CONTROL TRANSFER STATEMENTS

break, continue, goto

Designed by Parul Khurana, LIECA.

Control Transfer Statements

- These statements transfer the control from one part of the program to another part.
- C language provides the following statements:
  - **THE break STATEMENT**
  - **THE continue STATEMENT**
  - **THE goto STATEMENT**
  - **THE return STATEMENT (to be covered in Lecture #12)**

Control Transfer Statements

- THE break STATEMENT
  - **The break statement is used to terminate loops or to exit from a switch statement.**
  - **It can be used within a for, while, do-while, or switch statement.**

Control Transfer Statements

- THE break STATEMENT
  - **The break statement is written simply as:**
    - **break;**

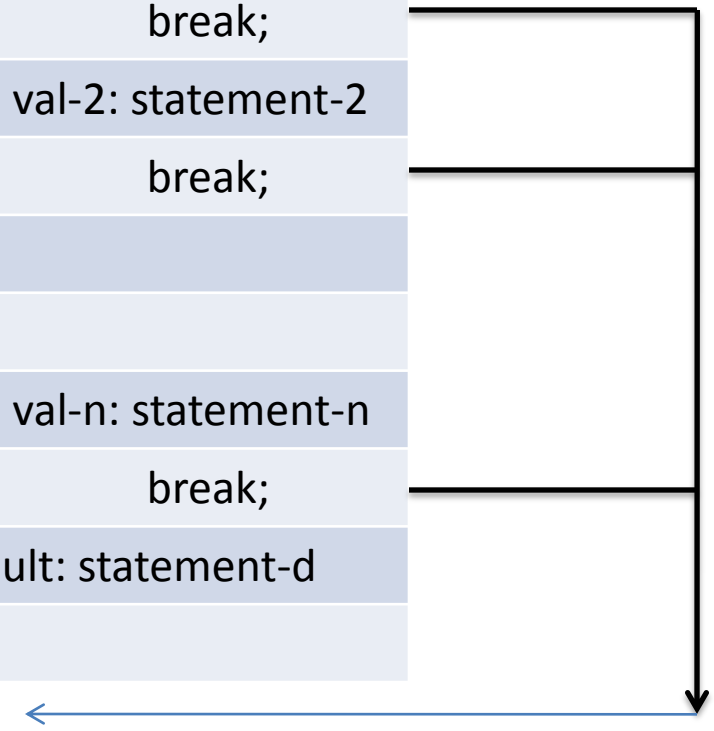
without any embedded expressions or statements.

The break statement causes a transfer of control out of the entire switch statement, to the first statement following the switch statement.

Control Transfer Statements

- THE break STATEMENT in ACTION in switch statement

```
switch ( expression)
{
case val-1: statement-1
            break;
case val-2: statement-2
            break;
:
:
case val-n: statement-n
            break;
default: statement-d
 }
```



- **In switch statement, it is used as the last statement of every case except the last one.**
- **When executed, it transfers the control out of switch statement and the execution of the program continues from the statement following switch statement.**

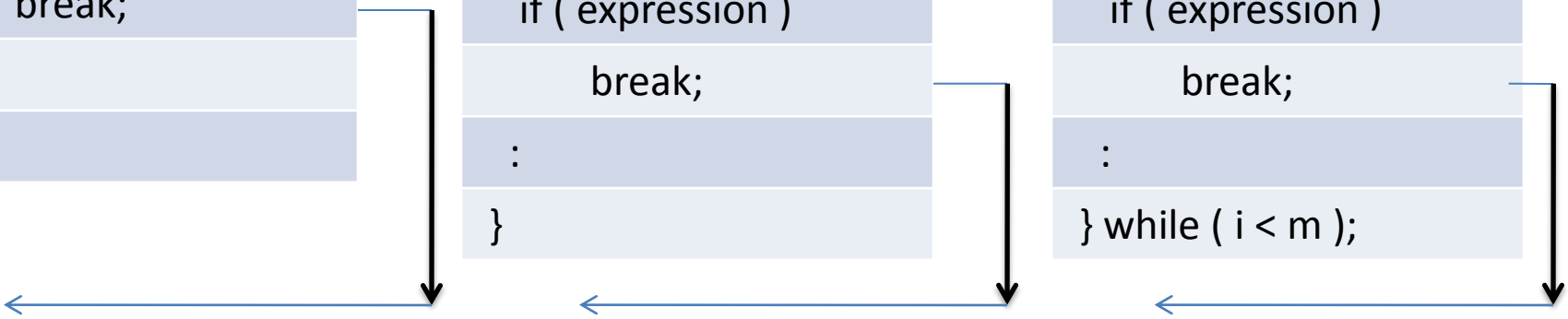
Control Transfer Statements

- **THE break STATEMENT in ACTION in for, while and do-while statement:**

```
for ( i=0; i<m; i++ )
{
  :
  if ( expression )
      break;
  :
}
```

```
i=0;
while ( i < m )
{
  :
  if ( expression )
      break;
  :
}
```

```
i=0;
do
{
  :
  if ( expression )
      break;
  :
} while ( i < m );
```



In for, while and do-while statements, it is always used in conjunction with if statement. Note that it is never used with if statement if it is not part of the body of the looping statement.

