

Lekcija 4 – Kontrola programa

Pregled

- 4.1 Uvod
- 4.2 Osnovni pojmovi ponavljanja
- 4.3 Petlja kontrolisana brojačem
- 4.4 for naredba
- 4.5 for naredba: napomene
- 4.6 Primjeri upotrebe for petlje
- 4.7 switch naredba
- 4.8 do...while naredba
- 4.9 break i continue naredbe
- 4.10 Logički operatori
- 4.11 Konfuzija sa (==) i (=) operatorima
- 4.12 Struktuirano programiranje - zaključak

Ciljevi lekcije

- U ovoj lekciji:
 - Naučićete kako da koristite `for` i `do...while` naredbe.
 - Shvatićete i koristiti `switch` naredbu.
 - Koristićete `break` i `continue` kontrolne naredbe
 - Naučićete upotrebu logičkih operatora.

4.1 Uvod

- U ovoj lekciji uvodimo:
 - Dodatne naredbe za ponavljanje
 - `for`
 - `do...while`
 - `switch` naredbu
 - `break` naredbu
 - Za izlazak iz kontrolanih struktura
 - `continue` statement
 - Za preskakanje ostatka tijela kontrolne strukture i nastavljjanje u sledećoj iteraciji

4.2 Osnovi ponavljanja

- Petlje
 - Grupa instrukcija koja se ponavlja dok je neki uslov tačan
- Petlja kontrolisana brojačem
 - Konačna petlja: poznat broj ponavljanja
 - Kontrolna promjenljiva broji ponavljanja
- Petlja kontrolisan sentinelom
 - Nedefinisan broj ponavljanja
 - Sentinel označava kraj unosa podataka

4.3 Osnovi ponavljanja

- Petlje kontrolisane brojačem zahtijevaju
 - Ime kontrolne promjenljive (brojač)
 - Početnu vrijednost brojača
 - Inkrement (ili dekrement) koji mijenja kontrolnu promjenljivu u svakoj iteraciji
 - Uslov koji testira vrijednost brojača

4.3 Osnovi ponavljanja

- Primjer:

```
int counter = 1;           // initialization
while ( counter <= 10 ) { // repetition condition
    printf( "%d\n", counter );
    ++counter;              // increment
}
```

- Naredba

```
int counter = 1;
```

- Daje ime brojaču counter
- Definiše da je cio broj
- Rezerviše memorijski prostor
- Postavlja početnu vrijednost na 1



Outline

fig04_01.c

```
1  /* Fig. 4.1: fig04_01.c
2      Counter-controlled repetition */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8      int counter = 1;           /* initialization */
9
10     while ( counter <= 10 ) {   /* repetition condition */
11         printf ( "%d\n", counter ); /* display counter */
12         ++counter;              /* increment */
13     } /* end while */
14
15     return 0; /* indicate program ended successfully */
16
17 } /* end function main */
```

Program Output

```
1
2
3
4
5
6
7
8
9
10
```

4.3 Osnovi ponavljanja

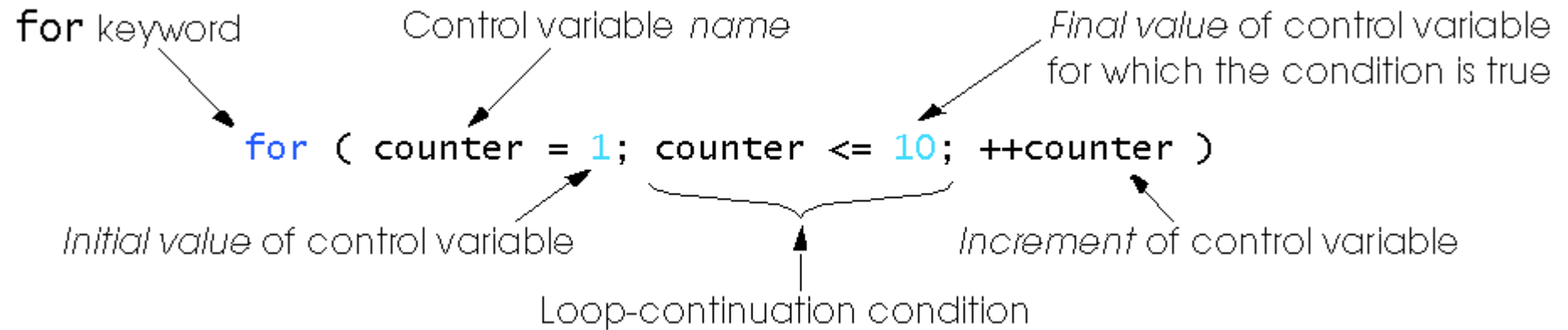
- Skraćeni kod
 - C programeri vole da pišu koncizne programe
 - Inicijalizovati counter na 0
 - ```
while (++counter <= 10)
 printf("%d\n", counter);
```





```
1 /* Fig. 4.2: fig04_02.c
2 Counter-controlled repetition with the for statement */
3 #include <stdio.h>
4
5 /* function main begins program execution */
6 int main()
7 {
8 int counter; /* define counter */
9
10 /* initialization, repetition condition, and increment
11 are all included in the for statement header. */
12 for (counter = 1; counter <= 10; counter++) {
13 printf("%d\n", counter);
14 } /* end for */
15
16 return 0; /* indicate program ended successfully */
17
18 } /* end function main */
```

## 4.4 for naredba



## 4.4 for naredba

- Format for petlje

```
for (initialization; loopContinuationTest; increment)
 statement
```

- Primjer:

```
for(int counter = 1; counter <= 10; counter++)
 printf("%d\n", counter);
```

- Štampa cijele brojeve od 1 do 10



Nema (;)



## 4.4 for naredba

- For petlja može se zapisati kao while petlja:

```
initialization;
while (loopContinuationTest) {
 statement;
 increment;
}
```

