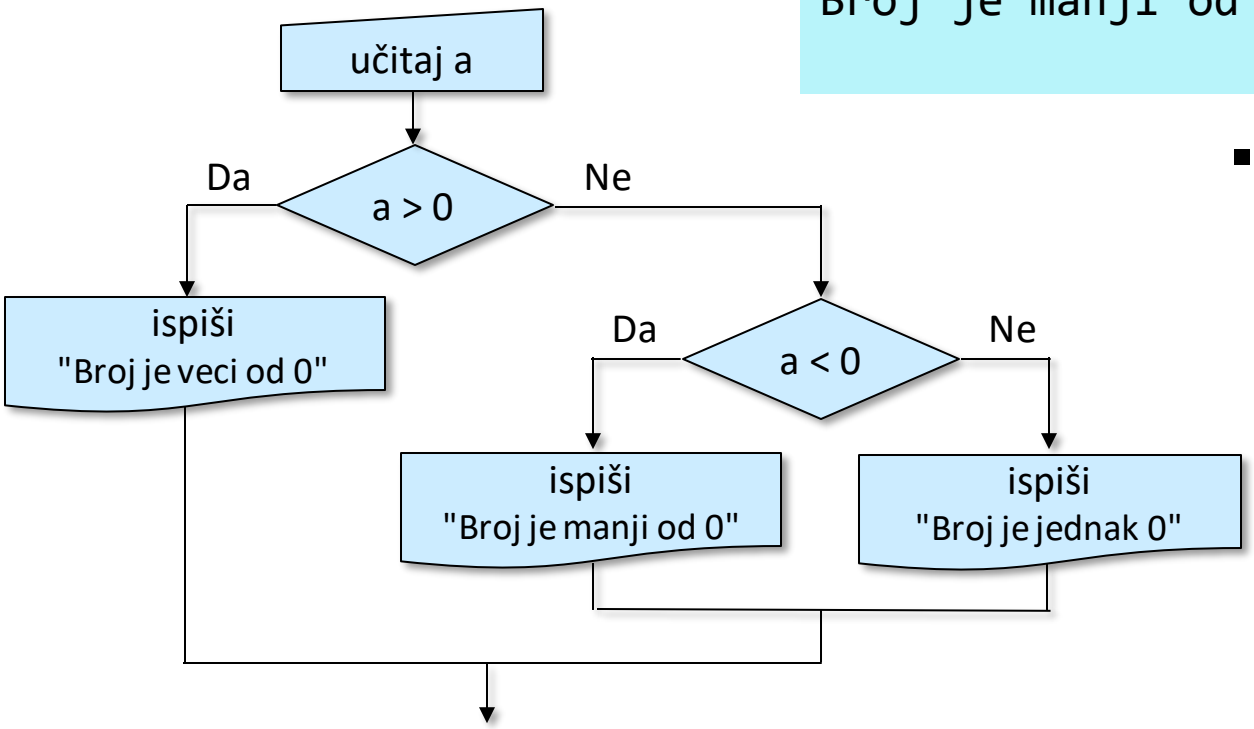
Selekcija

Kaskadna

Primjer

- Programski zadatak
  - učitati cijeli broj. Ovisno o učitanoj vrijednosti, na zaslon ispisati jednu od poruka "Broj je veci od 0", "Broj je jednak 0" ili "Broj je manji od 0"

```
Upisite cijeli broj > -47↵
Broj je manji od 0↵
```



- kaskadna selekcija: obavljanje niza testova koje se zaustavlja u prvom slučaju u kojem je neki od uvjeta zadovoljen

Rješenje

```
#include <stdio.h>

int main(void) {
   int a;

   printf("Upisite cijeli broj > ");
   scanf("%d", &a);

   if (a > 0) {
      printf("Broj je veci od 0\n");
   } else {
      if (a < 0) {
         printf("Broj je manji od 0\n");
      } else {
         printf("Broj je jednak 0\n");
      }
   }
   return 0;
}
```

Stil pisanja programa

Važna digresija

Stilovi pisanja programa

- praznine, tabulatori ili skokovi u novi red u programskom jeziku C nemaju specijalno značenje
  - npr. sljedeći program će se korektno prevesti:

```
#include <stdio.h>
int main(void){int a;scanf("%d",&a);if(a%2==0)printf(
"Paran");else printf("Neparan");return 0;}
```

  - tako napisane programe čovjek će vrlo teško razumjeti. Zato je dobro odabrati i pri pisanju programa čvrsto se držati neke od konvencija za pisanje koda

Thus, programs must be written for people to read, and only incidentally for machines to execute.

