COMPUTER SCIENCE 10TH – DETAILED QUESTION ANSWERS

➔ PROBLEM SOLVING AND ALGORITHM DESIGNING

Chapter # 01

Q.1: What is problem solving?

Ans. PROBLEM SOLVING:

Problem solving is a procedure to figure out the solution of complex problems. Problem solving is the main process in computer programming, where programmers first understand the problem, plan the solution and then understand how to translate algorithm into something a computer can do, and finally how to write the specific syntax or code required by a computer to get the job done.

Q.2: Define Problem?

Ans. PROBLEM:

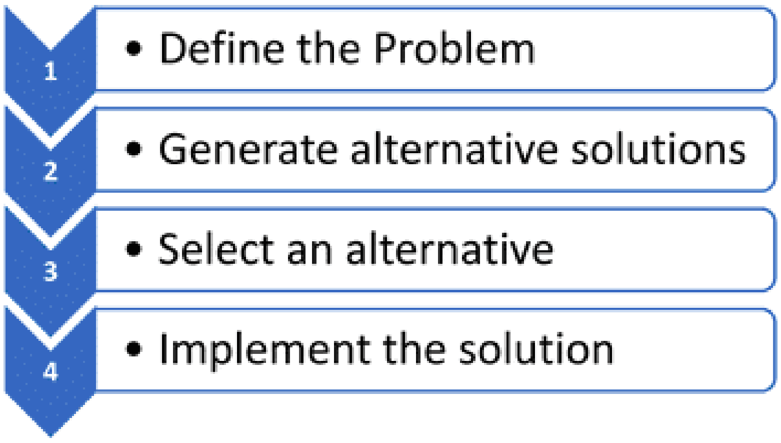
Problem sometimes referred as an issue, is any situation that occurs and is unexpected or prevents something from occurring. In programming a problem is considered as a matter which is difficult to solve, it is a complex task or routine.

Q.3: What are the strategies of problem solving?

Ans. PROBLEM SOLVING PROCESS:

Finding a suitable solution for issues can be accomplished by the following the basic four step problem-solving process given below.

1. Define the problem.
2. Generate alternative solutions.
3. Evaluate and select an alternative.
4. Implement and follow up on the solution.



1. DEFINE THE PROBLEM

Specifically, the task of defining the problem consists of identifying what you know (input-given data), and what you want to obtain (output-the result). During this step, it is more important to understand described problem.

2. GENERATE ALTERNATIVE SOLUTIONS

Many alternative solutions to the problem should be generated to achieve the desired results. Generating multiple alternatives can enhance the value of best solution. So, it is good to generate a list of all possible solutions and evaluate to decide which one is the best for that particular problem.

3. EVALUATE AND SELECT AN ALTERNATIVE

Good problem-solving approach uses a series of considerations when selecting the best alternative. In this step, evaluate all possible solutions related to a target standard and select the best alternative. A particular alternative will solve the problem without causing other unanticipated problems.

4. IMPLEMENT AND FOLLOW UP ON THE SOLUTION

The implementation of best solution also includes planning on what happens when something goes wrong with the selected solution. It is also important to follow up and track the results of solution at every stage of implementation.

Q.4: What is algorithm?

Ans. ALGORITHM

An algorithm is a procedure or formula for solving a problem. It can be defined as the set of well-defined step by step instructions to solve a program. In computer science an algorithm usually means a small procedure that solves a specific problem designs; algorithms are widely used throughout all areas of IT. There are two common methods to represent algorithm designs; pseudocode and flowcharts.

Q.5: Explain the role of algorithm in problem solving. OR Describe the advantages of algorithm.

Ans. ROLE OF ALGORITHM IN PROBLEM SOLVING

Algorithms give us the most ideal option of accomplishing a task in problem solving. Algorithms are used to find the best possible way of solving a problem. In doing so they improve the efficiency of a program. With the best algorithm, a computer program will be able to produce very accurate results. An algorithm can be used to improve the speed at which a program executes.

Q.6: What are the qualities of a good algorithm?

Ans. QUALITIES OF A GOOD ALGORITHM

1. Inputs and outputs should be defined exactly.
2. Every step in an algorithm must be clear, precise and unambiguous.
3. It should be effective among many different ways to solve a problem.
4. It shouldn't have computer code. Instead, the algorithm must be written in such a way that, it can be used in similar programming languages.

Q.7: Write some examples of algorithm.

Ans. ALGORITHM EXAMPLES

ALGORITHM 1: MAKING A CUP OF TEA

```
Step 1: Start.
Step 2: Put the teabag in a cup.
Step 3: Fill the kettle with water.
Step 4: Boil the water in the kettle.
Step 5: Pour some of the boiled water into the cup.
Step 6: Add milk to the cup.
Step 7: Add sugar to the cup.
Step 8: Stir the tea.
Step 9: Drink the tea.
Step 10: Stop
```

ALGORITHM 2: SUBTRACTION OF TWO NUMBERS

```
Step 1: Start
Step 2: Declare variables n1, n2 & sub
Step 3: Read values n1 & n2
Step 4: Subtract n1 & n2 and assign the result to sub
          sub = n1 - n2
Step 5: Display sub
Step 6: Stop
```

ALGORITHM 3: ADDITION OF TWO NUMBERS

```
Step 1: Start
Step 2: Declare variables n1, n2 & sum
Step 3: Read values n1 & n2
Step 4: Add n1 & n2 and assign the result to sum
          sum = n1 + n2
Step 5: Display sum
Step 6: Stop
```

ALGORITHM 4: MULTIPLICATION OF TWO NUMBERS

```
Step 1: Start
Step 2: Declare variables n1, n2 & mul
Step 3: Read values n1 & n2
Step 4: Multiply n1 & n2 and assign the result to mul
          mul = n1 * n2
Step 5: Display mul
Step 6: Stop
```

ALGORITHM 5: AREA OF CIRCLE

```
Step 1: Start
Step 2: Declare variable r and area
Step 3: Read value of r
Step 4: Apply formula {Area = (3.142) * r * r}
        area = (3.142) * r * r
Step 5: Display area
Step 6: Stop
```

Q.8: What is flowchart?

FLOWCHART

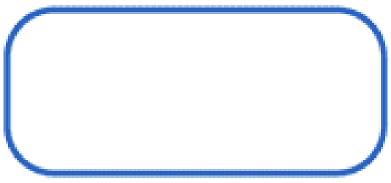
A flowchart is a pictorial representation of an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. It is used to show the sequence of steps and logic of solution to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Q.9: Define different symbols used in flowchart.

Ans. FLOWCHART SYMBOLS

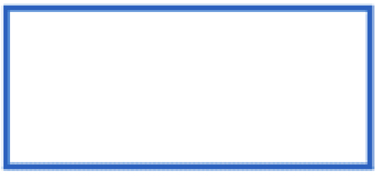
1. TERMINAL OR START / STOP

The terminal symbol is used to indicate the beginning (START) and ending (STOP) in the program logic flow.



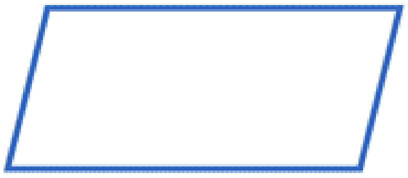
2. Process

Process symbol is used to illustrate a process, action or an operation. These are represented by rectangles.



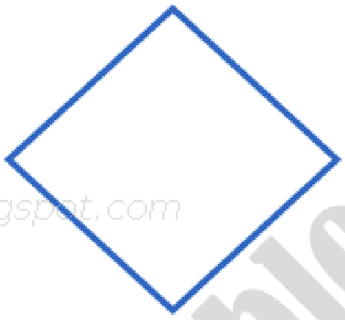
3. Input / Output

The input / output symbol is used to denote any type of input data or output information.



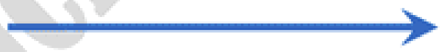
4. Decision

The decision symbol is used in a flowchart to indicate a point at which a decision has to be made and a branch to one of two or more alternative points are possible.



5. Arrows or Flowlines

Flowlines with arrowheads are used to indicate the flow of operation. It shows the exact sequence in which the instructions are to be executed.



Q.10: Write down the importance of flowchart in problem solving.

Ans. IMPORTANCE OF FLOWCHART IN SOLVING A PROBLEM

1. Communication

   Flowcharts are better way of communicating the logic of a system.

2. Effective Analysis

   With the help of flowchart, problem can be analyzed in more effective way therefore reducing cost and wastage of time.

3. Efficient Coding

   The flowcharts act as a guide or blueprint during the systems analysis and program development phase.

4. **Proper Debugging**

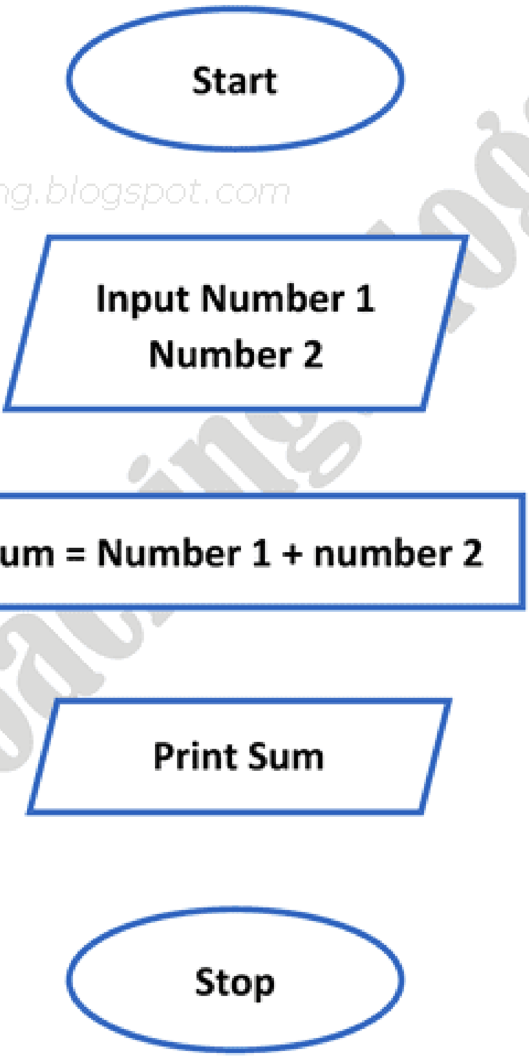
   The flowchart helps in debugging a process.

5. **Efficient Program Maintenance**

   The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part.

Q.11: Write down the example of flowchart.

Ans. FLOWCHART EXAMPLE



Q.12: What are the differences between algorithm and flowchart?

Ans. DIFFERENCES BETWEEN ALGORITHM & FLOWCHART

ALGORITHM	FLOWCHART
It is step by step solution of program	It is the diagrammatical representation which shows the flow of data.
Text in common language is used in algorithm.	Symbols and shapes are used in flowchart.
It is difficult to write and understand.	It is easy to construct and understand
Algorithm doesn't have any specific rules.	Flowchart have specific rules for its construction.

