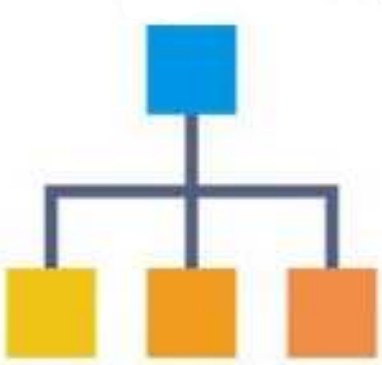
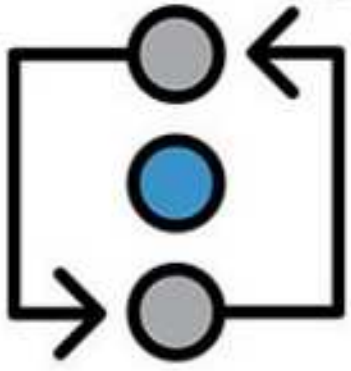
Unit 4

CONTROL STRUCTURE





- Recognize the various types of control statements:
- Define decision making structure
- Understand the syntax of If and If-Else statements
- Use If and If-Else statements in C++ programming
- Differentiate between If, If-Else and switch decision making structures
- Use switch statement in the programs

4.1 CONTROL STATEMENTS

A computer program is the set of instructions in sequential form. These instructions execute from top to bottom. We can control the flow of program with the help of Control Statements. Control Statements are used to control the direction of program by repeating any block of code, making a choice or simply transfer the control to any specific block of program. These statements help programmers to decide which part of code is executed at certain time.

C++ has three types of control statements:

1. Selection/Decision Making Structure
2. Iteration / Loops
3. Jump

4.2 SELECTION/DECISION MAKING STRUCTURE

They are used to decide whether a certain part of code is executed or not.

C++ has three decision making structures;

1 'if' statement
2 'if- else' statement
3 'switch' statement

4.2.1 "if" statement

It is the basic decision statement. The structure contains "**if**" keyword followed by a conditional expression in parenthesis and then its body of statements(s) also called if block. If there are more than one statement in "**if**" block or body we enclose all of them in braces.

If statement checks a condition, if the condition is true the statements in "**if**" block are executed and if condition is false it leaves the statements in **"if"** block and starts executing statements after "**if**" block.

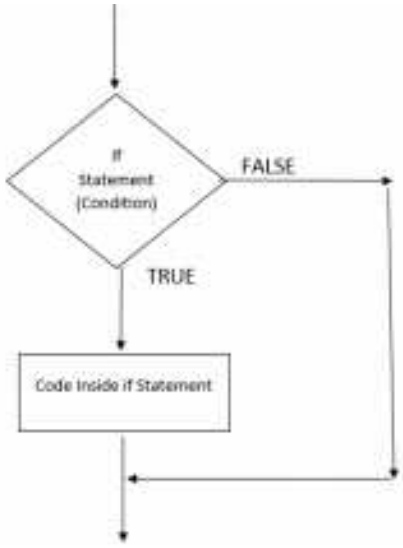


Fig: 4.1: if Statement's Flowchart

Syntax:

```
if (test condition)
   {
    Statement(s);
```

Following is an example of using if in a program. This program takes marks as input. If marks are greater than 60 then it adds word "good" between "you are a " and "student of this class

```
/* Program 1.   "if" statement example */
// writes good on basis of marks
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int marks;
      cout<<"\nEnter Marks ";
      cin>>marks;

      cout<<" You have secured "<<marks<< " marks ";

      if(marks>=70)
                  {
            cout<<" and A Grade ";
                  }
      cout<<" in 5th class";
return 0;
}
```

Nested "if" Statement

An 'if' statement which is part of block of another 'if' statement is called nested 'if'. The inner 'if' statement will only be tested if the outer 'if' is true.