H. Abelson, J. Sussman: Structure and Interpretation of Computer Programs; MIT Press, 1984.

Stilovi pisanja programa

- stilski korektan program
  - je lakše razumjeti - stoga je vjerojatnije da je ispravan
  - je lakše održavati - stoga je vjerojatnije da će i ostati ispravan
- postoje mnogi različiti stilovi pisanja C programa
  - ne postoji "najbolji" stil. Koji će se stil koristiti može ovisiti o različitim faktorima
    - kompanijskim pravilima, osobnim preferencijama, ...
  - primjeri različitih stilova pisanja jednostrane selekcije

```
if (x > y) {
   rez = x;
}
```

```
if (x > y)
{
   rez = x;
}
```

```
if (x > y)
   {
   rez = x;
   }
```

```
if (x > y)
   {
      rez = x;
   }
```

  - najvažnije je: odabrati jedan stil i konzistentno ga primjenjivati najmanje u okviru istog projekta

Stilovi pisanja programa

- na predmetu Uvod u programiranje **obavezno je** pridržavati se preporuka koje su prikazane u nastavku ovih predavanja
  - ovdje prikazane preporuke temelje se na standardima pisanja programskog koda u projektu LLVM
    - LLVM Coding Standards
    - https://llvm.org/docs/CodingStandards.html

- neki drugi poznati stilovi za pisanje C/C++ programa:
  - Google
  - WebKit
  - Microsoft Windows

Stilovi pisanja programa - preporuke

- svaku naredbu treba započeti u novom retku. Redak ne bi trebao biti dulji od 80 znakova
- blok naredbi koji je logički podređen treba uvući za određeni broj mjesta. Kolokvijalno se taj postupak naziva *indentacija*
  - na predmetu UPRO: 3 znaka praznine, ne koristiti tabulator
- oznaku početka bloka **{** napisati na kraju retka koji je neposredno nadređen tom bloku, a oznaku završetka bloka **}** točno ispod početka tog istog retka

```
if (x > y) {
    rez = x;
    if (rez == 0) {
        z = 10;
    }
    printf("%d", z);
}
```

Stilovi pisanja programa - preporuke

- ključna riječ else treba se nalaziti u istom retku u kojem se nalazi znak **}** kojim je zatvoren blok naredbi iza if. Slično vrijedi za ključnu riječ while u petlji do-while (objašnjeno kasnije)

```
if (x > y) {
   rez = x;
} else {
   rez = y;
}
```

```
do {
   ...
} while (brojac < 100);
```

Stilovi pisanja programa - preporuke

- praznim redcima razdvojiti logičke cjeline programa
  - direktive pretprocesoru
  - definicije varijabli
  - ostale ključne funkcionalne cjeline (čitanje, izračunavanje, …)

```
#include <stdio.h>

int main(void) {
   int a;
   float x, y;

   printf("Upisite cijeli broj > ");
   scanf("%d", &a);

   if (a > 0) {
      printf("Broj je veci od 0\n");
   ...
```

Stilovi pisanja programa - preporuke

- umetnuti prazninu prije znaka **{** kojim započinje blok

```
int main(void) {
   printf("%d", r);
```

- umetnuti prazninu iza zareza, ali ne prije zareza

```
scanf("%d %d", &m, &n);
```

- umetnuti prazninu iza ključne riječi koja upravlja programskim slijedom (if, switch, while, for)

```
if (x > y) {
```

Stilovi pisanja programa - preporuke

- umetnuti prazninu između operatora i operanada, operatora i zagrada, ali ne i između zagrada i operanada

```
if (m > n) {
   r = (m + n) * 10;
```

- izuzetak od pravila su unarni operatori i sljedeći binarni operatori
  - . ->
  - značenje tih operatora bit će objašnjeno kasnije

```
godina = p_student->god_rod;
s_student.prosj_ocj = 4.5f;
godina++;
a = -a;
```

Stilovi pisanja programa - preporuke

- ne stavljati praznine između imena funkcije i otvorene zagrade, nakon otvorene zagrade, prije zatvorene zagrade niti između zatvorene zagrade i znaka ;

```
scanf("%d %d", &m, &n);
```

- složene izraze logično razlomiti u više redaka i vertikalno uskladiti

```
if (mjesec == 1 && dan == 1 || mjesec == 5 && dan ==  1 || mjesec == 6 &&
    dan == 22 || mjesec == 6 && dan == 25 || mjesec == 8 && dan == 10) {
   printf("Drzavni praznik");
```

```
if (mjesec == 1 && dan ==  1 ||
    mjesec == 5 && dan ==  1 ||
    mjesec == 6 && dan == 22 ||
    mjesec == 6 && dan == 25 ||
    mjesec == 8 && dan ==  5) {
   printf("Drzavni praznik");
}
```

Primjer

- Programski zadatak
  - učitati cijeli broj koji predstavlja brojčanu ocjenu. Ovisno o učitanoj vrijednosti, na zaslon ispisati jednu od poruka:
    - izvrstan
    - vrlo dobar
    - dobar
    - dovoljan
    - nedovoljan
    - neispravna ocjena

Rješenje (odsječak programa)

```
if (ocj == 5) {
   printf("izvrstan");
} else {
   if (ocj == 4) {
      printf("vrlo dobar");
   } else {
      if (ocj == 3) {
         printf("dobar");
      } else {
         if (ocj == 2) {
            printf("dovoljan");
         } else {
            if (ocj == 1) {
               printf("nedovoljan");
            } else {
               printf("neispravna ocjena");
            }
         }
      }
   }
}
```