Q.13: Define data structures.

Ans. Data Structures:

Data structure is a particular means of organizing and storing data in computers in such a way that we can perform operations on the stored data more efficiently. Data structures have a wide and diverse scope of usage across the fields of Computer Science and Software engineering. Data structure may be linear or non-linear.

Q.14: Describe linear data structure.

Ans: LINEAR DATA STRUCTURE

Linear data structure arranges the data in a linear fashion. The data elements are arranged sequentially such that the element is directly linked to its previous and the next elements. This data structure supports single-level storage of data. And hence, transfer of the data is achieved through a single run only.

Examples of linear data structures are Stack, Queue, Array, etc.

Q.15: Define stack and its operations.

Ans. STACK

Stack is a linear data structure that stores information in such a way that the last item stored is the first item retrieved. It is based on the principle of LIFO (Last-in-first-out). The stack in digital computers is a group of memory locations with a register that holds the address of top of element. Stacks usually performs two operations.

Push: It is used to insert an element on top of stack.

Pop: It is used to delete an element from top of stack.

Q.16: Define queue data structure and its operations.

Ans. QUEUE:-

Queue is the type of linear data structure where the elements to be stored follow the rule of First in First Out (FIFO). In queue, the element that was added first is removed first) In this data structure, both ends are used for the insertion and the removal of data. The two main operations of queue are enqueue and dequeue.

Enqueue: It refers to the process where inserting an element is allowed to the collection of data.

Dequeue: It refers to the process where removal of elements is allowed, which is the first element of queue.

Q.17: Define array data structure and its operations.

Ans. ARRAY:

The array is the type of linear data structure that stores homogeneous (same type) elements at memory locations which are consecutive. The same types of objects are stored sequentially in an array. The main idea of an array is that multiple data of the same type can be stored together. Before storing the data in an array, the size of the array has to be defined. The location of elements stored in an array has a numerical value called index to identify the element. Some operations of array are:

Traverse: Go through the elements and print them.

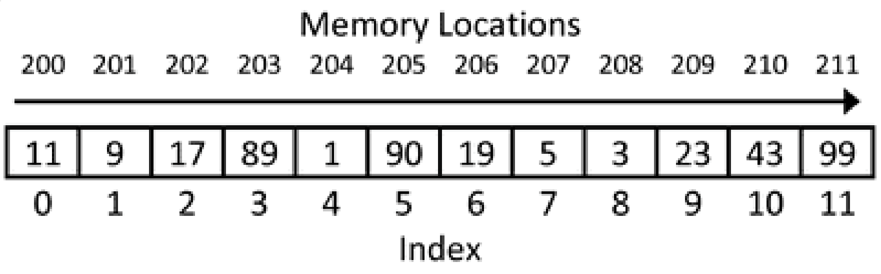
Search: Search for an element in the array by its value or its index.

Update: Update the value of an existing element at a given index.

Insert: Insert an element in array.

Deletion: Delete an element from array.

Sorting: Rearrange a given array or list of elements.



Q.18: Describe nonlinear data structure.

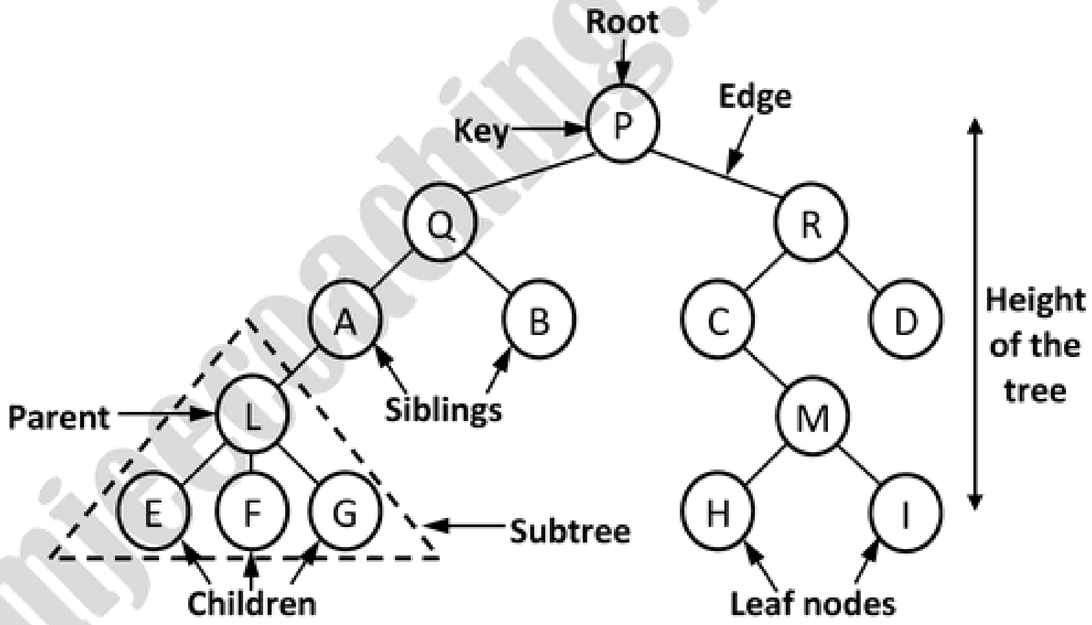
Ans. NON-LINEAR DATA STRUCTURES:

Non-linear data structures are those data structures in which data items are not arranged in a sequence. In this structure, each element can have multiple paths to connect to other elements. Non-linear data structures allow multi-level storage. Non-linear data structures are not easy to implement but are more efficient in utilizing computer memory. Examples of non-linear data structures are Tree, Graphs, etc.

Q.19: Define tree data structure.

Ans. TREE:

A tree can be defined as a finite set of data items (nodes) in which data items are arranged in branches and sub-branches. It consists of various linked nodes and has a hierarchical tree structure that forms a parent-child relationship. In tree structure, the first node is called root node. The node connected to root node are called parent node and another node connected to parent node is called child node. Various types of trees have been developed throughout the past decades, one of them are binary tree or binary search tree.



Q.20: Define graph data structure.

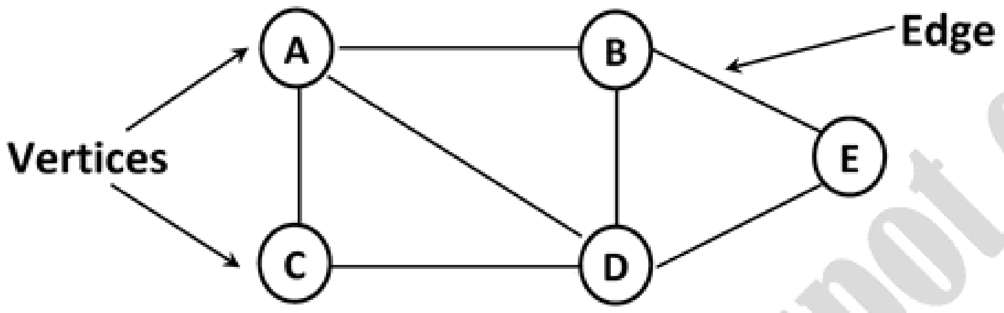
Ans. GRAPH

Graphs are used to solve many real-life problems. A graph is a non-linear data structure that has a finite number of vertices and edges, and these edges are used to connect the vertices. The vertices are used to store the data elements, while the edges represent the relationship between the vertices. The order of a graph is the number of vertices in the graph. The size of a graph is the number of edges in the graph. Two nodes are said to be adjacent if they are connected to each other by the same edge.

There are two types of graphs:

(a) Undirected graph

(b) Directed graph

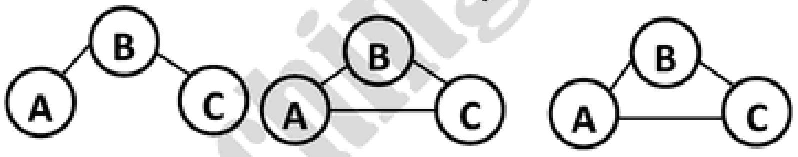


Q.21: Define undirected graph.

Ans. UNDIRECTED GRAPH

An undirected graph is a graph in which all its edges have no direction. It can go in both ways between the two vertices.

Undirected Graphs

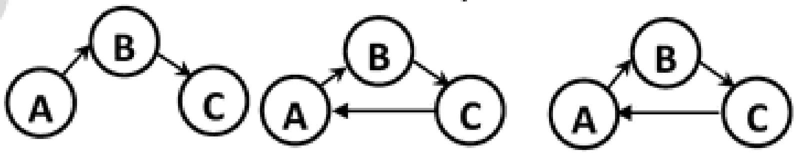


Q.22: Define directed graph.

Ans. DIRECTED GRAPH

A directed graph is a graph in which all its edges have a direction indicating what is the start vertex and what is the end vertex.

Directed Graphs



COMPUTER SCIENCE 10TH – DETAILED QUESTION ANSWERS

BASICS OF PROGRAMMING IN C++

Chapter # 02

Q.1: Define Computer Program?

Ans. COMPUTER PROGRAM:

A computer program is a set of instructions that performs a specific task when executed by a computer. A computer requires programs to function and typically executes the program's instructions in a central processing unit. A computer program is usually written by a computer programmer in a programming language.

Q.2: Define syntax OR What is syntax in programming?

Ans. SYNTAX IN PROGRAMMING LANGUAGE:

Syntax refers to the rules that define the structure of a language. Syntax in computer programming means a set of keywords and characters that a computer can understand, interpret and perform task associated with them. The syntax of a language must be followed, and if it is not followed, the code will not be understood by a compiler or interpreter. Different programming languages have different types of syntax.

Q.3: Describe classification of programming language.

CLASSIFICATION OF PROGRAMMING LANGUAGE:

Based on the accessibility of hardware, programming languages can be classified into the following categories:

1. Low-level language
2. Middle-level language
3. High level language

Q.4: Define low level language.

LOW-LEVEL LANGUAGE:

Low-level programming languages are those languages that are directly communicated with computer hardware. The two languages come under this category are Machine language and Assembly language.

Q.5: Define machine language.

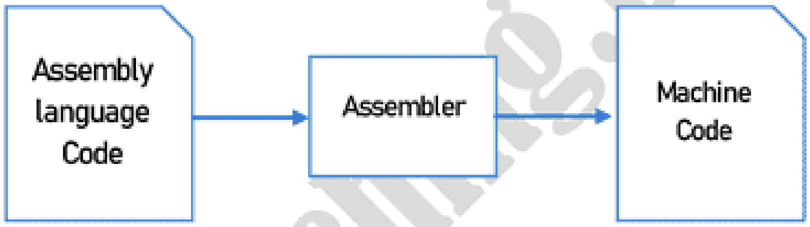
Ans. MACHINE-LEVEL LANGUAGE

Machine language is a collection of binary digits or bits that the computer reads and interprets. This language is the only language understood by computers while machine language is almost impossible for humans to use because they consist entirely of numbers (0s & 1s). Machine code doesn't require translator because machine code is directly executed by computer. It is also called first generation language.

