1. What is program? What do you mean by problem analysis?

2. What is algorithm and flowchart.

3. Create an algorithm and draw the corresponding flow chart to determine simple interest

given the principal, time and rate.

4. Create an algorithm and draw the corresponding flow chart to determine whether a given n

digit number is palindrome or not.

5. Explain basic structure of a c program with suitable example.

6. Why C program needs to be compiled? Explain compilation of a C program in detail.

7. What is identifier? What are rules for valid identifier in C?

8. Explain different primary data types with their respective ranges.

9. How to define variables in C. How Variables are different with Constants? Support your

answers with examples of each.

10. What is keyword in C?

1. **What is program? What do you mean by problem analysis?**

=) A **program** is a set of instructions written in a programming language that a computer can execute to perform specific tasks. These tasks can range from simple operations like calculations to complex processes like data analysis or controlling hardware.

- **Problem analysis** is the process of understanding a problem in detail to find the most effective way to solve it through programming. It involves breaking down the problem into smaller, manageable parts, identifying the inputs, processing requirements, and expected outputs.

**Steps in Problem Analysis:**

1. **Understand the Problem**: Identify what the problem is and what needs to be solved.
2. **Define Inputs:** Determine the data or information required to solve the problem.
3. **Identify Outputs:** Define what results are expected after solving the problem.
4. **Establish Logic:** Plan the steps or algorithm to transform inputs into outputs.
5. **Consider Constraints:** Account for limitations like time, resources, or specific conditions.
6. **What is algorithm and flowchart.**

**=)** An **algorithm** is a step-by-step procedure or a set of rules designed to solve a problem or perform a specific task. It is a logical and structured approach to achieving a goal, written in a way that can be understood and implemented.

**Example of an Algorithm:**

**Problem**: Find the largest of three numbers (a, b, c).

**Algorithm:**

1. Start.
2. Input three numbers: a, b, c.
3. If (a > b) and (a > c), print "a is largest".
4. Else if (b > c), print "b is largest".
5. Otherwise, print "c is largest".
6. Stop.

- A **flowchart** is a graphical representation of an algorithm or a process. It uses symbols like arrows, rectangles, diamonds, and ovals to illustrate the sequence of steps in solving a problem.

**Common Flowchart Symbols:**

1. **Oval**: Represents the start or end of a process.
2. **Rectangle**: Represents a process or instruction.
3. **Diamond**: Represents a decision-making step.
4. Arrow: Shows the flow of the process.
5. **Create an algorithm and draw the corresponding flow chart to determine simple interest given the principal, time and rate.**

**=)**

**Algorithm**

1. Start
2. Read Principal Amount, Rate and Time
3. Calculate Interest using formula SI= ((amount\*rate\*time)/100)
4. Print Simple Interest
5. Stop

**Flowchart**

1. **Create an algorithm and draw the corresponding flow chart to determine whether a given n digit number is palindrome or not.**

**Algorithm**

Step 1: Start

Step 2: Read the input number from the user

Step 3: Declare and initialize the variable reverse

and assign input to a temp variable tempNum=num

Step 4: Start the while loop until num !=0 becomes false

rem = num % 10

reverse\*= 10 + rem

num = num / 10

Step 5 : Check if reverse == tempNum

Step 6: If it’s true then the number is a palindrome

Step 7: If not, the number is NOT a palindrome

Step 8: Stop

1. **Explain basic structure of a c program with suitable example.**

Top of Form

Bottom of Form

**Basic Structure of C Program**

Whenever we create a program in C language, we can divide that program into six different sections. This section is as follows:

1. **Documentation (Documentation Section)**

=) Programmers write comments in the Documentation section to describe the program. The

compiler ignores the comments and does not print them on the screen. Comments are used

only to describe that program.

1. **Preprocessor Statements (Link Section)**

=) Within the Link Section, we declare all the Header Files that are used in our program. From the link

section, we instruct the compiler to link those header files from the system libraries, which we have

declared in the link section in our program.

Example -:#include <stdio.h>, #include <conio.h>, #include <string.h>, #include <math.h>

In addition to all these Header Files in the Link Section, there are a lot of Header Files which

we can link in our program if needed.

1. **Definition Section**

=) The definition of Symbolic Constant is defined in this section, so this section is called

Definition Section. Macros are used in this section.

Example -: #define PI 3.14

1. **Global Declarations Section**

=) Within the Global Declarations Section section, we declare such variables which we can use

anywhere in our program, and that variable is called Global Variables, we can use these

variables in any function.

In the Global Declaration section, we also declare functions that we want to use anywhere in

our program, and such functions are called Global Function.

Example -:

int area (int x); //global function

int n; // global Variable

1. **Main functions section**

=) Whenever we create a program in C language, there is one main() function in that program.

