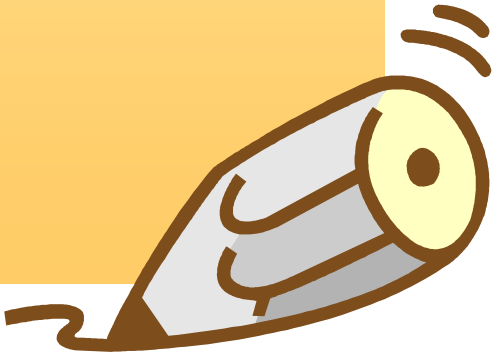
C++

Naredbe za kontrolu tijeka programa

(grananja, petlje)

Vrste tijeka programa

- **Pravocrtni tijek (linearno izvođenje)**
- **Grananja (uvjetne naredbe)**
- **Petlje (naredbe ponavljanja)**
  - sa unaprijed poznatim brojem ponavljanja
  - sa nepoznatim brojem ponavljanja



Blokovi naredbi i područje dosega (*scope*)

- **Blokovi naredbi**
  - dijelovi programa omeđeni parom vitičastih zagrada
  - pišu se uvučeno (radi preglednosti)
- **Lokalne i globalne varijable**
  - lokalne varijable: deklarirane u bloku - vidljive (dostupne) su samo unutar njega
  - globalne varijable: deklarirane izvan bloka ili funkcija (mogu biti dostupne cijeloj funkciji ili svim funkcijama)
- **Područje dosega varijable**
  - područje u kojem je varijabla dostupna, dohvatljiva, može joj se pristupiti

Naredbe za grananje

Naredba

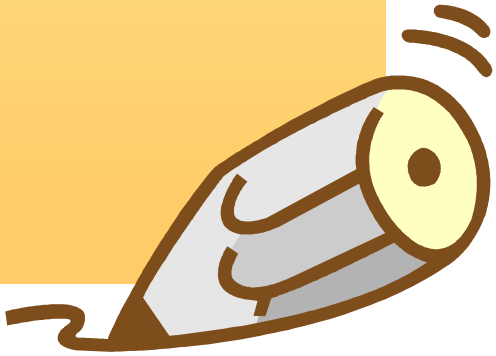
if ()

Grananje programa - naredba *if (ako)*

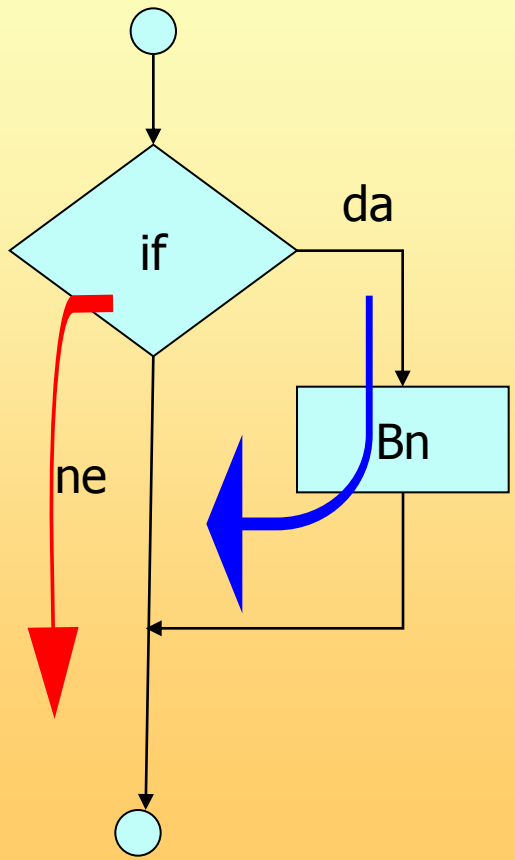
- **uvjetno grananje programa - ovisno o zadovoljenju uvjeta**