Q.6: Define assembly language.

Ans. ASSEMBLY LANGUAGE

A program written in assembly language consists of a series of instructions called mnemonics that correspond to a stream of executable instructions. Assembly language code is translated by a translator called assembler. Assembly language uses keywords and symbols much like English and are easy to read, write and maintain as compared to machine language. It is also called second generation programming language.



Q.7: Define middle level language.

Ans. MIDDLE-LEVEL LANGUAGE:

The middle-level language lies in between the low level and high-level language. Middle-level language actually binds the gap between a machine level language and high-level languages, these languages are now become obsolete and are not in used.

Q.8: What do you know about high level language?

Ans. HIGH-LEVEL LANGUAGE:

High-level languages are relatively new and drastically revolutionized the programming world. It allows us to write computer code using instructions resembling everyday spoken language, usually English (for example: print, if, while). Programs written in a high-level language need to be translated by a translator (compiler or interpreter) into machine language before execution.

Q.9: write down the advantages or benefits of high-level language.

Ans. ADVANTAGES OF IDGH-LEVEL LANGUAGE

- High level language is much closer to human language so it is more suitable to write code in high level language.
- It is more or less independent or portable of the particular type of computer used.
- It is easier to read, write and maintain.

Q.10: Write down the differences between low-level and high-level languages.

Ans. DIFFERENCES BETWEEN LOW-LEVEL LANGUAGE AND IDGH-LEVEL LANGUAGE

Low Level Language	High Level Language
In low level language machine codes (0 and 1) are used as an instruction to the computer.	In high level language English like words are used as an instruction to the computer.
The execution of programs is quite fast.	The execution of programs is not very fast.
Instructions are directly understood by the CPU	Instruction are not directly understood by the CPU
No need to translate program. In case of assembly language assembler is required.	Translation of program is required.
The programs written in low level languages are machine dependent and are difficult to modify.	The programs written in high level languages are machine independent and are easy to modify.
The examples of low-level languages are: (1) Machine language (2) Assembly language	The examples of high-level languages are: BASIC, FORTRAN, COBOL, PASCAL, C languages etc.

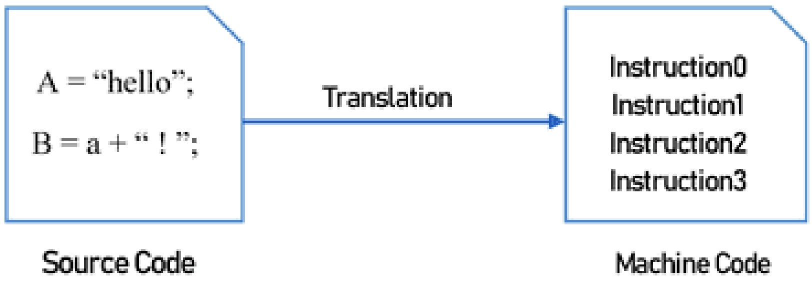
Q.11: Define translators or language translators.

Ans. TRANSLATORS OR LANGUAGE TRANSLATORS

Language translator is a computer program that converts high level language program into low level program (1s & 0s), or machine language. It transforms the source code into the object code (machine code) which understands directly by the computer processor. Translators also detect and report errors in the process of translation.

There are three types of language translators.

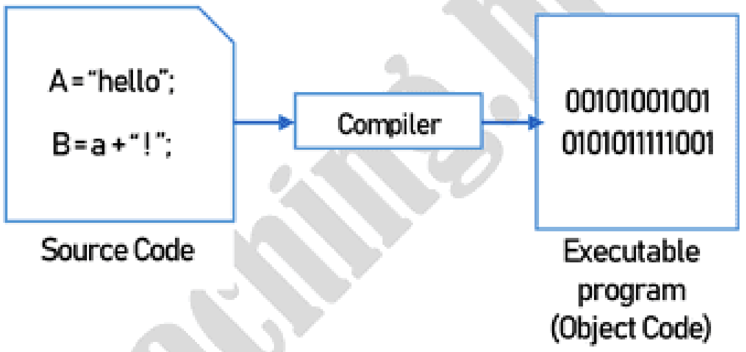
* Interpreter * Compiler * Assembler



Q.12: What is compiler?

Ans. COMPILER

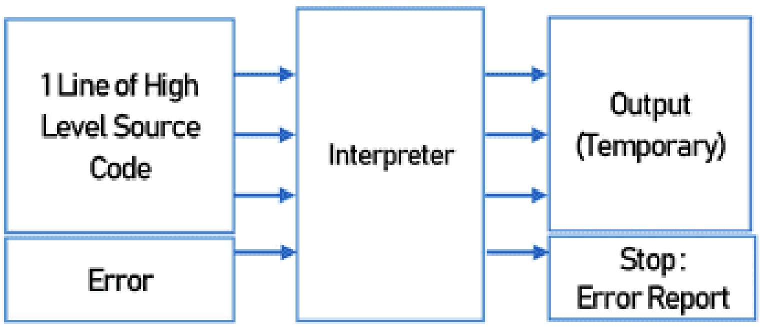
Compiler is a program that translates the high-level language program into machine language. Compiler translates the whole program at a time at once before it executed and makes a separate object file for the translated program. Translated program can be used multiple times without the need of retranslation of source codes. Each high-level language has its own compiler.



Q,13: Define Interpreter?

Ans. INTERPRETER

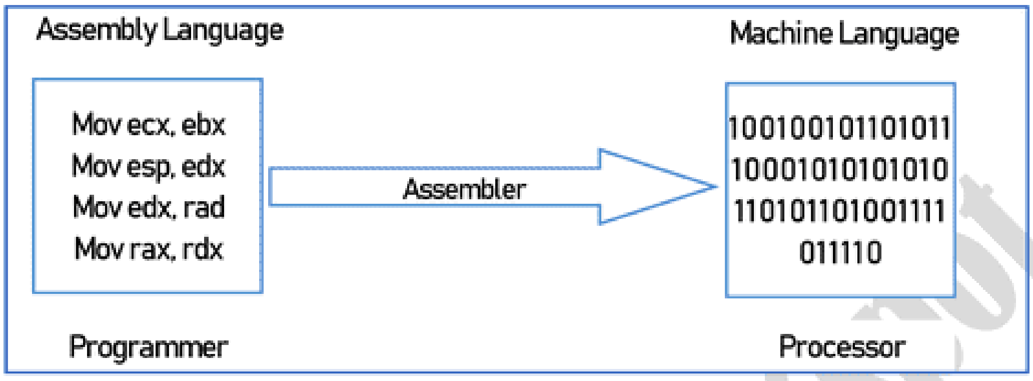
Interpreter is a language translator program, which converts high level program into machine language. It translates one instruction at a time. Interpreter does not make any object file and it translates the program every time when executed. An interpreter is faster than a compiler as it immediately executes the codes.



Q.14: Define Assembler.

Ans. ASSEMBLER

An assembler is a translator that converts assembly language program into machine language. An assembler translates assembly language code directly into machine code that can be understood by the CPU.



Q.15: Define source program or source code?

Ans. SOURCE PROGRAM / SOURCE CODE

A program or A program written by a programmer in any language other than machine language is called a source program or source code.

Q.16: Define object program or object code.

Ans. OBJECT PROGRAM / OBJECT CODE

Object program is a program or code that is converted into machine language. (OR) The output from a language translator, which consists of machine language instructions, is called the object program.

Q.17: Write down the differences between interpreter and compiler?

Ans. DIFFERENCES BETWEEN INTERPRETER AND COMPILER

Interpreter	Compiler
Interpreter translates high level language program into machine language line by line.	Compiler translates high level language program into machine language as a whole.
The interpreter translates the program every time when executed.	The compiler translates the program once at a time.

The interpreter does not make any object file.	The compiler makes a separate object file for the translated program
Errors are displayed for every instruction interpreted if any.	Errors are displayed after entire program is checked.
Examples: BASIC, LISP etc.	Examples: C Compiler, C++ compiler etc.

Q.18: Define errors or programming errors.

Ans. ERROR

An error describes any issue that arises unexpectedly that causes a program to not function properly. In general, there are three types of errors that occur in a computer program. Syntax Error, Logical Error and Runtime Error.

Q.19: What is syntax error?

Ans. SYNTAX ERROR

A syntax error is an error in the syntax due to violation of rules of whatever language program is being written. These are sometimes caused by simple typing mistakes.

Q.20: What is Logic error?

Ans. LOGIC ERROR

In computer programming, a logic error is a bug or error in planning the program's logic. They do not cause the program to crash or simply not work at all, they cause it to "misbehave" in some way, rendering wrong output of some kind. The computer does not tell you what is wrong.

Q.21: What is runtime error?

Ans. RUNTIME ERROR

A runtime error is a program error that occurs when a program is run on the computer and the result are not achieved due to some misinterpretation of a particular instruction. The code doesn't have any syntax or logic error but when it executes it cannot perform specific task.

Q.22: What is Programming Environment of C++ ?

Ans. PROGRAMMING ENVIRONMENT OF C++

Programming environment is an environment which supports execution of programming language smoothly and efficiently on a local computer to compile and run programs.

Q.23: Define IDE or integrated development environment?

Ans. INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

An integrated development environment (IDE) is software for building applications or programs that combines common developer tools into a single graphical user interface (GUI). IDE facilitates the development of applications designed to encompass all programming tasks in one application, one of the main benefits of an IDE is that they offer a central interface with all the tools, a developer needs. Dev-C++ is used for writing programs in C++ language, however there are many multiple language IDEs.

Q.24: Write down the benefits or advantages of Integrated development environment.

Ans. BENEFITS / ADVANTAGES OF INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

- IDE combines all tools that need for development. Programmers don't need to switch between different tools to design a layout, write the code, debug, build, etc.
- Many IDEs incorporate basic spelling checkers, so automatically check for errors to improve code.
- Libraries provide for functions in IDEs that are not included in the core part of the programming language.

Q.25: What are the components of Integrated Development Environment?

Ans. COMPONENTS OF INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

IDEs combining common tools that are necessary for a programmer to develop program, these are:

EDITING SOURCE CODE

Writing and editing source codes is a major part of programming. A text editor is used for writing and editing source codes with feature providing language specific autocompletion, and checking for bugs as code is being written.

SYNTAX HIGHLIGHTING

Syntax highlighting is a feature of IDEs that provides visual cues. keywords, and other words that have special meaning in languages are highlighted. This feature makes code easier to read or understand.

CODE COMPLETION

It is a feature of IDE that completely knows the language syntax that speeds up the process of coding by reducing typos and other mistakes. Autocompletion popups while typing, querying parameters of functions, query hints related to syntax errors.

COMPILER

Compiler is a component of IDEs that translate source code into machine code. IDEs provide automated build process. This feature can help automate developer tasks that are more common to save time.

Q.26: Define Linker, Loader and Debugging?

Ans. LINKER

Linker connects or links referenced library files with compiled code. It saved the linked objects into an executable file.