In following program, marks and age of a person are taken as input. First condition checks marks, if they are greater than or equal to 60 then next condition checks age. If age is also greater than 18 then it prints message "you got the job". "good luck" is always printed.

```
/* Program 2.    Nested "if" statement example  */
// job given message on basis of marks and age
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int marks,age;

      cout<<"\nEnter your Marks ";
      cin>>marks;
      cout<<"\nEnter your age ";
      cin>>age;

      if(marks>=60)
            {
            if(age>=18)
                  {
                  cout<<"you got the job";
                  }
            }
      cout<<" Good Luck";
return 0;
}
```

4.2.2 "if-else" statement

The structure of if- else contains 'if' keyword followed by a conditional expression in parenthesis and then its body of statements(s) then 'else' keyword and its body of statements. It checks the condition, if the condition is true the statements in if block are executed and in case of false condition, it executes statements in 'else' block. It means either 'if block statements or 'else' block statements must execute.

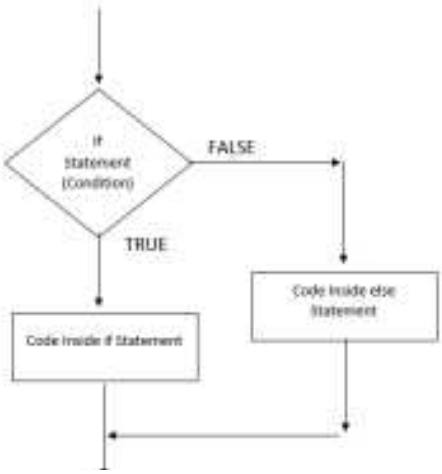


Fig: 4.2. if- else Statement's Flowchart

Syntax:

```
if (test condition)
{
        Statement(s);
}
Else
{
    Statement(s);
}
```

For example, following program takes marks as input and decides pass or fail on the basis of marks. If greater than or equal to 40 then pass otherwise fail.

```
/* Program 3. "if-else" statement example */
// shows pass or fail on the basis of marks
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int marks;
      cout<<"\nEnter your Marks ";
      cin>>marks;

      if(marks>=40)
            cout<<"You are pass";
      else
            cout<<" You are fail";
return 0;
}
```

4.2.3 else-if statement

In "if" statement if we use nesting deeply i.e. one if is nested to other and other is nested in another and so on then in indentation we see a ladder like structure which is difficult to understand like below. Here is an example. This program takes marks as input and then determines the grade by applying if condition multiple times.

```
/* Program 4. "else-if" statement example */
// shows Grade on the basis of marks
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int marks;

      cout<<"\nEnter your Marks ";
      cin>>marks;

      if(marks>=80)
            cout<<"Grade is A1";
      else
            if(marks>=70)
                  cout<<" Grade is A";
                    else
                      if(marks>=60)
                         cout<<" Grade is B";
                           else
                              if(marks>=50)
                                  cout<<" Grade is C";
                                       else
                                           cout<<" Fail";
return 0;
}
```

However, to make this program look simpler and more understandable, we may design it in another format by using "else-if" statement. This format only increases the readability of program while there is no any change for compiler. "else-if" statement for the above program is shown in next program where all "else" are in a single column.

```
if(marks>=80)
     cout<<"Grade is A1”;
else if(marks>=70)
                    cout<<" Grade is A";
      else if(marks>=60)
                    cout<<" Grade is B";
      else if(marks>=50)
                     cout<<" Grade is C";
      else
                     cout<<" Fail";\
```

Now, make a program in which user will input two integers and one arithmetic operator (+, -, *, /). Perform the given arithmetic operation on given numbers by using else- if statement and print the result on screen.

