COMPUTER SCIENCE 10TH – DETAILED QUESTION ANSWERS

➔ CONTROL STRUCTURE

Chapter # 04

Q.1: What are control structures?

Ans. CONTROL STRUCTURES/STATEMENTS

Control structures control the flow of execution in a program or function. Control structure are used to repeat any block of code, transfer control to specific block of code and make a choice by selection. There are three basic control structures in C++ programming.

1. Selection / Decision Making Control Structure
2. Loops / Iteration Control Structure
3. Jumps

Q.2: Define selection or decision making control structure and name its types.

Ans. SELECTION / DECISION MAKING CONTROL STRUCTURES / STATEMENTS

The selection control structure allows a number of conditions which lead to a selection of one out of several alternatives. There are three types of selection control structure:

1. If selection structure / statement
2. If - else selection structure / statement
3. Switch selection structure / statement

Q.3: Define if selection structure with syntax, flow diagram and example.

Ans. 1. if SELECTION STATEMENT

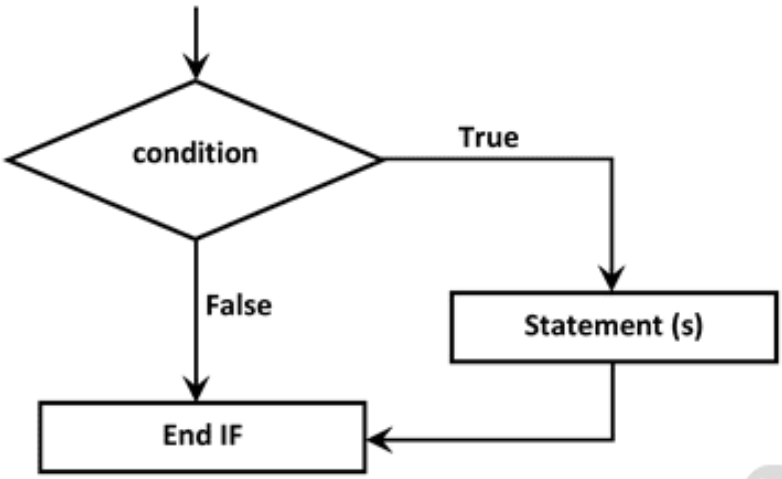
An if statement is a conditional statement that tests a particular condition. Whenever that condition evaluates as true, performs an action, but if it is not true, then the action is skipped.

SYNTAX

The general syntax of if statement is:

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

FLOW DIAGRAM



if SELECTION STATEMENT EXAMPLE

```
#include<iostream>
using namespace std;
int main( )
{
    int num;
    cout << "Enter an integer number: ";
    cin >> num;
    if(num > 0)
    {
        cout << "You entered a positive integer: " << num << "\n";
    }
    return 0;
}
```


OUTPUT

Enter an integer number : 10
You entered a positive integer : 10

Q.4: Define nested if selection structure with syntax and example.

Ans. NESTED if STATEMENT

An if condition can be written as deeply as needed within the body of another statement. This is called nested if statement.

SYNTAX.

The general syntax of nested if statement is:

```
if (condition 1)
{
if (condition 2)
{
   statements;
}
}
```

NESTED if STATEEMENT EXAMPLE

```
#include<iostream>
using namespace std;
int main( )
{
   int exp, status;
   cout << "Enter experience: ";
   cin >> exp;
   cout << " \n Enter status: ";
   cin >> status;
   if(exp >= 4)
   {
      if(status >= 2)
      {
       cout << "\n Bonus Given to Employee" << "\n";
      }
   }
   return 0;
}
```

OUTPUT

```
Enter experience: 6
Enter status: 3
Bonus Given to Employee
```

Q.5: Define if-else selection structure with syntax, flow diagram and example.