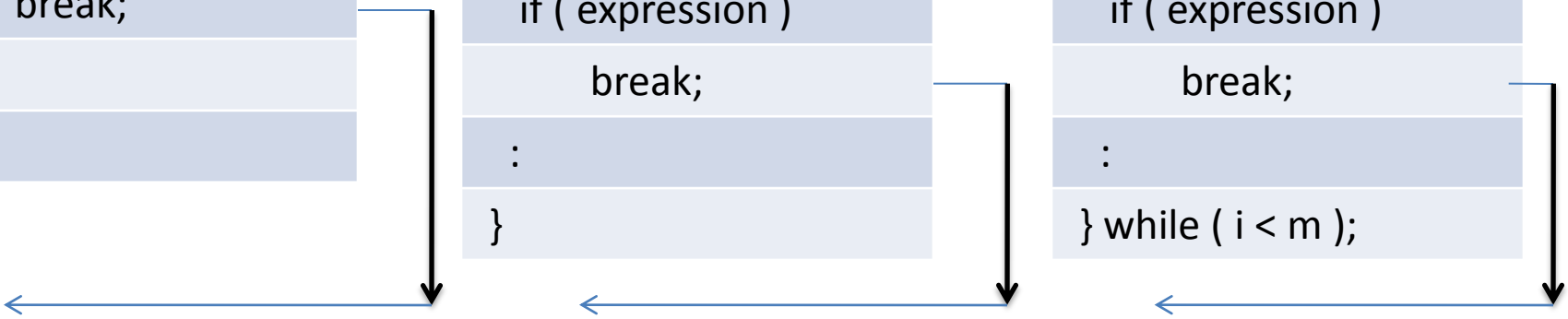
Control Transfer Statements

- **THE break STATEMENT in ACTION in for, while and do-while statement:**

```
for ( i=0; i<m; i++ )
{
  :
  if ( expression )
      break;
  :
}
```

```
i=0;
while ( i < m )
{
  :
  if ( expression )
      break;
  :
}
```

```
i=0;
do
{
  :
  if ( expression )
      break;
  :
} while ( i < m );
```



When executed, it transfers the control out of looping statement and the execution of the program continues from the statement following looping statement.

Control Transfer Statements

- **THE break STATEMENT** - Example
  - **Consider once again the switch statement originally presented in Lecture #9.**

```
switch ( choice )
{
    case 'R':
            printf("Red");
             break;
    case 'W':
            printf("White");
             break;
    default:
            printf(" Error ");
}
```

- **Notice** that each group of statements ends with a break statement, in order to transfer control out of the switch statement. The break statement is required in each of the first two groups, in order to prevent the succeeding groups of statements from executing.

Control Transfer Statements

- **THE break STATEMENT** - Example
  - **Consider once again the switch statement originally presented in Lecture #9.**

```
switch ( choice )
{
    case 'R':
            printf("Red");
             break;
    case 'W':
            printf("White");
             break;
    default:
            printf(" Error ");
}
```

- **The last group does not require a break statement, since control will automatically be transferred out of the switch statement after the last group has been executed.**

Control Transfer Statements

- THE continue STATEMENT
  - **The continue statement is used to bypass the remainder of the current pass through a loop. The loop does not terminate when a continue statement is encountered.**
  - **Rather, the remaining loop statements are skipped and the computation proceeds directly to the next pass through the loop.**

