SNS COLLEGE OF TECHNOLOGY

Coimbatore-35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

****

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMMING FOR PROBLEM SOLVING

I YEAR

UNIT 2 – C PROGRAMMING BASICS

DECISION MAKING AND BRANCHING

DECISION MAKING AND BRANCHING

Conditional or Decision Statements

- Decision control statements are used to alter the flow of a sequence of instructions.
- These statements help to jump from one part of the program to another depending on whether a particular condition is satisfied or not.
- The decision is described to the computer as a conditional statement that can be answered either true or false.
- Different types of control statements are:

1 . if statement

2 . if – else statement

3 . nested if Statement

4 . else if ladder

5 . Switch statement

1.If statement or Null Else Statement

- If statement is the simplest form of decision control statements that is frequently used in decision making.

Syntax:

```
if (test expression)
{
    statement 1;
    ..............
   statement n;
}
statement x;
```

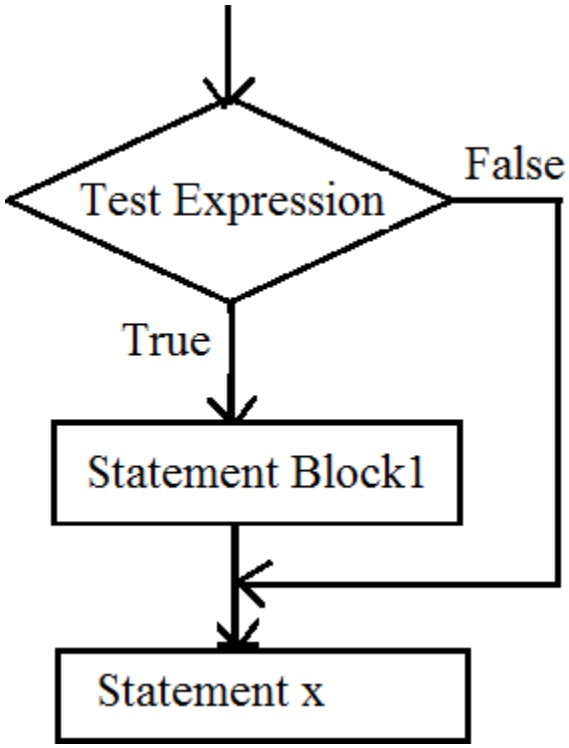
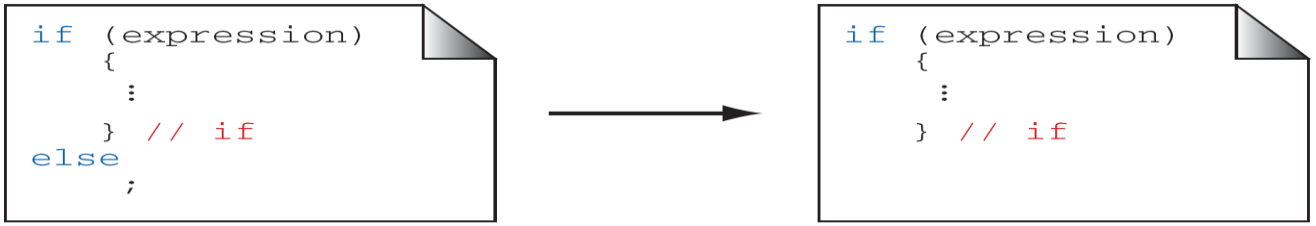