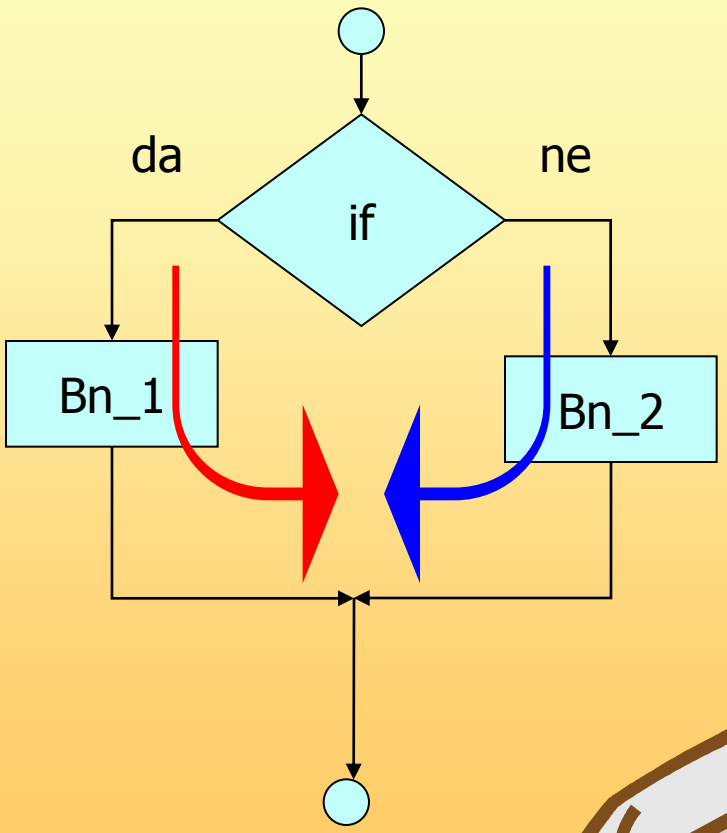
```
if (logičko/relacijski_izraz)
        {blok_naredbi}

if (logičko/relacijski_izraz)
        {blok_naredbi1}
else
        {blok_naredbi2}
```



