Lekcija 4 – Kontrola programa

Pregled

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 nastavljanje 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 ponavljajnja
  - 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 promjeniljivu 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

```c
/* Fig. 4.1: fig04_01.c
   Counter-controlled repetition */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int counter = 1;              /* initialization */

   while ( counter <= 10 ) {      /* repetition condition */
      printf ( "%d\n", counter ); /* display counter */
      ++counter;                  /* increment */
   } /* end while */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

Program Output

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

4.3 Osnovi ponavljanja

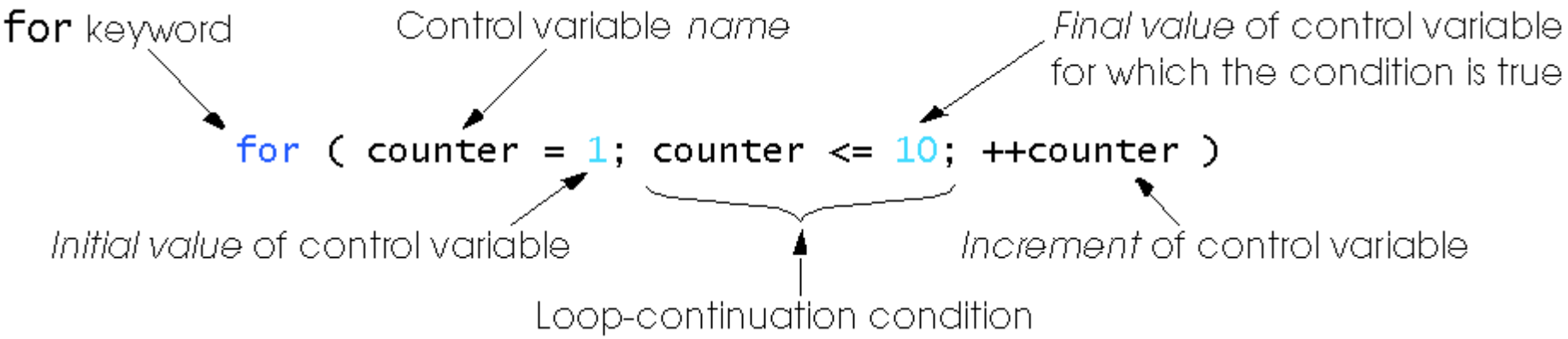
- Skraćeni kod
  - C programeri vole da pišu koncizne programe
  - Inicijalizovati `counter` na `0`
    -
```
while ( ++counter <= 10 )
        printf( “%d\n, counter );
```

Outline

fig04_02.c

```c
/* Fig. 4.2: fig04_02.c
   Counter-controlled repetition with the for statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int counter; /* define counter */

   /* initialization, repetition condition, and increment
      are all included in the for statement header. */
   for ( counter = 1; counter <= 10; counter++ ) {
      printf( "%d\n", counter );
   } /* end for */