Fig: If Statement flow chart

Statements

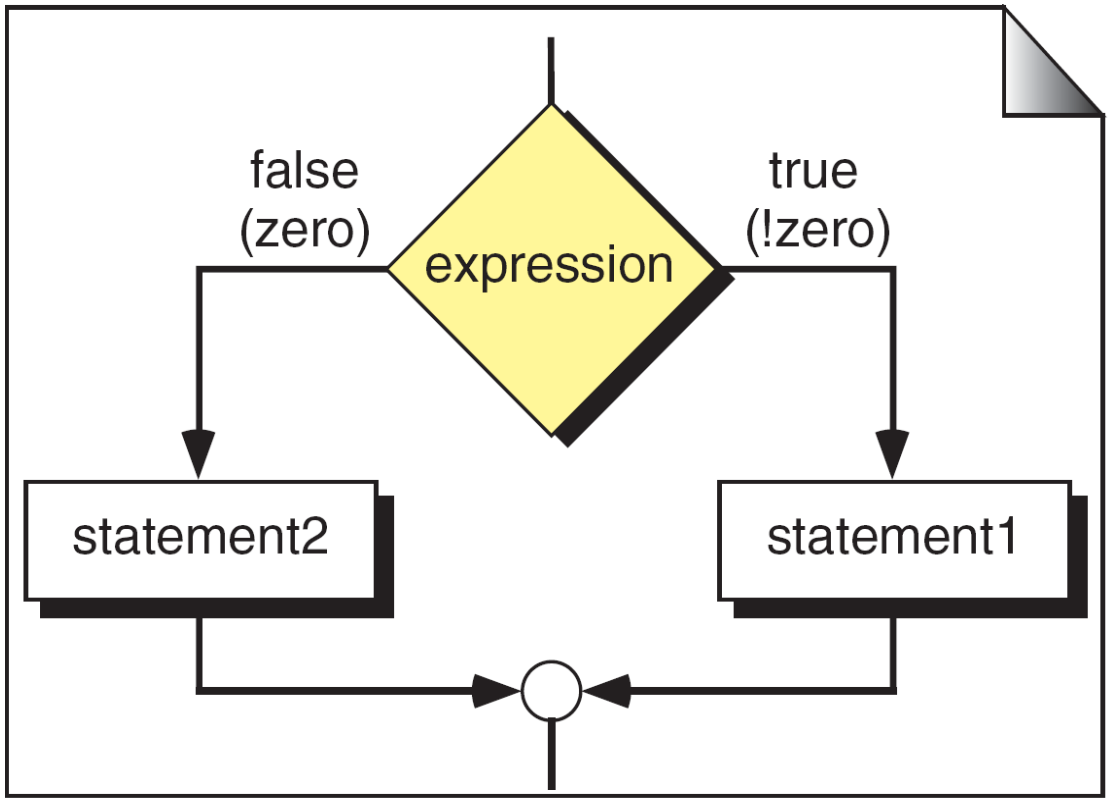
- First the test expression is evaluated. If the test expression is true, the statements of if block (statement 1 to n) are executed otherwise these statements will be skipped and the execution will jump to statement x.
- In this case the else statement is not required.



2.If- else Statement

- In the if-else construct, first the test expression is evaluated.
- If the expression is true, statement block 1 is executed and statement block 2 is skipped.
- Otherwise, if the expression is false, statement block 2 is executed and statement block 1 is ignored.
- In any case after the statement block 1 or 2 gets executed the control will pass to statement x.

Statements



(a) Logical Flow

```
if (expression)
     statement1
else
     statement2
```

(b) Code

Fig:if...else Logic Flow

Statements

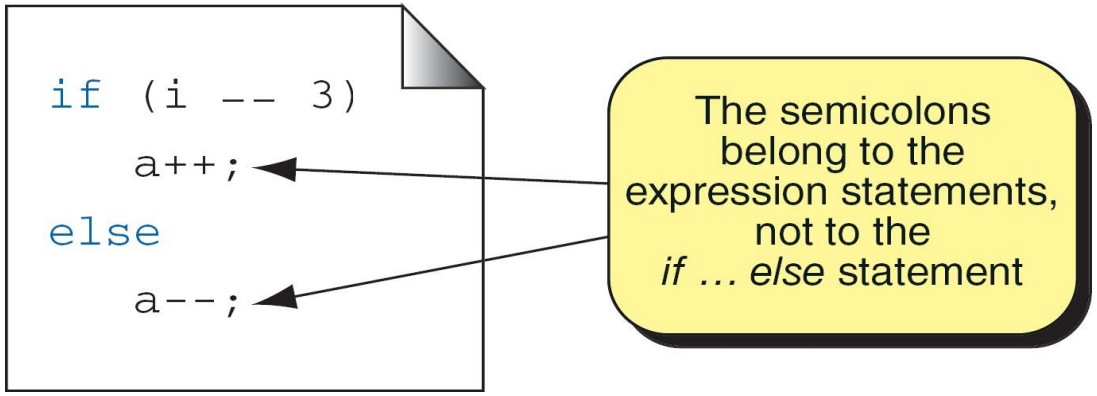
1. The expression must be enclosed in parentheses.
2. No semicolon (;) is needed for an *if…else* statement; statement 1 and statement 2 may have a semicolon as required by their types.
3. The expression can have a side effect.
4. Both the true and the false statements can be any statement (even another *if…else* statement) or they can be a null statement.
5. Both statement 1 and statement 2 must be one and only one statement. Remember, however, that multiple statements can be combined into a compound statement through the use of braces.
6. We can swap the position of statement 1 and statement 2 if we use the complement of the original expression.

Fig : Syntactical Rules for *if…else* Statements

Statements

- In the first rule the expression must be enclosed in parentheses.
- The second rule no semicolon for if else statement and statement 1 and statement 2 ends with semi colon.

Ex:



- The third rule it is common in c code expressions that have side effect.