- program je napisan stilski ispravno
- ipak, zbog velikog broja selekcija u kaskadi postaje težak za pisanje i čitanje. Lako je pogriješiti u broju zagrada i indentaciji za koju se troši previše prostora
- također primijetiti da su vitičaste zagrade iza else (osim zadnjeg) nepotrebne jer se iza else uvijek nalazi samo jedna naredba - sljedeća naredba if

- napišimo program malo drugačije: uklonimo nepotrebne vitičaste zagrade i suvišnu indentaciju

Rješenje (odsječak programa)

```
if (ocj == 5) {
   printf("izvrstan");
} else if (ocj == 4) {
   printf("vrlo dobar");
} else if (ocj == 3) {
   printf("dobar");
} else if (ocj == 2) {
   printf("dovoljan");
} else if (ocj == 1) {
   printf("nedovoljan");
} else {
   printf("neispravna ocjena");
}
```

dodatno: u ovom konkretnom primjeru smiju se ispustiti sve vitičaste zagrade

- Uočiti:
  - pojednostavljuje se pisanje i povećava preglednost koda
  - programeri će lako uočiti da se radi o ničem drugom nego o nizu ispitivanja uvjeta koji jedan drugog isključuju

Bolje rješenje jednog od prethodnih primjera

```
#include <stdio.h>

int main(void) {
   int a;

   printf("Upisite cijeli broj > ");
   scanf("%d", &a);

   if (a > 0) {
      printf("Broj je veci od 0\n");
   } else if (a < 0) {
      printf("Broj je manji od 0\n");
   } else {
      printf("Broj je jednak 0\n");
   }

   return 0;
}
```

Objašnjenje

- U nekim programskim jezicima postoji poseban oblik naredbe za kaskadnu selekciju. Npr. u jeziku Python:

```
if ocj == 5:
   print "izvrstan"
elif ocj == 4:
   print "vrlo dobar"
elif ocj == 3:
   print "dobar"
elif ocj == 2:
   print "dovoljan"
elif ocj == 1:
   print "nedovoljan"
else:
   print "neispravna ocjena"
```

Objašnjenje

- U programskom jeziku C ne postoji posebna naredba za kaskadnu selekciju
  - koristi se niz ugniježđenih dvostranih selekcija, koje se samo radi preglednosti pišu na karakterističan način. Praktički, radi se o stilu pisanja, a ne o posebnoj naredbi za kaskadnu selekciju

```
if (logički_izraz_1)
   naredba_1;
else if (logički_izraz_2)
   naredba_2;
else if (logički_izraz_3)
   naredba_3;
...
else
   naredba_n;
```

- prema potrebi, na mjestima pojedinačnih naredbi (naredba_1, naredba_2, ...) mogu se koristiti složene naredbe
- prema potrebi, posljednji else i pripadna naredba se mogu izostaviti

Primjer

- Programski zadatak
  - učitati realni broj T koji predstavlja izmjerenu tjelesnu temperaturu. Ovisno o učitanoj vrijednosti na zaslon ispisati jedno od upozorenja:
    - Temperatura je blago povisena za $37.0 \leq T < 38.0$
    - Temperatura je znacajno povisena za $38.0 \leq T < 39.0$
    - Temperatura je opasno povisena za $T \geq 39$
  - Primjer izvršavanja programa

```
Upisite temperaturu > 39↵
Temperatura je opasno povisena↵
```

Rješenje

```
#include <stdio.h>

int main(void) {
   float temp;

   printf("Upisite temperaturu > ");
   scanf("%f", &temp);

   if (temp >= 37.0f && temp < 38.0f) {
      printf("Temperatura je blago povisena\n");
   } else if (temp >= 38.0f && temp < 39.0f) {
      printf("Temperatura je znacajno povisena\n");
   } else if (temp >= 39.0f) {
      printf("Temperatura je opasno povisena\n");
   }
   return 0;
}
```

i u ovom konkretnom primjeru se sve vitičaste zagrade smiju ispustiti

nema "zadnjeg else" jer u konkretnom primjeru nije potreban

Moguće dileme u vezi dijela naredbe else

- **Važno pravilo:** else dio naredbe za selekciju "pripada" najbližoj if naredbi (koja već nema "svoj" else dio)
- Nepažljivo ugniježđena naredba za selekciju koja nema svoj else dio, može dovesti do kasnije teško uočljive logičke pogreške
  - Primjer: ako je cjelobrojna vrijednost b različita od nule, tada provjeriti je li cjelobrojna vrijednost a djeljiva s b, i ako jest, ispisati poruku "a je djeljiv s b". Inače (ako je b jednaka nuli), ispisati poruku "b je nula"

Pogrešno! Indentacija ne pomaže!

```
if (b != 0)
   if (a % b == 0)
      printf("a je djeljiv s b");
else
   printf("b je nula");
```

Ispravno

```
if (b != 0) {
   if (a % b == 0)
      printf("a je djeljiv s b");
} else
   printf("b je nula");
```