Osnovni oblici naredbe *if*

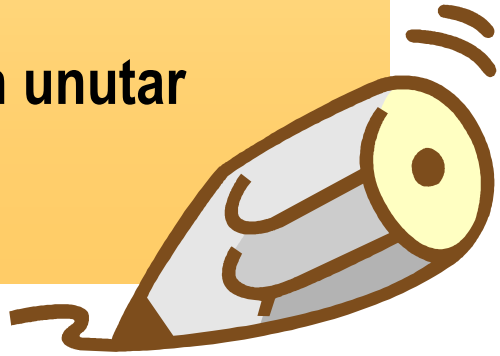




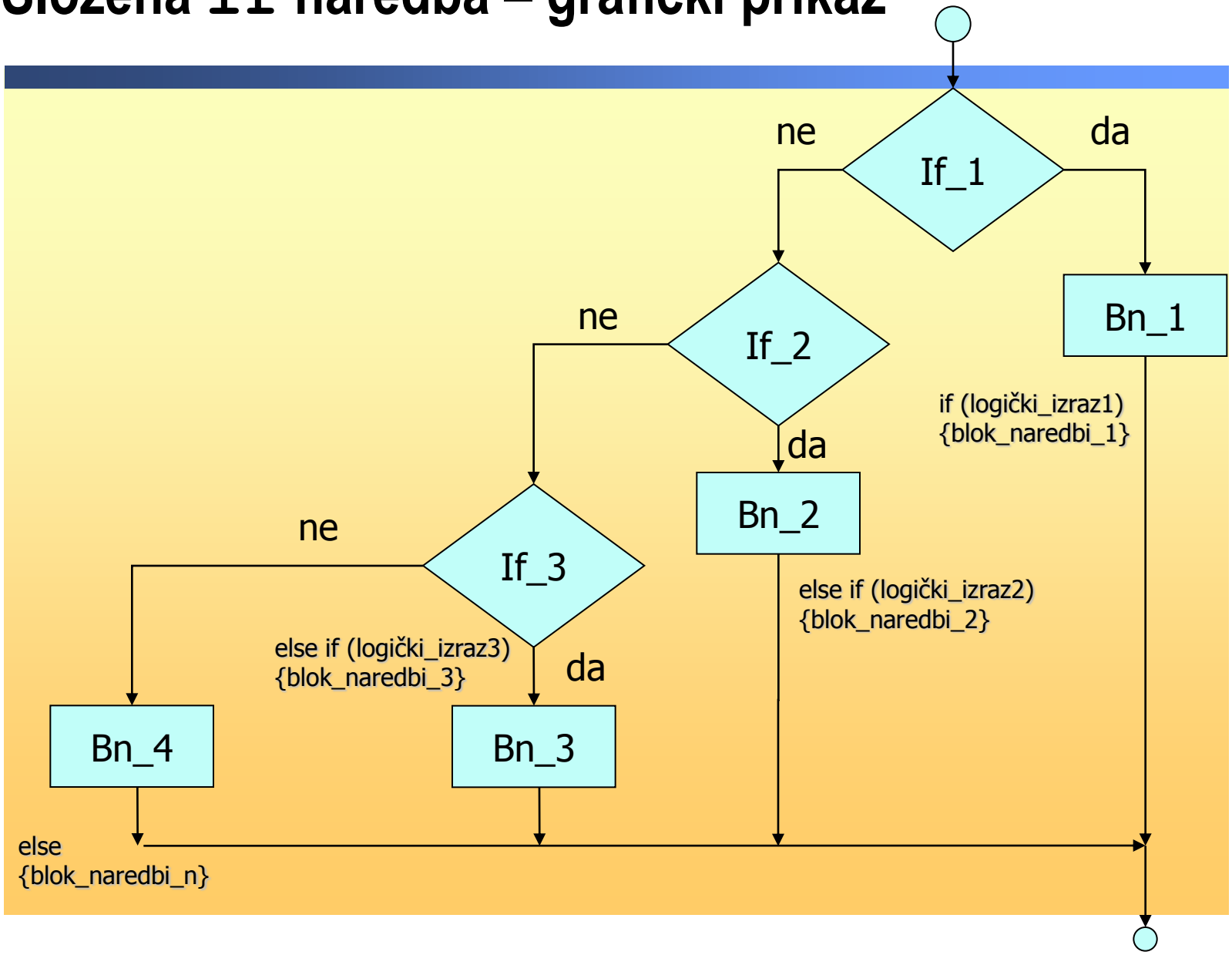
Složena `if` naredba

```
if (logički_izraz1)
        {blok_naredbi_1}
else if (logički_izraz2)
        {blok_naredbi_2}
else if (logički_izraz3)
        {blok_naredbi_3}
...
else
        {blok_naredbi_n}
```

Blokovi `if` naredbi mogu se ugnijezditi jedan unutar drugoga.



Složena `if` naredba – grafički prikaz



Uvjetni operator

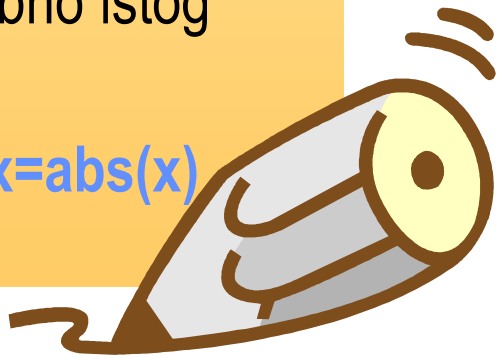
? :

Uvjetni operator ? :

- ne spada među naredbe za kontrolu toka programa
- po strukturi je sličan `if - else` bloku
- jedini **ternarni** operator
- sintaksa je: uvjet ? izraz1 : izraz2;

Ako se uvjet evaluira kao istina izračunava se izraz1, u protivnom izračunava se izraz2. Uvjet mora kao rezultat vratiti logički tip ili tip koji se može svesti na `bool`. Izrazi desno od znaka ? moraju davati rezultat međusobno istog tipa ili se moraju dati svesti na isti tip.