Ex: if(++lineCnt > 10){
pritf(—\n‖);
lineCnt=0;
}
else printf(….);

Statements

- Rules 4 and 5 are related .the multiple statements can be used through the braces.
- The sixth rule states that the true and false statements can be exchanged by complementing the expression.

Statements

3 . Nested if Statement

- An if else is included within an if else is called nested if statement.
- There is no limit to how many level can be nested , but if the number of levels are increased it becomes more difficult.

Else if ladder

- To make a multi way decision on the basis of a value that not an integral. We go for else if .
- Else if is not a c construct ,it is a style of coding to make a multi way selection based on a value that is not integral.

Syntax:

```
if( expression)
    {
        Statements;
    }
else if(expression2)
   {
       Statements;
    }
else
    {
        Statements;
    }
```

Statements

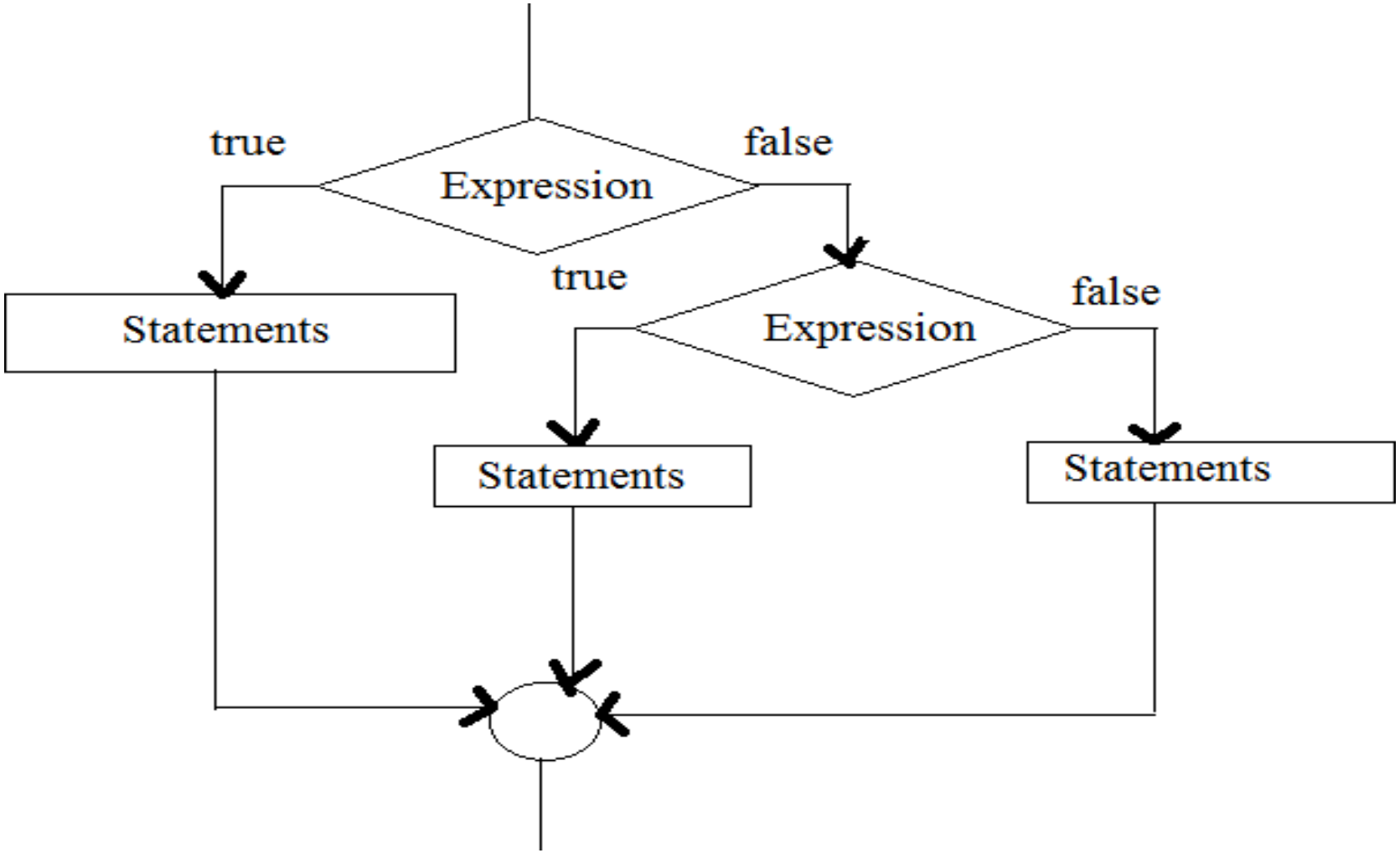


Fig : Flow chart for else if ladder

Statements

Switch

- Switch is a composite statement used to make a decision between many alternatives.
- The switch statement can be used only when the selection condition is reduced to an integral expression.