- Inicijalizacija i inkrementiranje

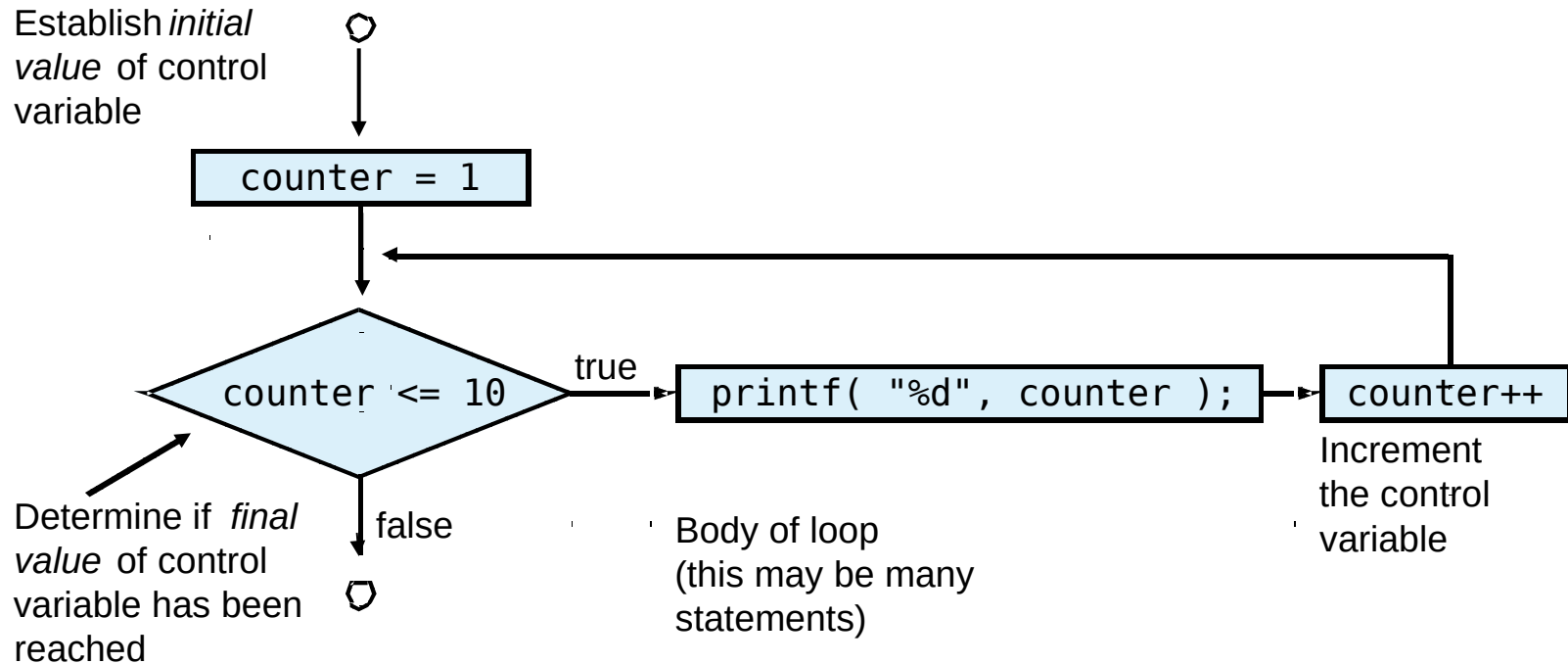
- Može biti lista naredbi razdvojena zarezima
- Primjer:

```
for (int i = 0, j = 0; j + i <= 10; j++, i++)
 printf("%d\n", j + i);
```

## 4.5 for naredba : napomene

- Aritmetički izrazi
  - Inicijalizacija, nastavak petlje i inkrementiranje mogu sadržati aritmetičke izraze. Ako je  $x$  jednako 2 i  $y$  jednako 10  
$$\text{for ( } j = x; j \leq 4 * x * y; j += y / x \text{ )}$$
je ekvivalentno sa  
$$\text{for ( } j = 2; j \leq 80; j += 5 \text{ )}$$
- Napomene u vezi for naredbe:
  - "Inkrementiranje" može biti negativno (dekrementiranje)
  - Ako je uslov u startu netačan (`false`)
    - Tijelo for naredbe se ne izvodi
    - Kontrola nastavlja sa naredbom iza for naredbe
  - Kontrolne promjenljive
    - Često se štampa ili koristi unutar tijela, ali ne obavezno

## 4.5 for naredba: napomene





## Outline

fig04\_05.c

```
1 /* Fig. 4.5: fig04_05.c
2 Summation with for */
3 #include <stdio.h>
4
5 /* function main begins program execution */
6 int main()
7 {
8 int sum = 0; /* initialize sum */
9 int number; /* number to be added to sum */
10
11 for (number = 2; number <= 100; number += 2) {
12 sum += number; /* add number to sum */
13 } /* end for */
14
15 printf("Sum is %d\n", sum); /* output sum */
16
17 return 0; /* indicate program ended successfully */
18
19 } /* end function main */
```

Sum is 2550

Program Output



## Outline

### fig04\_06.c (Part 1 of 2)

```
1 /* Fig. 4.6: fig04_06.c
2 Calculating compound interest */
3 #include <stdio.h>
4 #include <math.h>
5
6 /* function main begins program execution */
7 int main()
8 {
9 double amount; /* amount on deposit */
10 double principal = 1000.0; /* starting principal */
11 double rate = .05; /* interest rate */
12 int year; /* year counter */
13
14 /* output table column head */
15 printf("%4s%21s\n", "Year", "Amount on deposit");
16
17 /* calculate amount on deposit for each of ten years */
18 for (year = 1; year <= 10; year++) {
19
20 /* calculate new amount for specified year */
21 amount = principal * pow(1.0 + rate, year);
22
23 /* output one table row */
24 printf("%4d%21.2f\n", year, amount);
25 } /* end for */
26
```



```
27 return 0; /* indicate program ended successfully */
28
29 } /* end function main */
```

| Year | Amount on deposit |
|------|-------------------|
| 1    | 1050.00           |
| 2    | 1102.50           |
| 3    | 1157.63           |
| 4    | 1215.51           |
| 5    | 1276.28           |
| 6    | 1340.10           |
| 7    | 1407.10           |
| 8    | 1477.46           |
| 9    | 1551.33           |
| 10   | 1628.89           |



