

Programiranje

kontrolne naredbe

selekcije

Naredbe za promjenu programskog slijeda (kontrolne naredbe)

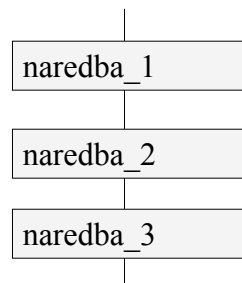
- Normalan programski slijed:

naredba_1

naredba_2

naredba_3

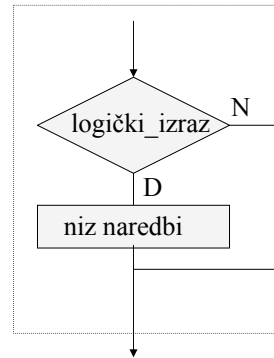
...



Kontrolna naredba `if` - jednostrana selekcija

- Pseudokôd
ako je `logički_izraz` tada
| naredbe
- U C-u
`if (logički_izraz) naredba;`

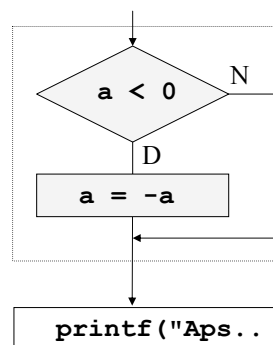
ili
`if (logički_izraz){`
 `niz naredbi`
`}`



3

Primjer za jednostranu selekciju

- Cjelobrojnoj varijabli `a` pridružiti njenu apsolutnu vrijednost i nakon toga ju ispisati



```
if ( a < 0 )  
  a = -a;  
printf("Apsolutna vrijednost je %d\n", a);
```

4

Primjer za jednostranu selekciju

- Ako je vrijednost cjelobrojne varijable b veća od 100, izračunati i ispisati njenu posljednju znamenku. (Komentirati ponuđeno rješenje)

```
int znam;  
...  
if ( b > 100 )  
    znam = b % 10;  
    printf("Posljednja znamenka je %d\n", znam);  
int znam;  
...  
if ( b > 100 ) {  
    znam = b % 10;  
    printf("Posljednja znamenka je %d\n", znam);  
}
```

5

Primjeri za jednostranu selekciju

- Što obavlja slijedeća naredba:
`if (a < 0) a = -a;`
- Postoji li bitna razlika sljedećeg rješenja u odnosu na prethodno:
`if (a < 0){
 a = -a;
}`
- Postoji li bitna razlika sljedećeg rješenja u odnosu na prethodno:
`if (a < 0); a = -a;`

6

Kontrolna naredba `if` - dvostrana selekcija

▪ Pseudokôd

ako je `logički_izraz` tada

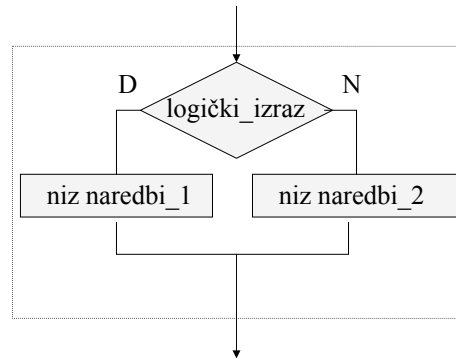
| `niz_naredbi_1`

inače

| `niz_naredbi_2`

▪ U C-u

```
if (logički_izraz) {  
    niz_naredbi_1  
} else {  
    niz_naredbi_2  
}
```



7

Primjer: Napisati program koji računa i ispisuje apsolutnu vrijednost unesenog broja

```
#include <stdio.h>  
int main(){  
    int broj, abs;  
    printf("Unesite broj: ");  
    scanf("%d", &broj);  
    if (broj < 0) {  
        abs = -broj;  
    } else {  
        abs = broj;  
    }  
    printf("Apsolutna vrijednost broja %d je %d\n",  
        broj, abs);  
    return 0;  
}
```

8

Primjer: Napisati program koji računa i ispisuje apsolutnu vrijednost unesenog broja

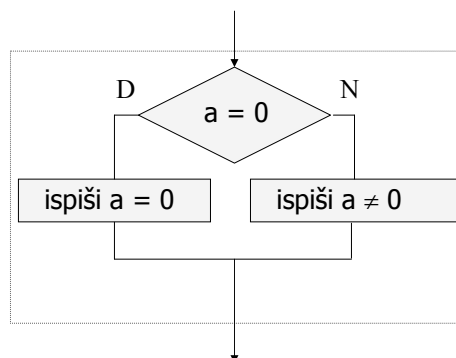
```
#include <stdio.h>
int main(){
    int broj, abs;
    printf("Unesite broj: ");
    scanf("%d", &broj);
    if (broj < 0)
        abs = -broj;
    else
        abs = broj;

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

9

Primjer za dvostranu selekciju

- Zadatak: učitati cijeli broj. Ovisno o učitanoj broju, ispisati "broj je jednak 0" ili "broj je različit od 0".



10

Primjer neispravnog korištenja operatora

- Primjer: Što će se ispisati sljedećim blokom naredbi za zadane vrijednosti varijabli:

1.) $a = 25$ i $b = 4$.

2.) $a = 0$, $b = 0$

```
if (a = b)
    printf ("Vrijednost varijabli je jednaka\n");
else
    printf ("Vrijednost varijabli nije jednaka\n");
```

11

Analiza prethodnog primjera

- Važno je napomenuti:

```
if (a = b) ...
```

... je ekvivalentno:

```
a = b; if (a) ...
```

- Pitanje: Što bi se ispisalo u prvom i drugom slučaju kada bi uvjet glasilo $(a == b)$?

12

Primjer: Napisati program koji učitava dva cijela broja, ispituje je li prvi broj djeljiv s drugim bez ostatka i ispisuje odgovarajuću poruku. Drugi broj ne smije biti 0 (zašto?).

```
#include<stdio.h>
int main() {
    int a, b;
    printf("Unesite a i b:");
    scanf("%d %d", &a, &b);
    if (b != 0) {
        if (a % b == 0) {
            printf("%d jest djeljiv s %d\n", a, b);
        } else {
            printf("%d nije djeljiv s %d\n", a, b);
        }
    }
    return 0;
}
```

13

Primjer: Za zadani radijus izračunati površinu kruga

```
#include <stdio.h>
#define PI 3.141592653589793

int main(){
    double radijus,povrsina;
    printf("Unesite radijus kruga: ");
    scanf("%lf",&radijus);
    if (radijus <= 0.0) {
        printf("Unijeli ste neispravan radijus!\n");
    } else {
        povrsina=radijus*radijus*PI;
    }
    printf("Povrsina kruga je: %20.15lf\n",povrsina);
    return 0;
}
```

14

Diskusija o rješenju

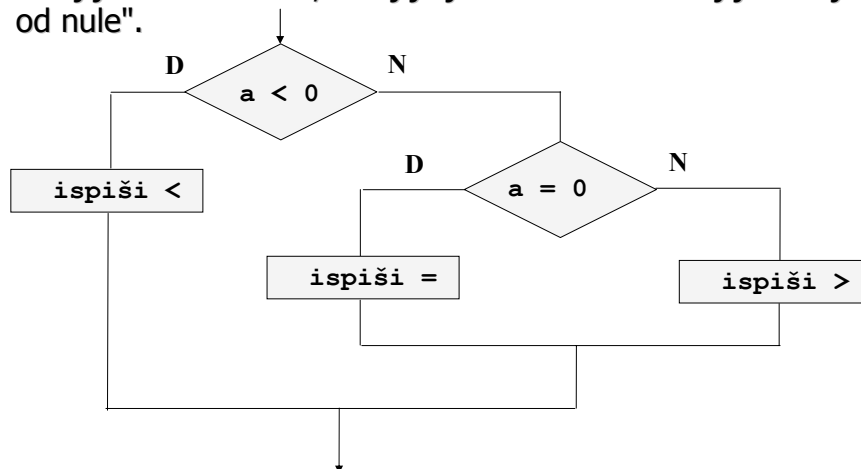
- Što će se ispisati ako se na upit "Unesite radijus kruga:" upiše vrijednost -10?
Unijeli ste neispravan radijus!
Povrsina kruga je: x.xxxxx (ovisno o sadržaju memorijskih lokacija dodijeljenih varijabli povrsina)
- Kako modificirati gornji program da se poruka o izračunatoj površini ispiše samo u slučaju kada se zaista i izračuna?
Naredbu `printf("Povrsina kruga je: %f\n",povrsina);` treba potpisati pod `else` dio:

```
if (radijus <= 0) {
    printf("Unijeli ste neispravan radijus!\n");
}
else {
    povrsina = radijus*radijus*PI;
    printf("Povrsina kruga je: %f\n",povrsina);
}
```

15

Primjer za dvostranu selekciju

- Zadatak: učitati cijeli broj. Ovisno o učitanoj broju, ispisati "broj je veći od nule", "broj je jednak nuli" ili "broj je manji od nule".



16

Rješenje:

```
int main() {
    int a;
    printf("Upisite cijeli broj:");
    scanf("%d", &a);
    if (a < 0) {
        printf("Broj je manji od 0\n");
    } else
        if (a == 0) {
            printf("Broj je jednak 0\n");
        } else {
            printf("Broj je veci od 0\n");
        }
    return 0;
}
```

17

Rješenje: napisano na malo drugačiji način (1)

```
int main() {
    int a;
    printf("Upisite cijeli broj:");
    scanf("%d", &a);
    if (a < 0) {
        printf("Broj je manji od 0\n");
    } else if (a == 0) {
        printf("Broj je jednak 0\n");
    } else {
        printf("Broj je veci od 0\n");
    }

    return 0;
}
```

18

Rješenje: napisano na malo drugačiji način (2)

U konkretnom primjeru vitičaste zagrade nisu bile nužne:

```
int main() {
    int a;
    printf("Upisite cijeli broj:");
    scanf("%d", &a);
    if (a < 0)
        printf("Broj je manji od 0\n");
    else if (a == 0)
        printf("Broj je jednak 0\n");
    else
        printf("Broj je veci od 0\n");

    return 0;
}
```

19

Višestrana selekcija pomoću naredbi if - else if - else

```
if (logički_izraz_1) {
    niz_naredbi_1
} else if (logički_izraz_2) {
    niz_naredbi_2
} else if (logički_izraz_3) {
    niz_naredbi_3
    ...
} else {
    niz_naredbi_0
}
```

20

Primjer s `else if` dijelom naredbe

- Primjer: Napisati programski odsječak koji će zbrojiti `int` varijable `a` i `b` ako je sadržaj `char` varijable `c` jednak `'+'`, oduzeti ako je sadržaj varijable `c` jednak `'-'`, a ispisati poruku o pogrešci ako varijabla `c` sadrži neki treći znak. Rezultat pohraniti u varijablu `r`.