Statements

```
switch (expression)
{
case value-1:
                block-1
                break;
……….
……….
default:
        default-block
        break;
}
statement-x;
```

Fig : switch Statement Syntax

Statements

- The switch expression can use any expression that reduces to an integral value.
- The selection alternatives known as case labels must be integral types.
- Every possible value in the switch expression a separate case label is defined.
- Every thing from a case label to the next case label is sequence.
- Case label simply provides an entry point to start the execution of the code.
- The **default label** is a special form of the case label . It is executed none of the other case values matches the value in the switch expression.
- Once the program enters into a case it executes the code for all the following cases until the end .

1. The control expression that follows the keyword *switch* must be an integral type.
2. Each *case* label is the keyword *case* followed by a constant expression.
3. No two *case* labels can have the same constant expression value.
4. But two *case* labels can be associated with the same set of actions.
5. The *default* label is not required. If the value of the expression does not match with any labeled constant expression, the control transfers outside of the *switch* statement. However, we recommend that all *switch* statements have a *default* label.
6. The *switch* statement can include at most one *default* label. The *default* label may be coded anywhere, but it is traditionally coded last.

Switch Statement Rules

LOOPING STATEMENTS

Loop

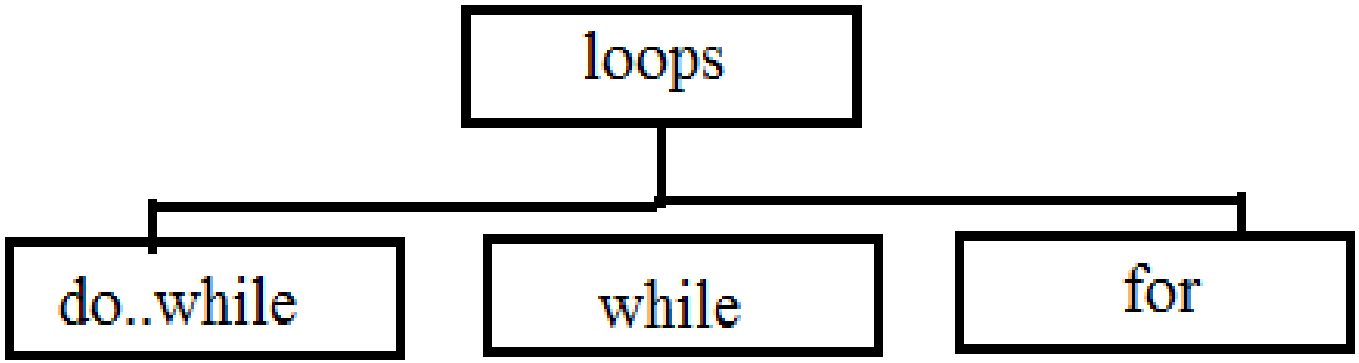
- To repeat an operation or a series of operation many times is called looping .
- The loop must terminate when the work is completed.
- Design the loop to check condition before or after the loop.

- There are three loop statements in c language

  1) do..while loop

  2) while loop

  3) for loop



C lopp Constructs

- The first one is post test loop and remaining two are pretest loops.
- All the three loops support event and counter controlled loops.
- While and do while are commonly used for event controlled and for is used for counter controlled.

Do…while loop

- The do while statement is a pretest loop.
- Do while statement test the expression after the execution of the body.
- Do while is concluded with a semicolon.

