COMPUTER SCIENCE 10TH – DETAILED QUESTION ANSWERS

➔ INPUT / OUTPUT HANDLING IN C++

Chapter # 03

ADAMJEECOACHING.BLOGSPOT.COM

Q.1: Describe basic structure of C++ program.

Ans. BASIC STRUCTURE OF C++ PROGRAM

C++ program is mainly divided in three parts:

1. Preprocessor Directives
2. Main Function Header
3. Body of program / Function

Basic structure of C++ program is given below:

```
1. #include<iostream>
2. using namespace std;
3. int main( )
4. {
5. statements;
6. return 0;
7. }
```

Q.2: Describe elements of basic structure.

Ans. EXPLANATION OF BASIC STRUCTURE

1. #include<iostream>

The statement starts with # symbol is called preprocessor directives. This statement tells the preprocessor to include the contents of iostream header file in the program before compilation. This file is required for input-output statements.

2. using namespace std;

This statement is used to instruct the compiler to use standard namespace. A namespace is a declarative location that provides a scope to the identifiers. Namespace std contains all the classes, objects and functions of the standard C++ library.

3. int main()

This statement is a function and used for the execution of C++ program. int means it returns integer type value.

4. **{**

This symbol represents the start of main function.

5. **statements;**

Statements are instructions that performs particular task. Statement terminator (;) is used to end every statement in C++ program.

6. **return 0;**

This statement is used to return the value to the operating system. By default, main function returns integer value 0.

7. **}**

This symbol represents the end of main function.

Q.3: Define comments in C++. Also define its types.

Ans. COMMENTS IN C++

Comments are special remarks that helps to understand different parts of the code in C++ program. Comments are ignored by the compiler. In C++, there are two types of comments statement.

1. Single line Comment
2. Multi line Comment

Q.4: Define single line comment In C++.

Ans. SINGLE LINE COMMENT

This type is used to write single line comment. Double slash (//) symbol is used at the start of each single line comment.

example

```
// This is my first C++ program
// This program displays a statement on screen
#include<iostream>
int main( )
{
puts("My First C++ Program");
return 0;
}
```

Q.5: Define multi line comment in C++.

Ans. MULTI LINE COMMENT

This type is used to write multi line comment. Symbols (/* and */) are used at the start and end of comment statements.

example

```
/* This is my first C++ program
This program displays a statement on screen */
#include<iostream>
int main( )
{
puts("My First C++ Program");
return 0;
}
```

Q.6: What do you mean by input / output statement in C++?

Ans. INPUT / OUTPUT STATEMENTS IN C++

Input and output statements are used to perform input & output operations in C++. These input / output statements are stored in header files like <iostream>. At the beginning of every program these header files must be included.

Q.7: Define output functions or statements in C++. Define cout statement and puts statement with syntax and example in C++.

Ans. OUTPUT FUNCTION / STATEMENTS IN C++

cout STATEMENT

cout stands for "Character Output". In C++, cout sends formatted output to standard output devices, such as the screen. cout object is used along with the insertion operator (<<) for displaying output.

syntax

```
        cont << variable; or cont << exp. / string;
```

example

```
        cont << "This is C++ Programming";
        cont << num1;
```

puts() STATEMENT

This function is used to print the string output. After printing the screen new line is automatically inserted.

syntax

```
puts("string constant");
```

example

```
puts("This is C++ Programming");
```

Q.8: Define input functions or statements in C++.

Ans. INPUT FUNCTION / STATEMENTS IN C++

following area the input functions / statement in C++.

- cin STATEMENT
- getchar( ) STATEMENT
- getch( ) STATEMENT
- getche( ) STATEMENT
- gets( ) STATEMENT

Q.9: Define cin statement with syntax and example in C++.