4.2.4 “switch” statement

The switch statement starts with **“switch”** keyword followed by a variable or expression in parenthesis, then a block of switch statement in braces. The block contains one or more case statement each followed by integer or character constant and then colon. After colon there may be many statements followed by “break” statement in the end. The switch variable or expression checks its value equal to these constants that follow case statement. If it matches to any of them then control is transferred to that **“case”** and all statements after colon are executed and switch is broken with “**break**” statement and control transfers to the next

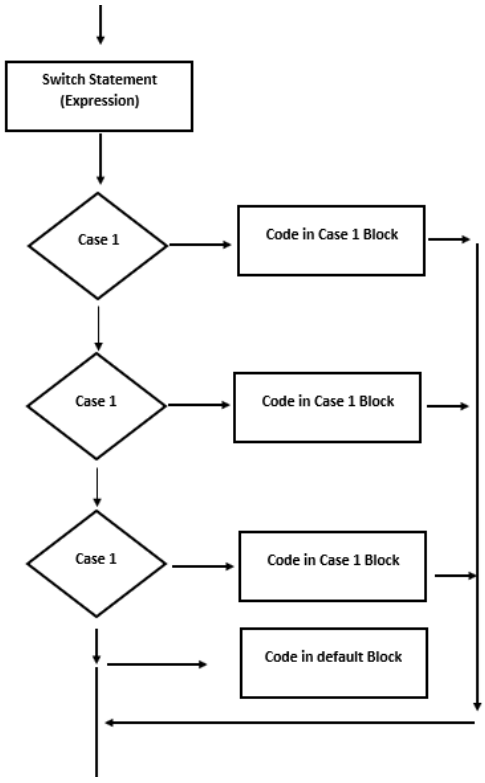


Fig: 4.3. switch Statement’s Flowchart

statement after switch. If switch variable does not match with any of the case constants control goes to keyword **"default"** (if it is present). It acts as **"else".** **"break"** and **"default"** keywords are optional.

Syntax:

```
switch(expression)

    {
    case constant 1:
       statement(s)
       break;
    case constant 2:
       statement(s)
    break;

   .   .   .
   .   .   .
    default:
       statement(s)
    }
```

In following example, the program takes number of the day of week (from 1 to 7) as input and on the basis of case matching prints the name of the day. If other number is given then invalid day message is shown through default statement.

```
/* Program 5.   "switch" statement example */
// prints name of day checking day number
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int dow;

      cout<<"\nEnter number of weekday 1 to 7 ";
      cin>>dow;

      switch(dow)
```

```
            {
            case 1: cout<<"Sunday"; break;
            case 2: cout<<"Monday"; break;
            case 3: cout<<"Tuesday"; break;
            case 4: cout<<"Wednesday"; break;
            case 5: cout<<"Thursday"; break;
            case 6: cout<<"Friday"; break;
            case 7: cout<<"Saturday"; break;
            default: cout<<"Invalid day number";
            }

return 0;
}
```

Can you make a program to input month's number and print its name on screen?

4.2.5 Difference between If, If-Else and switch decision making structures

"if"	**"if"** statement checks a condition using relational and other operators, if the condition is true the statements in if block execute. If condition is false, it leaves the statements in **"if"** block and starts executing statements after **"if"** block.
"if-else"	**"if-else"** statement checks a condition using relational and other operators, if the condition is true the statements in **"if"** block are executed and in case of false condition it executes statements in **'else'** block. It means either **"if"** block statements will execute or **"else"** block statements.
"switch"	**"switch"** statement checks 'switch' variable value equal to constant that follows case statement. If it matches to any of them then control is transferred to that case and all statement after colon are executed and switch is broken with **"break"** statement. Otherwise, control is transferred to **"default"** statement. It has some limitations. It only matches character and integer data type variables, it checks switch variable value only with case constants not with any variable and it also cannot use relational operators like less than (<) or greater than (>) and exactly matches value with case constants.