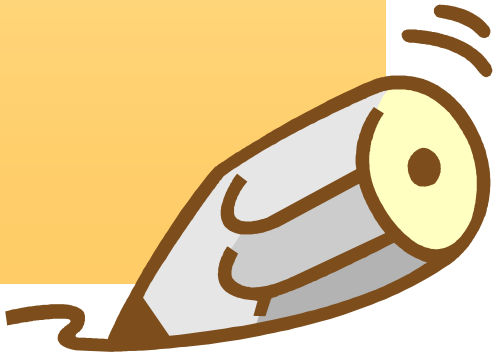
Pr. `x = (x<0) ? -x : x;` //x=abs(x)



Naredba `switch`

Grananje toka naredbom `switch`

- **kada rezultat izračunavanja u nekom uvjetu daje više cjelobrojnih rezultata, a za svaki od njih treba provesti različite odsječke programa, preglednije je koristiti tzv. `switch` grananje**
- prvo se izračunava neki izraz koji daje cjelobrojni rezultat, pa se tok programa preusmjerava na neku od grana unutar `switch` bloka naredbi, zatim se izvode sve naredbe koje slijede pripadajući `case` uvjet označen `case` oznakom (`case label` konstantom) do prve naredbe `break`, nakon čega se izlazi iz `switch` bloka

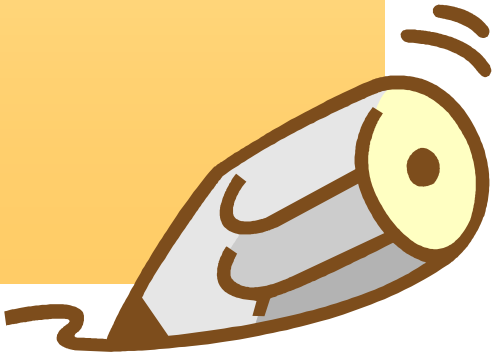


Sintaksa `switch` grananja

```
switch (cjelobrojni_izraz) {
  case rez_1:
       {blok_naredbi_1}
  case rez_2:
       {blok_naredbi_2}
       break;
  case rez_3:
  case rez_4:
       {blok_naredbi_3}
       break;
  default:   //nije obavezno, ali je uobičajeno
       {blok_naredbi_4}
}
```

Ključna riječ `default`

- Ako izraz daje rezultat koji nije naveden niti u jednom od `case` uvjeta, tada se izvodi blok naredbi iza ključne riječi - oznake `default` (može biti smješten bilo gdje unutar `switch` bloka).
- `switch` kontrolna struktura može tražiti samo jednakost; za upotrebu drugih relacijskih operatora treba koristiti `if-else-if`



Savjeti

- **Ne preporuča se koristiti operator == kod usporedbe brojeva realnog tipa (`float, double...`)**

  Kako uspoređivati realne brojeve?

  - Upotrebom funkcije `fabs` i znaka `<`.
  - Npr. `if (fabs(a-b)<1.0e-10)`
  - konstanta za usporedbu uzima se kao proizvoljno mali broj.