```
int a,b,r;
char c;
if( c == '+' )
    r = a + b;
else if( c == '-' )
    r = a - b;
else
    printf("Pogrešna operacija");
```

21

Primjer: Kotao ima radnu temperaturu u intervalu [50°C-80°C]. Napisati program koji provjerava je li s tipkovnice zadana vrijednost temperature kotla u dopuštenom intervalu i ispisuje odgovarajuću poruku.

```
#include <stdio.h>
#define donja 50.0
#define gornja 80.0
int main(){
    float temp;
    printf("\n Unesite temperaturu kotla: ");
    scanf("%f",&temp);
    if (temp < donja) {
        printf("Temperatura je pala ispod dopustene!\n");
    } else if (temp > gornja) {
        printf("Temperatura je porasla iznad dopustene!\n");
    } else {
        printf("Temperatura kotla OK\n");
    }
    return 0;
}
```

22

Određivanje sjecišta dvaju pravaca

- Odrediti sjecište dvaju pravaca (ili riješite sustav od dvije jednačbe s dvije nepoznanice!?). Parametre pravaca učitati programom. Ako sjecište ne postoji, ispisati odgovarajuću poruku.

Postupak:

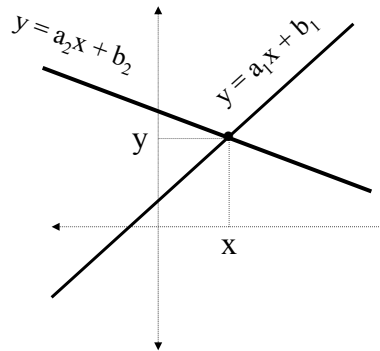
$$y = a_1 \cdot x + b_1$$

$$y = a_2 \cdot x + b_2$$

$$a_1 \cdot x + b_1 = a_2 \cdot x + b_2$$

$$x = (b_2 - b_1) / (a_1 - a_2)$$

$$y = a_1 \cdot x + b_1$$



23

Određivanje sjecišta dvaju pravaca Rješenje u pseudokodu

```
učitaj (a1,b1,a2,b2)
ispiši (a1,b1,a2,b2)
izračunaj nazivnik izraza
ako su pravci paralelni onda
| ispiši poruku da su pravci paralelni
inače
|   izračunaj x,y
|   ispiši (x,y)
kraj
```

24

Određivanje sjecišta dvaju pravaca Rješenje u C-u (I dio):

```
📁SjecistePravaca
#include <stdio.h>
int main () {

    float a1, a2, b1, b2, x, y, anaz;
    scanf ("%f %f %f %f", &a1, &b1, &a2, &b2);
    printf("a1=%f b1=%f a2=%f b2=%f\n"
           , a1, b1, a2, b2);

    /* izračunaj nazivnik izraza */
    anaz = a1 - a2;
```

25

Određivanje sjecišta dvaju pravaca Rješenje u C-u (II dio)

```
if(anaz == 0.) {
    /* ako su pravci paralelni onda */
    printf ("Pravci su paralelni\n");
} else {
    /* inače */
    printf ("Nazivnik izraza: %f\n", anaz);
    x = (b2 - b1) / anaz;
    y = a1 * x + b1;
    printf("Koordinate sjecišta su(%f,%f)\n",
           x, y);
}
return 0;
}
```

26

Određivanje sjecišta dvaju pravaca Prvo testiranje programa

- Ulazni podaci 1:

1 2 3 4

- Ispis na zaslonu 1:

```
a1=1.000000 b1=2.000000 a2=3.000000 b2=4.000000
Nazivnik izraza: -2.000000
Koordinate sjecišta su (-1.000000, 1.000000)
```

27

Određivanje sjecišta dvaju pravaca Drugo testiranje programa

- Ulazni podaci 2:

1 2 1.000001 3

- Ispis na zaslonu 2:

```
a1=1.000000 b1=2.000000 a2=1.000001 b2=3.000000
Nazivnik izraza: -0.000001
Koordinate sjecista su (-1048576.000000, -1048574.000000)
```

Pravi rezultat: (-1000000, -999998)

28

Moguće dileme u korištenju `else` u dvostranim selekcijama

Pogrešno "uvučeno"

```
if( uvjet1 )
    if ( uvjet2 )
        naredba2;
else
    naredba3;
```

Ispravno "uvučeno"

```
if( uvjet1 )
    if ( uvjet2 )
        naredba2;
else
    naredba3;
```

Ispravno "uvučeno" ali to nije isti odsječak kao gore

```
if( uvjet1 ) {
    if ( uvjet2 )
        naredba2;
}
else
    naredba3;
```

Pravilo: *else* pripada "najbližem" *if-u* koji "nema svoj *else*"

29

Moguće dileme u korištenju `else` u dvostranim selekcijama

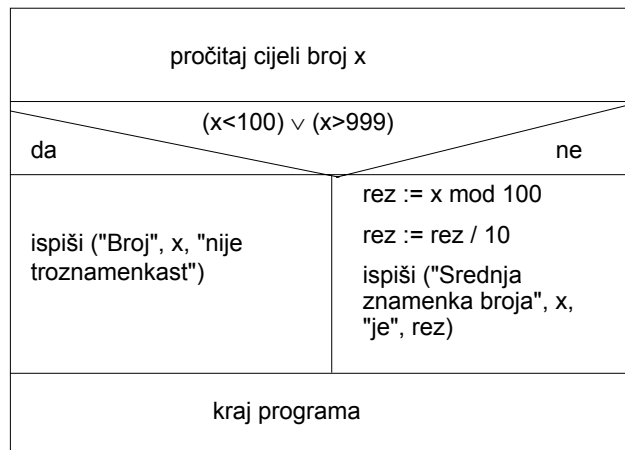
- Primjer: Napisati programski odsječak koji će izračunati presjek dvije dužine na pravcu realnih brojeva (pretpostavlja se da se prva dužina na pravcu nalazi lijevo, a druga dužina desno)

```
int a1, a2, b1, b2; /* dužine [a1, a2] i [b1, b2] */
int r1, r2;        /* rezultat [r1, r2] */
if( a2 > b1 )      /* znači da se sijeku */
    if( a2 > b2 ) { /* cijela dužina B je unutar duž. A */
        r1 = b1;
        r2 = b2;
    }
else {
    r1 = b1;
    r2 = a2;
}
```

- Obratiti pozornost na to da `else` pripada drugoj `if` naredbi, a ne prvoj. Ako `else` dio treba pridružiti prvoj (vanjskoj) `if` naredbi, mora se cijeli blok iza prve `if` naredbe staviti u vitičaste zagrade.

30

Primjer: Pročitati neki troznamenkasti cijeli broj x. Ako je učitani broj troznamenkast treba pronaći srednju znamenku i ispisati pročitani broj i nađenu znamenku, a ako nije treba uz odgovarajuću poruku završiti program.



31

Srednja znamenka troznamenkastog broja Realizacija u C-u

```

SrednjaZnamenka
#include <stdio.h>
int main ( ) {
    int x, rez;
    printf("Unesite troznamenkasti cijeli broj > ");
    scanf("%d", &x);
    if( (x<100) || (x>999) ) {
        printf("Broj %d nije troznamenkast\n", x);
    } else {
        rez = x % 100;
        rez = rez /10;
        printf("Srednja znamenka broja %d je %d\n", x, rez);
    }
    return 0;
}

```

Programski odsječak za pronalaženje srednje znamenke (još neke mogućnosti)

```
. . .  
} else {  
    rez = ( x % 100) / 10;  
    printf("Srednja znamenka broja %d je %d\n",x, rez);  
}
```

ili