Ans. **if-else SELECTION STATEMENT**

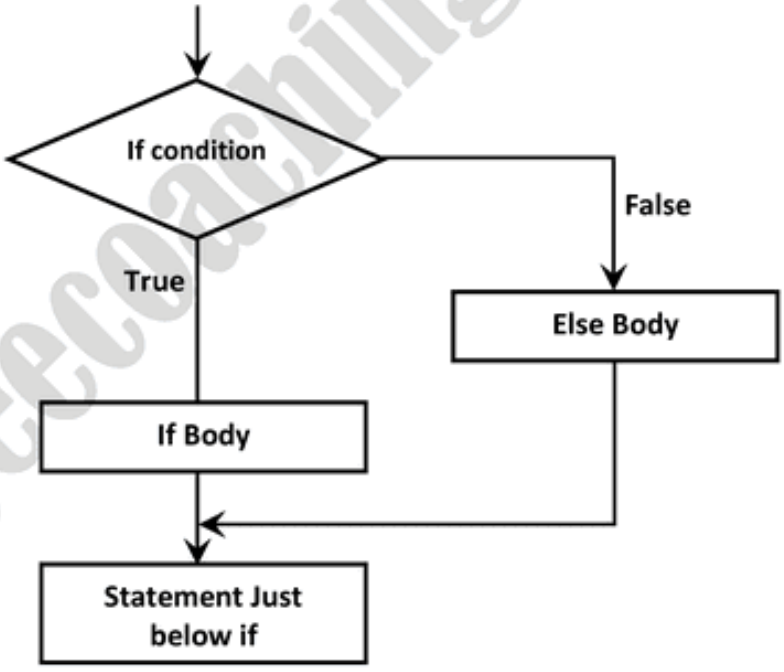
An if-else selection structure performs certain action when the condition is true and some different action when the condition is false.

SYNTAX

The general syntax of if-else statement is:

```
if (condition)
{
 statement(s);
}
else
{
 statement(s);
}
```


Flow Diagram



if-else SELECTION STATEMENT EXAMPLE

```
#include<iostream>
using namespace std;
int main( )
```

```
{
  int number;
  cout << "Enter an integer: ";
  cin >> number;
  if (number >= 0)
  {
    cout << "The number is a positive integer: " << number << "\n";
  }
  else
  {
    cout << "The number is a negative integer: " << number << "\n";
  }
  cout << "This line is always printed.";
  return 0;
}
```


OUTPUT
Enter an integer : -5 The number is a negative integer: -5 This line is always printed.

Q.6: Define else-if selection structure with example.

Ans. **else-if SELECTION STATEMENT**

Nested if-else statements test for multiple conditions by placing if-else statements inside if-else statements. When a condition is evaluated as true, the corresponding statements are executed and rest of the structure is skipped. This structure is also referred as the if-else-if ladder.

else-if SELECTION STATEMENT EXAMPLE

```
#include<iostream>
using namespace std;
int main( )
{
  int per;
  cout << "\n Enter your percentage: ";
  cin >> per;
  if (per >= 80)
```

```
 {
  cout << "Your grade is A+ :";
 }
 else if (per >= 70)
 {
  cout << "Your grade is A:";
 }
 else if (per >= 60)
 {
  cout << "Your grade is B:";
 }
 else if (per >= 50)
 {
  cout << "Your grade is C:";
 }
 else
 {
  cout << "Failed.";
 }
 return 0;
}
```


OUTPUT

Enter your percentage : 70
Your grade is A:

Q.7: Define switch statement with syntax, flow diagram and example.

Ans. SWITCHSJATEMENT

Switch statement is a control statement that allows to select only one choice among the many given choices. The expression in switch evaluates to return an integer or character value, which is then compared to the values present in different cases. It executes that block of codes which matches the case value. If there is no match, then default block is executed (if present).