   return 0; /* indicate program ended successfully */

} /* 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`

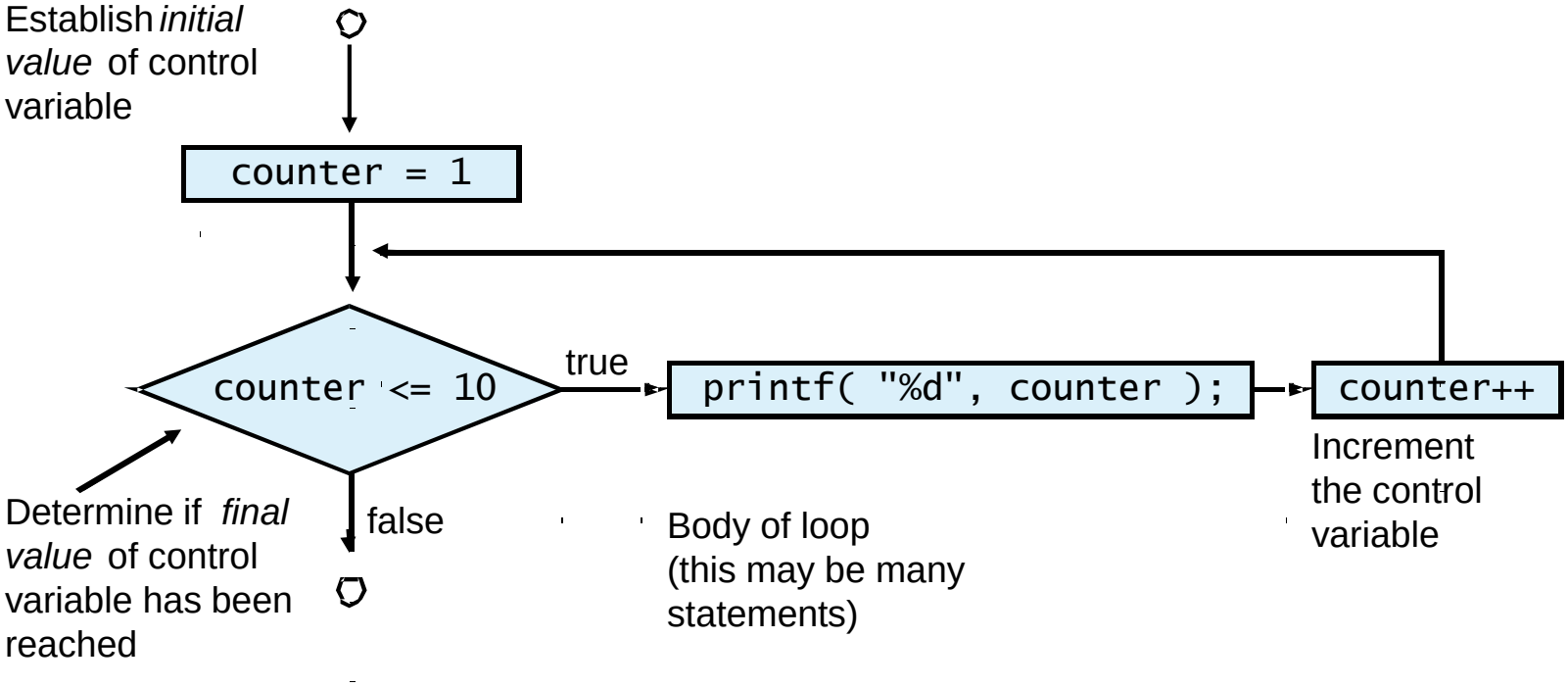
```
for ( j = x; j <= 4 * x * y; j += y / x )
```

je ekvivalentno sa

```
for ( j = 2; j <= 80; j += 5 )
```

- 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

```c
/* Fig. 4.5: fig04_05.c
   Summation with for */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int sum = 0; /* initialize sum */
   int number;  /* number to be added to sum */

   for ( number = 2; number <= 100; number += 2 ) {
      sum += number; /* add number to sum */
   } /* end for */

   printf( "Sum is %d\n", sum ); /* output sum */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

Program Output

```
Sum is 2550
```

Outline

fig04_06.c (Part 1 of 2)

```c
/* Fig. 4.6: fig04_06.c
   Calculating compound interest */
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main()
{
   double amount;             /* amount on deposit */
   double principal = 1000.0; /* starting principal */
   double rate = .05;         /* interest rate */
   int year;                  /* year counter */

   /* output table column head */
   printf( "%4s%21s\n", "Year", "Amount on deposit" );

   /* calculate amount on deposit for each of ten years */
   for ( year = 1; year <= 10; year++ ) {

      /* calculate new amount for specified year */
      amount = principal * pow( 1.0 + rate, year );

      /* output one table row */
      printf( "%4d%21.2f\n", year, amount );
   } /* end for */

```