LOADER

This is the operating system's program that loads executable files into memory and directs the CPU to start running the program as directed by the IDE.

DEBUGGING

The process of removing errors from a program is known as debugging. Debugging

Q.27: What do you know about Dev C++?

Ans. INTRODUCTION TO DEV-C++

Dev-C++ is a fully featured graphical IDE (Integrated Development Environment) for programming in CIC++. Dev-C++ is developed by Bloodshed software. It was originally developed by Colin Laplace and first released in 1998. It is written in Delphi. With Dev C++ programmer can write Windows or console-based C/C++ programs easily.

Q.28: Write down the installation procedure of Dev C++.

Ans. INSTALLING AND CONFIGURINGDEV-C++ IDE

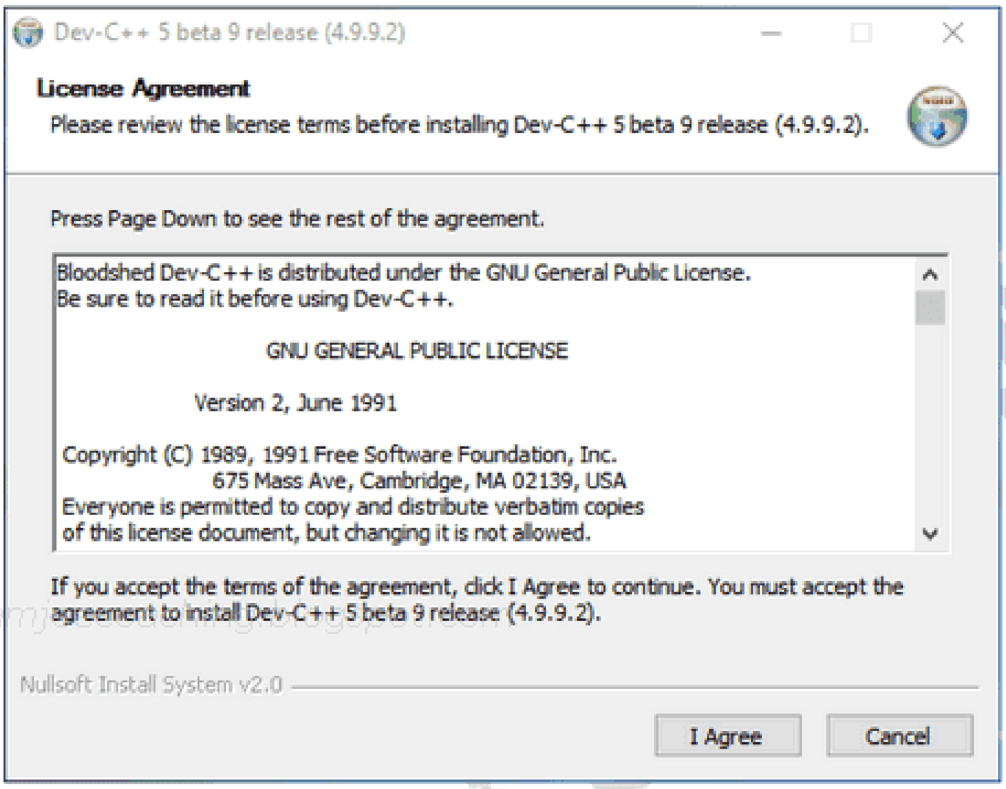
Dev-C++ is free available for download on Internet. After downloading the package begin the installation process. Following are the steps for installing Dec-C++ IDE.

STEP 1:

Select "English" as the language to be used for installation process.

STEP 2

Press "I Agree" button to agree the license agreement.



Dev-C++ 5 beta 9 release (4.9.9.2)

License Agreement

Please review the license terms before installing Dev-C++ 5 beta 9 release (4.9.9.2).

Press Page Down to see the rest of the agreement.

Bloodshed Dev-C++ is distributed under the GNU General Public License.
Be sure to read it before using Dev-C++.

GNU GENERAL PUBLIC LICENSE

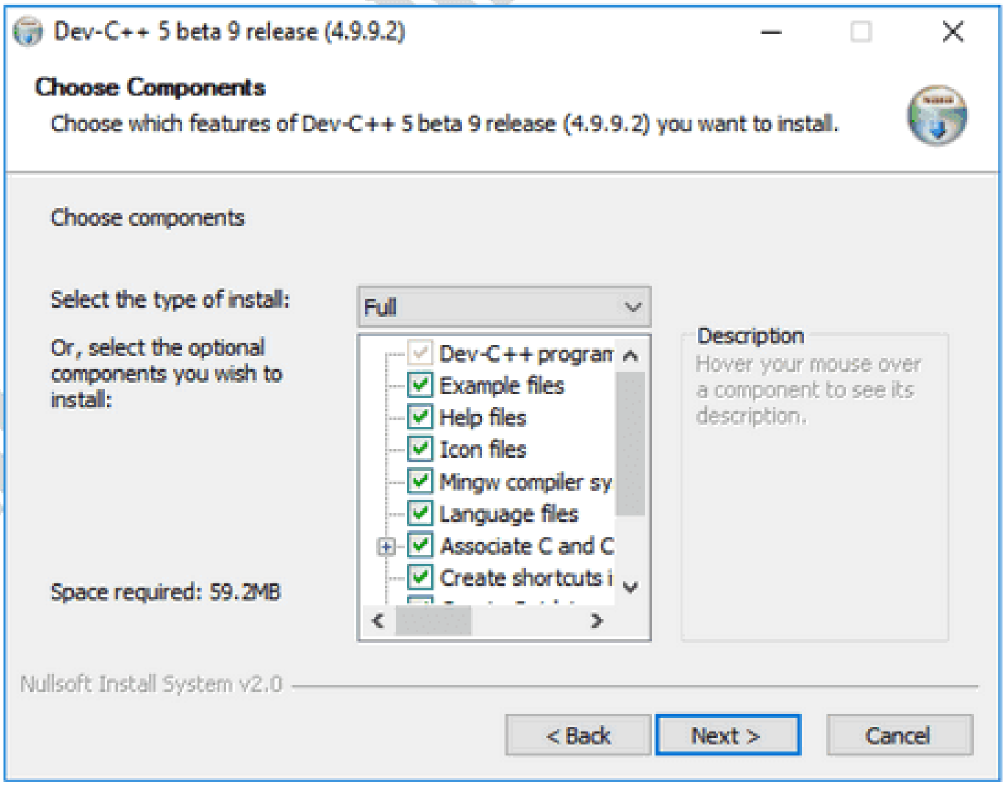
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
675 Mass Ave, Cambridge, MA 02139, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Dev-C++ 5 beta 9 release (4.9.9.2).

STEP 3

Select "Full" from the dropdown menu for "type of install". Click on "Next" to proceed.



Dev-C++ 5 beta 9 release (4.9.9.2)

Choose Components

Choose which features of Dev-C++ 5 beta 9 release (4.9.9.2) you want to install.

Choose components

Select the type of install: Full

Or, select the optional components you wish to install:

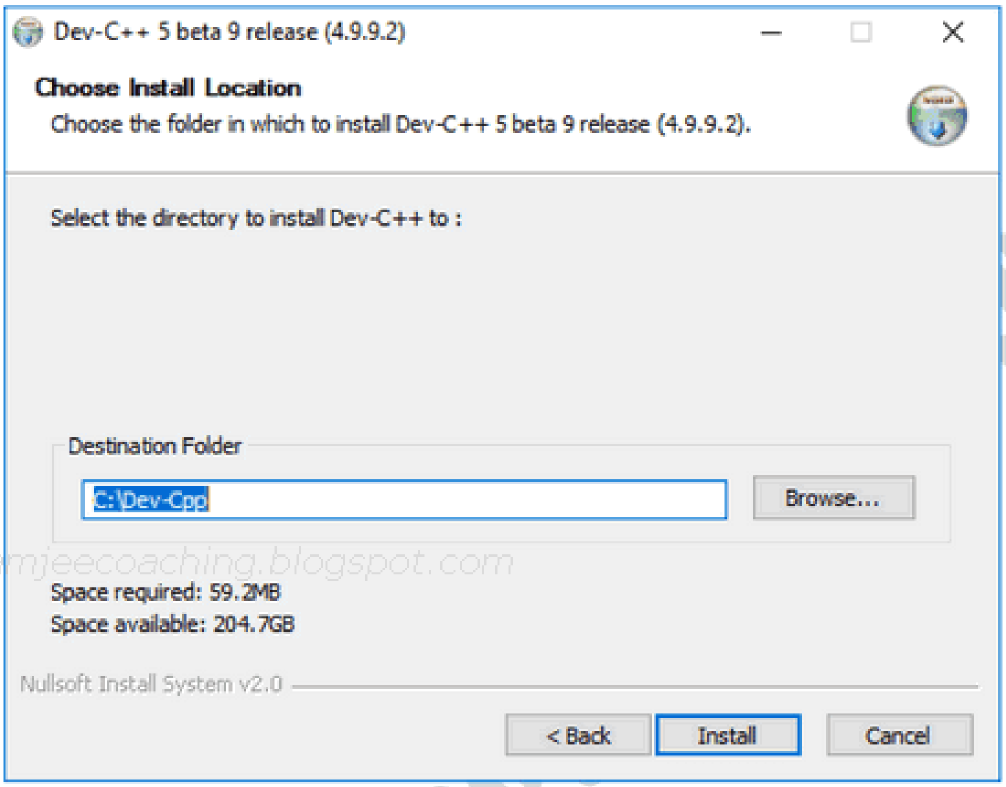
- ✓ Dev-C++ progran
- ✓ Example files
- ✓ Help files
- ✓ Icon files
- ✓ Mingw compiler sy
- ✓ Language files
- ✓ Associate C and C
- ✓ Create shortcuts i

Space required: 59.2MB

Description
Hover your mouse over a component to see its description.

STEP 4

Select the installation folder where Dev-C++ files and libraries will be installed. Click on "Install" to start installation.



Dev-C++ 5 beta 9 release (4.9.9.2)

Choose Install Location

Choose the folder in which to install Dev-C++ 5 beta 9 release (4.9.9.2).