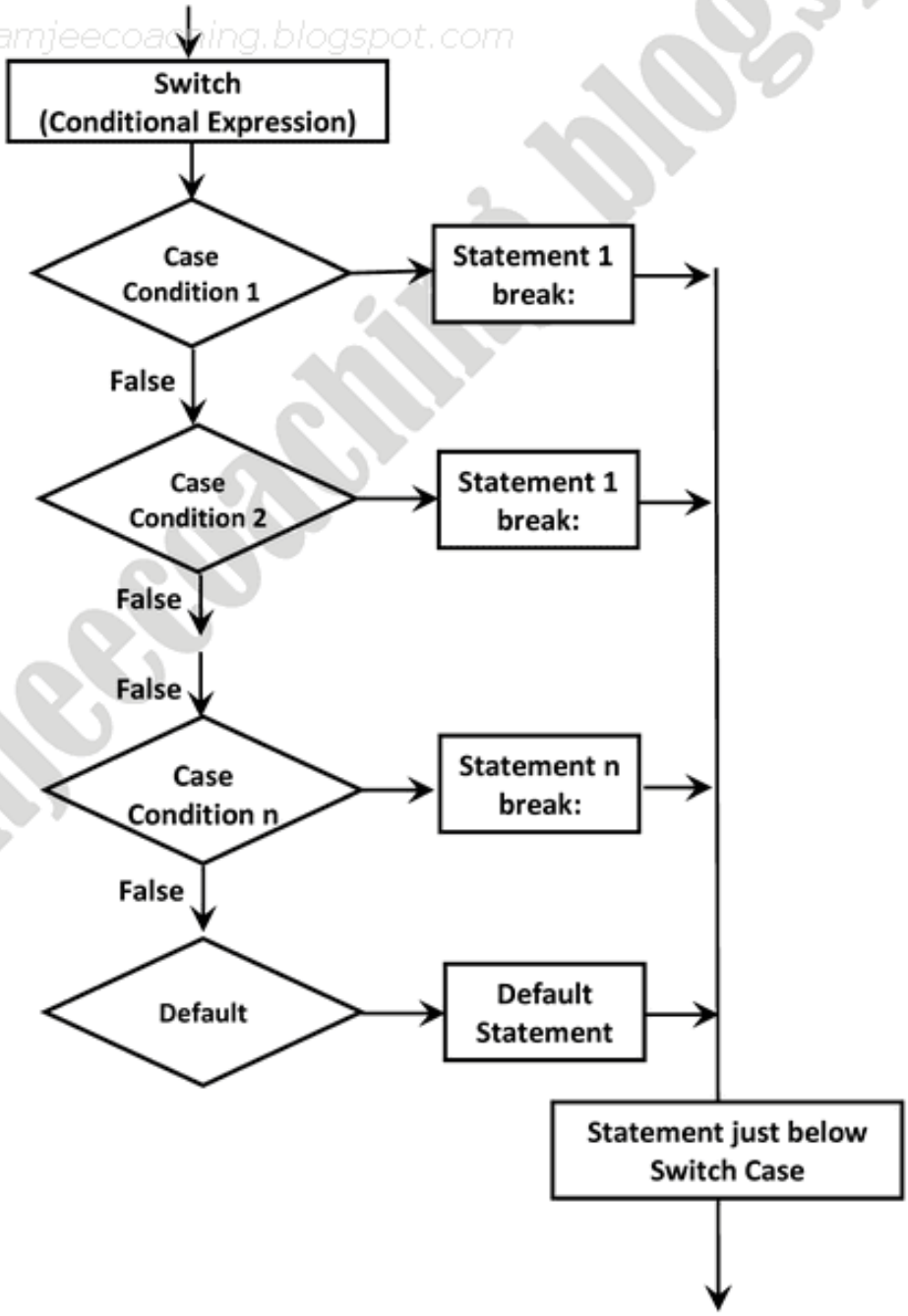
SYNTAX

The general form of switch statement is,

```
switch(variable)
{
 case constant 1:
{
```

```
  statement (s);
  break;
}
  case constant 2:
{
  statement (s);
  break;
}
  default:
{
  statement (s);
  break·
}
}
```