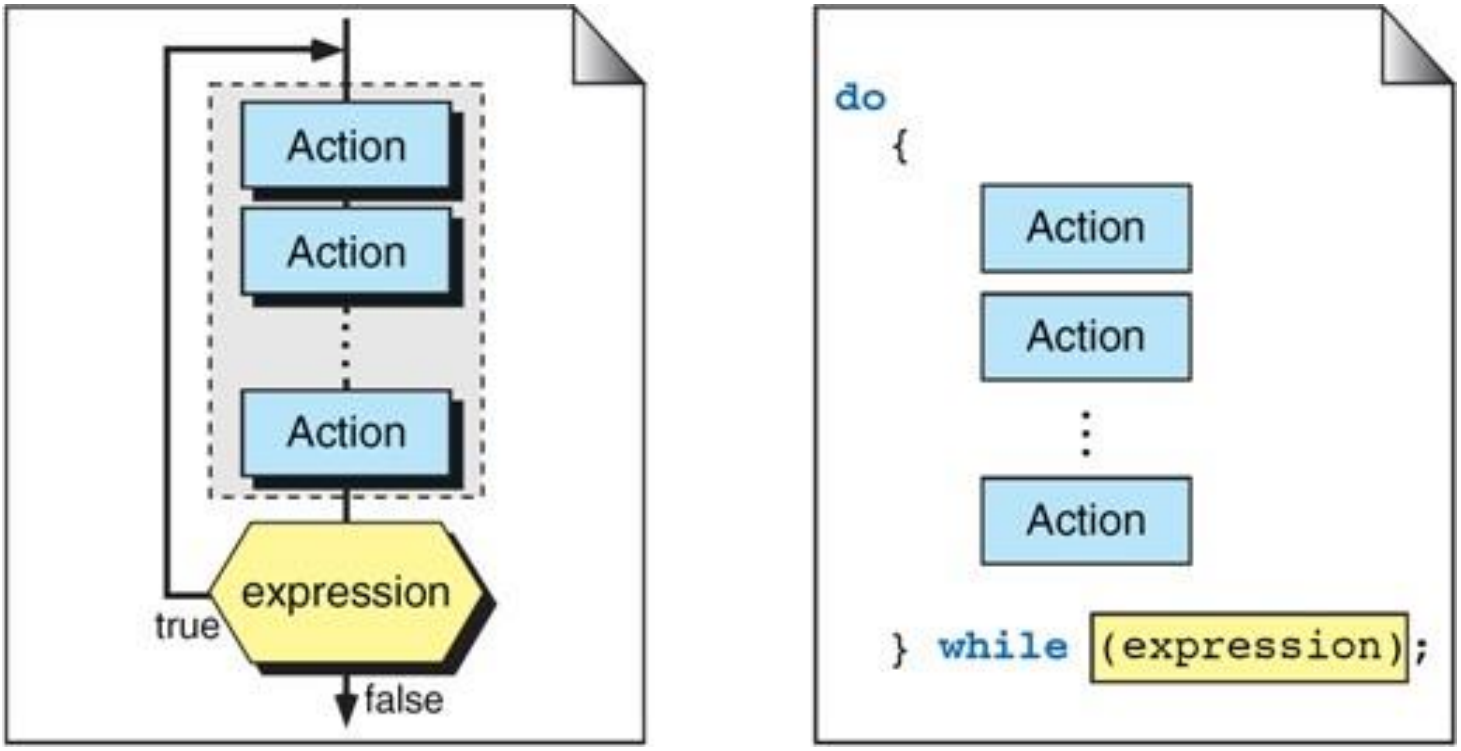


Fig : do…while Statement

Statements

While loop

- The while loop is a pretest loop.
- It test the expression before every iteration of the loop.
- No semicolon is needed at the end of the while statement.
- Hear the body of the loop must be only one statement.

Statements

- To include multiple statements in the body , we must put them in a compound statement.

Statements

For Loop

- It is a pretest loop.
- It uses three expressions.
- The first expression contains any initialization statements , the second contains the limit-test expression , and the third contains the updating expression.