## Outline



**fig04\_06.c (Part 2  
of 2)**

**Program Output**

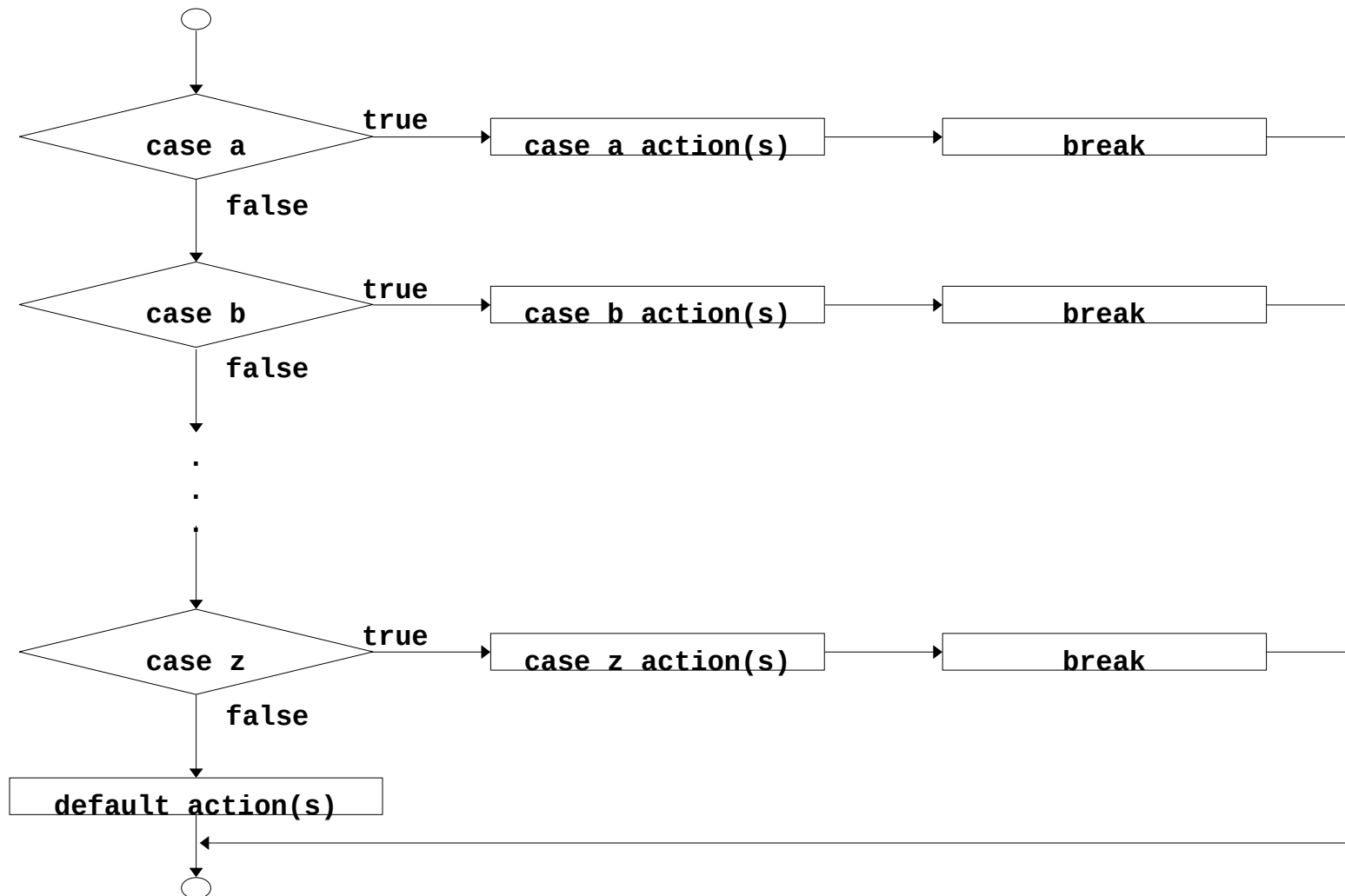
## 4.7 switch naredba

- **switch**
  - Koristi se kada se promjenljiva ili izraz testira na sve vrijednosti koje može imati i za svaku vrijednost se preduzima različita akcija
- **Format**
  - Niz case labela i opcionalni default

```
switch (value){
 case '1':
 actions
 case '2':
 actions
 default:
 actions
}
```
  - `break;` izlazi iz naredbe

## 4.7 switch naredba

- Flowchart za switch naredbu





## Outline

### fig04\_07.c (Part 1 of 3)

```
1 /* Fig. 4.7: fig04_07.c
2 Counting letter grades */
3 #include <stdio.h>
4
5 /* function main begins program execution */
6 int main()
7 {
8 int grade; /* one grade */
9 int aCount = 0; /* number of As */
10 int bCount = 0; /* number of Bs */
11 int cCount = 0; /* number of Cs */
12 int dCount = 0; /* number of Ds */
13 int fCount = 0; /* number of Fs */
14
15 printf("Enter the letter grades.\n");
16 printf("Enter the EOF character to end input.\n");
17
18 /* loop until user types end-of-file key sequence */
19 while ((grade = getchar()) != EOF) {
20
21 /* determine which grade was input */
22 switch (grade) { /* switch nested in while */
23
24 case 'A': /* grade was uppercase A */
25 case 'a': /* or lowercase a */
26 ++aCount; /* increment aCount */
27 break; /* necessary to exit switch */
28
```