FLOW DIAGRAM



SWITCH STATEMENT EXAMPLE

```
#include<iostream>
using namespace std;
int main( )
{
  char op;
  float num1, num2;
  cout << "Enter an operator (+ , - , * , / ) : ";
  cin >> op;
  cout << "Enter two numbers: " << "\n";
  cin >> num1 >> num2;
  switch (op)
  {
    case '+' :
    cout << num1 << " + " << num2 << " = " << num1 + num2;
    break;
    case '-':
    cout << num1 << " - " << num2 << " = " << num1 - num2;
    break;
    case ' *' :
    cout << num1 << " * " << num2 << " = " << num1 * num2;
    break;
    case 'I':
    cout << num1 << " / " << num2 << " = " << num1 / num2;
    break;
    default:
    cout << "Error! The operator is not correct";
  }
  return 0;
}
```


OUTPUT

```
Enter an operator : +
Enter two numbers :
10 15
10 + 15 = 25
```

Q.8: Write down the difference between if-else and switch statement.

Ans. DIFFERENCES BETWEEN if- else & SWITCH STATEMENT

if-else	SWITCH
If-else statement is used to select among two alternatives.	The switch statement is used to select among multiple alternatives.
If-else can have values based on constraints.	Switch can have values based on user choice.
Float, double, char, int and other data types can be used in if-else condition.	Only int and char data types can be used in switch block.
It is difficult to edit the if-else statement, if the nested if-else statement is used.	It is easy to edit switch cases as, they are recognized easily.

Q.9: What are loops or iteration and define its types?

Ans. LOOP / ITERATION CONTROL STRUCTURE

Iteration or loop in computer programming, is a process wherein a set of instructions or structures are repeated in a sequence a specified, number of times until a condition is true. When the set of instructions is executed again, it is called an iteration. A loop statement allows us to execute a statement or a group of statements multiple times

C++ provides the following types of loops to handle looping requirements.

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

Q.10: Describe for loop with syntax, flow diagram and example.

Ans. for LOOP

A for loop is a repetition or iteration control structure that repeats a statement or block of statements for a specified number of times. The for-loop statement includes the initialization of the counter, the condition, and the increment. The for loop is commonly used when the number of iterations is known.

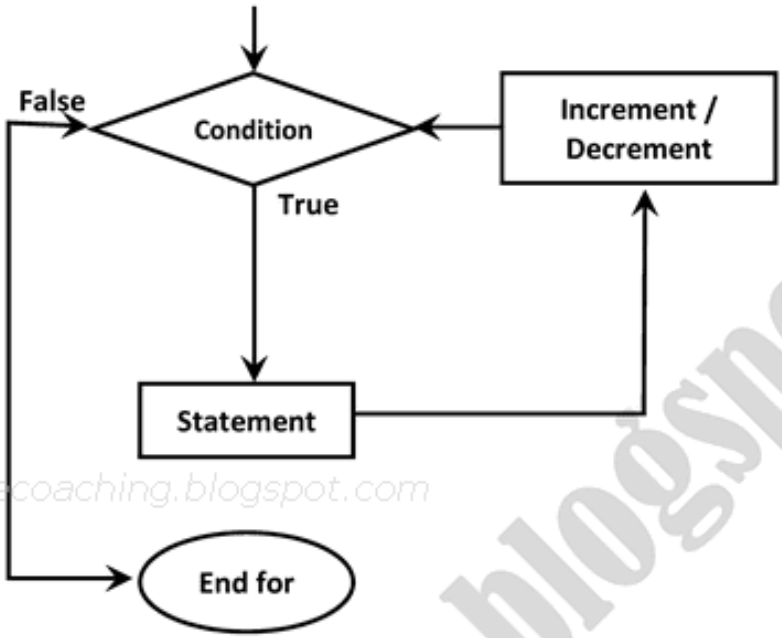
SYNTAX

The general syntax of for loop is:

for (initialization; test condition; increment/decrement)

```
{
  statement(s);
}
```

Flow Diagram



for Loop Example

```
#include<iostream>
using namespace std;
int main ( )
{
  int i;
  for (i = 1; i <= 10; i++)
  {
    cout << i << " " ;
  }
  return 0;
}
```

OUTPUT
1 2 3 4 5 6 7 8 9 10

Q.11: Describe while loop with syntax, flow diagram and example.