Initialization

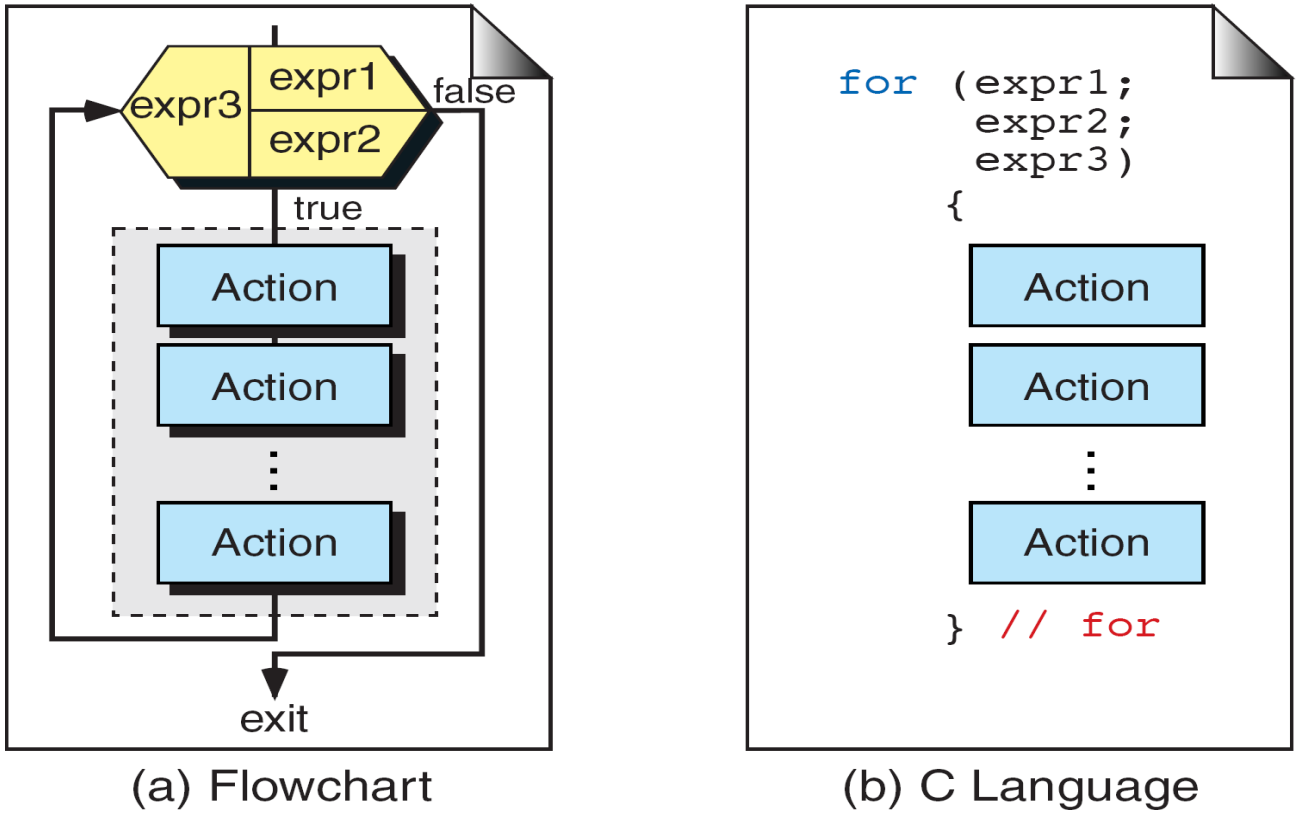
Updating

Syntax:

condition

```
for( exp1  ; exp2   ; exp3 )
```

- The body of the for loop must be one and only one statement.
- To include more statements in the body we must code them in a compound statement.



(a) Flowchart (b) C Language

Fig : Compound for Statement

- A for loop is used when a loop is to be executed a known number of times .
- The same thing can be implemented with a while loop but the for loop is easier to read and more natural for counting loops.