## Outline

### fig04\_07.c (Part 2 of 3)

```
29 case 'B': /* grade was uppercase B */
30 case 'b': /* or lowercase b */
31 ++bCount; /* increment bCount */
32 break; /* exit switch */
33
34 case 'C': /* grade was uppercase C */
35 case 'c': /* or lowercase c */
36 ++cCount; /* increment cCount */
37 break; /* exit switch */
38
39 case 'D': /* grade was uppercase D */
40 case 'd': /* or lowercase d */
41 ++dCount; /* increment dCount */
42 break; /* exit switch */
43
44 case 'F': /* grade was uppercase F */
45 case 'f': /* or lowercase f */
46 ++fCount; /* increment fCount */
47 break; /* exit switch */
48
49 case '\n': /* ignore newlines, */
50 case '\t': /* tabs, */
51 case ' ': /* and spaces in input */
52 break; /* exit switch */
53
```



```
54 default: /* catch all other characters */
55 printf("Incorrect letter grade entered.");
56 printf(" Enter a new grade.\n");
57 break; /* optional; will exit switch anyway */
58 } /* end switch */
59
60 } /* end while */
61
62 /* output summary of results */
63 printf("\nTotals for each letter grade are:\n");
64 printf("A: %d\n", aCount); /* display number of A grades */
65 printf("B: %d\n", bCount); /* display number of B grades */
66 printf("C: %d\n", cCount); /* display number of C grades */
67 printf("D: %d\n", dCount); /* display number of D grades */
68 printf("F: %d\n", fCount); /* display number of F grades */
69
70 return 0; /* indicate program ended successfully */
71
72 } /* end function main */
```

[Outline](#)**Program Output**

```
Enter the letter grades.
Enter the EOF character to end input.
a
b
c
C
A
d
f
C
E
Incorrect letter grade entered. Enter a new grade.
D
A
b
^Z

Totals for each letter grade are:
A: 3
B: 2
C: 3
D: 2
F: 1
```

## 4.8 do...while naredba

- do...while naredba
  - Slična while naredbi
  - Uslov se testira poslije izvršavanja tijela petlje
    - Sve akcije izvrše se bar jednom
  - Format:

```
do {
 statement;
} while (condition);
```



## 4.8 do...while naredba

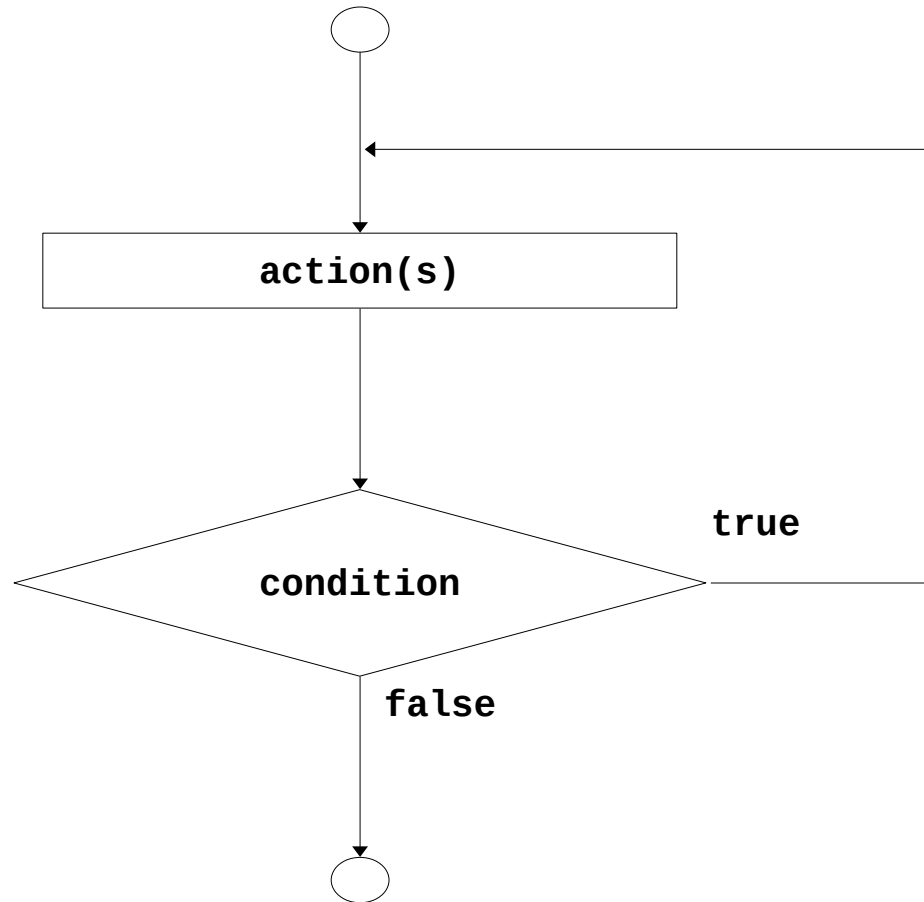
- Primjer (pretpostavimo da je counter = 1):

```
do {
 printf("%d ", counter);
} while (++counter <= 10);
```

  - Štampa cijele brojeve od 1 do 10

## 4.8 do...while naredba

- Flowchart do...while naredbe





## Outline

fig04\_09.c

```
1 /* Fig. 4.9: fig04_09.c
2 Using the do/while repetition statement */
3 #include <stdio.h>
4
5 /* function main begins program execution */
6 int main()
7 {
8 int counter = 1; /* initialize counter */
9
10 do {
11 printf("%d ", counter); /* display counter */
12 } while (++counter <= 10); /* end do...while */
13
14 return 0; /* indicate program ended successfully */
15
16 } /* end function main */
```