- **Logički operatori mogu se koristiti za povezivanje dva ili više relacijskih izraza**
  - npr. `if (x==0 && y==0)`

- **Relacijskim izrazima C++ daje numeričke vrijednosti**
  - za `false` - vrijednost je 0, za `true` - vrijednost je 1

Naredbe za petlje

`for, while, do-while`

-upravljačke strukture zadužene za

ponavljano izvođenje naredbe ili grupe naredbi

- - provode “iteracije”

Naredba – petlja `for`

Petlja `for`

- Kad u programima treba ponavljati dijelove koda, a broj ponavljanja **je poznat** prije ulaska u petlju
- neka se varijabla inicijalizira, testira-ispituje i mijenja

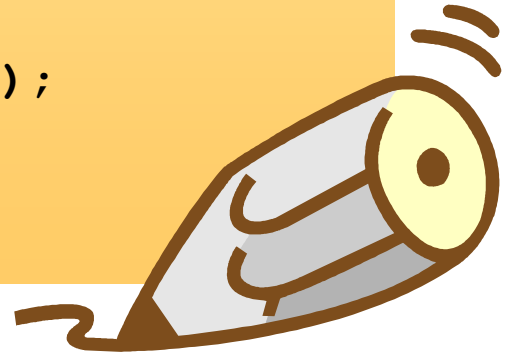
```
for (poč_vrij_brojača; uvjet_ponav;izraz_promj_brojača)
   { blok naredbi }
```

Primjer:

```
for (int i=2; i<=n; i++)
            fakt *=i;
```

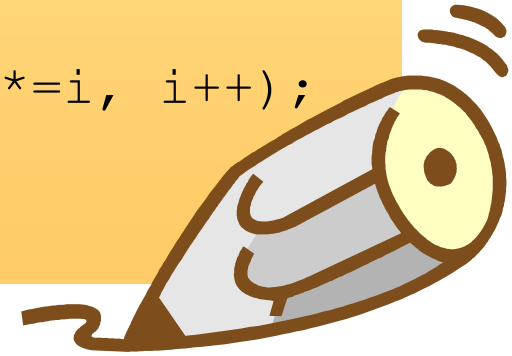
Primjer:

```
for (int i=1; i<=100; zbroj+=++i);
```



Značajke

- bilo koji od tri izraza u `for` naredbi se može izostaviti, jedino su oba znaka `;` obavezna
- ako se izostavi uvjet izvođenja, podrazumijevana vrijednost je `true` i petlja postaje beskonačna.
- treba izbjegavati mijenjanje vrijednosti kontrolne varijable (brojača) unutar bloka naredbi `for` petlje.
- `for` petlje mogu biti ugniježđene jedna unutar druge
- početni izraz i izraz promjene brojača mogu se sastojati od više izraza odvojenih zarezima.
  - Npr. `for(i=2, fakt=1; i<=n; fakt *=i, i++);`



Naredba - petlja `while`

Naredba `while`

- Koristi se uglavnom za ponavljanje segmenta koda kad broj ponavljanja naredbi u bloku nije unaprijed poznat

- `while (uvjet_izvođenja)`

  `{blok_naredbi}`

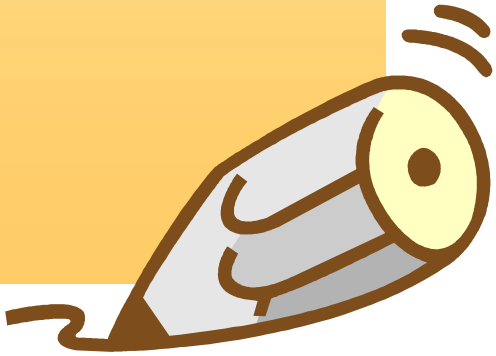
- Uvjet izvođenja je izraz čija je vrijednost logičkog tipa (`bool`) ili se da svesti na taj tip