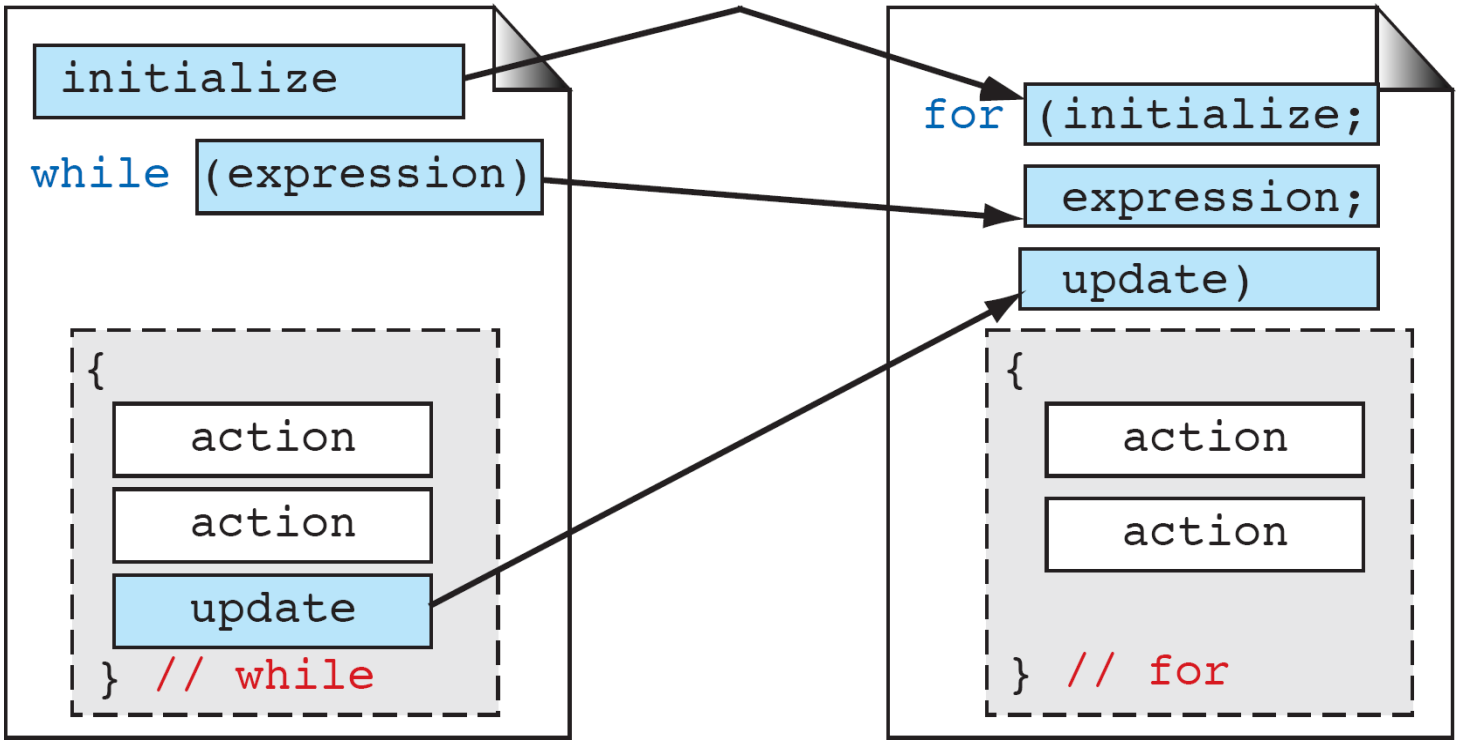


Fig : Comparing for and while Loops

Nested For Loops

- Including the for loop with in the body of another for loop.
- By using nested for loops we create looping applications.

The comma Expression

- The comma expression is a complex expression made up of two expressions separated by a comma.
- The expressions are evaluated left to right.
- The value and type of the expression are the value and type of the right expression.
- The comma expression has the lowest priority of all expressions , i.e is 1.

Ex:

```
for( sum=0,i=1; i<=20; i++)
        sum=sum+i;
```

Fig : Nested Comma Expression

Statements

Break

- The break statement causes a loop to terminate.
- The break statement can be used in any of the loop statements like while , do-while , for and in the selection switch statement.