Ans. While Loop

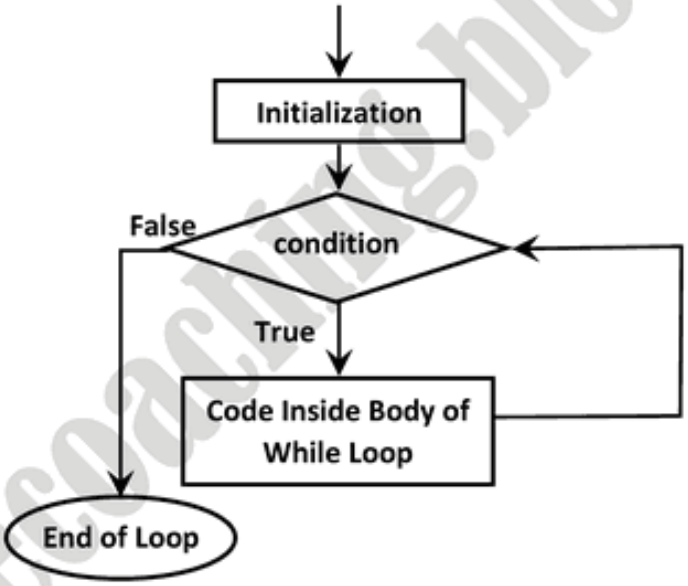
The while loop allows the programmer to specify that an action is to be repeated while some conditions remain true. It is used when the exact number of loop repetitions before the loop execution begins are not known.

Syntax

The general syntax of a while loop is:

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


Flow Diagram



while Loop Example

```
#include<iostream>
using namespace std;
int main( )
{
int i = 1;
while (i <= 10)
{
```

```
cout << I << "  " ;
i++,
}
return 0;
}
```

Q.12: Describe do while loop with syntax, flow diagram and example.

Ans. **do while Loop**

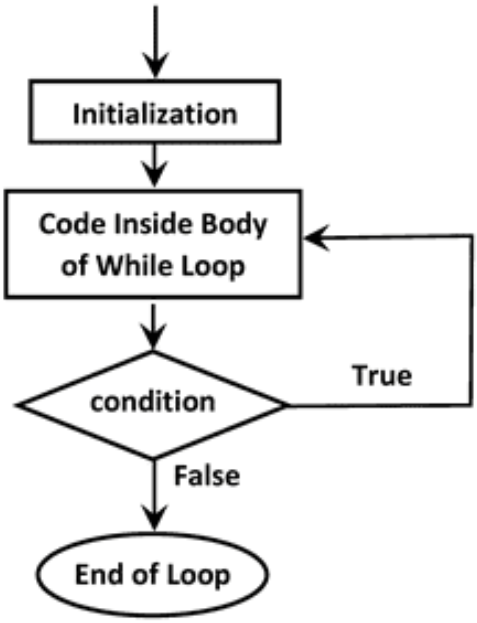
A do-while loop is similar to a while loop, except that a do-while loop is guaranteed to execute at least one time, even if the condition fails for the first time. Unlike for and while loops, which test the loop condition at the start of the loop, the do while loop, the do while loop checks its condition at the end of the loop.

Syntax

The general syntax of a do-while loop is:

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

Flow Diagram



do while Loop Example

```
#include<iostream>
using namespace std;
int main( )
{
 int i = 1;
do
{
 cout << i << "  ";
 i++;
}
 while (i <= 10);
 return 0;
}
```


OUTPUT

1 2 3 4 5 6 7 8 9 10

Q.13: Describe nested loop with example.