```c
   return 0; /* indicate program ended successfully */

} /* 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
```


fig04_06.c (Part 2 of 2)

Program Output

4.7 switch naredba

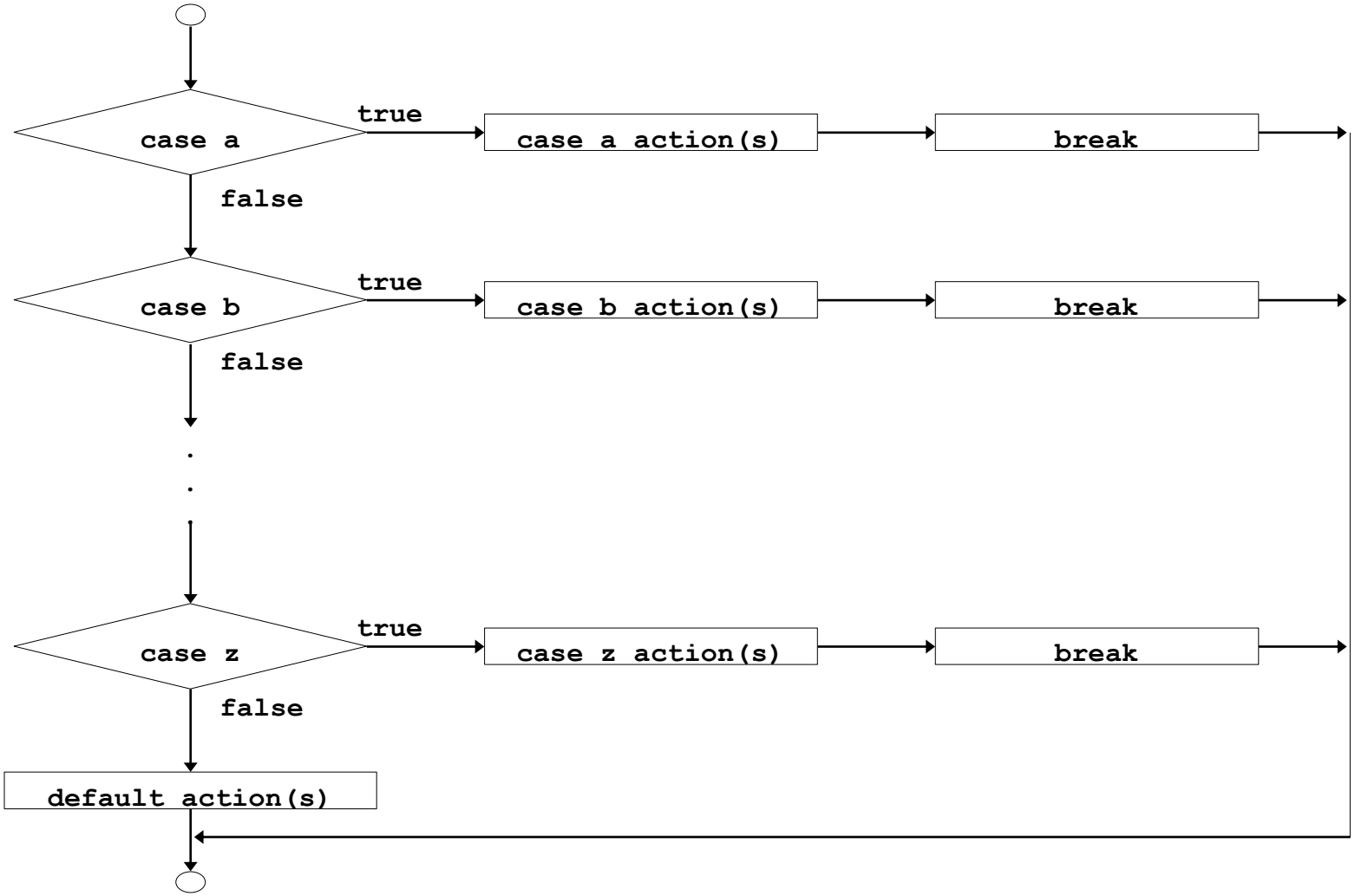
- `switch`
  - Koristi se kada se promjenljiva ili izraz testira na sve vrijednosti koje može imati i za svaku vrijednost se preduzima različita akcija
- Format
  - Niz `case` labela i opcionalni `default`

```
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)

```c
/* Fig. 4.7: fig04_07.c
   Counting letter grades */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int grade;      /* one grade */
   int aCount = 0; /* number of As */
   int bCount = 0; /* number of Bs */
   int cCount = 0; /* number of Cs */
   int dCount = 0; /* number of Ds */
   int fCount = 0; /* number of Fs */

   printf( "Enter the letter grades.\n" );
   printf( "Enter the EOF character to end input.\n" );

   /* loop until user types end-of-file key sequence */
   while ( ( grade = getchar() ) != EOF ) {

      /* determine which grade was input */
      switch ( grade ) { /* switch nested in while */

         case 'A':    /* grade was uppercase A */
         case 'a':    /* or lowercase a */
            ++aCount; /* increment aCount */
            break;    /* necessary to exit switch */