```
. . .  
} else {  
    rez = ( x - x/100*100) / 10;  
    printf("Srednja znamenka broja %d je %d\n",x, rez);  
}
```

33

Programiranje ostali operatori

Unarni operatori

Operator	Značenje	Primjer
<code>sizeof</code>	zauzeće memorije	<code>sizeof(int)</code>
<code>(tip)</code>	pretvorba tipa (<i>cast</i>)	<code>(double) i</code>
<code>!</code>	logičko ne	<code>!(5 > 2)</code>
<code>+</code>	unarni plus	<code>+4 / 2</code>
<code>-</code>	unarni minus	<code>-5 * 3</code>
<code>++</code>	uvećavanje za 1	<code>++i</code> ili <code>k++</code>
<code>--</code>	umanjenje za 1	<code>--j</code> ili <code>k--</code>
<code>~</code>	inverzija bitova → u poglavlju o operatorima nad bitovima	<code>int x = ~0xFFFF;</code>
<code>*</code>	indirekcija	<code>*p</code>
<code>&</code>	adresni operator → u poglavlju o pokazivačima	<code>scanf("%d", &n);</code>

35

Unarni + i -

```
float x, y;
x = 5.0;
y = 2.0;
```

```
+ x + y           → 7.0
+x + +y           → 7.0
+x + + y          → 7.0
- x + - y         → -7.0
-x - -y           → -3.0
- x - - y         → -3.0
```

+x NIJE apsolutna vrijednost od x

```
x = -5.0;
+x           → -5.0
```

36

Za vježbu: Napišite program za izračunavanje rješenja (korijena) kvadratne jednadžbe

- Napomene uz program:
 - Prva `#include <stdio.h>` naredba "uključuje" datoteku zaglavlja (header) s opisom funkcija i konstanti (dakle, ne implementacija tj. izvorni kôd tih funkcija) koje se odnose na rad s ulazom izlazom (čitanje s tipkovnice i ispis na zaslon)
 - Druga `#include <math.h>` naredba "uključuje" datoteku zaglavlja s opisom matematičkih funkcija i konstanti (dakle, ne implementacija tj. izvorni kôd tih funkcija)
 - Napomena: Za izračunavanje drugog korijena treba koristiti funkciju `pow` za koja računa x^y , a koja se poziva na slijedeći način:
`korijen=pow((double)x, (double)y);`
 - Budući da se pri prevođenju (točnije: povezivanju-linking) s MS Visual C++ 6.0 po definiciji uključuje datoteka s implementacijom matematičkih funkcija, pri prevođenju nije potrebno navesti nikakve dodatne parametre. Pri radu na nekim inačicama Unix operacijskog sustava, C prevodilac ne uključuje po definiciji tu datoteku te je potrebno dodati i opciju `"-lm"` pri pokretanju prevođenja.

37

Rješenje I dio

```
#include <stdio.h>
#include <math.h>
int main() {
    float a, b, c, x1, x2, q, d, x1r, x2r, x1i, x2i;
    printf("Zadajte koeficijente kvadratne jednadzbe a,b,c:");
    scanf("%f %f %f", &a,&b,&c);

    d = b*b -4.0*a*c; /* diskriminanta */
    if (d > 0) {
        /* Rjesenja su realna */
        q = pow (d, 1./2);
        x1 = (-b + q)/(2*a);
        x2 = (-b - q)/(2*a);
        printf ("X1=%f   X2=%f\n", x1, x2);
    }
}
```

38

Rješenje II dio

```
} else if (d == 0) {
    /* postoji samo jedno rješenje */
    x1 = -b/(2*a);
    printf ("X1=X2=%f\n", x1);
} else {
    /* Rjesenja su konjugirano kompleksni broj */
    q = pow(-d, 1./2);
    x1r = -b/(2*a) ;
    x2r = x1r;
    x1i = q/(2*a);
    x2i = -x1i;
    printf ("X1 = (%f,%f)\n", x1r, x1i);
    printf ("X2 = (%f,%f)\n", x2r, x2i);
}
return 0;
}
```

39

Operatori povećanja i smanjenja za 1

Razlog njihovog postojanja je djelotvornije izvršavanje (postoje posebne procesorske instrukcije za vrlo brzo obavljanje tih operacija):

Operator povećanja za 1

```
int m = 7;
float x = -15.2;
++m;          ili      m++;
++x;          ili      x++;
printf("%d  %f\n", m, x);      → 8  -14.2
```

Operator smanjenja za 1

```
int m = 7;
float x = -15.2;
--m;          ili      m--;
--x;          ili      x--;
printf("%d  %f\n", m, x);      → 6  -16.2
```

40

Operatori povećanja i smanjenja za 1

- prefiksni oblik (operator ispred varijable)
`++a; --a;`
- postfiksni oblik (operator iza varijable)
`a++; a--;`
- u oba oblika varijabla `a` se uvećava/umanjuje za 1, ali:

Prefiksni operator: u izrazima se vrijednost varijable prvo uveća/umanji, a tek onda "iskoristi" njena vrijednost

<code>int i, j;</code>	<u>i</u>	<u>j</u>
<code>i = 2;</code>	2	?
<code>j = ++i;</code>	3	3

<code>int i, j;</code>	<u>i</u>	<u>j</u>
<code>i = 2;</code>	2	?
<code>j = --i;</code>	1	1

41

Operatori povećanja i smanjenja za 1

Postfiksni operator: u izrazima se vrijednost varijable prvo "iskoristi", a nakon toga se varijabla uveća/umanji

<code>int i, j;</code>	<u>i</u>	<u>j</u>
<code>i = 2;</code>	2	?
<code>j = i++;</code>	3	2

<code>int i, j;</code>	<u>i</u>	<u>j</u>
<code>i = 2;</code>	2	?
<code>j = i--;</code>	1	2

Nema garancije da će se varijabla uvećati/umanjiti ODMAH nakon što se "iskoristi" njena vrijednost, ali se garantira da će se varijabla uvećati/umanjiti prije nego započne izvršavanje sljedeće naredbe (vidjeti primjer: *doing to much at once*).

42

Operatori povećanja i smanjenja za 1

- **Primjer:**

```
a = 5;  
b = ++a * 2;  
c = b++;
```

- Nakon izvođenja ovih naredbi
 - a ima vrijednost 6
 - b ima vrijednost 13
 - c ima vrijednost 12

43

Operatori povećanja i smanjenja za 1

- Izbjegavati dvoznačnosti oblika:

```
int i = 7;  
i = 2 * i++;      (doing too much at once)
```

- ako se ++ obavi nakon pridruživanja, rezultat je 15
- ako se ++ obavi prije pridruživanja, rezultat je 14
- **Koristan savjet:** izbjegavati dvije ili više promjena sadržaja iste varijable unutar iste naredbe. Još jedan loš primjer: `k = ++i * --i;`
- Sljedeća naredba ne uzrokuje takve probleme:
`k = m++ + n--;`

- **Nije dopušteno:**

```
i = ++5;  
m = (i+j) --;
```

44

Binarni operatori

<i>Operator</i>	<i>Značenje</i>	<i>Primjer</i>
* / %	množenje, dijeljenje	
+ -	zbrajanje, oduzimanje	
<<	pomak bitova u lijevo	<code>n = n << 2;</code>
>>	pomak bitova u desno	<code>n = n >> 1;</code>
< > <= >=	relacijski operatori	
== !=	operatori jednakosti	
&	logički I po bitovima	<code>znam & '0'</code>
^	isključivi ILI po bitovima	<code>f = f ^ f</code>
	uključivi ILI po bitovima	<code>f = f 0x80</code>
&&	logički I i ILI	

45

Operatori nad bitovima

- Operatori `&`, `|`, `^` su binarni, a odnose se na usporedbu bitova dvaju operandata

`&` AND
`|` OR
`^` XOR

b1	b2	b1 & b2	b1 b2	b1 ^ b2
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

- Operator `~` je unarni, a odnosi se na operand s desna

`~` NOT

b1	~b1
0	1
1	0

- Operatori posmaka bitova su binarni, a odnose se na posmak bitova lijevog operandata za iznos određen desnim operandom

`<<` SHIFT LEFT
`>>` SHIFT RIGHT

46

Primjer operacija s bitovima

- Izračunati izraze: $3 \& 5$, $3 | 5$, $3 \wedge 5$, ~ 3

```
0000 0000 0000 0000 0000 0000 0000 0011 (3)
& 0000 0000 0000 0000 0000 0000 0000 0101 (5)
```

```
-----
0000 0000 0000 0000 0000 0000 0000 0001 (1)
```

```
0000 0000 0000 0000 0000 0000 0000 0011 (3)
| 0000 0000 0000 0000 0000 0000 0000 0101 (5)
```

```
-----
0000 0000 0000 0000 0000 0000 0000 0111 (7)
```

```
0000 0000 0000 0000 0000 0000 0000 0011 (3)
^ 0000 0000 0000 0000 0000 0000 0000 0101 (5)
```

```
-----
0000 0000 0000 0000 0000 0000 0000 0110 (6)
```

```
~ 0000 0000 0000 0000 0000 0000 0000 0011 (3)
```

```
-----
1111 1111 1111 1111 1111 1111 1111 1100 (-4)
```

47

Primjer operacija s bitovima

- Pretvoriti znak u broj

'7' == 48 + 7