Ans. cin STATEMENT

cin stands for "Character Input". In C++, cin reads formatted data as input from keyboard. cin object is used along with the extraction operator (>>) to accept data from standard input device.

syntax

```
cin >> variable;
```

example

```
#include<iostream>
using namespace std;
int main( )
{
int b;
cin >>b; // cin takes input in "b" variable
return 0;
}
```

Q.10: Define getchar() statement with example in C++.

Ans. getchar() STATEMENT

The getchar( ) function reads the available character from the keyboard. This function reads only single character at a time. When the user presses the key, getchar( ) requires Enter key to be pressed. This function is defined in <stdio.h> header file.

example

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main( )
{
char b;
cout << "\n Enter a character:";
b = getchar( );
cout << " \n Input character is " << b;
return 0;
}
```

Q.11: Define getch( ) statement with example in C++.

Ans. getch() STATEMENT

The getch( ) function reads the character from the keyboard. This function reads only single character and not printed on the screen. This function takes input and does not requires Enter key to be pressed. This function is defined in <conio.h> header file.

example

```
#include<iostream>
#include<conio.h>
using namespace std;
int main( )
{
char ch;
cout << "\n Enter a character:";
ch = getch( );
cout << " \n Input character is " << ch;
return 0;
}
```

Q.12: Define getche( ) statement with example in C++.

Ans. getche() STATEMENT

The getche( ) function reads the character from the keyboard and echoes on the screen. It reads only single character and displays on the screen. This function takes input and does not require Enter key to be pressed. This function is defined in <conio.h> header file.

example

```
#include<iostream>
#include<conio.h>
using namespace std;
int main( )
{
char ch;
cout << "\n Enter a character:";
ch = getche( );
cout << " \n Input character is " << ch;
return 0;
}
```

Q.13: Define gets( ) statement with syntax and example in C++.

Ans. gets() STATEMENT

This function is used to reads characters or the string input and stores them until a newline character found. This function is defined in <cstdio.h> header file.

syntax

```
gets("variable");
```

example

```
#include<iostream>
#include<conio.h>
using namespace std;
int main( )
{
char name [25];
cout << "\n Enter your name:";
gets(name);
cout << "\n Your Name is " << name;
return 0;
}
```

Q.14: What is statement terminator?

Ans. STATEMENT TERMINATOR (;)

In C++, statement terminator is used to end the statement. Statements are terminated with semicolon (;) symbol. Every statement in C++ must be terminated otherwise and error message will occur.

Q.15: Write down in detail about escape sequences in C++.

Ans. ESCAPE SEQUENCES

Escape sequences are used to control the cursor moves on screen by using special codes. An escape sequence is a special non-printing characters consists of the escape character (the backslash "\") and a second (code) character. The list of the escape sequences is given below:

Escape Sequence	Explanation with Example
\n	**Newline**. Position the cursor at the beginning of the next line. **Example**: cout << "\n";
\t|	**Horizontal tab**. It move the cursor to the next tab stop. **Example**: cout << "\t";
\\	**Backslash**. Insert a backslash character in a string. **Example**: cout << "\\";
\a	**Alert**. Produces a beep sound or visible alert. **Example**: cout << "\a";
\b	**Backspace**. It moves the cursor backspace. **Example**: cout << "\b";
\r	**Carriage Return**. Moves the cursor to the beginning of the current line. **Example**: cout << "\r";
\’	**Single Quotation**. It is used to print apostrophe sign ('). **Example**: cout << "\'";
\”	**Double Quotation**." It is used to print quotation mark ('). **Example**: cout << " \" ";

Q.16: What are operators? Write down the name of operators used in C++.

Ans. OPERATORS IN C++

Operators are the symbols which tell the computer to execute certain mathematical or logical operations. A mathematical or logical expression is generally formed with the help of an operator. C++ programming offers a number of operators which are classified into the following categories.

1. Arithmetic Operators
2. Increment Operators
3. Decrement Operators
4. Relational Operators
5. Logical/Boolean Operators
6. Assignment Operators
7. Arithmetic Assignment Operators

Q.17: What are arithmetic operators?

Ans. ARITHMETIC OPERATORS

Arithmetic operators are used to perform mathematical operations. All operators used integer & floating-point data type except remainder or modulas operator.

Operator	Operation	Example
+	Addition: It is used to perform addition.	a + b
-	Subtraction: It is ·used to perform subtraction.	a – b
*	Multiplication: It is used to perform multiplication.	a * b
/	Division; It is used to perform division.	a / b
%	Remainder Or Modulas: Find remainder after integer division.	a % b

Q.18: Define all arithmetic operators with examples.

Ans. SIMPLE CALCULATOR PROGRAM IN C++ USING ARITHMETIC OPERATORS