Program Output

1 2 3 4 5 6 7 8 9 10

## 4.9 break i continue naredbe

- break
  - Trenutni izlazak iz while, for, do...while ili switch naredbe
  - Program nastavlja izvršavanje sa prvom naredbom poslije kontrolne strukture
  - Uobičajene upotrebe break naredbe
    - Rani izlazak iz petlje
    - Preskakanje ostatka switch naredbe



## Outline

fig04\_11.c

```
1 /* Fig. 4.11: fig04_11.c
2 Using the break statement in a for statement */
3 #include <stdio.h>
4
5 /* function main begins program execution */
6 int main()
7 {
8 int x; /* counter */
9
10 /* loop 10 times */
11 for (x = 1; x <= 10; x++) {
12
13 /* if x is 5, terminate loop */
14 if (x == 5) {
15 break; /* break loop only if x is 5 */
16 } /* end if */
17
18 printf("%d ", x); /* display value of x */
19 } /* end for */
20
21 printf("\nBroke out of loop at x == %d\n", x);
22
23 return 0; /* indicate program ended successfully */
24
25 } /* end function main */
```

```
1 2 3 4
Broke out of loop at x == 5
```

Program Output

## 4.9 break i continue naredba

- `continue`
  - Preskače ostatak naredbi u tijelu `while`, `for` ili `do...while` naredbe
    - Nastavlja sa sledećom iteracijom petlje
  - `while` i `do...while`
    - Uslov za nastavak petlje se izračunava neposredno po izvršavanju `continue` naredbe
  - `for`
    - Izraz inkrementiranja se izvršava, pa se zatim izvršava provjera uslova



## Outline

fig04\_12.c

```
1 /* Fig. 4.12: fig04_12.c
2 Using the continue statement in a for statement */
3 #include <stdio.h>
4
5 /* function main begins program execution */
6 int main()
7 {
8 int x; /* counter */
9
10 /* loop 10 times */
11 for (x = 1; x <= 10; x++) {
12
13 /* if x is 5, continue with next iteration of loop */
14 if (x == 5) {
15 continue; /* skip remaining code in loop body */
16 } /* end if */
17
18 printf("%d ", x); /* display value of x */
19 } /* end for */
20
21 printf("\nUsed continue to skip printing the value 5\n");
22
23 return 0; /* indicate program ended successfully */
24
25 } /* end function main */
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing the value 5
```

**Program Output**

## 4.10 Logički operatori

- `&&` ( logičko AND )
  - Vraća `true` ako su oba uslova tačna (`true`)
- `||` (logičko OR )
  - Vraća `true` ako je bar jedan od uslova tačan (`true`)
- `!` (logičko NOT, logička negacija )
  - Preokreće tačnost izraza
  - Unarni operator
- Najčešće kao uslovi u kontrolnim strukturama

| <u>Expression</u>                  | <u>Result</u>      |
|------------------------------------|--------------------|
| <code>true &amp;&amp; false</code> | <code>false</code> |
| <code>true    false</code>         | <code>true</code>  |
| <code>!false</code>                | <code>true</code>  |



## 4.10 Logički operatori

| expression1 | expression2 | expression1 && expression2 |
|-------------|-------------|----------------------------|
| 0           | 0           | 0                          |
| 0           | nonzero     | 0                          |
| nonzero     | 0           | 0                          |
| nonzero     | nonzero     | 1                          |

Fig. 4.13 Truth table for the && (logical AND) operator.

| expression1 | expression2 | expression1    expression2 |
|-------------|-------------|----------------------------|
| 0           | 0           | 0                          |
| 0           | nonzero     | 1                          |
| nonzero     | 0           | 1                          |
| nonzero     | nonzero     | 1                          |

Fig. 4.14 Truth table for the logical OR (||) operator.

| expression | ! expression |
|------------|--------------|
| 0          | 1            |
| nonzero    | 0            |

Fig. 4.15 Truth table for operator ! (logical negation).

## 4.10 Logički operatori

| Operators |    |    |    |    |        | Associativity | Type           |
|-----------|----|----|----|----|--------|---------------|----------------|
| ++        | -- | +  | -  | !  | (type) | right to left | unary          |
| *         | /  | %  |    |    |        | left to right | multiplicative |
| +         | -  |    |    |    |        | left to right | additive       |
| <         | <= | >  | >= |    |        | left to right | relational     |
| ==        | != |    |    |    |        | left to right | equality       |
| &&        |    |    |    |    |        | left to right | logical AND    |
|           |    |    |    |    |        | left to right | logical OR     |
| ?:        |    |    |    |    |        | right to left | conditional    |
| =         | += | -= | *= | /= | %=     | right to left | assignment     |
| ,         |    |    |    |    |        | left to right | comma          |

Fig. 4.16 Operator precedence and associativity.

## 4.11 Konfuzija sa (==) i (=) operatorima

- Opasna greška
  - Najčešće se ne prijavljuje kao sintaksna greška
  - Bilo koji izraz koji daje vrijednost može se koristiti u kontrolnim strukturama
  - Nenulta vrijednost kao `true`, nula kao `false`
  - Primjer upotrebe `==`:

```
if (payCode == 4)
 printf("You get a bonus!\n");
```

    - Provjeri `payCode`, ako je 4 tada se dodjeljuje bonus

## 4.11 Konfuzija sa (==) i (=) operatorima

- Primjer, zamjena == sa =:

```
if (payCode = 4)
 printf("You get a bonus!\n");
```

- payCode postaje 4
  - 4 nije nula, pa je izraz tačan i bonus se dodjeljuje bez obzira na vrijednost promjenljive payCode
- Logička greška, a ne sintaksna

