

Unit 1

Introduction

Problem Solving

Problem Solving is a systematic approach to find and implement the solution to a problem.

The computer is the symbol manipulating machine that follows the set of instructions called program.

Program

A set of instructions to solve the problem or the specification of the sequence of computational steps in a particular programming language is called program. The task of developing program is called programming.

Problem Solving Techniques involve the following steps

1. Problem Definition

- To solve a problem, the first step is to identify and define the problem.
- The problem must be stated clearly, accurately and precisely.

2. Problem Analysis

The problem analysis helps in designing and coding for that particular problem.

- Input specifications: The number of inputs and what forms the input are available
- Output specifications: The number of outputs and what forms the output should be displayed.

3. Designing a program

Formulate an algorithm and flowchart to solve the problem.

- **Algorithm:** step by step procedure of solving a problem
- **Flowcharts:** It is the graphical representation of the algorithm.

4. Coding

Writing instructions in a particular programming language to solve a problem. The compiler will convert the program code to the machine language which the computer can understand.

5. Program testing

After writing a program, programmer needs to test the program for completeness, correctness, reliability and maintainability. There are different types of tests:

- Unit testing
- Program Testing
- Verification Testing
- Validation Testing etc.

6. Installation and Maintenance

- Installation of a computer program is the act of making the program ready for execution.
- Maintenance means periodic review of the programs and modifications based on user requirements.

Algorithm

An algorithm is a step by step descriptions of the procedure written in human understandable language for solving given problem.

The characteristics of an algorithm are:

- Algorithm must have finite number of steps.
- An algorithm should be simple.
- An algorithm must take at least one or more input values.
- An algorithm must provide at least one or more output values.

Advantages of Algorithms:

- An algorithms are very easy to understand.
- Algorithm is programming language independent.
- Algorithm makes the problem simple, clear, correct.

E.g.✓ ***Algorithm to find largest number among three numbers:***

step 1: start

step 2: input a,b,c

Step 3: if (a>b) and (a>c) then print "a is greater".

Else if (b>a) and (b>c) then print "b is greater".

Else print "c is greater".

Step 4: stop

✓ ***Algorithm to find sum and average of first n natural numbers:***

1. Start
2. n = input an integers
3. $sum = 1 + 2 + 3 + \dots + n$
4. $avg = sum/n$
5. print sum, avg
6. stop

✓ ***Algorithm to find simple interest:***

1. Start
2. Input p, t, r
3. Calculate $I = p * t * r / 100$
4. Print I
5. END

✓ ***Algorithm to find sum of two numbers:***

1. Start
2. Input two numbers n_1 and n_2
3. Calculate $sum = n_1 + n_2$
4. Print sum
5. END

Flowchart

- A flow chart is a step by step diagrammatic representation of the logic paths to solve a given problem.
- A flowchart is graphical representation of an algorithm.

Advantages :

- The flowchart shows the logic of a problem displayed in pictorial fashion.
- It is useful for debugging and testing of programs.
- Program could be coded efficiently using flowcharts.
- The Flowchart is good means of communication to other users.








Disadvantages:

- It is not useful to represent complex program logic
- For any alterations, the flowcharts have to be redrawn completely.

Rules for writing flowcharts :

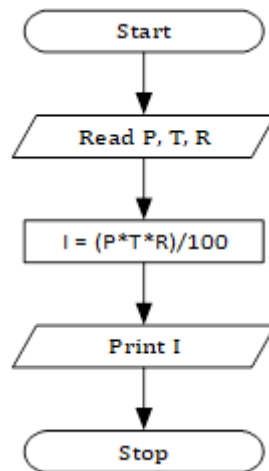
- The flow chart should be clear, neat and easy to follow.
- It should be drawn from top to bottom.
- A flowchart always begins with start symbol and ends with stop symbol.
- Flow lines are used to join the symbols
- Decision box should have one entry point and two exit points.
- For lengthy flowcharts, connectors are used to join them.