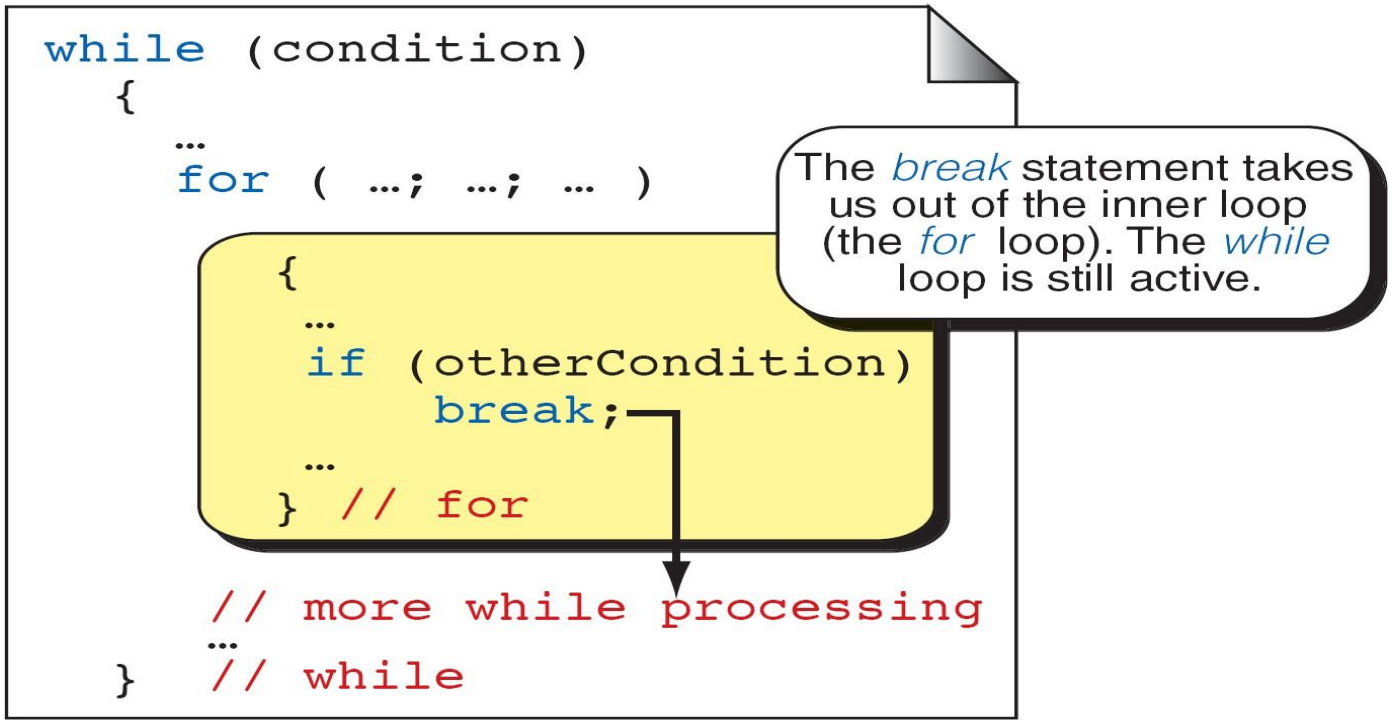


Fig : break and Inner Loops

Statements

```
1  // A bad loop style
2  for ( ; ; )
3     {
4      …
5      if (condition)
6         break;
7     } // for
```

```
1  // A better loop style
2  for ( ; !condition ; )
3     {
4       …
5     } // for
```

```
1  while (x)
2     {
3      …
4      if (condition)
5         break;
6      else
7         …
8     } // while
```

```
1  while  (x && !condition)
2     {
3      …
4      if (!condition)
5           …;
6     } // while
```

The for and while as Perpetual Loops

Statements

```
1  breakFlag = 0;
2  while (!breakFlag)
3      {
4       …
5       if (x && !y || z)              // Complex limit test
6           breakFlag = 1;
7       else
8           …;
9      } // while
```

Using a break Flag

Continue

- The continue statement does not terminate the loop.
- It simply transfer to the testing expression in while and do-while statements and transfer to the updating expression in a for statement.

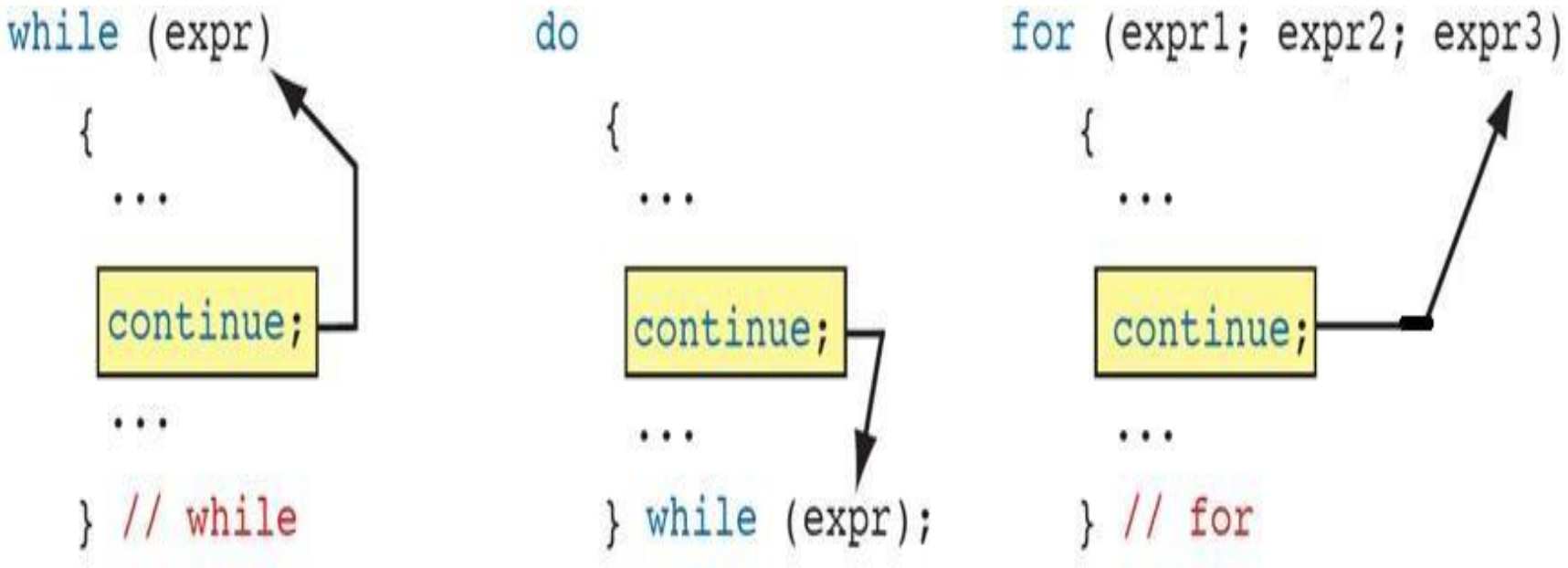


Fig : The continue Statement

goto

- The goto statement is used to transfer control to a specified label.
- Label is an identifier that specifies the place where the branch is to be made. Label can be any valid variable name that is followed by acolon (:).
- label can be placed anywhere in the program either before or after the goto statement. Whenever the goto statement is encountered the control is immediately transferred to the statements following thelabel.
- If the label is placed after the goto statement then it is called a forward jump and in case it is located before the goto statement, it is said to bea backward jump.

THANK YOU