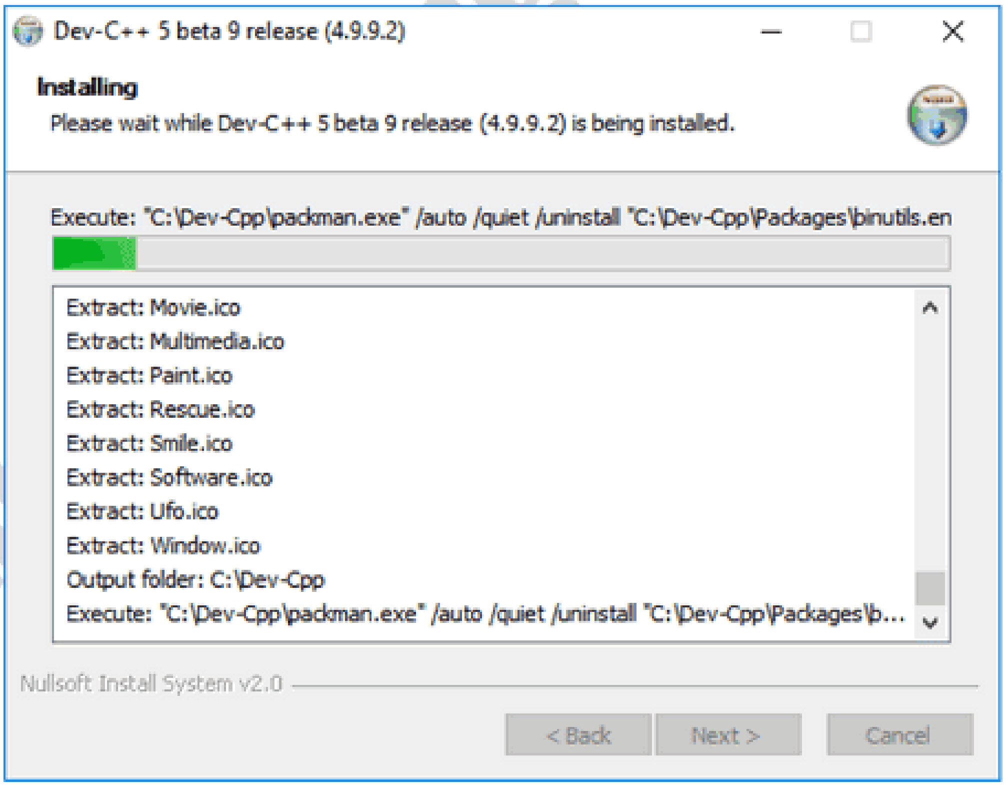
Select the directory to install Dev-C++ to :

Destination Folder

C:\Dev-Cpp

Space required: 59.2MB

Space available: 204.7GB



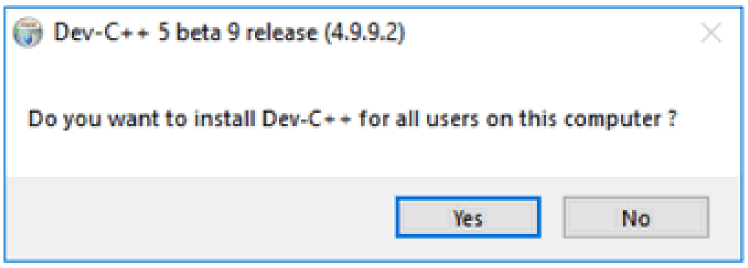
Dev-C++ 5 beta 9 release (4.9.9.2)

Installing

Please wait while Dev-C++ 5 beta 9 release (4.9.9.2) is being installed.

```
Execute: "C:\Dev-Cpp\packman.exe" /auto /quiet /uninstall "C:\Dev-Cpp\Packages\binutils.en
```

```
Extract: Movie.ico
Extract: Multimedia.ico
Extract: Paint.ico
Extract: Rescue.ico
Extract: Smile.ico
Extract: Software.ico
Extract: Ufo.ico
Extract: Window.ico
Output folder: C:\Dev-Cpp
Execute: "C:\Dev-Cpp\packman.exe" /auto /quiet /uninstall "C:\Dev-Cpp\Packages\b...
```

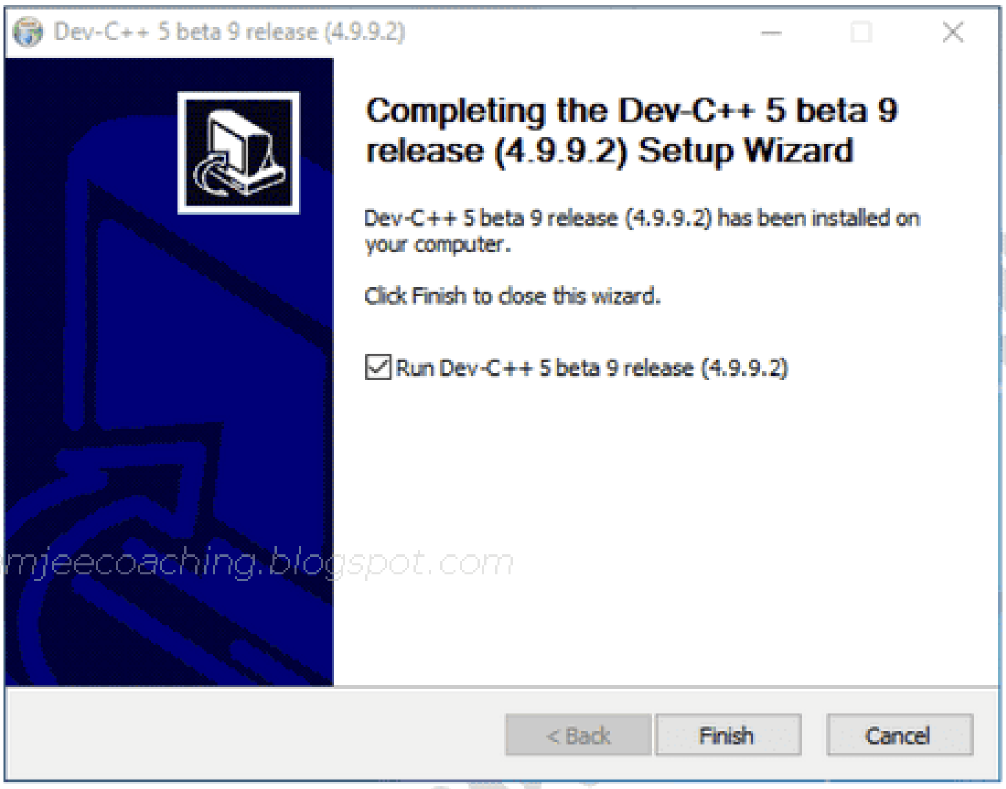


Dev-C++ 5 beta 9 release (4.9.9.2)

Do you want to install Dev-C++ for all users on this computer ?

STEP 5

After completing process, it will show a "Finish" dialog box. Click "Finish" burton. This will automatically start Dev-C++ after installation completes.



Dev-C++ 5 beta 9 release (4.9.9.2)

Completing the Dev-C++ 5 beta 9 release (4.9.9.2) Setup Wizard

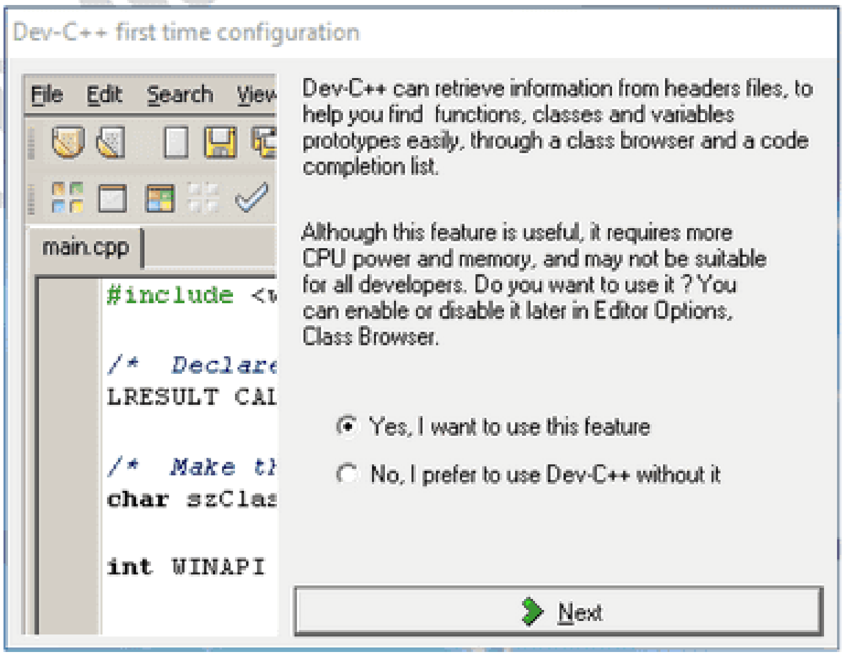
Dev-C++ 5 beta 9 release (4.9.9.2) has been installed on your computer.

Click Finish to close this wizard.

☑ Run Dev-C++ 5 beta 9 release (4.9.9.2)

CONFIGURING DEV-C++

Dev-C++ will require some configuration when it runs first time. Set" English (Original)" as default language and click "Next" to continue. On the "Theme" selection dialog box leave the default setting and click on "Next" and "OK" to continue.



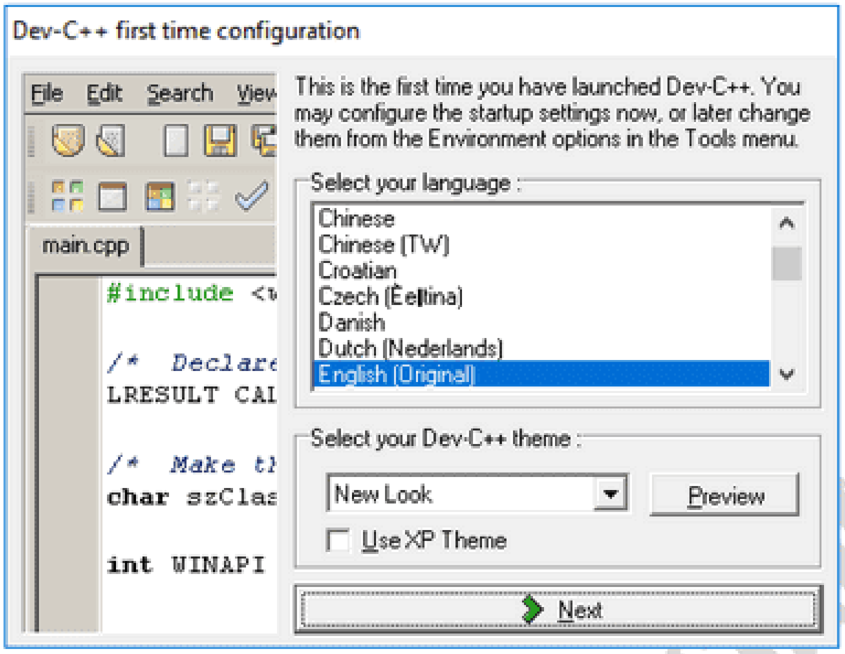
Dev-C++ first time configuration

```
#include <w

/*  Declare
LRESULT CAl

/*  Make th
char szClas

int WINAPI
```

Dev-C++ can retrieve information from headers files, to help you find functions, classes and variables prototypes easily, through a class browser and a code completion list.

Although this feature is useful, it requires more CPU power and memory, and may not be suitable for all developers. Do you want to use it ? You can enable or disable it later in Editor Options, Class Browser.

[✓ Yes, I want to use this feature]
No, I prefer to use Dev-C++ without it


Dev-C++ first time configuration

This is the first time you have launched Dev-C++. You may configure the startup settings now, or later change them from the Environment options in the Tools menu.