- Explain the concept of loop structure
- Explain for, while and do-while loop structures
- Differentiate between for, while and do-while loop structures and their use
- Use these three loop structures into C++ programming
- Explain the concept of nested loops

4.3 ITERATION/ LOOP

Normally statements are executed sequentially, one after the other. In some situations, we need to execute a block of statements several number of times. Loops allow us to execute a statement or a group of statements several numbers of times. A group or block is made by enclosing statements in braces. There are three types of loops in C++.

```
1. for
2. while
3. do- while
```

4.3.1 "for" loop

This loop executes a sequence of statements multiple times. It is usually used in situations where at the start of loop we know that how many times loop block will execute. It is a pre-test loop as condition is tested at the start of loop. It starts with keyword **"for"** followed by loop expression in parenthesis. Loop expression has three parts separated by semicolon.

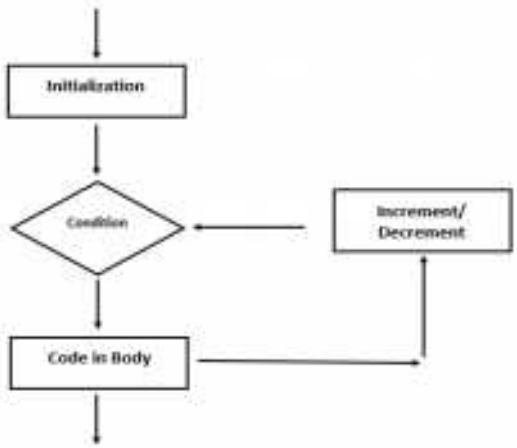


Fig: 4.4. for Loop Statement's Flowchart

First part is initialization which initializes loop variable; second part is test expression which tests loops variable and third part in increment or decrement in loop variable. Then comes the body of the loop which may have one or more statements. If there are more than one statements then they are enclosed in braces.

When loop starts, first part is executed, only one time. Then second part tests the condition, if it is true it enters the loop otherwise transfers control to statement after loop. If it enters in loop then after executing all statements in loop block, third part is executed and again condition is tested. It is also called counter controlled loop or definite repetition loop since the number of iterations is known before loop execution.

Syntax:

```
for (initialization; condition testing; increment/decrement)
{
   statement(s);
}
```

The following program shows even numbers from two to twenty. It initializes loop variable count with two and then print numbers by adding two each time to this variable until count variable remains less than twenty.

```
/* Program 6. "for" loop example */
// shows even numbers from 1 to 20
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int count;


      for(count=2;count<=20;count=count+2)
            {
            cout<<"\n Number= ";
            cout<<count;
            }
return 0;
}
```

4.3.2 "while" loop

It is a pre-test loop. It tests the condition at the start of loop before executing the body of loop. It is usually used in situations where we do not know at the start of loop, how many times loop block will execute. So, it is also called indefinite repetition loop. It starts with keyword "while" followed by a conditional expression like if statement in parenthesis. Then comes the body of the loop which may have one or more statements in braces.

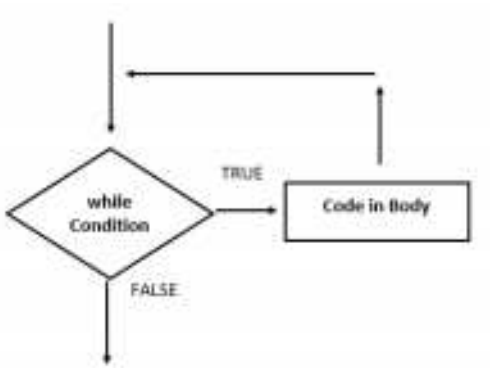


Fig: 4.5. while Loop Statement's

When loop starts, it tests a condition, if it is true it enters in the loop otherwise transfers control to statement after loop. If it enters in loop then after executing all statements in loop block again condition is tested.