```
char znak = '7';
```

```
int broj;
```

```
broj = znak & 0x0f;
```

48

```
0000 0000 0000 0000 0000 0000 0011 0111 (znak '7')
& 0000 0000 0000 0000 0000 0000 0000 1111 (0x0f)
-----
0000 0000 0000 0000 0000 0000 0000 0111 (710)
```

Napomena: prije obavljanja operacije &, vrijednost varijable znak pretvorena je u tip podatka int.

48

Rad s operatorima pomaka bitova

- Operatori \ll i \gg služe za pomak svih bitova vrijednosti varijable u lijevo ili u desno. Pomak bitova u varijabli za jedno mjesto u lijevo odgovara množenju vrijednosti varijable sa 2, dok pomak za jedno mjesto u desno rezultira dijeljenjem vrijednosti varijable sa 2.
- Elektronička računala u skupu svojih strojnih naredbi u pravilu imaju naredbe za pomak vrijednosti u registru i tako izvršeno množenje ili dijeljenje s višekratnikom broja 2 bitno je brže u odnosu na klasično množenje i dijeljenje.
- Broj pomaka u lijevo ili desno određen je parametrom.

49

Primjeri s pomakom bitova

- Primjer: Izračunati izraze $2 \ll 1, 37 \gg 2$
0000 0000 0000 0000 0000 0000 0000 0010 **(2)**
- Nakon pomaka u lijevo za jedno mjesto, rezultat je umnožak vrijednosti s 2:
0000 0000 0000 0000 0000 0000 0000 0100 **(4)**

0000 0000 0000 0000 0000 0000 0010 0101 **(37)**
- Nakon pomaka u desno za dva mjesta, rezultat je cjelobrojno dijeljenje s 4:
0000 0000 0000 0000 0000 0000 0000 1001 **(9)**

50

Primjeri s pomakom bitova

- **Opres:** U slučaju da je podatak spremljen u jedan oktet bez bita za predznak, izračunati $128 \ll 1$:
1000 0000 (128)
- Nakon pomaka za jedno mjesto:
0000 0000 (0)
- U slučaju da je podatak spremljen u jedan oktet s bitom za predznak, izračunati $64 \ll 1$:
0100 0000 (64)
- Nakon pomaka za jedno mjesto:
1000 0000 (-128)

51

Prioritet operatora

	OPERATORI	PRIDRUŽIVANJE
Viši prioritet ↑	()	L → D
	! ~ ++ -- sizeof & * unarni + -	D → L
	(cast)	D → L
	* / %	L → D
	+ -	L → D
	<< >>	L → D
	< <= > >=	L → D
	== !=	L → D
	&	L → D
	^	L → D
Niži prioritet ↓		L → D
	&&	L → D
		L → D
	? :	D → L
	= *= /= %= += -= &= ^= = <<= >>=	D → L
	,	L → D

52

Složeniji primjer s logičkim operatorima

- **Primjer:** Što će se ispisati sljedećim programskim odsječkom?

```
int a, b, c, d;
a = 0;
b = 4;
c = (a++) + b;          /* (a++) +b      0+4  4*/
printf("a = %d, b = %d, c = %d " , a, b, c); /* 1, 4, 4 */
d = c && b + 3 * a;
printf("d = %d", d);
```

- Budući da je prioritet aritmetičkih veći od prioriteta logičkih operatora:

```
d = c && b + 3 * a /* ovo je ekvivalentno izrazu u
sljedećem retku */
d = c && (b + (3 * a))
d = 4 && (4 + (3 * 1))
d = 4 && 7
d = 1
```

53

Poteškoće sa složenim logičkim uvjetima

- Složeni izrazi često se pogrešno napišu zbog "doslovnog prepisivanja" izrečenog uvjeta:
- Izraz "ako je x veći od 20 i manji od 100" (uočiti da se pod "manji od 100" podrazumijeva "x manji od 100"), često se "doslovno prepíše" u:

```
if ( x > 20 && < 100 )
```

što dovodi do pogreške pri prevođenju. Osim toga česta je pogreška da se umjesto operatora "I" stavi operator ",", te se prethodna naredba napiše:

```
if ( x > 20, x < 100)
```

što dovodi do "logičke" pogreške (koju je puno teže otkriti jer ju prevodilac ne prijavi).

Izraz "x>20, x<100" odgovara kao da je napisano "x<100".

54

Primjeri

- Koja je vrijednost varijable d nakon sljedećeg programskog odsječka:

```
short int a=4, b=2, c=8, d;  
d = a < 10 && 2 * b < c;
```

- Rješenje:

```
d = ( a < 10 ) && ( ( 2 * b ) < c )  
d = 1 && ( 4 < 8 )  
d = 1 && 1  
d = 1
```

- Što će se ispisati sljedećim programskim odsječkom?

```
int a = 5, b = -1, c = 0;  
c = ( a = c && b ) ? a = b : ++c;  
printf("a = %d, b = %d, c = %d: \n", a, b, c);
```

- Rezultat:

```
a = 0 , b = -1 , c = 1
```

55

Ternarni operatori

Operator za uvjetno pridruživanje

- Operator za uvjetno pridruživanje (? :) je ternarni operator (zahtijeva tri operanda), a koristi se u pojedinim situacijama umjesto if-else naredbi. Oblik izraza u kojem se koristi operator je :

```
uvjetni_izraz ? izraz1 : izraz2;
```

- uvjetni_izraz se izračunava prvi. Ako je rezultat TRUE, izračunava se izraz1 i to je konačni rezultat cijelog izraza. Ako je uvjetni_izraz FALSE, izračunava se izraz2 i to je konačni rezultat cijelog izraza.

56

Operator za uvjetno pridruživanje

- *Primjer:* U znakovnu varijablu `r` pohraniti vrijednost 'Z' ako je u varijabli `c` znamenka, a inače pohraniti vrijednost 'N' .

```
if (c >= '0' && c <= '9') {  
    r = 'Z';  
} else {  
    r = 'N';  
}
```



```
r = c >= '0' && c <= '9' ? 'Z' : 'N';
```

57

Primjer: Napisati program koji učitava dva broja, ispituje je li prvi broj djeljiv s drugim bez ostatka i ispisuje odgovarajuću poruku.

```
#include<stdio.h>  
int main(){  
    int a,b;  
    printf("Unesite a i b:");  
    scanf("%d %d", &a, &b);  
    if (b != 0) {  
        printf("%d %s djeljiv s %d\n",  
                a,  
                a % b ? "nije" : "jest",  
                b);  
    }  
    return 0;  
}
```

58

Skraćeni izrazi pridruživanja

- U programiranju se često nova vrijednost varijable izračunava na temelju stare vrijednosti te iste varijable. Zbog toga u C-u postoje **skraćeni izrazi pridruživanja**.

- **Primjer**

Izraz

```
i = i + 5;
```

može se pisati kao:

```
i += 5;
```

Izraz

```
i = i / (a + b);
```

može se pisati kao:

```
i /= a + b;
```

59

Skraćeni izrazi pridruživanja

- **Služi li to ičemu?**

Članu niza `niz` s indeksom `3*i+2*j-m*k` uvećaj vrijednost za 9.

```
niz[3*i+2*j-m*k] = niz[3*i+2*j-m*k] + 9;
```

ili

```
niz[3*i+2*j-m*k] += 9;
```

60

Skraćeni izrazi pridruživanja

<code>i = i + 7;</code>	<code>⇒</code>	<code>i += 7;</code>
<code>j = j - k;</code>	<code>⇒</code>	<code>j -= k;</code>
<code>a = a * (3 + 2);</code>	<code>⇒</code>	<code>a *= 3 + 2;</code>
<code>b = b / (c * 2);</code>	<code>⇒</code>	<code>b /= c * 2;</code>
<code>d = d % 2;</code>	<code>⇒</code>	<code>d %= 2;</code>
<code>a = a & b;</code>	<code>⇒</code>	<code>a &= b;</code>
<code>a = a ^ b;</code>	<code>⇒</code>	<code>a ^= b;</code>
<code>a = a b;</code>	<code>⇒</code>	<code>a = b;</code>
<code>a = a << b;</code>	<code>⇒</code>	<code>a <<= b;</code>
<code>a = a >> b;</code>	<code>⇒</code>	<code>a >>= b;</code>

61

Primjer: ako je `i > j`, uvećati `i` za 1 i pridružiti dobivenu vrijednost varijabli `k`; inače, uvećati `j` za 1 i pridružiti dobivenu vrijednost varijabli `k`.

```
int i, j, k;
...
k = i > j ? ++i : ++j;
```

Ako je `i > j`, `i` se uvećava za 1, `j` se ne mijenja. U suprotnom, `j` se uvećava za 1, `i` se ne mijenja.

62

Višestruko pridruživanje

- Operator pridruživanja obavlja pridruživanje vrijednosti izraza s desne strane (*r-value*) operandu s lijeve strane (*l-value*). Razlika između *l-value* i *r-value* je u tome što *l-value* operand mora imati dobro definiranu adresu i nakon izračunavanja izraza.
- Primjer
`a = b = c = 0;`
- Sve tri varijable inicijaliziraju se na vrijednost 0. Pridruživanje je s desna na lijevo, tj. kao da je napisano:
`a = (b = (c = 0));`
- Prvo se varijabli c pridružuje 0 i vrijednost tog cijelog izraza (`c = 0`) poprima vrijednost 0; zatim se varijabli b pridružuje vrijednost tog izraza, zatim cijeli taj izraz poprima vrijednost 0 koja se na kraju pridružuje varijabli a.
`a = b = c + 3;`
- Može li?
`a = b + 3 = c = d * 3;` NE MOŽE!!! → b+3 nije l-value

63

Odvajanje naredbi zarezom

- Zarez se kao operator koristi za odvajanje naredbi obično tamo gdje je dopuštena samo jedna naredba. Na primjer:

```
float x, y;  
x = 7, y = 3;
```

64

Programiranje

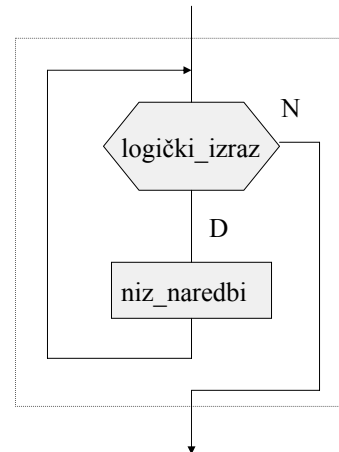
kontrolne naredbe (programske petlje)

Programske petlje

- Programske petlje služe za obavljanje određenog programskog odsječka (tijelo petlje) više puta.
- Dijelimo ih na
 - programske petlje s ispitivanjem uvjeta na početku
 - ovisno o početnim uvjetima može se dogoditi da se tijelo petlje uopće ne izvršava
 - programske petlje s ispitivanjem uvjeta na kraju
 - tijelo petlje se obavezno izvršava barem jedan put
 - programske petlje s poznatim brojem ponavljanja
 - broj ponavljanja može se unaprijed izračunati i ne ovisi o izvršavanju tijela petlje

Programska petlja s ispitivanjem uvjeta ponavljanja na početku

- Pseudokod_
dok je (logički_izraz)
| niz_naredbi
- U C-u
`while(logički_izraz) naredba`
ili
`while(logički_izraz) {`
 `niz_naredbi`
`}`



67

Primjer: Napisati program koji će ispisivati ostatke uzastopnog dijeljenja učitano^g nenegativnog cijelog broja s 2. Prekinuti postupak kad se dijeljenjem dođe do 0.