Select your language :
Chinese
Chinese (TW)
Croatian
Czech (Èeština)
Danish
Dutch (Nederlands)
English (Original)

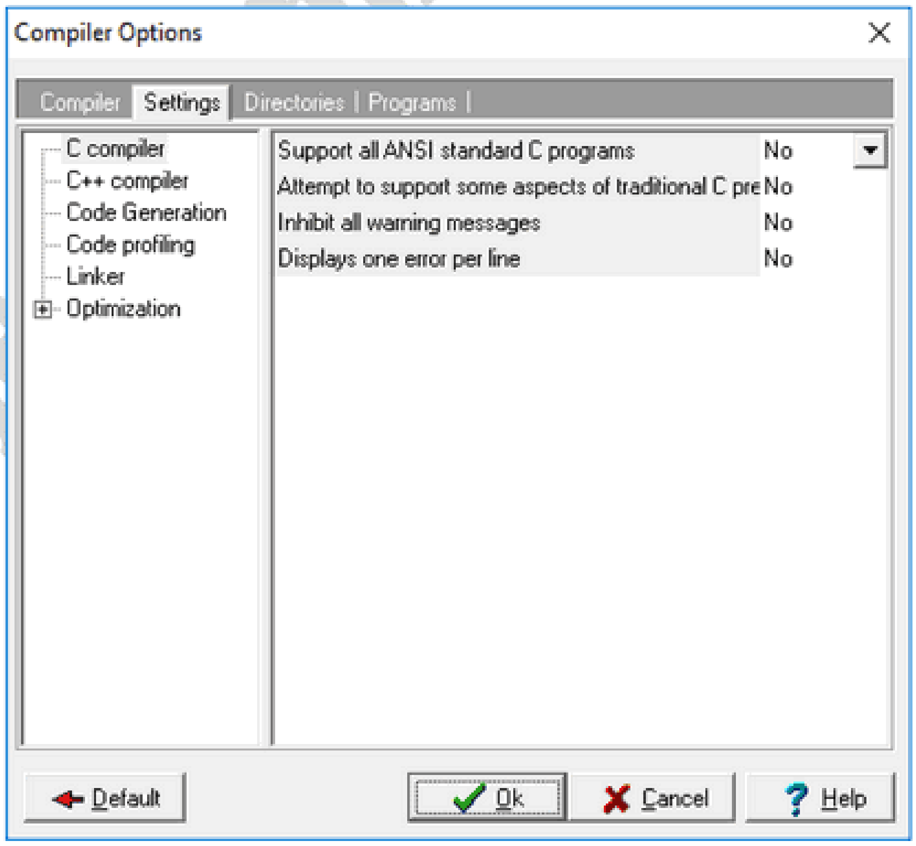
Select your Dev-C++ theme :
New Look
Use XP Theme

```
#include <w

/*  Declare
LRESULT CAL

/*  Make th
char szClas

int WINAPI
```

LINKER SETTING FORDEBUGGING

Linker setting for debugging is required first time to obtain information about problems in source code. The following steps are used to enable this configuration.

1. Click on Tools then Compiler Options and open the Settings tab.
2. Under Settings tab, open Linker tab. In Linker tab change the Generate Debugging Information (-g3) options to Yes.
3. Click on OK to save settings.


Compiler Options

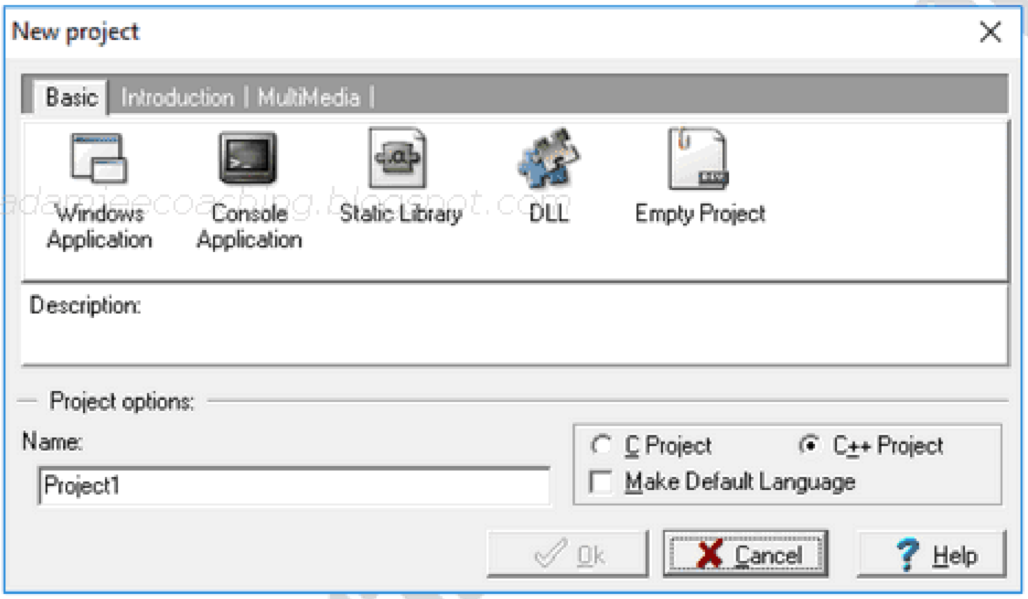
Support all ANSI standard C programs	No
Attempt to support some aspects of traditional C pre	No
Inhibit all warning messages	No
Displays one error per line	No

Q.29: Write down the procedure to develop a program in Dev C++.

Ans. DEVELOPING PROGRAM IN DEV-C++

Development of C++ program requires writing source code and saving those files for compilation. The steps to create a new project in Dev-C++ are:

1. Click on File then click New » Project.
2. In New Project dialog, Select Empty Project then select C++ Project. Also enter Name for project.
3. Click on OK. Dev C++ will ask path to save the new project, enter path and save project.



Name: Project1

C Project [✓ C++ Project]

Make Default Language

ADD NEW FILES TO PROJECT IN DEV-C++

The steps to create a new file are:

1. Click on Project » New File.
2. Click on Yes on the confirm dialog box.
3. To save file, click on File» Save. Enter the path and provide its name. Click on Save to Store the file.

COMPILE & EXECUTE PROJECT IN DEV-C++

The steps to compile and execute project are:

1. To compile, click on Execute » Compile or press F9 key.
2. After successfully compiling the project, run it by clicking on Execute » Run or by pressing F10 key.
3. A console will open and show the output of the program.

Q.30: What do you know about C++ programming language?

Ans. C++ PROGRAMMING LANGUAGE

C++ is a powerful general-purpose programming language. It was created by Bjarne Stroustrup in 1979 at Bell Laboratories. It is used to develop operating systems, browsers, games, and other applications. C++ supports mainly support programming like object oriented. C++ is a flexible language aims to make writing programs easier and more pleasant for the individual programmer.

Q.31: Define reserved words in C++.

Ans. RESERVED WORDS IN C++

Reserved words are keywords that have standard predefined meanings in C++ language. These keywords can only be used for their intended purpose; they cannot be used as programmer defined identifiers.

The following list the keywords or reserved words of the C++ language:

asm	else	new	this
auto	enum	Operator	Throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	Using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	and
or	not	requires	nullptr

Q.32: Define data types in C++.

Ans. C++ DATA TYPES

Data values passed in a program may be of different types. Each of these data types are represented differently within the computer's memory and have different memory requirements. These data types can be augmented by the use of data type qualifiers / modifiers.

The data types supported in C ++ are described below:

DATA TYPE	KEYWORD	SIZE	RANGE
Boolean	bool	1 Bytes	0 (false), 1 (true)
Character	char	1 Bytes	-127 to 127 OR 0 to 255
Integer	int	4 Bytes	-2.147483648 to 2147483647
Floating Point	float	4 Bytes	1.5×10^{-45} to 3.4×10^{38}
Double Floating Point	double	8 Bytes	5.0×10^{-345} to 1.7×10^{308}

Q.33: What is constant? Also define its types.

Ans. CONSTANT

A constant is an identifier whose value remains unchanged throughout the program. Constants are used in two ways, they are:

1. Literal Constants
2. Defined Constants

Q.34: Define literal constant.

Ans. LITERAL CONSTANT

Literal constants are data used for representing fixed values. They can be used directly in the code.

Example: 1,2.5, 'c', "good" etc.

Q.35: Define symbolic constant.

Ans. DEFINED OR SYMBOLIC CONSTANT

In C++, we can create symbolic constant whose value remains unchanged but used as a variable. A symbolic constant can be created using the #define preprocessor directive or const keyword.

Example:

```
const int LIGHT_SPEED = 299792458;
#define LIGHT SPEED 299792458
```

Q.36: . Define variable and rules for naming variables.

Ans. VARIABLE

A variable is nothing but a name given to a storage area that our programs can manipulate. Its value can change during program execution. Each variable in C++ has a specific data type, which determines the size and layout of the variable's memory.

RULES FOR NAMING VARIABLE

- A variable name contains alphabets, numbers, and the underscore.
- A variable name must start with a letter or an underscore.
- Variable names are case sensitive. (Sum and sum are different)
- A variable name cannot be a keyword.
- A variable name cannot be longer than 32 characters.

Q.37: Define Declaring (Creating) and Initializing Variables.

Ans. DECLARATION (CREATING) VARIABLES

Variable declaration is a process in which we create storage space for variable in memory. A variable declaration consists of data type and name of the variable written as follow:

```
data_type variable_name;
int sum;
```

INITIALIZATION

Assign initial value to a variable is known as variable initialization. It can be initialized during declaration or separately. The equal sign is used to assign value written as follows:

```
data_type variable_name = value;
int sum = 3;
```

Q.38: Define strings in C++.

Ans. STRINGS IN C++

Variables that can store alphanumeric value that consist of multiple characters are called strings. In C++, strings are used by one-dimensional array of characters, which is terminated by a null character \0.