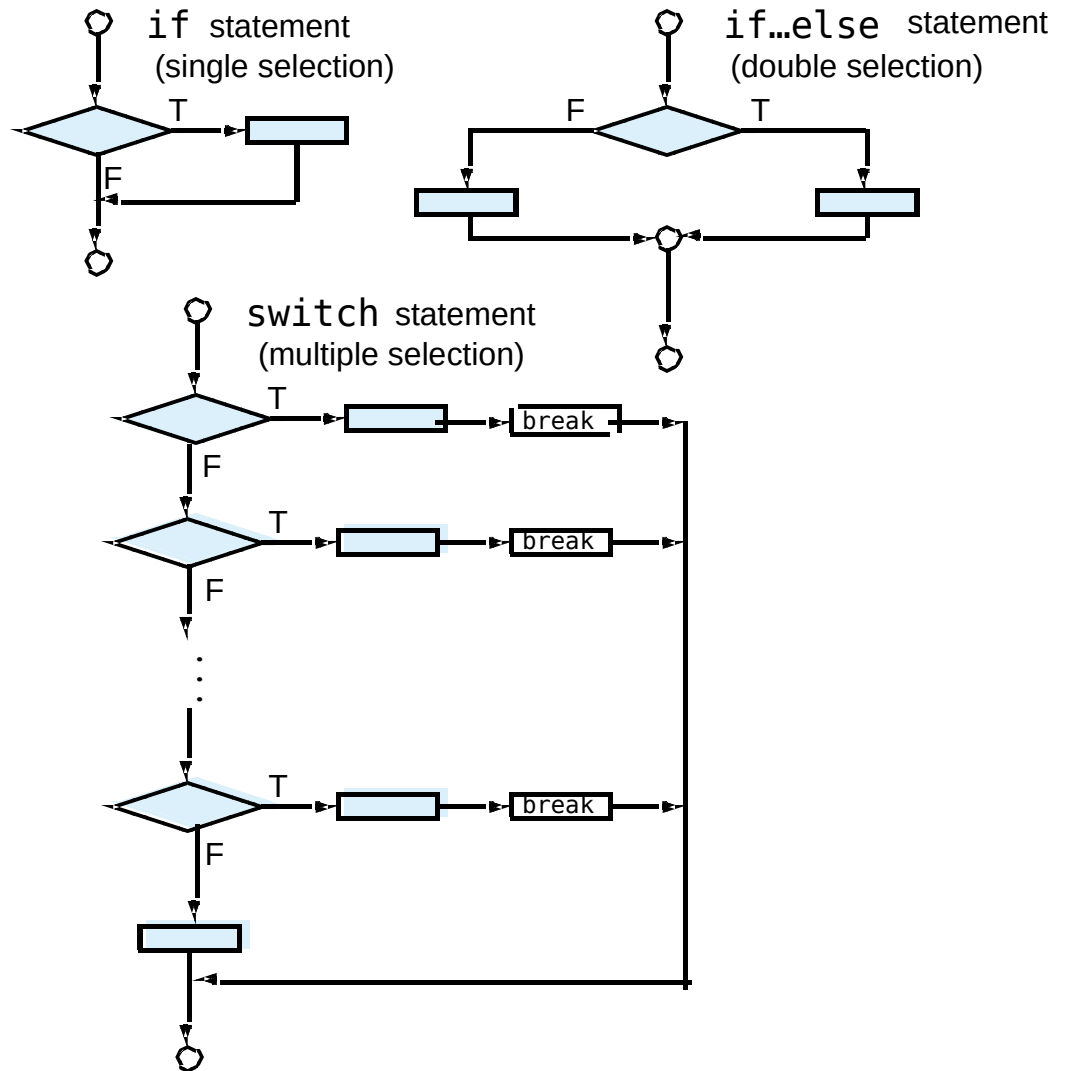
## 4.11 Konfuzija sa (==) i (=) operatorima

- lvalues
  - Izrazi koji se mogu pojaviti na lijevoj strani pri dodjeli
  - Njihova vrijednost može biti promijenjena
    - `x = 4;`
- rvalues
  - Izrazi koji se mogu pojaviti samo na desnoj strani pri dodjeli
  - Konstante, kao što su brojevi
    - Ne možemo napisati `4 = x;`
    - Moramo napisati `x = 4;`
  - lvalues može biti upotrebljena kao rvalues, ali obrnuto nije moguće
    - `y = x;`

## 4.12 Struktuirano programiranje - pregled

Sequence

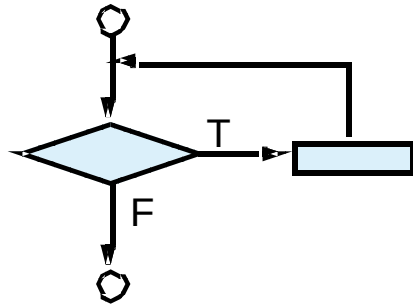
Selection



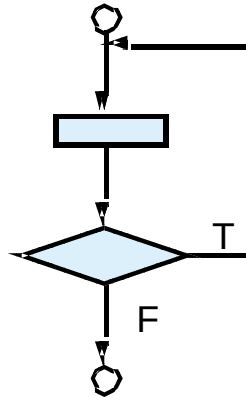
## 4.12 Struktuirano programiranje - pregled