```
#include<iostream>
#include<conio.h>
#include<stdio.h>
using namespace std;
intmain( )
{
int a, b, add, sub, mul, rem;
float div;
cout << "\n \t SIMPLE CALCULATOR";
cout << "\n \t Enter the value of a... ";
cin >> a;
cout << "\n \t Enter the value of a...";
cin >> b;
add = a + b;
cout << " \n \t Addition of "<< a << "and" << b << "is" << add;
sub = a - b;
cout << " \n \t Subtraction of "<< a << "and" << b << "is"<< sub;
mul = a * b;
cout << " \n \t Multiplication of "<< a << "and" << b << "is"<< mul;
div = a / b;
cout << " \n \t Division of " a << "and" << b << "is" << div;
rem=a % b;
cout << " \n \t Remainder of "<< a << "and" << b << "is" << rem;
return 0;
}
```

OUTPUT

```
SIMPLE CALCULATOR
Enter the value of a 30
Enter the value of b 20
Addition of 30 and 20 is 50
Subtraction of 30 and 20 is 10
Multiplication of 30 and 20 600
Division of 30 and 20 is 1
Remainder of 30 and 20 is 10
```

Q.19: Define increment operators with example in C++.

Ans. INCREMENT OPERATORS

C++ provides the unary increment operator. It is used to be incremented a variable by 1. Increment operator represented by ++ (double plus sign). The increment operators are used in two ways (postfix & Prefix) summarized below:

Operators	**Explanation**
++a (Prefix)	Increment a by 1, then use the new value of a in the expression in which a resides.
a++ (Postfix)	Use the current value of a in the expression in which a resides, then increment a by 1.

EXAMPLE (PREFIX)

```
#include<iostream>
using namespace std;
int main( )
{
int ch = 5;
cout << "\n Value of ch is :" << ++ch;
return 0;
}
```

OUTPUT

```
Value of ch is 6
```

EXAMPLE (POSTFIX)

```
#include<iostream>
using namespace std;
int main( )
{
int ch = 5 ;
cout << "\n Value of ch is :" << ch++;
return 0;
}
```

OUTPUT

```
Value of ch is 5
```

Q.20: Define decrement operator in C++.

Ans. DECREMENT OPERATORS

C++ also provides the unary decrement operator. It is used to be decremented a variable by 1. decrement operator represented by -- (double minus sign). The decrement operators are used in two ways (postfix & Prefix) summarized below:

Operators	**Explanation**
--a (Prefix)	Decrement a by 1, then use the new value of a in the expression in which a resides.
a-- (Postfix)	Use the current value of a in the expression in which a resides, then decrement a by 1.

Q.21: Define Relational operators with example in C++.

Ans. RELATIONAL OPERATORS

Relational operators are used when we have to make comparisons. It is used to test the relation between two values. The result of comparison is True (1) or False (0). C++ programming offers following relational operators:

Operator	**Operations**	**Example**
<	It checks the value on left is less than value on right.	a < b
>	It checks the value on left is greater than value on right.	a > b
<=	It checks the value on left is less than or equal to value on right.	a <= b
>=	It checks the value on left is greater than or equal to value on right.	a >= b
==	It checks the equality of two values.	a == b
!=	It checks the value on left is not equal to value on right.	a != b

PROGRAM USING RELATIONAL OPERATORIN C++