```

fig04_07.c (Part 2 of 3)

```
      case 'B':    /* grade was uppercase B */
      case 'b':    /* or lowercase b */
         ++bCount; /* increment bCount */
         break;    /* exit switch */

      case 'C':    /* grade was uppercase C */
      case 'c':    /* or lowercase c */
         ++cCount; /* increment cCount */
         break;    /* exit switch */

      case 'D':    /* grade was uppercase D */
      case 'd':    /* or lowercase d */
         ++dCount; /* increment dCount */
         break;    /* exit switch */

      case 'F':    /* grade was uppercase F */
      case 'f':    /* or lowercase f */
         ++fCount; /* increment fCount */
         break;    /* exit switch */

      case '\n':   /* ignore newlines, */
      case '\t':   /* tabs, */
      case ' ':    /* and spaces in input */
         break;    /* exit switch */

```

```
        default:     /* catch all other characters */
           printf( "Incorrect letter grade entered." );
           printf( " Enter a new grade.\n" );
           break;    /* optional; will exit switch anyway */
      } /* end switch */

   } /* end while */

   /* output summary of results */
   printf( "\nTotals for each letter grade are:\n" );
   printf( "A: %d\n", aCount ); /* display number of A grades */
   printf( "B: %d\n", bCount ); /* display number of B grades */
   printf( "C: %d\n", cCount ); /* display number of C grades */
   printf( "D: %d\n", dCount ); /* display number of D grades */
   printf( "F: %d\n", fCount ); /* display number of F grades */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

fig04_07.c (Part 3 of 3)

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

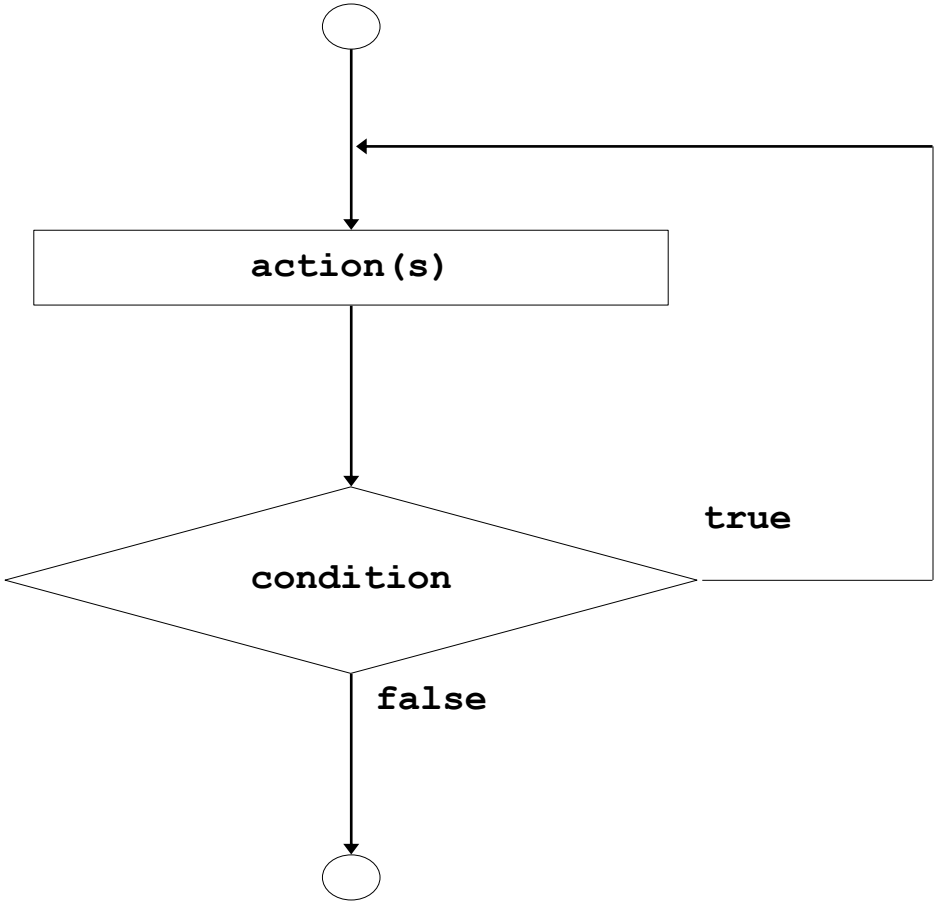
- Primjer (pretpostavimo da je counter = 1):