Ans. **Nested Loops**

When one for, while or do while loop is existed within another loop, they are said to be nested. The inside loop is completely repeated for each repetition of the outside loop.

Nested Loop Example

```
#include<iostream>
using namespace std;
int main( )
{
int row, column;
for (row = 1 ;  row <= 5 ; row++)
{
for (column = 1 ; column <= 3 ; column++)
{
cout <<  " * ";
}
```

```
cout <<  " \n ";
}
return 0;
}
```

OUTPUT

```
* * *
* * *
* * *
* * *
* * *
```

Q.14: Write down the differences between for while and do while loop.

Ans. Differences Between for, while AND do while loop

	for loop	while loop	do while loop
1.	It is pre-test loop because condition is tested at the start of loop	It is pre-test loop because condition is tested at the start of loop	It is post-test loop because condition is tested at the end of loop which executes loop at least once.
2.	It is known as entry controlled loop	It is known as entry controlled loop	It is known as exit controlled loop
3.	If the condition is not true first time then control will never enter in a loop	If the condition is not true first time then control will never enter in a loop.	Even if the condition is not true for the first time the control will enter in a loop.
4.	There is no semicolon; after the condition in the syntax of the for loop.	There is no semicolon; after the condition in the syntax of the while loop.	There is a semicolon; after the condition in the syntax of the do while loop.
5.	Initialization and updating is the part of the syntax.	Initialization and updating is not the part of the syntax.	Initialization and updating is not the part of the syntax

Q.15: What are jumps statements?

Ans. Jump Statements

These statements change the normal execution of program and jumps over the specific part of program.

Following are the jumps statements used in C++.

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

Q.16: Define break statement with example.

Ans. break STATEMENT

The break statement allows you to exit a loop or a switch statement from any point within Its body, bypassing its normal termination expression. It can be used within any C++ structure.

break EXAMPLE

```
#include<iostream>
using namespace std;
int main( )
{
  int count;
  for(count = 1; count <= 100; count++)
  {
    cout << count;
    if(count = 10)
    break;
  }
  return 0;
}
```

Q.17: Define continue statement with example.

Ans. continue STATEMENT

The continue statement is just the opposite of the break statement. Continue forces the next iteration of the loop to take place, skipping the remaining statements of its body.

continue EXAMPLE

```
#include<iostream>
using namespace std;
int main( )
```

```
{
  int count;
  for(count = 1; count <= 10; count++)
  {
    if(count = 5)
    continue;
    cout << count;
  }
  return 0;
}
```

Q.18: Define goto statement with Example.

Ans. goto STATEMENT

In c++ programming, the goto statement is used for altering the normal sequence of program execution by transferring control to some other parts of the program.

goto EXAMPLE

```
# include<iostream>
using namespace std;
int main( )
{
  float num, average, sum = 0.0;
  int i, n;
  cout << "Maximum number of inputs: ";
  cin >> n;
  for(i = 1; i <= n; i++)
  {
  cout << "Enter number" << I << ": ";
  cin >> num;
  if(num < 0.0)
  {
  // Control of the program move to jump:
  goto jump;
  }
  sum += num;
  }
  Jump:
  average = sum / (n);
```

```
  cout << "\n Average =" << average;
  return 0;
}
```

Q.19: Define return statement with syntax.

Ans. return STATEMENT

The return statement returns the flow of the execution to the function from where it is called. This statement does not mandatorily need any conditional statements. As soon as the statement is executed, the flow of the program stops immediately and return the control from where it was called.

SYNTAX

return (expression / value);

Q.20: Define exit( ) statement with Syntax.

Ans. exit() STATEMENT

The exit function, declared in <stdlib.h>, terminates a C++ program. The value supplied as an argument to exit is returned to the operating system as the program's return code or exit code.

SYNTAX

void exit (int);