Syntax:

```
while
{
    
}
```

Following program takes characters as input through getche() in loop. When user presses enter key equal to '\r' in C++ loop condition is false and it shows total number of typed characters.

```
/* Program 7. "while" loop example */
// counts number of characters typed
#include<stdio.h>
#include <iostream.h>
#include<conio.h>
using namespace std;
int main(void)
{
      int num=0;
      char ch;

      clrscr();
      cout<<"Type any word or text, Press enter to terminate -> ";
```

```
        ch=getche();
        while(ch!='\r')
              {
              num++;
              ch=getche();
              }
        cout<<"\n Total number of characters typed "<<num;
return 0;
}
```

4.3.3 "do while" loop

It is similar to while loop, except that it tests the condition at the end of the loop body and so it is also called post-test loop. Its statements block is executed at least once. For second time condition is tested. It starts with keyword "do" followed by a body of loop which may have one or more statements in braces. Then "while" keyword and conditional expression in parenthesis. It must be terminated with semi colon.

When loop starts it executes all statement in block once and then tests the condition, after "while" keyword. If it is true it enters the loop again otherwise transfers control to the next statement after loop. It is also indefinite repetition loop.

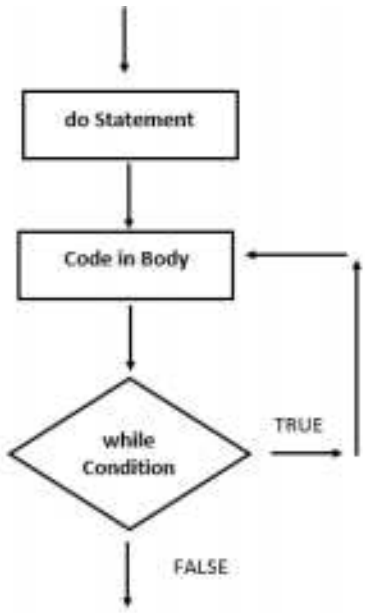


Fig: 4.6:
while Loop Statement's Flowchart

Syntax:

```
do
{
   Statement 1;
   Statement 2;
   Statement 3;
.
.

}
while( condition );
```

Following program takes salaries of employees as input in a loop. We press y to takes more salaries as input, any other character to end. Finally it shows total salary paid to all employees.

```
/*  Program 8.  "do-while" loop  example */
// calculates total salary paid to all employees
#include<stdio.h>
#include <constream.h>
using namespace std;
int main(void)
{
      long count=1;
      float tot_sal=0,salary;
      char ch;

      clrscr();
      do
      {
      cout<<"\n Enter salary of employee e"<<count<<"->";
      cin>>salary;
      tot_sal=tot_sal+salary;
      cout<<" Press Y to enter more salaries ";
      ch=getche();
      count++;
      }
      while(ch=='y');

      cout<<"\n Total salary paid is "<<tot_sal;
return 0;
}
```

Nested loops

You can use one or more loop inside any another 'for', 'while' or 'do while' loop. If a loop exists in the body of another loop then it is called nested loop. The inner nested loop is completely executed every time for each repetition of outer loop.

The following program is of nested for loop. It prints numbers from 1 to 10 in five rows. The outer loop shows row number, then inner loop prints numbers from 1 to 10 in one row then outer loop changes the row using '\n'. The outer loop runs five times, so inner loop which prints 1 to 10 numbers will run five times and prints numbers in five rows.

```
/* Program 9.   nested loops example */
// prints numbers from 1 to 10 in five rows
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
      int i,j;

      clrscr();
      for(i=1;i<=5;i++)
            {
            cout <<" Row no. "<<i<<" -> ";
            for(j=1;j<=10;j++)
                  cout<<j<<"  " ;
            cout<<"\n";
            }
return 0;
}
```

4.3.4 Difference between for, while and do-while loop structures

"for" Loop	"while" Loop	"do while" Loop
"for" loop is usually used in situations where we know at the start of loop that how many times loop block will execute. It is also called definite repetition loop.	It is usually used in situations where we do not know at the start of loop that how many times loop block will execute. It is also called indefinite repetition loop.	It is usually used in situations where we do not know at the start of loop that how many times loop block will execute. It is also called indefinite repetition loop.
It is called counter controlled loop as loop is controlled by a counter value, at each iteration counter value will increase or decrease	It does not need a counter value for its execution.	It does not need a counter value for its execution.
It has three parts first initialization, second condition testing and third increment or decrement.	It has only one part which is condition testing. If needed initialization is done before loop and increment or decrement is done in loop body.	It has only one part which is condition testing. If needed initialization is done before loop and increment or decrement is done in loop body.
It is pre-test loop as condition is tested at start of loop.	It is pre-test loop as condition is tested at start of loop.	It is post-test loop as condition is tested at the end of loop. So this loop executes at least once.