```
#include <stdio.h>
int main() {
    int broj, ostatak;
    printf("Upisite pozitivan cijeli broj:");
    scanf("%d", &broj);
    printf("Upisali ste: %d\n", broj);
    while (broj != 0) {
        ostatak = broj % 2;
        printf("Ostatak je: %d\n", ostatak);
        broj = broj / 2;
    }
    return 0;
}
```

68

Primjer: Napisati program koji će od učitano početnog cijelog broja uzastopno oduzimati cijele brojeve koje upisuje korisnik. Zaustaviti se kad se oduzimanjem dobije broj manji od 0. U svakom koraku ispisati "koliko je preostalo". Uočite da se kao početni broj može učitati i broj koji je manji od 0.

```
#include <stdio.h>
int main() {
    int trenutniBroj, oduzetiBroj;
    printf("Upisite cijeli broj:");
    scanf("%d", &trenutniBroj);
    printf("Upisali ste: %d\n", trenutniBroj);
    while( trenutniBroj >= 0 ) {
        printf("Upisite cijeli broj za oduzimanje:");
        scanf("%d", &oduzetiBroj);
        trenutniBroj = trenutniBroj - oduzetiBroj;
        printf("Preostalo je: %d\n", trenutniBroj);
    }
    return 0;
}
```

Primjer: Treba ispisati površinu kruga, ako se r mijenja od 1 do zadanog g s korakom 1. Ispis prekinuti i ako r premaši vrijednost 20.

📁 PovrsinaKrug

```
#include <stdio.h>
int main ( ) {
    float g, r=1.0f, pi=3.14159f;
    printf ("Unesite gornju granicu za r > ");
    scanf ("%f", &g);
    while ( (r <= g) && (r <= 20.0f)) {
        printf("%2.0f %10.7f\n", r, r*r*pi);
        r = r + 1;
    }
    return 0;
}
```

70

Rezultati izvođenja za g=55

```
1 3.1415901
2 12.5663605
3 28.2743111
4 50.2654419
5 78.5397530
6 113.0972443
7 153.9379158
8 201.0617676
9 254.4687996
10 314.1590118
11 380.1324043
12 452.3889771
13 530.9287300
14 615.7516632
15 706.8577766
16 804.2470703
17 907.9195442
18 1017.8751000
19 1134.1140327
20 1256.6360474
```

Komentirajte točnost prikazivanja
8. važeće znamenke

Komentirajte zašto je petlja obavila samo 20 prolaza

71

Primjer: Učitavati pozitivne cijele brojeve sve dok njihova suma ne premaši dopušteni raspon za short int. Ispisati posljednju valjanu sumu.

```
#include <stdio.h>
int main() {
    short int x, suma = 0, gotovo = 0;

    while (!gotovo) {
        printf("Unesite broj : ");
        scanf("%d", &x);
        if ((signed short)(suma+x) >= suma) { /* zašto treba cast? */
            suma = suma + x;
        } else {
            gotovo = 1;
        }
    }
    printf("Suma: %d\n", suma);
    return 0;
}
```

Primjer testiranja programa:

```
Unesite broj : 30000
Unesite broj : 10
Unesite broj : 10000
Suma: 30010
```

Je li u ovom primjeru unaprijed poznat broj ponavljanja petlje?

72

Primjer: Učitati pozitivan cijeli broj i izračunati slijed brojeva na sljedeći način: ako je broj paran podijeliti ga s 2. Ako je neparan pomnožiti s 3 i dodati 1. Ponavljati postupak dok broj ne postane jednak 1. U svakom koraku ispisati trenutnu vrijednost broja. Ispisati ukupan broj operacija nad učitanim brojem.

```
#include <stdio.h>
int main() {
    int broj, brKoraka = 0;
    printf("Upisi pozitivan broj:");
    scanf("%d", &broj);
    while (broj > 1) {
        brKoraka = brKoraka + 1;
        if (broj % 2){
            broj = broj * 3 + 1;
        }
        else {
            broj /= 2;
        }
        printf ("U %d. koraku broj = %d \n",brKoraka, broj);
    }
    printf ("Ukupno %d koraka. \n",brKoraka);
    return 0;
}
```

73

Ispis rezultata

```
Upisi pozitivan broj: 9
U 1. koraku broj = 28
U 2. koraku broj = 14
U 3. koraku broj = 7
U 4. koraku broj = 22
U 5. koraku broj = 11
U 6. koraku broj = 34
U 7. koraku broj = 17
U 8. koraku broj = 52
U 9. koraku broj = 26
U 10. koraku broj = 13
U 11. koraku broj = 40
U 12. koraku broj = 20
U 13. koraku broj = 10
U 14. koraku broj = 5
U 15. koraku broj = 16
U 16. koraku broj = 8
U 17. koraku broj = 4
U 18. koraku broj = 2
U 19. koraku broj = 1
Ukupno 19 koraka.
```

74

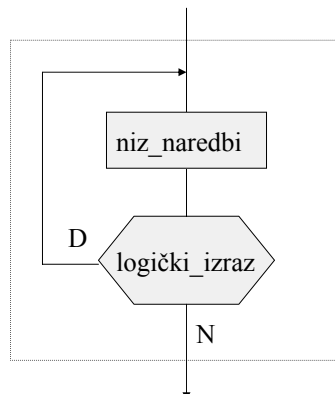
Programska petlja s ispitivanjem uvjeta ponavljanja na kraju

- U nekim programskim jezicima (npr. FORTRAN, PASCAL) postoji oblik:
ponavlajaj
| niz_naredbi REPEAT...UNTIL
dok ne bude (logički_izraz)
- Pseudokod za C
ponavlajaj
| niz_naredbi
dok je (logički_izraz)

75

Programska petlja s ispitivanjem uvjeta ponavljanja na kraju

- U C-u
do naredba while (logički_izraz);
ili
do
naredba
while (logički_izraz);
do {
niz_naredbi
} while (logički_izraz);



76

Primjer: Napisati program koji će učitavati pozitivne cijele brojeve sve dok se ne unese broj 0, i zatim ispisati koji je najveći uneseni broj. Zanimariti negativne brojeve zadane pomoću tipkovnice.

```
...
int maks = 0;
int x;
do {
    printf("Unesite broj : ");
    scanf("%d", &x);
    if (x > maks)
        maks = x;
} while (x != 0);
if (maks > 0)
    printf("Najveci uneseni broj je %d\n", maks);
else
    printf("Nije unesen niti jedan broj veci od 0\n");
...
```

77

Primjer 4. Napisati program koji učitava brojeve iz intervala [-100,100] ili [800,1000]. Program treba ispisati koliko je bilo pozitivnih brojeva, koliko negativnih i koliko vrijednosti nula.

```
#include <stdio.h>
int main(){
    int broj;
    int brojpozitivnih=0, brojnegativnih=0, brojnula=0;

    printf("Unesite brojeve: ");
    do {
        scanf("%d", &broj);
        if(broj>=-100 && broj<=100 || broj>=800 && broj<=1000) {
            if (!broj) brojnula = brojnula + 1;
            else if (broj>0) brojpozitivnih = brojpozitivnih + 1;
            else brojnegativnih = brojnegativnih + 1;
        }
    } while(broj>=-100 && broj<=100 || broj>=800 && broj<=1000);

    printf("Broj pozitivnih je %d, broj negativnih je %d,"
           " brojnula je %d\n", brojpozitivnih,
           brojnegativnih, brojnula);
    return 0;
}
```

78

Primjer: Izračunaj aritmetičku sredinu brojeva koji se redom čitaju s tipkovnice sve dok njihova suma ne premaši neku zadanu gornju granicu.