COMPUTER SCIENCE 10TH – DETAILED QUESTION ANSWERS

INPUT / OUTPUT HANDLING IN C++

Chapter # 03

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
ll	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
```

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

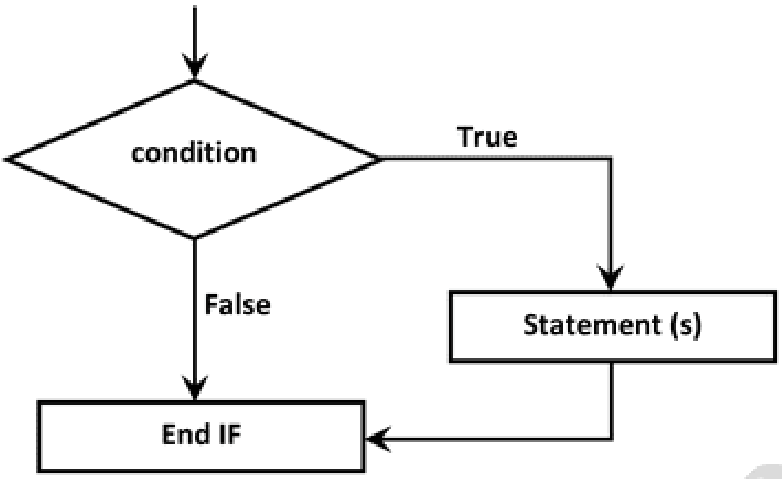
An if statement is a conditional statement that tests a particular condition. Whenever that condition evaluates as true, performs an action, but if it is not true, then the action is skipped.

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

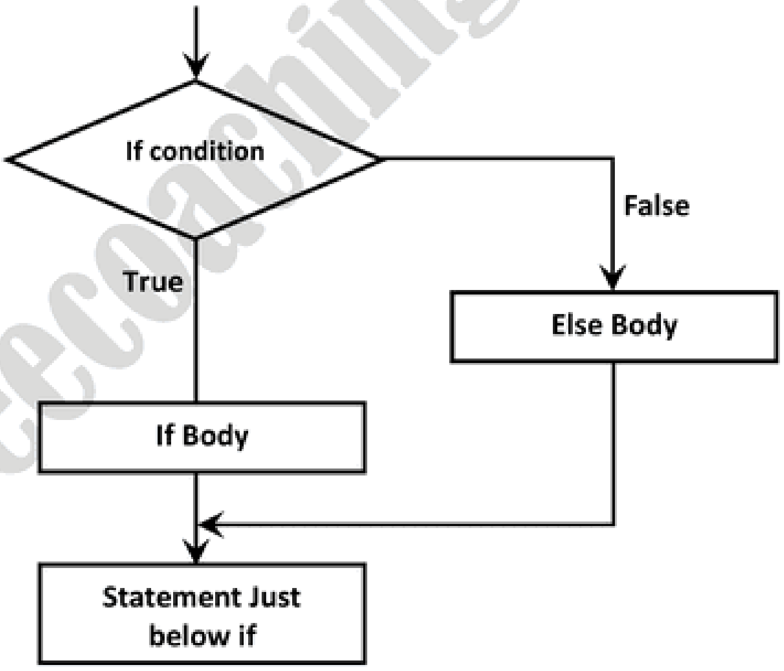
An if-else selection structure performs certain action when the condition is true and some different action when the condition is false.

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

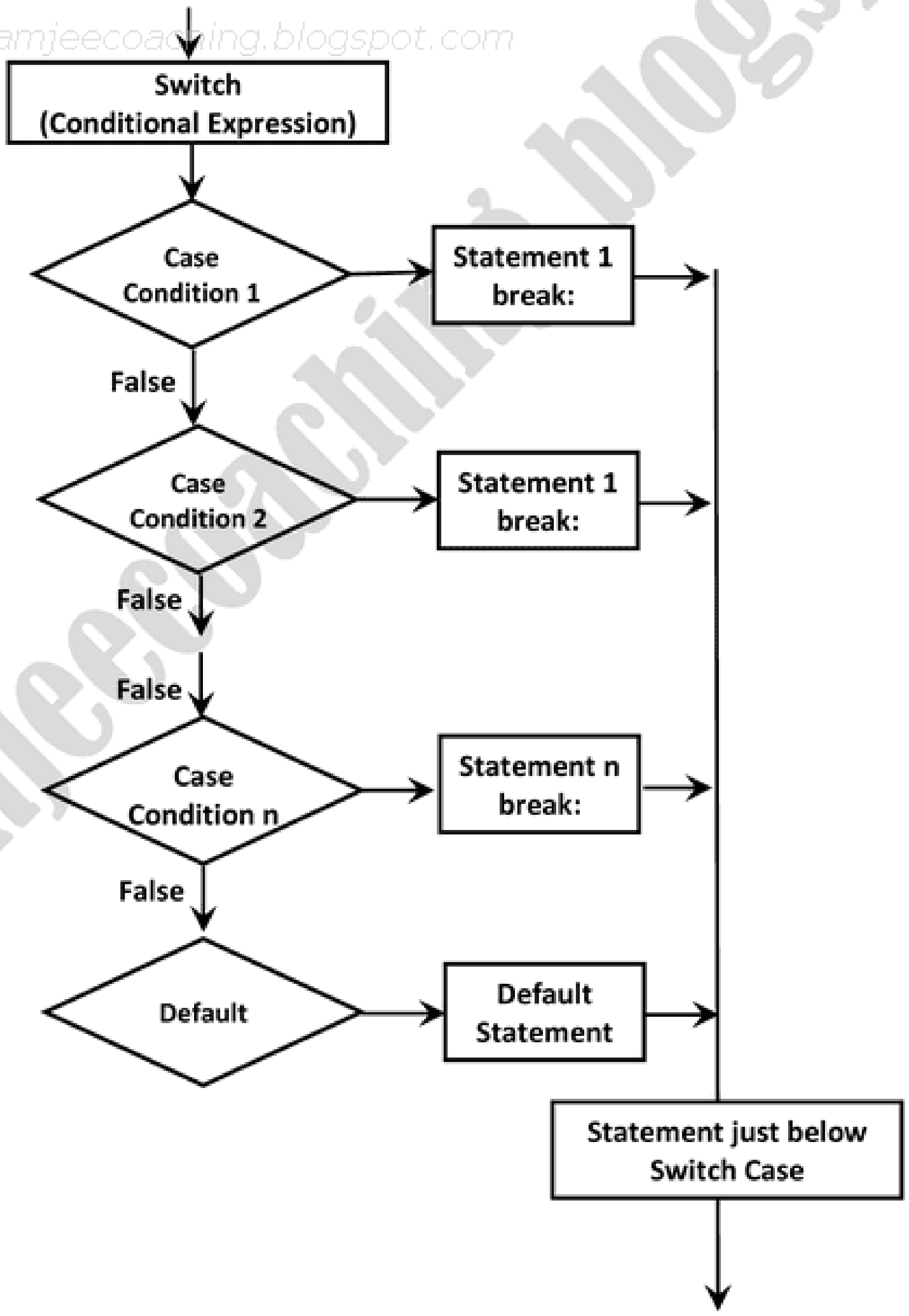
Switch statement is a control statement that allows to select only one choice among the many given choices. The expression in switch evaluates to return an integer or character value, which is then compared to the values present in different cases. It executes that block of codes which matches the case 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·
}
}
```

FLOWDIAGRAM



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.

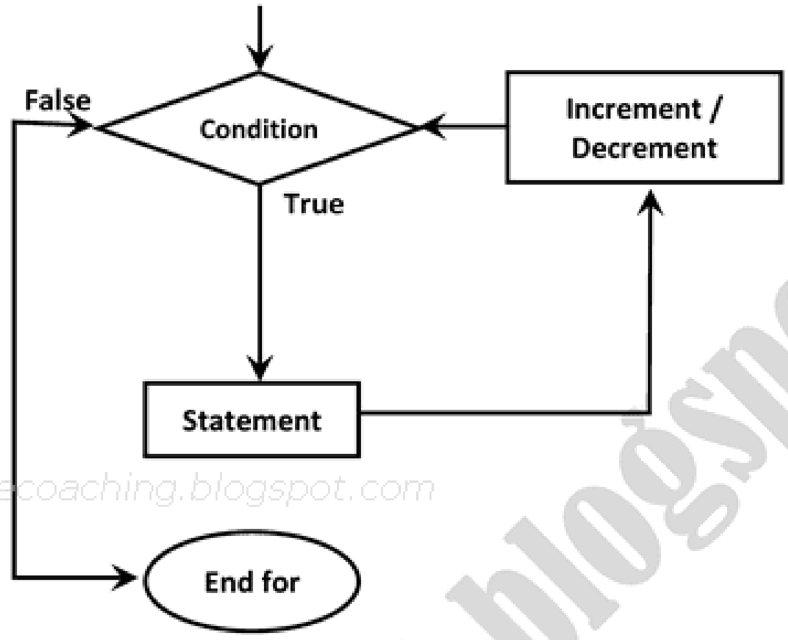
SYNTAX

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

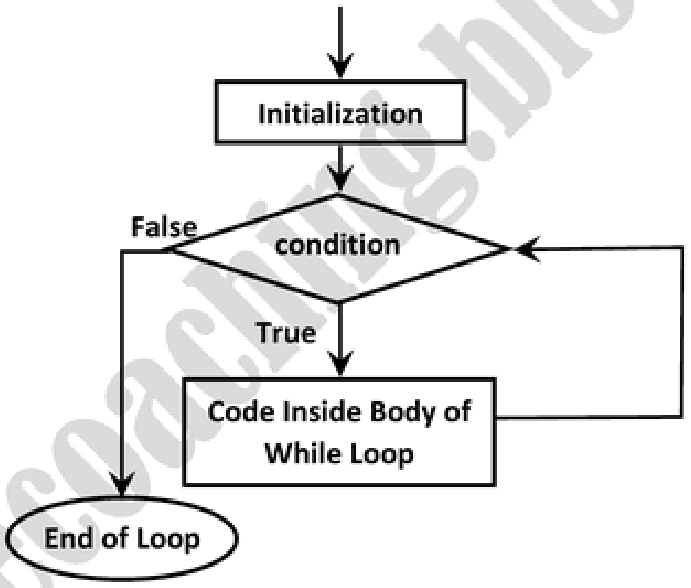
The while loop allows the programmer to specify that an action is to be repeated while some conditions remain true. It is used when the exact number of loop repetitions before the loop execution begins are not known.

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

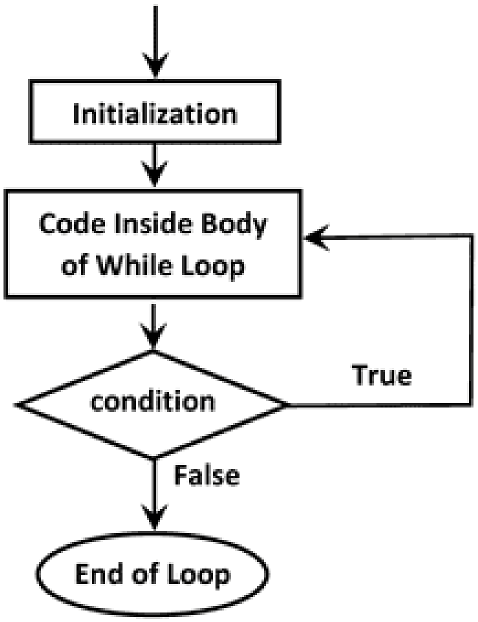
A do-while loop is similar to a while loop, except that a do-while loop is guaranteed to execute at least one time, even if the condition fails for the first time. Unlike for and while loops, which test the loop condition at the start of the loop, the do while loop, the do while loop checks its condition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

COMPUTER SCIENCE 10TH – DETAILED QUESTION ANSWERS

FUNCTIONS

Chapter # 05

Q.1: What do you know about functions?

Ans: INTRODUCTION To FUNCTIONS

A function is a block of code that performs a particular task. It is also called a method or a sub-routine or a procedures etc. There are some situations when we need to write a particular block of codes for more than once in our program. This may lead to bugs and irritation for the programmer. C++ language provides an approach in which you need to declare and define a group of statements once and that can be called and used whenever required. This saves both time and space. Every C++ program has at least one function, which is main( ), and all the programs can define additional functions.