Symbols used in flowcharts:

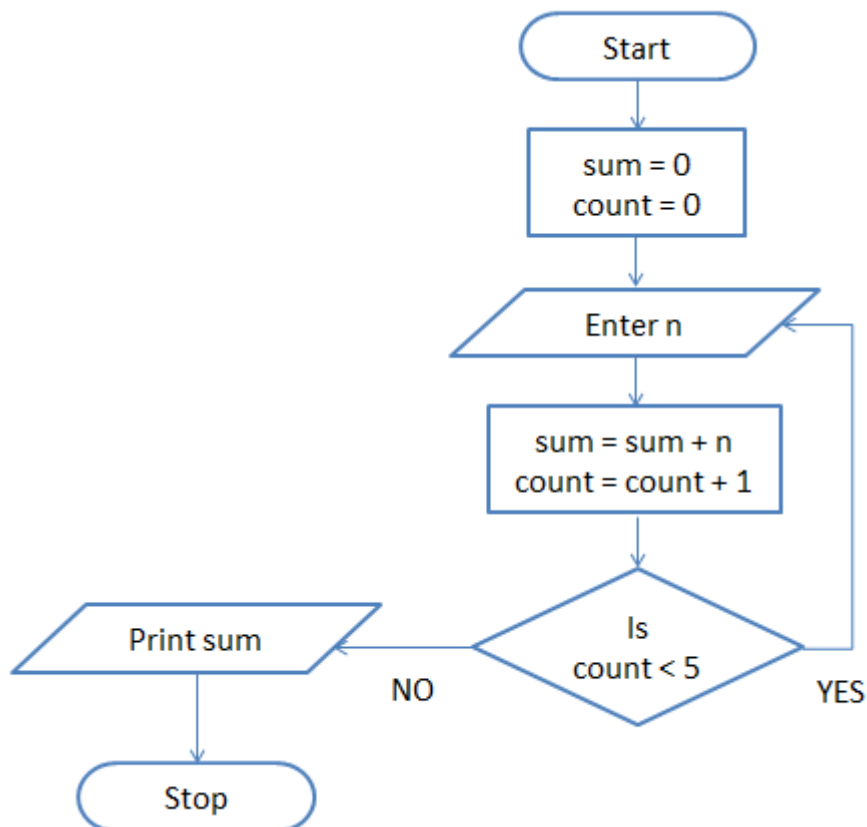
Symbol	Name	Meaning
	Terminal	Used to represent the beginning (START) or the END of a task.
	Input/Output	Used for input (reading) and output (printing) operations. The data to be read or printed are written inside the symbol.
	Processing	Used for arithmetic and data manipulation operations. The instructions are listed inside the symbol.
	Decision	Used to indicate a point at which a decision has to be made, and a branch to one of two or more alternatives is possible. The decision symbol has one entry and two exit paths. The path chosen depends on whether the answer to a question is 'yes' or 'no'.
	Flowlines	Used to indicate the flow of operation.
	Connector	Used to join different flow lines.
	Off-page connector	Used to indicate that the flowchart continues to a second page.

E.g.

✓ *Flowchart to compute simple interest:*



✓ *Flow chart to print the sum of five numbers:*



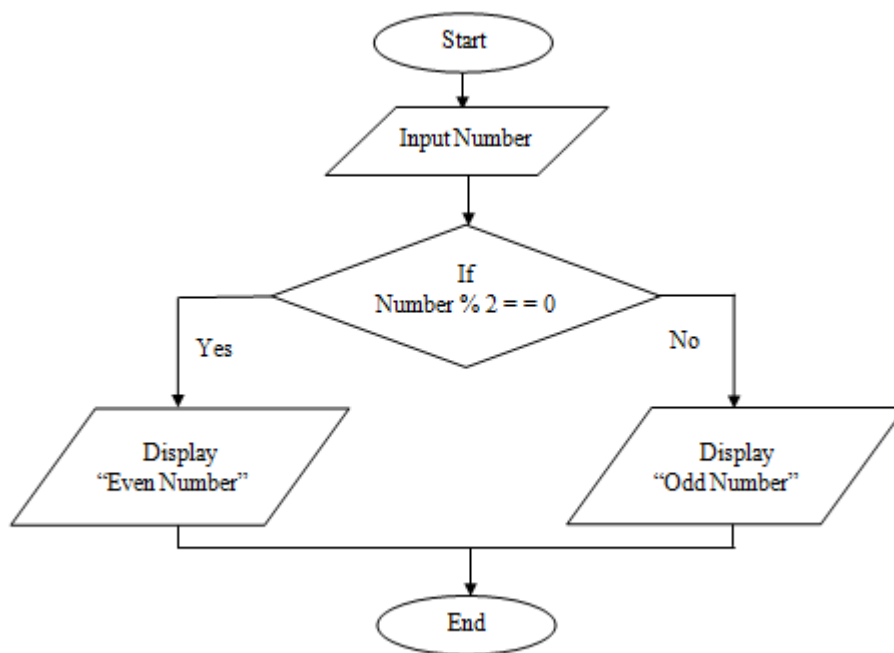
Q. Write an algorithm and draw flowchart to test a number for even or odd.