```
učitaj gornju granicu gg
brojač n := 0
s := 0
ponavljaj
    učitaj i
    s := s + i
    n := n+1
dok ne bude s > gg
as := s / n
tiskaj( s, n, as )
```

79

Rješenje zadatka u C-u

```
📁AritmetickaSredinaPunoBrojeva
#include <stdio.h>
int main ( ) {
    /* definicija varijabli */
    int n=0, s=0, i, gg; float as;
    /* citanje gornje granice za sumu */
    scanf("%d", &gg);

    do {
        scanf("%d", &i);
        s = s + i;
        n = n + 1;
    } while (s <= gg); ← Uvjet iz pseudokoda je negiran
    as = (float) s / n;
    printf("%d %d %10.3f\n", s, n, as);
    return 0;
}
```

80

Primjer: Napisati program koji će učitavati pozitivne cijele brojeve sve dok se ne unese broj 0, i zatim ispisati koji je najmanji uneseni broj. Zanimariti negativne brojeve zadane pomoću tipkovnice.

- Napomena: kod traženja najmanjeg (najvećeg) člana niza koristimo sljedeći algoritam:
 - prvi član niza proglasimo najmanjim i njegovu vrijednost pohranimo u pomoćnu varijablu koja predstavlja trenutni minimum
 - pretražujemo preostale članove niza i ukoliko je neki od njih manji od trenutnog minimuma, ažuriramo trenutni minimum na tu vrijednost
- Nakon što smo pretražili cijeli niz u pomoćnoj varijabli se nalazi minimalni element niza.
- Primjer:

niz: 5, 6, 3, 9, 4, 7, 2, -1, 5.

5		6	3	9	4	7	2	-1	5
		6<5?	3<5?	9<3?	4<3?	7<3?	2<3?	-1<2	5<-1?
min=5			min=3				min=2	min=-1	

81

Rješenje u C-u s while petljom

```
#include <stdio.h>
int main() {
    int min, x = 0;

    while(x <= 0) {
        printf("Unesite broj : ");
        scanf("%d", &x);
        min = x; // pretpostavljamo da je prvi uneseni najmanji
    }
    while(x != 0) {
        printf("Unesite broj : ");
        scanf("%d", &x);
        if (x > 0 && x < min) {
            min = x;
        }
    }
    printf("Najmanji uneseni broj je %d\n", min);
    return 0;
}
```

82

Alternativno rješenje, da se izbjegne dvostruko učitavanje

```
#include <stdio.h>
int main() {
    int min=0, x=1;
    // u prvom prolasku x=1
    while (x != 0) {
        printf("Unesite broj : ");
        scanf("%d", &x); //pri prvom učitavanju x postaje != 1
        if (x > 0) {
            if ((min==0) || (x < min)) min = x;
        }
    }
    printf("Najmanji uneseni broj je %d\n", min);
    return 0;
}
```

83

Rješenje u C-u s petljom do-while

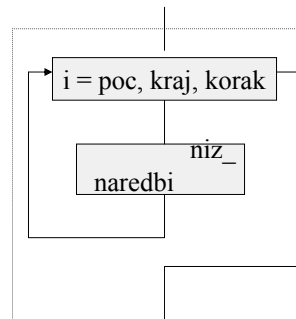
```
...
int maks = 0;
int x;
do {
    printf("Unesite broj : ");
    scanf("%d", &x);
    if (x > maks)
        maks = x;
} while (x != 0);
if (maks > 0)
    printf("Najveci uneseni broj je %d\n", maks);
else
    printf("Nije unesen niti jedan broj veci od 0\n");
...
```

84

Programska petlja s poznatim brojem ponavljanja

- Pseudokod

```
za i = poc do kraj (korak k)
  niz_naredbi
```



- U C-u

```
for(i = poc; i <= kraj; i = i + k) naredba;
    ili
for (i = poc; i <= kraj; i = i + k) {
    niz_naredbi
}
```

85

Programska petlja s poznatim brojem ponavljanja

- Sintaksa:

```
for (izraz1; izraz2; izraz3) {
    .
    .
}
```

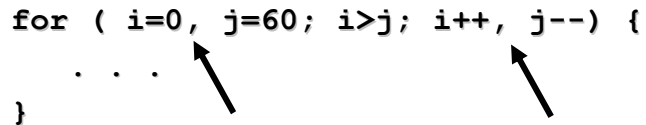
- **izraz1** je izraz koji će se izvršiti samo jednom, prije ulaska u prvu iteraciju. Najčešće se koristi za inicijalizaciju brojača. Ako je potrebno napisati više naredbi u izrazu, one se odvajaju zarezom.
- **izraz2** se izračunava kao logički izraz (0 - false, !=0 - true), te se petlja izvršava dok je **izraz2** zadovoljen (istinit). Provjera uvjeta obavlja se prije svake iteracije.
- **izraz3** se obavlja nakon svake iteracije (prolaska kroz petlju). Najčešće se koristi za povećavanje brojača. Ako je potrebno napisati više naredbi u izrazu, one se odvajaju zarezom.
- Bilo koji od izraza (izraz1, izraz2, izraz3) se može izostaviti. Ako je izostavljen izraz2, petlja se izvodi kao da je logička vrijednost izraza2 istinita (true).

86

Odvajanje naredbi zarezom u for petlji

- Zarez se kao operator koristi za odvajanje naredbi obično tamo gdje je dopuštena samo jedna naredba. Na primjer:

```
for ( i=0, j=60; i>j; i++, j-- ) {  
    . . .  
}
```



87

Primjeri petlji s poznatim brojem ponavljanja

- Primjer for petlje koja uzlazno mijenja kontrolnu varijablu :

```
for ( i = poc; i <= kraj; i = i + k ) {  
    .  
    .  
    .  
}
```
- Primjer for petlje koja silazno mijenja kontrolnu varijablu :

```
for ( cv = 10; cv >0; cv = cv - 1 ) {  
    .  
    .  
    .  
}
```
- Primjer for petlje s dva brojača:

```
for ( si = 100, uz = 0; si >= uz; si = si-1, uz = uz + 1 )  
{  
    .  
    .  
    .  
}
```

88

Napisati program koji će izračunati aritmetičku sredinu za n učitanih brojeva.

```
#include <stdio.h>
int main() {
    int    i, n, suma, x;
    float  arit_sred;
    printf("Za koliko brojeva želite izračunati"
           " aritmetičku sredinu : ");
    scanf("%d", &n );
    suma = 0;
    for(i = 0; i < n; i = i + 1) {
        printf("Unesite %d. broj : ", i);
        scanf("%d", &x);
        suma = suma + x;
    }
    arit_sred = (float) suma / n;
    printf("Aritmetička sredina učitanih brojeva"
           " je %f\n", arit_sred);
    return 0;
}
```

89

Napisati program koji će ispisati realne brojeve od 0 do n s korakom od 0.1

```
#include <stdio.h>
int main() {
    int n;
    float i;
    printf("Do kojeg broja zelite ispis :");
    scanf("%d", &n );
    for( i=0; i<=n; i=i+0.1 ) {
        printf("%f\n",i);
    }
    return 0;
}
```

90

Napisati program koji će ispisati brojeve djeljive sa 7, 13 ili 19, a manje od učitanoj broja n. Brojeve treba ispisati od najvećeg prema najmanjem.

```
#include <stdio.h>
int main() {
    int i, n;
    printf("Upisite broj n : ");
    scanf("%d", &n );
    /* za sve brojeve od n do 1 */
    for( i = n -1; i > 0; i = i -1 ) {
        if(( i % 7 == 0 ) ||
           ( i % 13 == 0 ) ||
           ( i % 19 == 0)) printf(" %d \n", i);
    }
    return 0;
}
```

91

Beskonačne petlje

- Beskonačna petlja
 - Tijelo petlje izvodi se beskonačno mnogo puta ako ne sadrži naredbu za izlazak iz petlje (**break**), naredbu za povratak iz funkcije (**return**), poziv funkcije za završetak programa (**exit**) ili **goto** naredbu.
- Primjeri:

```
while(1 == 1) { ... }
while(1) { ... }
do { ... } while(1 == 1);
...
```
- Primjer (loš):

```
for(;;) { ... }
```

92

Česte pogreške

Kod	Ispis
<pre>for(i=1; i = 10; i=i+1){ printf("%d\n",i); }</pre>	neprekidno ispisuje broj 10 (beskonačna petlja)
<pre>for(i=1; i == 10; i=i+1){ printf("%d\n",i); }</pre>	neće ništa ispisati (naredba u tijelu petlje se neće ni jednom obaviti)
<pre>for(i=1; i<=10; i=i+1); { printf("%d\n",i); }</pre>	jednom ispisuje broj 11

93

Napisati program koji će ispisati tablicu potencija 2^n i 2^{-n} za brojeve od n od 1 do 16

```
PotencijeBroja2
#include <stdio.h>
int main ( ) {
    short int i, x = 1;
    double y = 1.0;
    printf ( "%2d %12d %.16lf\n", 0, x, y);
    for( i = 1; i <= 16; i = i+1) {
        x = x*2;
        y = y/2;
        printf ( "%2d %12d %.16lf\n", i, x, y);
    }
    return 0;
}
```

94

Rezultati izvođenja programa

```

0          1 1.0000000000000000
1          2 0.5000000000000000
2          4 0.2500000000000000
3          8 0.1250000000000000
4         16 0.0625000000000000
5         32 0.0312500000000000
6         64 0.0156250000000000
7        128 0.0078125000000000
8        256 0.0039062500000000
9        512 0.0019531250000000
10       1024 0.0009765625000000
11       2048 0.0004882812500000
12       4096 0.0002441406250000
13       8192 0.0001220703125000
14      16384 0.0000610351562500
15     -32768 0.0000305175781250
16          0 0.0000152587890625
    