### Repetition

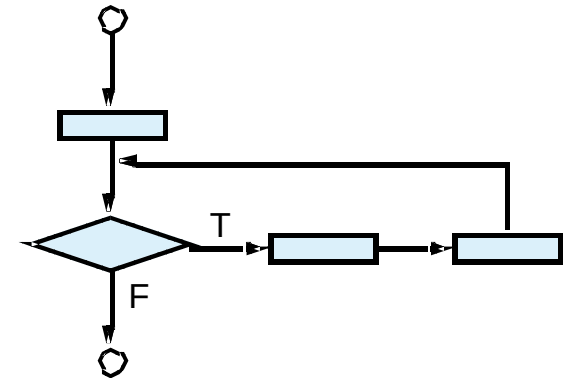
while statement



do..while statement



for statement



## 4.12 Struktuirano programiranje - pregled

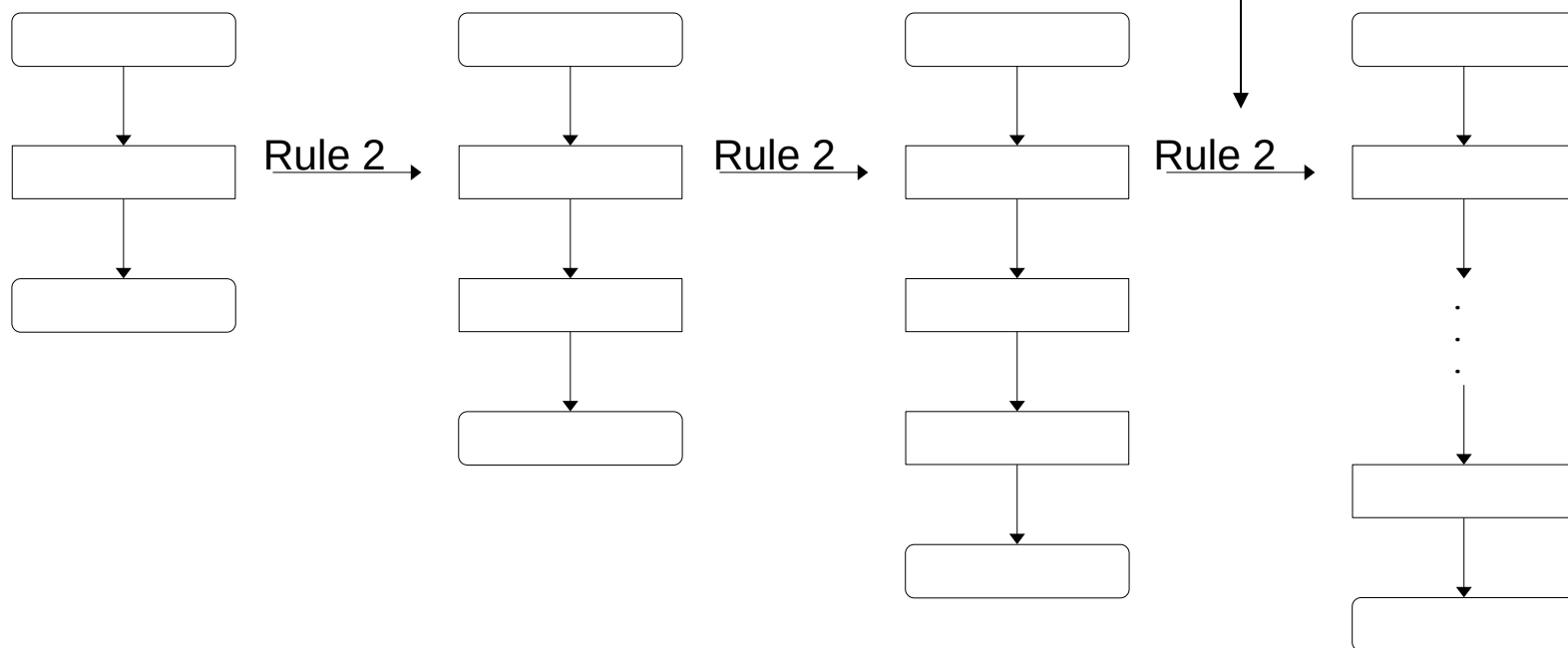
- Struktuirano programiranje
  - Lakše za razumijevanje, lakše za testiranje, otklanjanje grešaka i mijenjanje programa
- Pravila za struktuirano programiranje
  - Pravila koja je uspostavila programerska zajednica
  - Koriste se samo single-entry/single-exit kontrolne strukture
  - Pravila:
    1. Počinjemo sa “simplest flowchart”
    2. Pravilo slaganja (stacking rule): svaki pravougaonik (akcija) može se zamijeniti sa 2 uzastopna pravougaonika (akcije)
    3. Pravilo ugnježdavanja (nesting rule): svaki pravougaonik (akcija) može biti zanižen bilo kojom kontrolnom strukturom (sekvencijalnom, `if`, `if...else`, `switch`, `while`, `do...while` ili `for`)
    4. Pravila 2 i 3 možemo primjenjivati u bilo kom redu više puta