Solⁿ:

Algorithm:

1. Start
2. Input a number which is to be tested for even or odd.
3. If $number \% 2 == 0$ then print "The number is even".
Else print "The number is odd".
4. End

Flowchart:

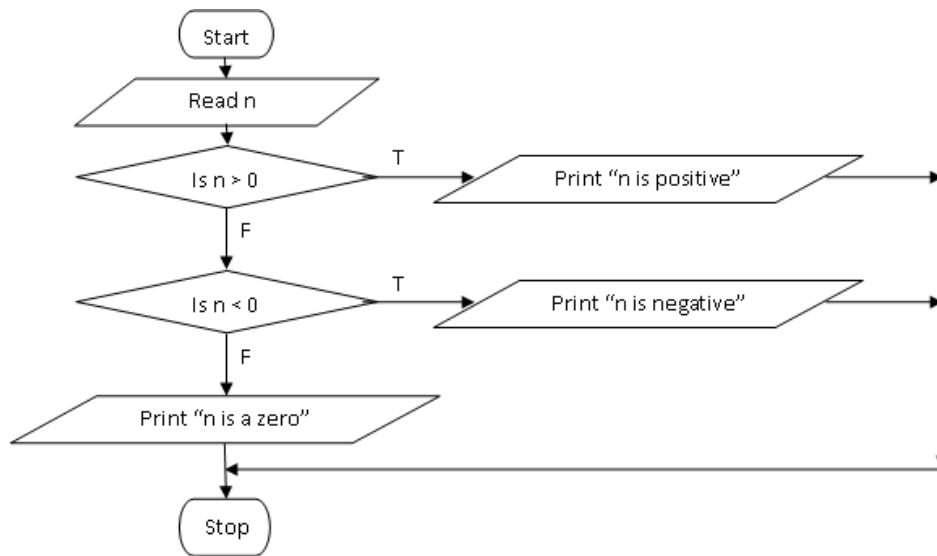


Q. Write an algorithm and flowchart to find out whether a given integer is zero, positive, or negative.

Solⁿ:

Algorithm:

1. Start
2. Print "Enter a number".
3. Read n
4. If $n > 0$ then print "The number is positive"
Else if $n < 0$ print "The number is negative"
Else "The number is zero".
5. Stop

Flowchart:

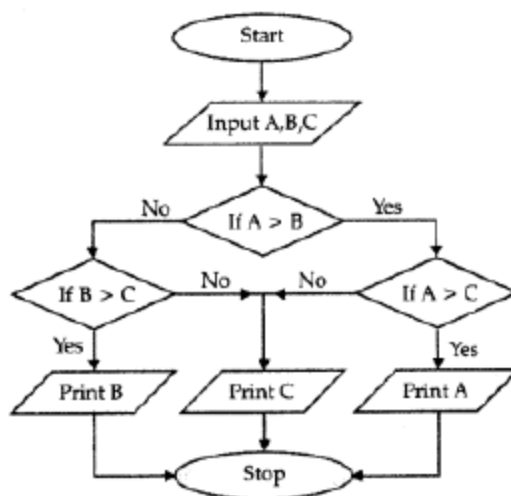
Q. Write an algorithm and flow chart for finding largest of three numbers.