Q.2: Write down the advantages of functions.

Ans: ADVANTAGES OF FUNCTIONS

There are some advantages of using functions.

1. The complexity of the entire program can be divided into simple subtasks and function subprograms can be written for each subtask.
2. Functions help us to avoid unnecessary repetition of codes. It helps in code reusability.
3. Functions can have inputs and outputs and can process information.
4. The functions are short, easier to write.
5. The function are understandable and can be debugged.

Q.3: Define the types of functions.

Ans: TYPES OF FUNCTIONS

There are two types of functions in C++ programming

1. redefined / Built-in functions
2. User-defined Functions

Q.4: What is predefined or built in function?

Ans: PREDEFINED / BUILT IN FUNCTIONS

The built-in functions are standard library functions to handle tasks such as mathematical computations, I/O processing, string handling etc. These functions are defined in the header file and don't have need to declare and define. The definitions of most common functions are found in the cmath and cstdlib libraries.

Q.5: Describe user defined functions.

Ans: USER-DEFINED FUNCTIONS

C++ allows programmers to define functions to do a task relevant to their programs. Such functions created by the user are called user-defined functions. These functions need declaration and definition. A user can create as many user-defined functions as needed.

Q.6: Write down the major elements of user-defined functions

Ans. The user-defined function consists of two parts:

1. Function declaration (Function prototype)
2. Function definition

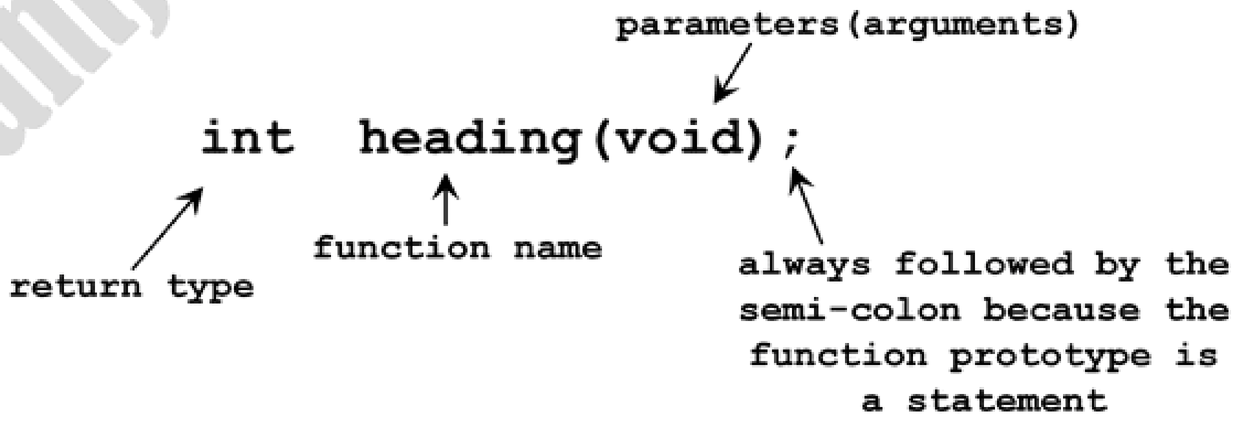
Q.7: Define function declaration.

Ans: FUNCTION DECLARATION / PROTOTYPE

The declaration of a function is called its prototype. Using function in programs requires that we have to declare the function first. It is declared before the main( ) function.

The general structure of the function prototype is:

```
return_datatype function_name(arguments)
```


parameters(arguments)
int heading(void);
return type
function name
always followed by the semi-colon because the function prototype is a statement

It has four main components. These are:

1. Return data type
2. Name of the function
3. Arguments (parameters)
4. Statement terminator

RETURN DATA TYPE

The return data type of the function is the type of return value. If function does not return a value, the type is defined as void.

FUNCTION NAME

The function name is any identifier followed by parenthesis without any spaces in between.

ARGUMENTS /PARAMETERS

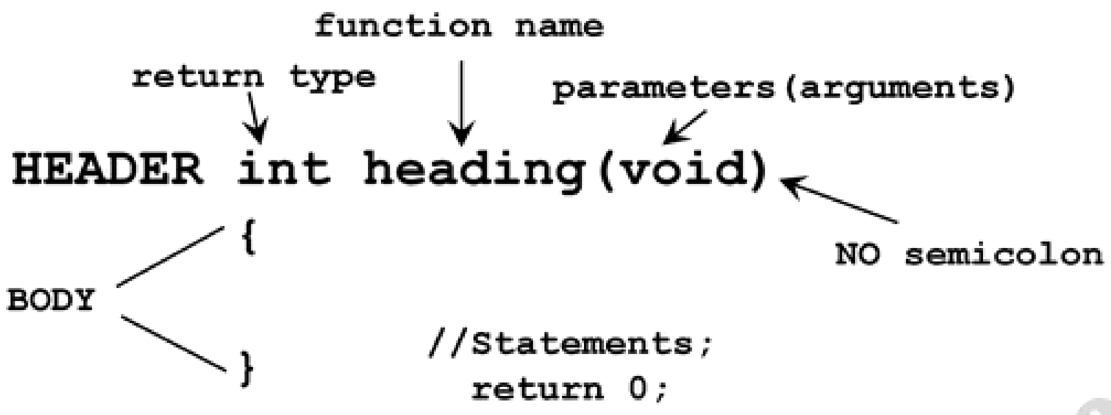
The arguments or parameters come inside the parenthesis, preceded by their types and separated by commas. If the function does not use any arguments, the word void is used inside the parentheses.

STATEMENITTERNUNATOR

Semicolon (;) is used as statement terminator at the end of function declaration or prototype.

FUNCTION DEFINITION

A function definition is the function itself. Function definition consists of a function header, a function body and code block. The definition begins with a header which is exactly same as the function prototype except it must not be terminated with semicolon (;). It has three main components; return type of the function, name of the function, arguments I parameters. Body of the function includes statements in braces and defines before or after main function.



Q.8: Define function calling.

Ans: FUNCTION CALL

A user defined function is called from the main program simply by using its name, including the parentheses which follow the name. The parentheses are necessary so that compiler knows you are referring to a function.

GENERAL SYNTAX

function_name( );

Q.9: Define function arguments.

Ans. FUNCTION PASSING ARGUMENTS ORPARAMETERS

Sending data to a function is called passing arguments. It is basically sending variables, constants, or expression whose value are needed by the function. Actual values that are passing to function as argument with function call statement are known as actual arguments. These values are received in variables of the header of the function definition. These receiving variable or arguments are called formal arguments.

Q.10: Define function return value.

Ans: RETURNING VALUE FROM FUNCTIONS

In C++, when a function completes its execution, it can return a value to the calling function using return keyword. Return type must be defined with the function header in declaration and definition.

Q.11: Write down the differences between function definition and function call.

Ans: DIFFERENCES BETWEEN FUNCTION DEFINITION AND FUNCTION CALL

FUNCTION DEFINITION	FUNCTION CALL
The function definition is the part of function where function is actually defined.	Invoke the code of the function is called function call.
A user defined function may define before or after the main function.	A user defined function is called from the main program simply by using its name.

Syntax: data_type function_name (arguments)
{
statements;
}
	Syntax:
	variable- name function- name (arguments);

Example:
int_sum(int a, int b)
{
int c;
c = a + b;
return(c);
}
	Example:
	z = sum(x, y)

Q.12: Define functions types based on arguments and return value.

Ans. DIFFERENT WAYS TO USE USER-DEFINED FUNCTION BASED ON ARGUMENTS AND RETURN TYPE

Function can be used in four variations in C++ based on arguments and return value from the functions.

- **No return value and no passing arguments**

  void function_name(void);

Return value but no passing arguments

  int/float/char function_name(void);

- **No return value but passing arguments**

  void function_name(int, float, char);

- **Return value and passing arguments**

  int/float/char function_name(int, float, char);

Q.13: Write a program using predefined functions.

Ans: EXAMPLE PREDEFINED FUNCTIONS

```
#include<iostream>
#include<cmath.h>
using namespace std;
int main( )
{
  int sq;
  cout << "Enter an integer to find square root: ";
  cin >> sq;
  cout << " \n The Square root of a given integer is: " << sqrt(sq);
  return 0;
}
```

Q.14: Write a program using user-defined functions.

Ans: EXAMPLE USER-DEFINED FUNCTIONS

```
#include<iostream>
using namespace std;
// function prototype
int add(int a, int b);
int main( )
{
  int x, y, sum;
  cout << "Enter 1st number: ";
  cin >> x;
  cout << "Enter 2nd number: ";
  cin >> y;
  sum = add(x, y);
  cout << "The sum of two numbers is = " << sum << "\n";
  return 0;
}
// function definition
```

```
int add(int a, int b)
{
  int c;
  c = a + b;
  return (c);
}
```

Q.15: Write down the differences between predefined function and user-defined function.

Ans: DIFFERENCES BETWEEN PREDEFINED AND USERDEFINED FUNCTIONS

PREDEFINED FUNCTION	USER-DEFINED FUNCTION
It is also called built in functions or library functions.	These are called end-user functions created by programmer.
It cannot be edited or modified	It can be edited or modified by programmer.
Function definition is not required because definition is a part of compiler.	Function declaration and definition are needed in user-defined function.
Example : pow( ), sqrt( ), getche( ), etc.	**Example:** add( ), int sum( ); float avg(float a, float b), etc.

Q.16: Define variables in C++.

Ans: VARIABLES IN USER-DEFINED FUNCTIONS

In C++ structure programming, there are two types of variables used:

1. Local variable
2. Global variable

Q.17: What is local variable? Also write its example.

Ans: LOCAL VARIABLES

Local variable is defined as a type of variable declared within programming block or functions. It can only be used inside the function or code block in which it is declared. The local variable exists until the block of the function is under execution. After that, it will be destroyed automatically.

LOCAL VARIABLES EXAMPLE

```
#include<iostream>
using namespace std;
int main( )
{
  int b;                  // local variable
  cout << "Enter an integer: ";
  cin >> b;
  cout << " \n The given integer is: " << b;
  return 0;
}
```

Q.18: What is global variable? Also write its example.

Ans: GLOBAL VARIABLES

A global variable in the program is a variable defined outside the subroutine or function. It has a global scope means it holds its value throughout the lifetime of the program. Hence, it can be accessed throughout the program by any function defined within the program.

GLOBAL VARIABLES EXAMPLE

```
#include<iostream>
using namespace std;
int add(int a);
int b = 10;             // Global Variable
int main( )
{
  int x, sum;           // Local Variables
  cout << "Enter 1st number: ";
  cin >>  x;
  sum = add(x);
  cout << "The sum of two numbers is = " << sum << "\n";
  return 0;
  }
  // function definition
```

```
int add(int a)
{
int c;                    // Local Variable
c = a + b;
return (c);
}
```