## 4.12 Struktuirano programiranje - pregled

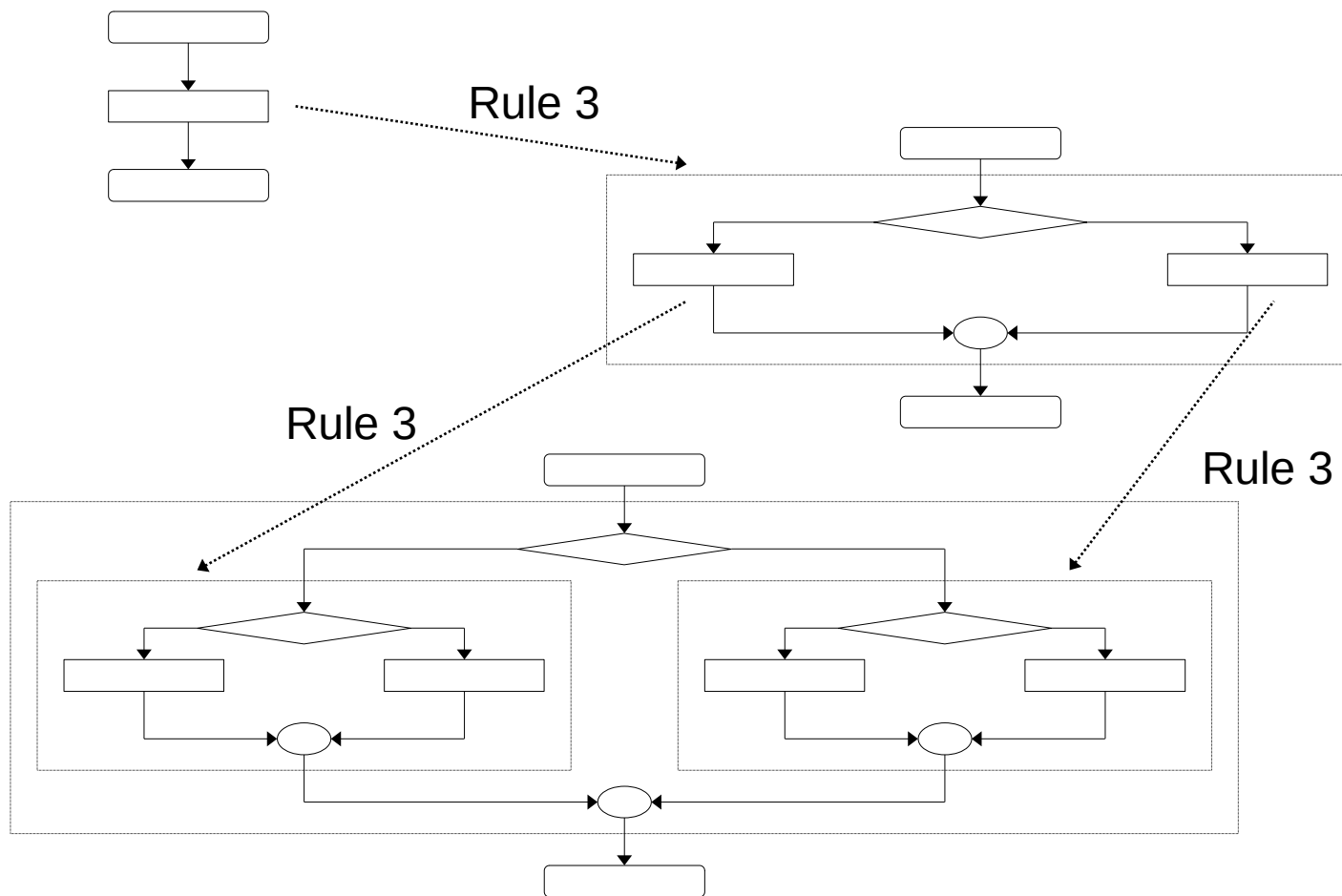
Pravilo 1 – Počinjemo najprostijom šemom

Pravilo 2 – svaku akciju možemo zamjeniti sa 2 uzastopne (sekvencijalne) akcije



## 4.12 Struktuirano programiranje - pregled

Pravilo 3 – Svaku akciju možemo zamjeniti kontrolnom strukturom

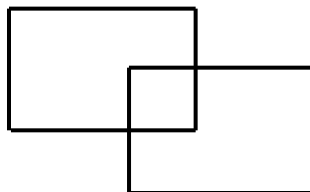


## 4.12 Struktuirano programiranje - pregled

Stacked building blocks

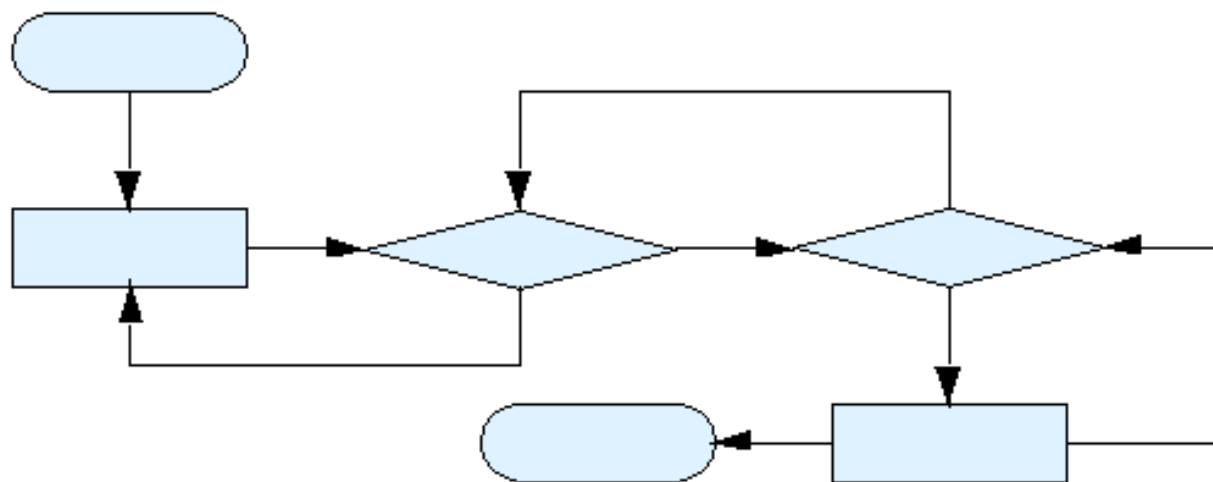
Nested building blocks

Overlapping building blocks  
(Illegal in structured programs)



## 4.12 Struktuirano programiranje - pregled

Figure 4.23 Nestruktuirana šema (flowchart).



## 4.12 Struktuirano programiranje - pregled

- Svi programi mogu biti razbijeni na 3 kontrolne strukture
  - Sekvencijana – automatski je obrađuje kompajler
  - Selekcija – `if`, `if...else` ili `switch`
  - Repeticija – `while`, `do...while` ili `for`
    - Mogu biti kombinovane na samo 2 načina
      - Ugnježdavanje (pravilo 3)
      - Slaganje (pravilo 2)
  - Svaka selekcija može biti pretvorena u `if` naredbu, i svaka repeticija može biti pretvorena u `while` naredbu