```
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

```c
/* Fig. 4.9: fig04_09.c
   Using the do/while repetition statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int counter = 1; /* initialize counter */

   do {
      printf( "%d  ", counter ); /* display counter */
   } while ( ++counter <= 10 ); /* end do...while */

   return 0; /* indicate program ended successfully */

} /* 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

```c
/* Fig. 4.11: fig04_11.c
   Using the break statement in a for statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

       /* if x is 5, terminate loop */
       if ( x == 5 ) {
          break; /* break loop only if x is 5 */
       } /* end if */

       printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );

    return 0; /* indicate program ended successfully */

 } /* end function main */
```

Program Output

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

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

fig04_12.c

```c
/* Fig. 4.12: fig04_12.c
   Using the continue statement in a for statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

       /* if x is 5, continue with next iteration of loop */
       if ( x == 5 ) {
          continue; /* skip remaining code in loop body */
       } /* end if */

       printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );

    return 0; /* indicate program ended successfully */

 } /* end function main */
```

Program Output

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

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

Expression	Result
`true && false`	`false`
`true || false`	`true`
`!false`	`true`

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 strukurama
  - 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

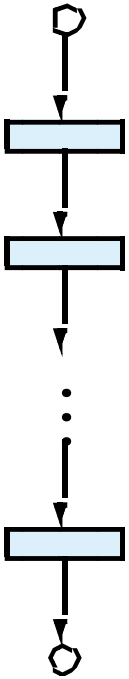
4.11 Konfuzija sa (==) i (=) operatorima

- 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;`

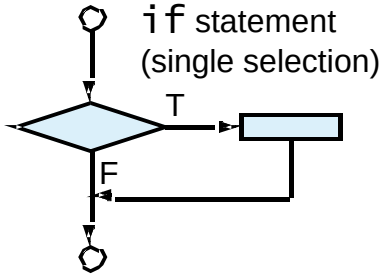
4.12 Struktuirano programiranje - pregled

Selection

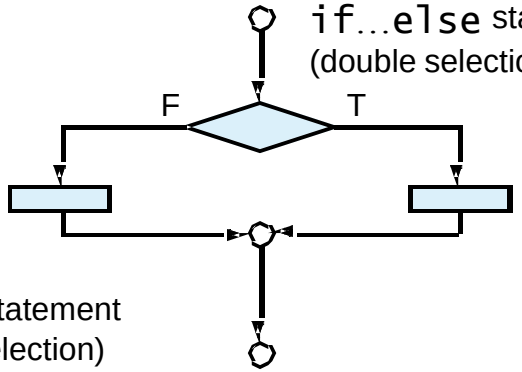
Sequence



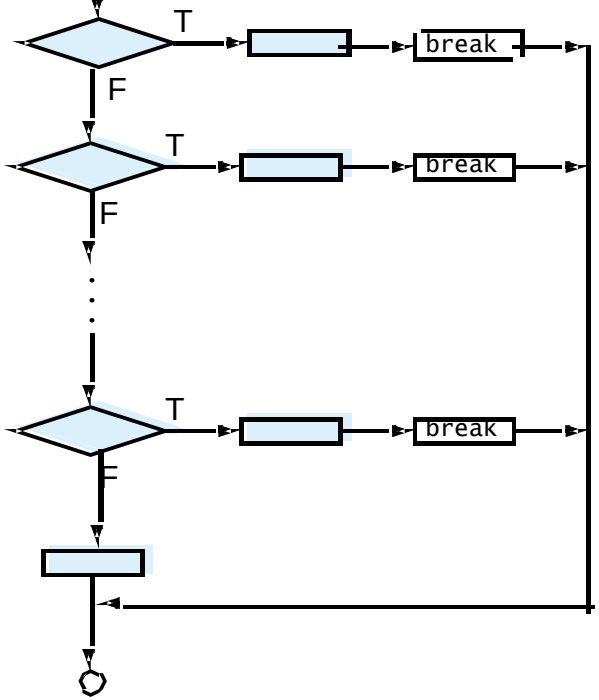
`if` statement
(single selection)



`if...else` statement
(double selection)



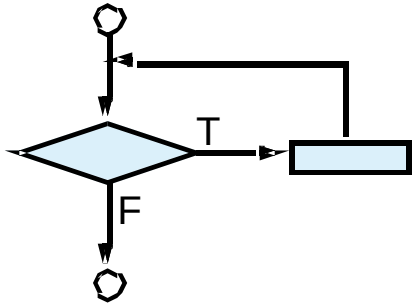
`switch` statement
(multiple selection)



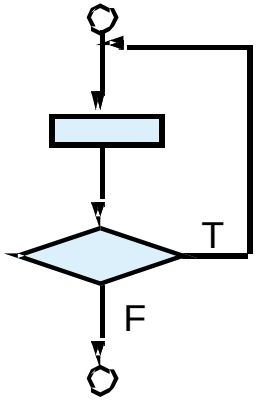
4.12 Struktuirano programiranje - pregled

Repetition

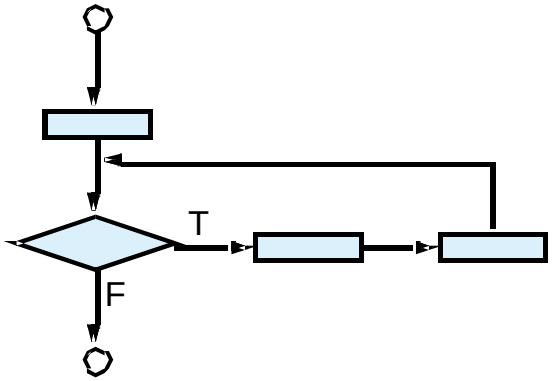
`while` statement



`do..while` statement



`for` statement



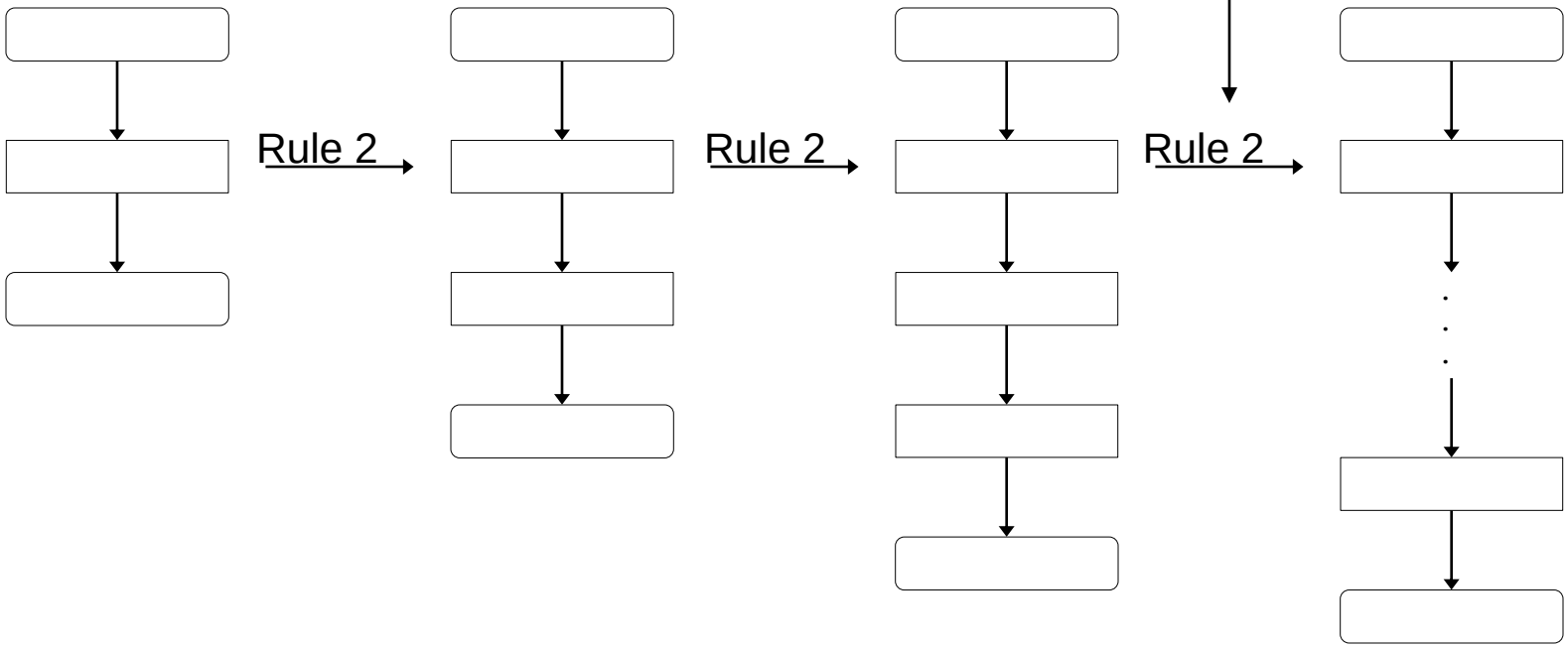