```

Zašto nije
dobar rezultat?

95

Komentar rezultata izvođenja programa

- Množenje s 10 u dekadskom sustavu:

$$153_{10} * 10_{10} = 1530_{10}$$

- Množenje s 2 u binarnom sustavu:

$$101_2 * 10_2 = 1010_2$$

- Što se je dogodilo?

$$0000000000000001_2 * 10_2$$

$$0000000000000010_2 * 10_2$$

...

$$0100000000000000_2 * 10_2$$

$$1000000000000000_2 * 10_2$$

$$0000000000000000_2$$

96

Primjer s operatorima nad bitovima

Primjer:  IzdvajanjeBitova

```
...
int a, i;
do {
    printf("Upisite cijeli broj a: ");
    scanf ("%d", &a);
    for (i = 31; i >= 0; i--) {
        printf("%d", (unsigned)a >> i & 0x1);
    }
    printf ("\n");
} while (a != 0);
...
```

97

Komentar rezultata izvođenja programa

- 0000...01011001 >> 2 & 0x1 → 0000...00000000
- 0000...01011001 >> 3 & 0x1 → 0000...00000001
- 0000...01011001 >> 4 & 0x1 → 0000...00000001

Zašto treba biti (unsigned)?

Zato jer se kod *signed* operanda prvi bit ("predznak") nikad ne posmiče na desno - prvi bit "ostaje na mjestu". Što se upisuje na mjesto drugog bita, ovisi o implementaciji:

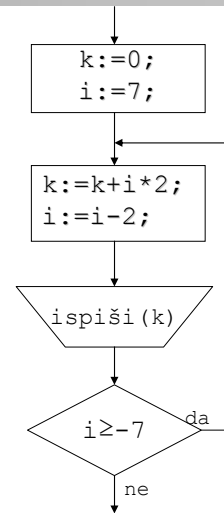
101110...01011001 >> 1 → 1101110...0101100

98

Primjeri realizacije istog algoritma raznim vrstama programskih petlji

- Sljedeći programski odsječak prikazan dijagramom toka treba nadomjestiti s:

- ispitivanjem uvjeta ponavljanja na početku
- ispitivanjem uvjeta ponavljanja na kraju
- unaprijed poznatim brojem ponavljanja



99

1. rješenje s `while`

```
k=0;
i=7;
while (i>=-7) {
    k=k+i*2;
    i=i-2;
    printf ("%d\n", k);
}
```

Napomena: ako bi prije ulaska u petlju bio $i < -7$ petlja se ne bi obavila niti jedanput.

100

2. rješenje s do-while

```
k=0;
i=7;
do {
    k=k+i*2;
    i=i-2;
    printf ("%d\n", k);
} while (i >= -7);
```

Napomena: ako bi prije ulaska u petlju bio $i < -7$ petlja bi se obavila jedanput.

101

3. rješenje s for

```
for (i = 7, k = 0; i >= -7; i = i - 2){
    k = k + i * 2;
    printf ("%d\n", k);
}
```

Prednost for petlje je u tome što se početna inicijalizacija brojača, ispitivanje uvjeta i korak brojača nalaze na jednom mjestu u kodu.

102

Program za izračunavanje N faktorijela 1. način

- Rješenje s programskom petljom u kojoj se uvjet ispituje na početku

```
učitaj (n)
fakt:=1.
i:=1
dok je i <= n
    fakt := fakt*i
    i := i+1
ispiši (n,fakt)
```

103

Rješenje u C-u (1.način)

```
ⓁFaktorijeliPocetak
#include <stdio.h>
int main () {
    int n, i;
    double fakt;
    scanf ("%d", &n);
    fakt = 1.;    i = 1;
    while (i <= n) {
        fakt = fakt * i;
        i = i + 1;
    }
    printf ("%d! = %g\n", n, fakt);
    return 0;
}
```

104

Program za izračunavanje N faktorijela 2. način

- Rješenje s programskom petljom u kojoj se uvjet ispituje na kraju:

```
ucitaj (n)
fakt := 1.
i := 1
ponavljaj
    fakt := fakt * i
    i := i+1
dok je i <= n
ispisi (n,fakt)
```

105

Rješenje u C-u (2. način)

```
ⓁFaktorijeliKraj
#include <stdio.h>
int main () {
    int n, i;
    double fakt;
    scanf ("%d", &n);
    fakt = 1.; i = 1;
    do {
        fakt = fakt * i;
        i = i + 1;
    } while (i <= n);
    printf ("%d! = %g\n", n, fakt);
    return 0;
}
```

106

Rješenje u C-u (2. način, uz korištenje operatora ++):

```
/* ne pisati ovako! */  
i=1; fakt=1.; do fakt *= i++; while (i <= n);
```



```
i = 1; fakt = 1.0;  
do {  
    fakt *= i++;  
} while (i <= n);
```



```
i = 1; fakt = 1.0;  
do {  
    fakt *= i;  
} while (++i <= n);
```

107

Program za izračunavanje N faktoriijela 3. način

- Rješenje s programskom petljom s poznatim brojem ponavljanja:

```
učitaj (n)  
fakt := 1.  
  
za i = 1 do n  
| fakt := fakt * i  
  
ispiši (n, fakt)
```

108

Program u C-u (3. način)

```

FaktorijeliPoznato
#include <stdio.h>
int main () {
    int n, i;
    double fakt;
    scanf ("%d", &n);
    fakt = 1.;
    for (i = 1; i <= n; i = i + 1){
        fakt = fakt * i;
    }
    printf ("%d! = %g\n", n, fakt);
    return 0;
}

```

109

Rezultati testiranja

<u>Ulaz:</u>	<u>Rezultat</u>
0	0! = 1
1	1! = 1
5	5! = 120
10	10! = 3628800
100	100! = 9.33262e+157
150	150! = 5.71338e+262
170	170! = 7.25742e+306
171	Floating point error: Overflow.

110

Primjer: Napisati program koji će ispisati tablicu množenja do 100.

```
/* 1*/ int main() {
/* 2*/ int i,j;
/* 3*/
/* 4*/ for (i = 1; i <= 10; ++i) {
/* 5*/     for (j = 1; j <= 10; ++j){
/* 6*/         printf("%3d",i*j);
/* 7*/     }
/* 8*/     printf("\n");
/* 9*/ }
/*10*/ return 0;
/*11*/}
```

111

Dodavanjem jedne linije koda postići to da se u tablici nalaze samo parni brojevi.

- Samo parne brojeve može se dobiti na dva načina:
 - Uvjet da su oba broja neparna
/*6*/ if ((i%2!=0) && (j%2!=0)) continue;
 - Uvjet da barem jedan od brojeva paran
/*6*/ if ((i%2==0) || (j%2==0))

112

Primjer : Napisati program koji će ispisivati prvih N Fibonaccijevih brojeva. N učitavati pomoću tipkovnice. Algoritam za računanje Fibonaccijevih brojeva:
Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2)
Fibonacci(0) = Fibonacci(1) = 1 → 1, 1, 2, 3, 5, 8, 13, 21,...

```
#include <stdio.h>
int main () {
    int N, i, f0 = 1, f1 = 1, f = 1;
    printf ("\n Upisite broj Fibonacci-jevih brojeva : \n");
    scanf ("%d",&N);
    for (i = 0; i <= N; i++) {
        if (i > 1) {
            f = f1 + f0;
            f0 = f1;
            f1 = f;
        }
        printf ("Fibonnaci (%d) = %d \n", i , f);
    }
    return 0;
}
```

113

Naredbe **break** i **continue**

- U C-u postoje dvije naredbe za kontrolu toka petlje
break prekida izvođenje najbliže vanjske programske petlje
continue unutar **while** i **do** petlje prebacuje izvođenje programa na ispitivanje uvjeta, a unutar **for** petlje prebacuje izvođenje programa na korak **for** petlje i potom na ispitivanje uvjeta (također se odnosi na najbližu vanjsku petlju)

114

Izračunati prosjek unaprijed nepoznatog broja pozitivnih cijelih brojeva (brojevi se učitavaju dok se ne unese 0) - rješenje bez **break** i **continue**

```
ProsjekBrojevaA
#include <stdio.h>
int main () {
    int suma, broj, n;
    suma = 0; n = 0;
    scanf ("%d", &broj);
    while (broj != 0) {
        if (broj > 0) {
            suma += broj;
            ++n;
        }
        scanf ("%d", &broj);
    }
    if (n > 0) printf("Prosjek =%f\n", (float) suma/n);
    return 0;
}
```

inicijalno čitanje

sva ostala čitanja

115

Izračunati prosjek unaprijed nepoznatog broja pozitivnih cijelih brojeva (brojevi se učitavaju dok se ne unese 0) - rješenje s **do-while**, bez **break** i **continue**

```
ProsjekBrojevaB
#include <stdio.h>
int main () {
    int suma, broj, n;
    suma = 0; n = 0;
    do {
        scanf ("%d", &broj);
        if (broj > 0) {
            suma += broj;
            ++n;
        }
    } while (broj != 0);
    if (n > 0) printf("Prosjek =%f\n", (float) suma/n);
    return 0;
}
```

najbolje rješenje

116

Izračunati prosjek unaprijed nepoznatog broja pozitivnih cijelih brojeva (brojevi se učitavaju dok se ne unese 0) -
rješenje s while i break i continue

```
ProsjekBrojevaC
#include <stdio.h>
int main () {
    int suma, broj, n;
    suma = 0; n = 0;
    while (1) {
        scanf ("%d", &broj);
        if (broj == 0) break;
        if (broj < 0) continue;
        suma += broj;
        ++n;
    }
    if (n > 0) printf("Prosjek =%f\n", (float)suma/n);
    return 0;
}
```

beskonačna petlja

117

Napisati program koji će provjeriti je li zadani broj prost broj

```
...
inti, n, prost=1;          /* prost je true */
printf("Upisite prirodan broj :");
scanf("%d", &n);

for ( i = 2; i <= n-1; i++) {
    if( n % i == 0 ) {      /* moze if(!(n % i)) */
        prost = 0;        /* prost je false */
        break;
    }
}
if( prost )
    printf("%d jest prost broj !\n", n);
else
    printf("%d nije prost broj !\n", n);
...
```

118

Napisati program koji će provjeriti je li zadani broj prost broj
Poboljšanja prethodnog rješenja

- Moguća poboljšanja algoritma (smanjivanje broja iteracija petlje):
 - Dovoljno je s petljom ići samo do \sqrt{n} (C funkcija `sqrt(n)`)
 - Ispitati djeljivost broja s 2, te ako nije djeljiv s 2, unutar petlje ispitivati djeljivost samo s neparnim brojevima većim od 2
-
- načiniti za vježbu!

119

Primjer : Što će se ispisati sljedećim programskim odsječkom?