Teachers Note	Teacher are supposed to show step by step output of nested loops.

Recognize the use of jump statements:

- Break Statement
- Continue Statement
- Goto Statement
- Return Statement

4.4 JUMP STATEMENTS

Jump statements change execution of program from its normal sequence.

Following are the jump statements used in C++

1. break
2. continue
3. goto
4. return
5. exit ()

1. "break" Statement:

It terminates the loop or switch statement and transfers control to the statement immediately following the loop or switch statement.

e.g.

```
/* Program 10.   'break'  statement example */
// Adds five numbers if 0 is given ends program
#include<stdio.h>
#include <constream.h>
using namespace std;
int main(void)
{
int  i, num, sum;
i=0; sum=0;

cout<<"\n Enter five numbers to add. Enter 0 to terminate -> ";
      while(i<5)
      {
            cin>>num;
            if(num==0)
            {
                  cout<<"\n Ending program ";
                  break;
            }
```

```
            sum=sum+num;
            i++;
      }
cout<<"\n sum of  " <<i<< " number(s) is " <<sum;
return 0;
}
```

2. "continue" Statement:

It causes the loop to skip the remaining statements of its body and immediately transfers control to the top of the loop i.e. first statement.
e.g.

```
/* Program 11.   'continue' statement example */
// Adds five positive numbers. It does not take -ve numbers
#include<stdio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
int i, num, sum;
i=0; sum=0;
  cout<<"\n Enter five positive numbers-> ";
      while(i<5)
      {
            cin>>num;
            if(num<=0)
            {
                  cout<<"\n Enter positive number";
                  continue;
            }
            sum=sum+num;
            i++;
      }
cout<<"\n sum of  five positive numbers is " <<sum;
return 0;
}
```

3.“goto” Statement :

A “goto” statement jumps or transfers control unconditionally from the “goto” to a labeled statement in the same function. A labeled statement is any identifier followed by a colon (:). It is not advised to use “goto” statements in programs.

e.g.

```
If ( marks<20)
          goto warning;    // warning is label
            …..
warning: cout<<” Need very hard work in examination”;
```

Control of program may be transferred to another position in program by two other ways; using return statement or exit () function.

4. “return” statement :

It terminates the execution of a function and transfers program control to the statement just after the function call statement in the calling function (or to the operating system if you transfer control from the main function). It can also return a value from the current function, if return type is not “void”.

Syntax : return [expression/value];

The value of the expression clause is returned to the calling function.

5 .”exit ( )” Function:

The exit() is used to terminate a C++ program and closes program whenever executed. It is defined under “stdlib.h” header file.

Syntax: void exit (int);

- C++ has three types of control statements: Selection/Decision Making Structure, Iteration / Loops and Jump.
- C++ has three decision making structures; 'if' statement, 'if-else' statement and 'switch' statement.
- If statement checks a condition, if it is true the statements in if block are executes and if it is false, it leaves the statements in if block and starts executing statements after the block.
- If else checks a condition, if it is true the statements in if block are executed and in case it is false, it executes statements in 'else' block.
- Switch statement checks different constants after case statement with switch variable; if matches it executes statements after it otherwise goes to default statement if present.
- Loops allow us to execute a statement or a group of statements several numbers of times.
- **"for"** loop execute a sequence of statements multiple times. And is usually used in situations where we know at the start of loop that how many times loop body will execute. Condition is tested at the start of loop.
- Like for loop **"while"** loop also repeats a statement or group of statements several numbers of times while a given condition is true. It tests the condition at start of loop and is usually used in situations where we do not know at the start of loop that how many times loop block will execute.
- **"do while"** loop is similar to **"while"** loop, except that it tests the condition at the end of the loop body. So its statements block is executed at least one time.
- If a loop exists in the body of another loop then it is called nested loop.
- A **"break"** statement terminates the loop or switch statement and transfers control to the statement immediately following the loop or switch statement.