Naredba - petlja `do-while`

Petlja `do-while`

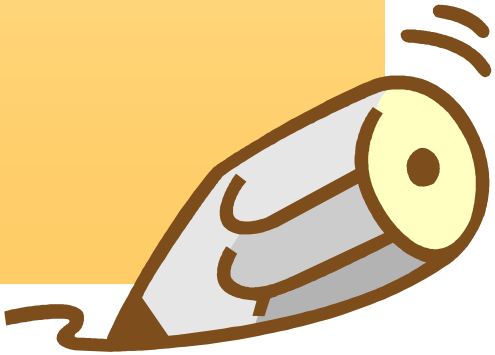
- `for` i `while` naredba ispituju uvjet izvođenja prije izvođenja bloka naredbi pa se može dogoditi da se blok naredbi ne izvede niti jednom
- kada je neophodno da se **prvo izvede neka operacija**, pa da se ovisno o njenom ishodu, ta operacija eventualno ponavlja koristimo `do-while` petlju

```
do
      {blok_naredbi}
while (uvjet_ponavljanja);
```



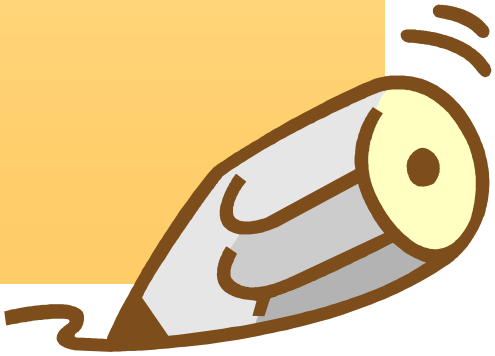
Upute za korištenje

- `for` naredbu preporučljivo je koristiti kada je broj ponavljanja petlje poznat i kontroliran cjelobrojnim brojačem
- `while` naredbu praktičnije je koristiti kada je uvjet ponavljanja određen nekim logičkim uvjetom
- unutar bloka naredbi `do-while` i `while` petlje potrebno je osigurati promjenu varijable uključene u izraz ispitivanja (brojača) kako bi se osigurali od beskonačne petlje



Naredbe `break` i `continue`

- Naredba `break;` može se koristiti samo u petljama i u `switch` grananjima. U petljama prekida izvođenje-ponavljanje okolne petlje.
- Naredba `continue;` uzrokuje skok programa na kraj okolne petlje, ali se potom ponavljanje petlje nastavlja. Preskaču se naredbe do kraja okolne petlje.
- **treba izbjegavati učestalo korištenje ovih naredbi u petljama jer narušavaju strukturiranost programa**



Ostale naredbe za skok

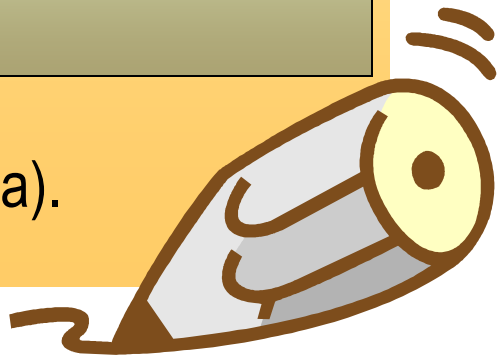
- Naredba `goto` omogućava bezuvjetni skok na neku drugu naredbu unutar iste funkcije

  `goto` *ime_oznake;*

- *ime_oznake* je simbolički naziv (identifikator) koji se mora nalaziti ispred naredbe na koju se želi prenijeti kontrola, odvojeno znakom :

U pravilno strukturiranom programu naredba `goto` uopće nije potrebna

- Naredba `return` (za prekid izvođenja funkcija).



Strukturiranje izvornog koda – doprinos preglednosti i čitljivosti izvornog koda

- uvlačenje blokova naredbi
- pravilan raspored vitičastih zagrada
- sažimanje izvornog koda često olakšava prevoditelju generiranje kraćeg i bržeg (efikasnijeg) izvedbenog koda (npr. višekratna uporaba operatora pridruživanja u istoj naredbi), no ostaje opasnost da se poveća nečitljivost

jer su mnoge naredbe bliske strojnim instrukcijama mikroprocesora

- `if (a!=0)` **<=>** `if (a)`
- `if (a==0)` **<=>** `if (!a)`