```
i = 1;
while (i < 5) {
    if (i==3) {
        printf ("\n Hello world %d.x!", i);
        continue;
    } else if (i==4) {
        printf ("\n Goodbye %d.x!", i);
        continue;
    }
    i++;
}
```

Rješenje:

Hello world 3.x!

Hello world 3.x!

... i tako beskonačno puta. Kada i dostigne vrijednost 3 izvršava se blok naredbi pod "i==3". Zbog `continue` se i ne povećava nego se program grana na uvjetni izraz (`i < 5`).

120

Primjer kontrole učitanih brojeva

- Napisati program koji će učitavati cijele brojeve s tipkovnice i postupati prema sljedećem pravilu: ako je učitani broj manji od nule, treba ispisati poruku o pogrešci i prestati s učitavanjem brojeva. Ako je učitani broj veći od 100, treba ga zanemariti i prijeći na sljedeći broj, a inače treba ispisati taj broj. Osim u slučaju pogreške s učitavanjem brojeva prestati kada se učita (i ispiše) 0.

121

Primjer kontrole učitanih brojeva

```
...           Za vježbu riješiti bez break i continue. Dobit
int x;        ćete jednostavnije i znatno bolje rješenje!
do {
    printf ("Upisite broj :\n");
    scanf ("%d", &x );
    if (x < 0) {
        printf ("Nedopustena vrijednost\n");
        break;    /* Izlazak iz petlje */
    }
    if (x > 100) {
        printf ("Zanemarujem vrijednost\n");
        continue;    /* Skok na novu iteraciju */
    }
    printf ("Upisani broj je : %d\n", x);
} while (x != 0);
...
```

122

Naredba goto

Opći oblik naredbe:

```
goto oznaka_naredbe_1;
. . .
oznaka_naredbe_1:
programski odsječak
```

123

Naredba goto

- Primjer: Napisati programski odsječak koji će učitavati pozitivne brojeve dok su manji ili jednaki 100. Ukoliko se unese negativni broj poduzeti odgovarajuće korake.

```
. . .
/* programska petlja za čitanje pozitivnih brojeva */
scanf("%d", &x);
while (x <= 100) {
    if (x < 0) goto pogreska;
    . . .
    scanf("%d", &x);
}
. . .
/* odsječak koji rješava problem pogreške */
pogreska:
printf("POGREŠKA - NEGATIVAN BROJ");
. . .
```

124

Naredba `goto` i strukturirano programiranje

Primjer neprihvatljive uporabe `goto`

```
for (i = 0; i < 10; i++) {  
    programski odsječak;  
}
```

najbolje rješenje

↓

```
i = 0;  
while (i < 10) {  
    programski odsječak;  
    i++;  
}
```

prihvatljivo rješenje

↓

```
i = 0;  
opet:  
    if (i >= 10) goto dalje;  
    programski odsječak;  
    i++;  
    goto opet;  
dalje:  
...
```

neprihvatljivo rješenje

125

Naredba `goto` i strukturirano programiranje

Primjer prikladan za uporabu `goto`

```
for (...; uvjet1; ...) {  
    while (uvjet2) {  
        do {  
            ...  
            if (uvjet) nastaviti s odsječkom X;  
            ...  
        } while (uvjet3);  
    }  
}  
odsječak X
```

126

Naredba `goto` i strukturirano programiranje

Primjer korektne uporabe `goto`

```
for (...; uvjet1; ...) {
    while (uvjet2) {
        do {
            ...
            if (uvjet) goto van;
            ...
        } while (uvjet3);
    }
}
van:
odsječak X;
```

127

Naredba `goto` i strukturirano programiranje

Rješenje bez `goto`

```
int gotovo = 0;
for (...; uvjet1 && !gotovo; ...) {
    while (uvjet2 && !gotovo) {
        do {
            ...
            if (uvjet) {
                gotovo = 1;
                break;
            }
            ...
        } while (uvjet3);
    }
}
van:
odsječak X;
```

128

Naredba switch - skretnica

- može se upotrijebiti umjesto višestране if selekcije
- sintaksa:

```
switch( cjelobrojni_izraz ) {
    case const_izraz1: naredbe1;
    [case const_izraz2: naredbe2;]
    ...
    [default : naredbeN;]
}
```
- izraz unutar zagrada iza switch-a **mora** biti cjelobrojan
- izrazi iza **case** moraju biti cjelobrojni i sadrže samo konstante (npr. $3*7+'a'$ je O.K., $3*i$ ne valja)
- obratiti pozornost: ako se unutar case bloka ne navede ključna riječ **break**; nastavak programa je sljedeći case blok u listi!

129

Primjer

```
CaseBezBreak
#include <stdio.h>
int main () {
    char c;
    scanf ("%c", &c);
    switch (c) {
        case 'A':
            printf ("c = 'A'\n");
        case 'B':
            printf ("c = 'B'\n");
        default:
            printf ("Pogreška\n");
    }
    return 0;
}
```

Ulazni podatak: **A**
Rezultat izvođenja:

```
c = 'A'
c = 'B'
Pogreška
```

Ulazni podatak: **B**
Rezultat izvođenja:

```
c = 'B'
Pogreška
```

130

Rješenje sa switch i break

```
CaseSBreak
...
char c;
scanf ("%c", &c);
switch (c) {
    case 'A':
        printf ("c = A\n");
        break;
    case 'B':
        printf ("c = 'B'\n");
        break;
    default:
        printf ("Pogreška\n");
        break;
}
...
```


Ulazni podatak: B
Rezultat izvođenja:

```
c = 'B'
```

131

Primjer: realizacija skretnice koja sadrži break, korištenjem naredbe if - else if - else

```
switch (vr) {
    case C1:
        naredbe_1;
        break;
    case C2:
        naredbe_2;
        break;
    case Cn:
        naredbe_n;
        break;
    default:
        naredbe_n_plus1;
}
}
```



```
if (vr == C1) {
    naredbe_1;
} else if (vr == C2) {
    naredbe_2;
} else if ...
...
} else if (vr == Cn) {
    naredbe_n;
} else {
    naredbe_n_plus1;
}
```

132

Primjer: realizacija skretnice koja ne sadrži **break**, korištenjem naredbe **if**

```
switch (vr) {
  case C1:
    naredbe_1;
  case C2:
    naredbe_2;
  case Cn:
    naredbe_n;
  default:
    naredbe_n_plus1;
}

nadjen = 0;
if (vr == C1) {
  naredbe_1;
  nadjen = 1;
}
if (vr == C2 || nadjen) {
  naredbe_2;
  nadjen = 1;
}
...
if (vr == Cn || nadjen) {
  naredbe_n;
}
naredbe_n_plus1;
```

→

CaseBezBreakIf

133

Učitati s tipkovnice brojevu ocjenu (od 1 do 5), a ispisati njezinu opisnu vrijednost

```
...
int ocjena;
printf ("Upisite ocjenu :");
scanf ("%d", &ocjena);
switch (ocjena) {
  case 1:
    printf ("Nedovoljan\n"); break;
  case 2:
    printf ("Dovoljan\n"); break;
  case 3:
    printf ("Dobar\n"); break;
  case 4:
    printf ("Vrlo dobar\n"); break;
  case 5:
    printf ("Izvrstan\n"); break;
  default:
    printf ("Unijeli ste nepostojecu ocjenu\n");
    break; /* može i bez naredbe break */
}
...
```

134

Komentar rješenja

- **Napomena:** Ako bi izostavili `break;` unutar svih `case` blokova, za učitanu vrijednost npr. 3 ispis bi izgledao ovako:
Dobar
Vrlo dobar
Odlican
Unijeli ste nepostojeću ocjenu
- **Napomena:** Ako je `default` blok posljednji blok `switch` naredbe nije unutar njega potrebno pisati naredbu `break`.

135

"Propadanje" po `case` labelama može se iskoristiti onda kada se "ispod" nekoliko labela nalazi isti programski kôd

```
switch (znak) {  
    case 'a': brsamoglasnika++; break;  
    case 'e': brsamoglasnika++; break;  
    case 'i': brsamoglasnika++; break;  
    case 'o': brsamoglasnika++; break;  
    case 'u': brsamoglasnika++; break;  
    default: brostalih++; break;  
}
```



```
switch (znak) {  
    case 'a':  
    case 'e':  
    case 'i':  
    case 'o':  
    case 'u': brsamoglasnika++; break;  
    default: brostalih++; break;  
}
```

136