- A **"continue"** statement causes the loop to skip the remaining statements of its body and immediately transfers control to the top of the loop.
- A **"goto"** statement jumps or transfers control unconditionally from the **"goto"** to a labeled statement in the same function.
- A **"return"** statement terminates the execution of a function and transfers program control to the statement just after the function call statement in the calling function.
- The **exit()** is used to terminate a C++ program.

1. Encircle the correct answer:

i) Loop within a loop is called _____ loop.
   a. inner b. outer
   c. enclosed d. nested

ii) "case" and _____ are also part of "switch" statement.
   a. have b. default
   c. for d. if

iii) "for" Loop expression has _____ parts.
   a. one b. two
   c. three d. four

iv) exit( ) function is used to _____.
   a. close function b. close loop
   c. close program d. close switch

v) "continue" statement takes control to the _____ .
   a. top of loop b. end of loop
   c. top of function d. end of function

vi) In "goto" statement label is followed by _____ character.
   a. colon (:) b. semi colon (;)
   c. single quote (') d. double quote (")

vii) To send value to the calling function we use _____ statement.
   a. throw b. return
   c. send d. back

viii) "break" statement is used with _____.

a. if b. switch
c. for d. while

ix) Using "else" is _____ with "if" statement.

a. prohibited b. advised
c. compulsory d. optional

x) "if" and loop expressions use _____ operators to test condition.

a. arithmetic b. relational
c. insertion d. bitwise

2. Answer the following questions:

1. What is the purpose of "**default**" statement in C++?
2. Can we use "**while**" loop in place of "**for**" loop, if yes then how?
3. What is the main difference between **while** and **do while** loops?
4. Write the function of **for** loop.
5. Why we make block of statements using braces?
6. Which data type variables can be used in "**switch**" statement?
7. What is the purpose of jump statements?
8. Write the purpose of following statements:
   a. else if
   b. switch
   c. goto
   d. exit

3. Match the column:

S.no	A	S.no	B
(a)	if	(i)	Relational operators
(b)	loop	(ii)	break
(c)	Conditional expression	(iii)	switch
(d)	Loops and switch	(iv)	operator
(e)	do	(v)	iteration
(f)	>>	(vi)	else
(g)	case	(vii)	while

LAB ACTIVITIES

1. Write a program that takes a number as input and print whether it is odd or even.
2. Write a program to add numbers from 1 to 20.
3. Write a program that take month number as input (from 1 to 12) and print number of days in that month. If wrong number is given then show error message.
4. Input a number up to six digits and show each digit in separate line.
5. Take input a character, number of rows and number of columns. Draw a square box filled with that character with given number of rows and columns.
6. Write a programs that generate the following outputs

*	1	1
**	12	22
***	123	333
****	1234	4444
*****	12345	55555

7. Write a program that takes a number as input and print whether it is prime or not.
8. Take salary as input and on its basis show different levels of designations in an organization like manager, supervisor, worker etc.
9. Write a program that prints square of numbers from 1 to 10.
10. Take a number and print its table from 1 to 10 using while loop according to the following format.

$2 \times 1 = 2$
$2 \times 2 = 4$
$2 \times 3 = 6$
.
.
.
$2 \times 10 = 20$

Teachers Note	Teachers are supposed to encourage students to develop different programs by using the concepts of input/ output statements, loops and selection statements.