Control Transfer Statements

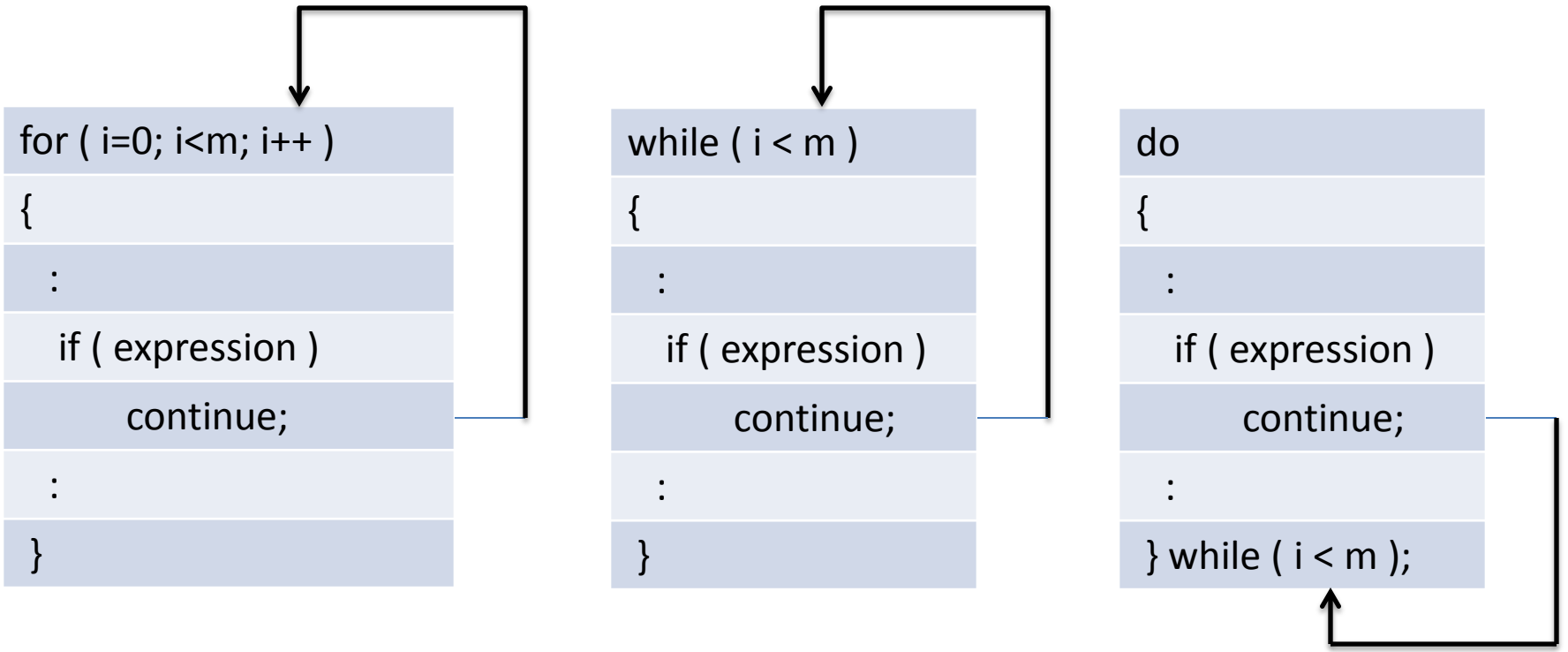
- THE continue STATEMENT
  - **The continue statement can be included within a while, a do-while or a for statement.**
  - **It is written simply as:**

    » **continue;**

    without any embedded statements or expressions.

Control Transfer Statements

- **THE continue STATEMENT in ACTION in for, while and do-while statement:**



The continue statement transfers the control to the beginning of the next iteration of the loop thus bypassing the statements which are not yet executed. Note that the continue statement is always used in conjunction with the if statement.

Control Transfer Statements

- **THE continue STATEMENT** - Example do-while statement
  - **Here are some illustrations of loops that contain continue statements.**

```
do
{
    scanf ( " %f", &x);
     if ( x < 0 )   {
        printf ( " ERROR - NEGATIVE VALUE FOR x" );
        continue; }
} while ( x <= 100 );
```

  - **The processing of the current value of x will be bypassed if the value of x is negative. Execution of the loop will then continue with the next pass.**

Control Transfer Statements

- **THE continue STATEMENT** - Example for statement
  - **Here are some illustrations of loops that contain continue statements.**

```
for ( count = 1; count <= 100; ++count)
{
    scanf ( " %f", &x);
     if ( x < 0 )   {
        printf ( " ERROR - NEGATIVE VALUE FOR x" );
        continue;
                    }
}
```

  - **The processing of the current value of x will be bypassed if the value of x is negative. Execution of the loop will then continue with the next pass.**

Control Transfer Statements

- THE goto STATEMENT
  - **The goto statement is used to alter the normal sequence of program execution by transferring control to some other part of the program.**

Control Transfer Statements

- THE goto STATEMENT
  - **In its general form, the goto statement is written as:**
    - » **goto label;**

Where label is an identifier that is used to label the target statement to which control will be transferred.

Control may be transferred to any other statement within the program.