The main () function starts with curly brackets and also ends with curly brackets. In the main

() function, we write our statements.

The code we write inside the main() function consists of two parts, one Declaration Part and

the other Execution Part. In the Declaration Part, we declare the variables that we have to

use in the Execution Part, let’s understand this with an example.

Example -:

int main (void)

{

int n = 15; // Declaration Part

printf ("n = %d", n); // Execution Part

return (0);

}

1. **User-Defined Functions or Sub Program Section**

**=)** User-defined functions are called in this section of the program. The control of the program is shifted to the called function whenever they are called from the main or outside the main() function. These are specified as per the requirements of the programmer.

**Example:-**

int sum(int x, int y)

{

return x+y;

}

1. **Why C program needs to be compiled? Explain compilation of a C program in detail.**

=) C programs need to be compiled because they are written in a high-level language that is understandable by humans but not directly understandable by the computer's hardware. A compiler translates the C source code into machine code (binary instructions) that can be executed by the computer's processor. This process ensures the program can interact with the hardware effectively and perform as intended.

The compilation process for a C program is a series of steps that converts human-readable C source code into machine-executable binary code. The process involves four main phases:

1. **Preprocessing (**program.i**)**

=) The preprocessor expands macros, includes header files, and handles conditional compilation.

1. **Compilation (**program.s**)**

=) The compiler translates the preprocessed code into assembly language or machine code, producing object files.

1. **Assembly (**program.o**)**

=) The assembler converts the object files to machine code specific to the target architecture.

1. **Linking** (program.exe)

=) The linker combines object files and libraries, resolves references, and creates the final executable.

The compilation process only generates output if the source file passes through all the components in the build pipeline without error. Even a minor failure in any of the components can cause a compile or link failure and will generate an error message.

1. **What is identifier? What are rules for valid identifier in C?**

In C, an identifier is a name given to a variable, function, type, array, or other user-defined object. Identifiers must follow certain rules, including:

1. An identifier can include letters (a-z or A-Z), and digits (0-9).
2. An identifier cannot include special characters except the ‘\_’ underscore.
3. Spaces are not allowed while naming an identifier.
4. An identifier can only begin with an underscore or letters.
5. We cannot name identifiers the same as keywords because they are reserved words to perform a specific task. For example, printf, scanf, int, char, struct, etc. If we use a keyword’s name as an identifier the compiler will throw an error.
6. The identifier must be unique in its namespace.
7. C language is case-sensitive so, ‘name’ and ‘NAME’ are different identifiers.
8. **Explain different primary data types with their respective ranges.**

**=)** Primary data types in C are the fundamental types provided by the language for defining variables. They represent numeric values, characters, and floating-point values. Here are the primary data types in C along with their ranges and characteristics:

1. **integer datatype**

The integer datatype in C is used to store the integer numbers (any number including positive, negative and zero without decimal part). Octal values, hexadecimal values, and decimal values can be stored in int data type in C.

Range: -2,147,483,648 to 2,147,483,647

Size: 4 bytes

Format Specifier: %d

1. **Character data type**

Character data type allows its variable to store only a single character. The size of the character is 1 byte. It is the most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.

Range: (-128 to 127) or (0 to 255)

Size: 1 byte

Format Specifier: %c

1. **float data type**

In C programming float data type is used to store floating-point values. Float in C is used to store decimal and exponential values. It is used to store decimal numbers (numbers with floating point values) with single precision.

Range: 1.2E-38 to 3.4E+38

Size: 4 bytes

Format Specifier: %f

1. **Boolean data type**: Stores logical values, either true or false.
2. **How to define variables in C. How Variables are different with Constants? Support your answers with examples of each.**

**=)** type variableName = value; Where type is one of C types (such as int ), and variableName is the name of the variable (such as x or myName). The equal sign is used to assign a value to the variable.

|  |  |
| --- | --- |
| **Variables** | **Constant** |
| **A variable is a name associated with some memory location.** | **A constant is a variable or value that cannot be altered once defined.** |
| **The variables are stored inside a data segment, heap, or stack depending on the environment it is declared in.** | **The constants are generally stored in the text segment as they are read-only** |
| **We can assign value to the variable anytime** | **We can only assign a value to the constant while defining it.** |
| **A variable can only be defined using the standard variable definition syntax.** | **A constant can be defined by using #define or const keyword.** |
| **Example: int var = 25;**  **var = 10;** | **Example: #define pi 3.14**  **const int pi = 3.14;** |
|  |  |
|  |  |

1. **What is keyword in C?**

**=)** A keyword in C is a reserved word with a specific meaning and purpose that is used to write programs in the C programming language