4.12 Struktuirano programiranje - pregled

- 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 zanijenjen 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 4.12 Struktuirano programiranje - pregled

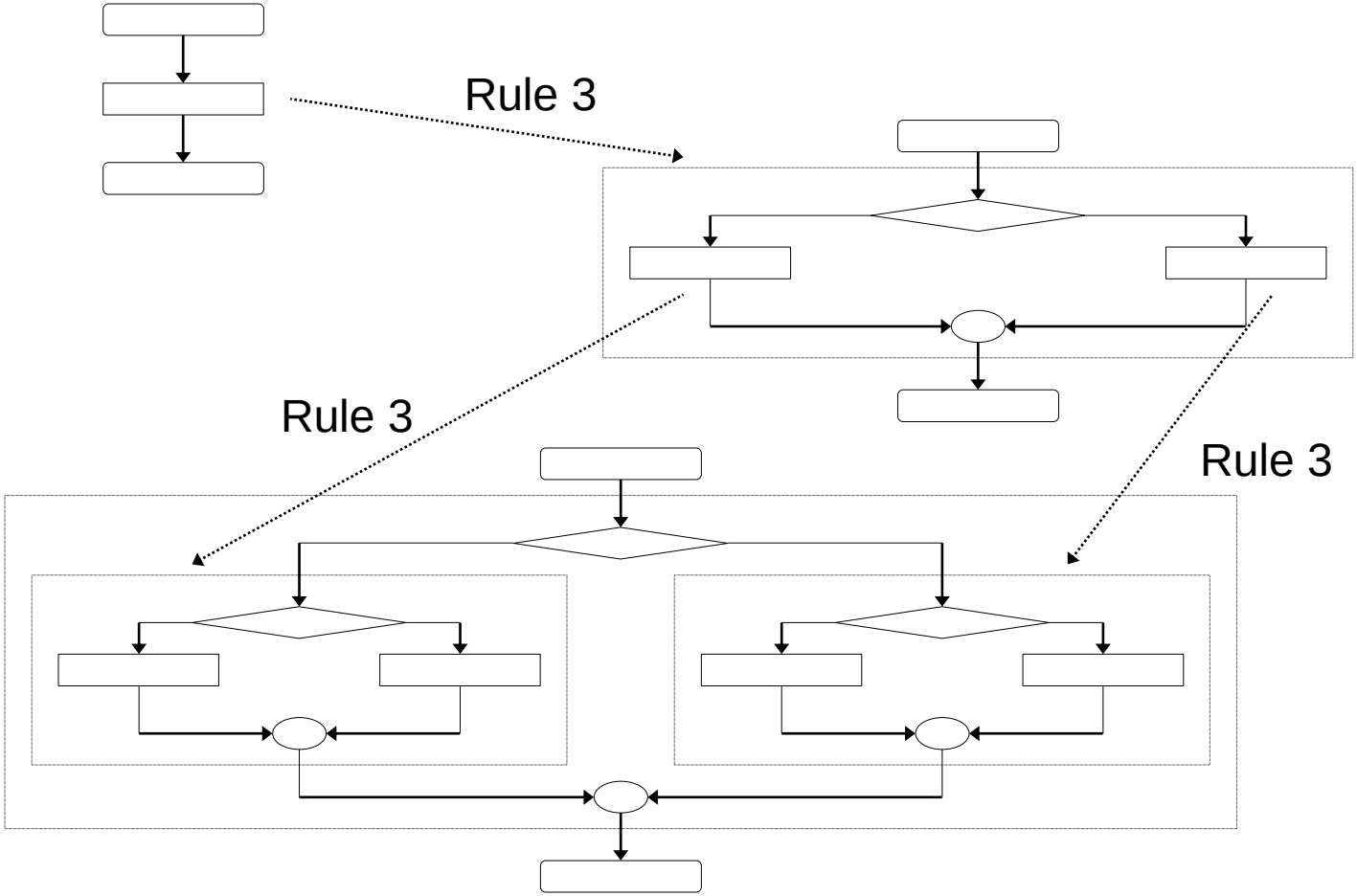
Pravilo 1 – Počinjemo naprostijom šemom

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



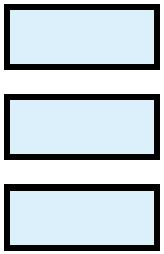
4.12 Struktuirano programiranje - pregled

Pravilo 3 – Svaku akciju možemo zamjeniti kontrolnom strukturom

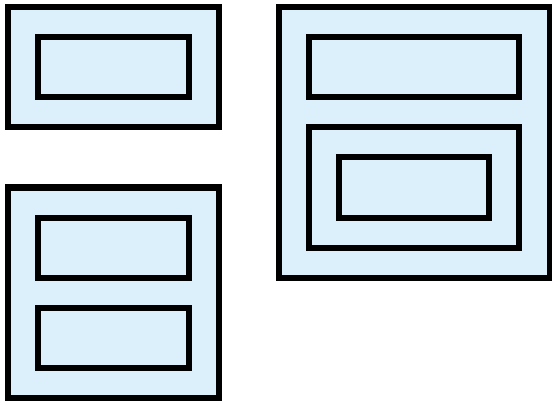


4.12 Struktuirano programiranje - pregled

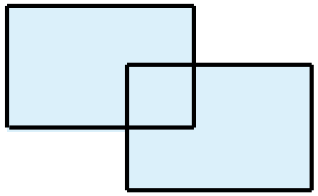
Stacked building blocks



Nested building blocks

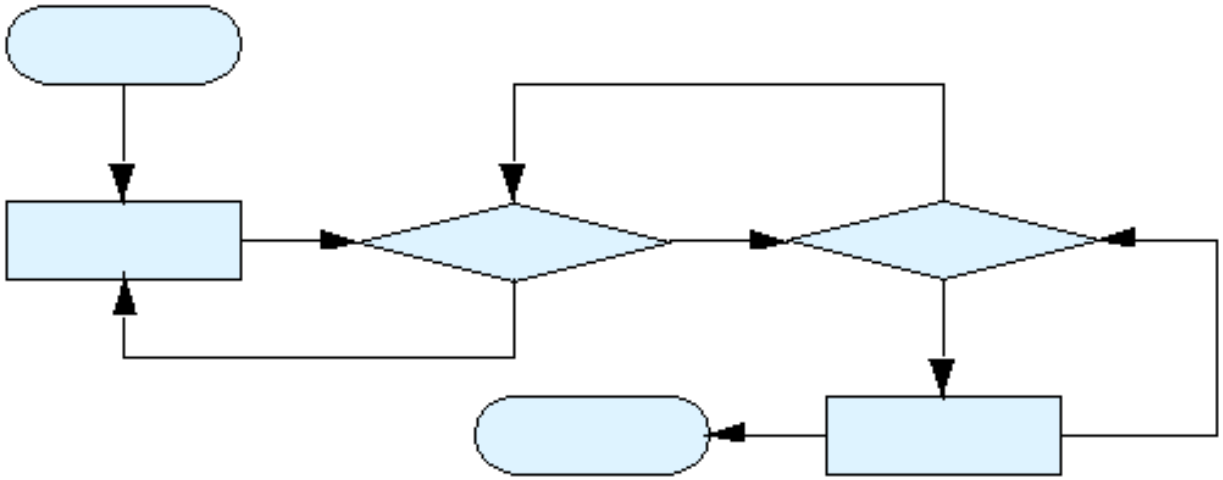


Overlapping building blocks
(Illegal in structured programs)



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