Control Transfer Statements

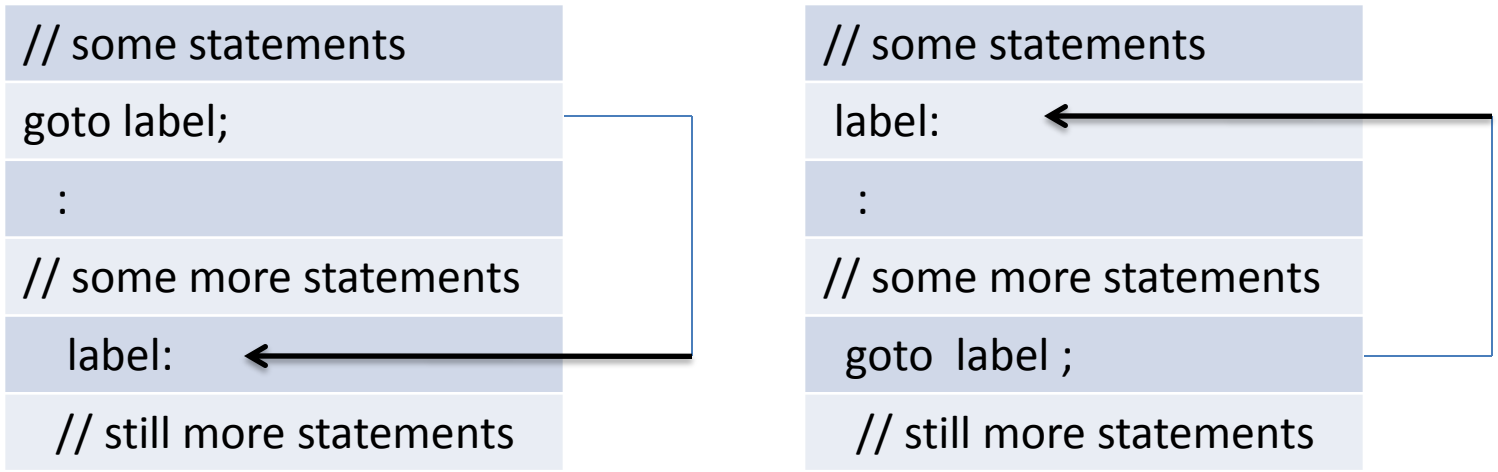
- THE goto STATEMENT
  - **In its general form, the goto statement is written as:**
    - » **goto label;**

      The target statement must be labeled, and the label must be followed by a colon. Thus, the target statement will appear as:

    - » **label: statement;**
    - » **Each labeled statement within the program must have a unique label; i.e., no two statements can have the same label.**

Control Transfer Statements

- **THE goto STATEMENT in ACTION**



The goto statement can transfer the control to any part of the program.

The target destination of the goto statement is marked by a label, where a label is any name or integer number followed by colon (character ':').

Control Transfer Statements

- **THE goto STATEMENT** - Example for statement
  - The following skeletal outline illustrates how the goto statement can be used to transfer control out of a loop if an unexpected condition arises.

```
scanf ( " %f", &x);
while ( x <= 100 ){
    . . . . . .
     if ( x < 0 )    goto errorcheck;
     . . . . . .
        scanf( "%f", &x);
                    }
errorcheck:    {
         printf ( " ERROR - NEGATIVE VALUE FOR x" );
                    }
```

In this example control is transferred out of the while loop, to the compound statement whose label is errorcheck, if a negative value is detected for the input x.

Practice Questions

- Predict the result of the following code segments:

```
x = 3, counter = 0;
while ((x - 1))
{
  ++counter;
  x--;
}
```

```
int x = 0;
for(;;)
{
if (x++ == 4)
 break;
continue;

}
    printf("x = %d\n", x);
```

```
int i = 4;
switch (i)
{
 default:
  ;
 case 3:
  i += 5;
  if (i == 8)
  {
   i++;
   if (i == 9) break;
   i *= 2;
  }
  i -= 4;
  break;
 case 8:
  i += 5;
  break;
}
printf("i = %d\n", i);
```

Practice Questions

- Predict the result of the following code segments:

```
void main (void)
{
int i,  j, x = 0;
for ( i=0; i < 5; i++)
   for ( j=0; j < i; j++)
     {
       x += ( i + j - 1);
       printf ( " %d" ,x );
     }
  printf ( " \n%d" ,x );
}
```

```
void main (void)
{
int i,  j, x = 0;
for ( i=0; i < 5; i++)
   for ( j=0; j < i; j++)
     {
       x += ( i + j - 1);
       printf ( " %d" ,x );
       break;
     }
  printf ( " \n%d" ,x );
}
```