```
#include<iostream>
using namespace std;
int main( )
{
int x = 20, y = 10;
if(x > y)
cout << "X is greater than Y";
else
cout << "Y is greater than X";
return 0;
}
```

OUTPUT

```
X is greater than Y
```

Q.22: Define logical operators with example in C++.

Ans. LOGICAL OPERATORS

Logical operators are used when more than one conditions are to be tested and based on that result, decisions have to be made. C++ programming offers three logical operators. They are:

Operator	Operations	Expression
&&	Logical AND. The condition will be true if both expressions are true.	1 if a == b && c == d; else 0
||	Logical OR. The condition will be true if anyone of the expressions are true.	1 if a == b | | c > d; else 0
!	Logical NOT. The condition, will be inverted, False becomes true & true becomes false.	1 if !(a == 0); else 0

PROGRAM USING LOGICAL OPERATORS IN C++

```
#include<iostream>
#include<conio.h>
using namespace std;
int main( )
{
  int num1 = 30, num2 = 40;
cout << "Logical Operators Example \n";
if(num1<=40 && num2>=40)
{
cout << "Num1 is less than and Num2 is greater than or equal to 40 \n";
}
if(num1 >= 40 || num2 >= 40)
{
  cout << "Num 1 or Num 2 is greater than or equal to 40 \n";
}
getch();
return 0;
}
```

OUTPUT

```
Logical Operators Example
Num1 is less than and Num2 is greater than or
equal to 40
Num 1 or Num 2 is greater than or equal to 40
```

Q.23: Write down the difference between relational and logical operators.

Ans. DIFFERENTIATE BETWEEN RELATIONAL OPERATOR AND LOGICAL OPERATOR

RELATIONAL OPERATOR

- Relational operators compare any values in the form of expressions.
- Relational operators are binary operators because they require two operands to operate.
- Relational operators return results either 1 (TRUE) or 0 (FALSE).

LOGICAL OPERATOR

- Logical operators perform logical operations on boolean values 1 (TRUE) and 0 (FALSE).
- Logical operator is usually used to compare one or more relational expressions.
- Logical operator also return output as 1 (TRUE) or 0 (FALSE).

Q.24: Define assignment operator.

Ans. ASSIGNMENT OPERATOR

Assignment operator (=) are used to assign result of an expression or a value to a variable. The associativity of assignment operators is right to left means value or expression at the right is assigned to the left side variable.

Q.25: Define arithmetic assignment operators with example.

Ans. ARITHMETIC ASSIGNMENT OPERATOR

Arithmetic assignment operator is a combination of arithmetic. and assignment operators. This operator first performs an arithmetic operation on the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.

OPERATOR	DESCRIPTION
+= (Addition-Assignment)	Adds the right operand to the left and assigns the result to the left operand.
-= (Subtraction-Assignment)	Subtracts the right operand to the left and assigns the result to the left operand.
*= (Multiplication-Assignment)	Multiplies the right operand to the left and assigns the result to the left operand.
/= (Division-Assignment)	Divides the right operand to the left and assigns the result to the left operand.

PROGRAM USING ASSIGNMENT & ARITHMETIC ASSIGNMENT OPERATORS IN C++

```
#include<iostream>
#include<conio.h>
using namespace std;
int main( )
{
int a = 10;
cout << "Value of a using assignment operator is "<< a << "\n";
a + = 10;
cout << "Value of a using addition assignment operator is "<< a << "\n";
a -= 10;
cout << "Value of a using subtraction assignment operator is "<< a << "\n";
a * = 10;
cout << "Value of a using multiplication assignment operator is "<< a << "\n";
a /= 10;
cout << "Value of a using division assignment operator is "<< a << "\n";
return 0;
}
```

```
OUTPUT
Value of a using assignment operator is 10
Value of a using addition assignment operator is 20
Value of a using subtraction assignment operator is 10
Value of a using multiplication assignment operator is 100
Value of a using division assignment operator is 10
```