Solⁿ:

Algorithm:

1. Start
2. Input A,B,C
3. If $(A > B)$ and $(A > C)$ then print "A is greater".
Else if $(B > A)$ and $(B > C)$ then print "B is greater".
Else print "C is greater".
4. Stop

Flowchart:



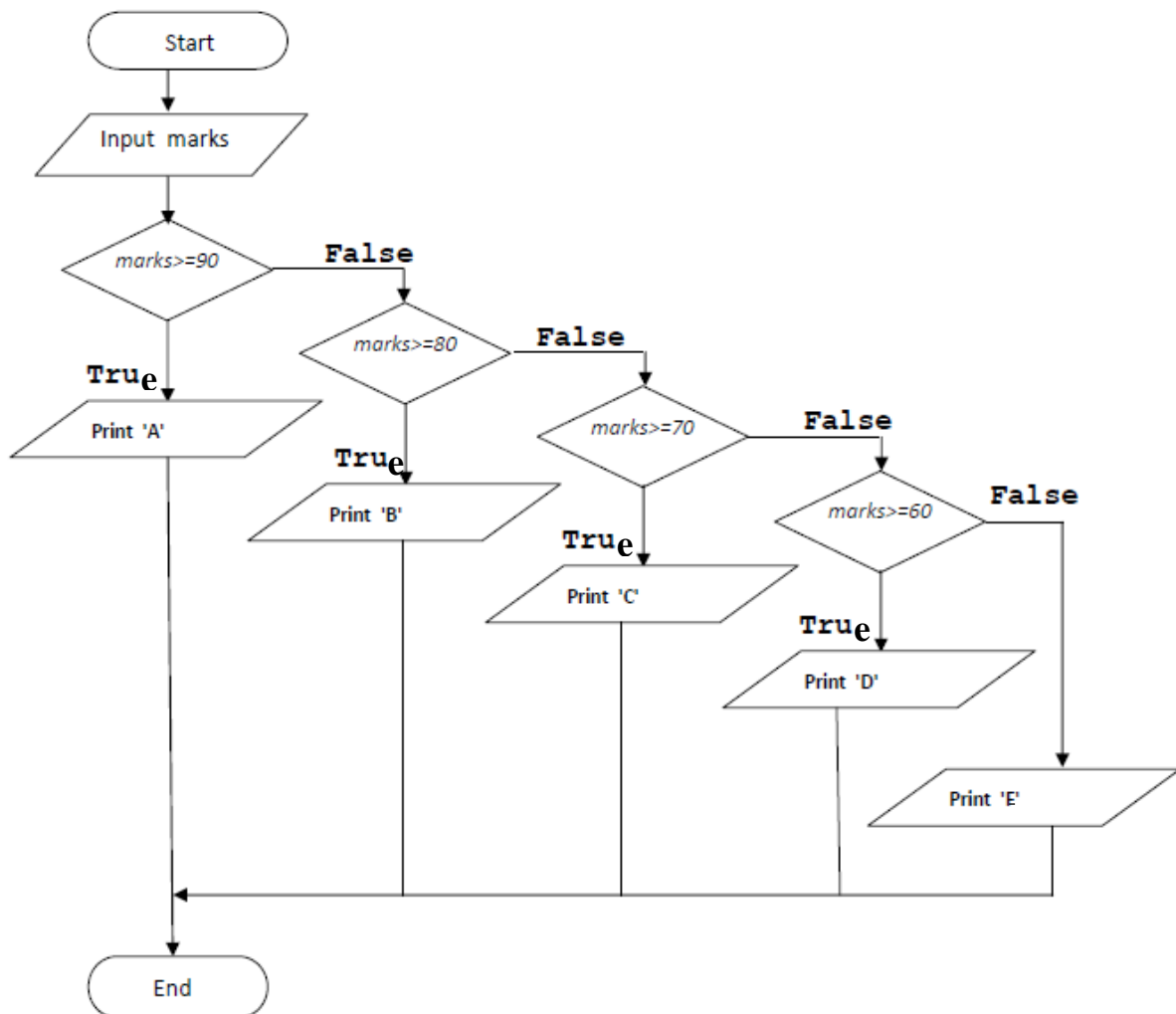
Q. Write the algorithm for obtaining the final grade of student based on mark and draw the flow chart.

Solⁿ:

Algorithm:

1. START
2. Input marks of students
3. If marks ≥ 90
 Print "Grade = A"
 else if marks ≥ 80
 Print "Grade = B"
 else if marks ≥ 70
 Print "Grade = C"
 else if marks ≥ 60
 Print "Grade = D"
 else
 Print "Grade = E"
4. END

Flowchart:



Program Development Cycle

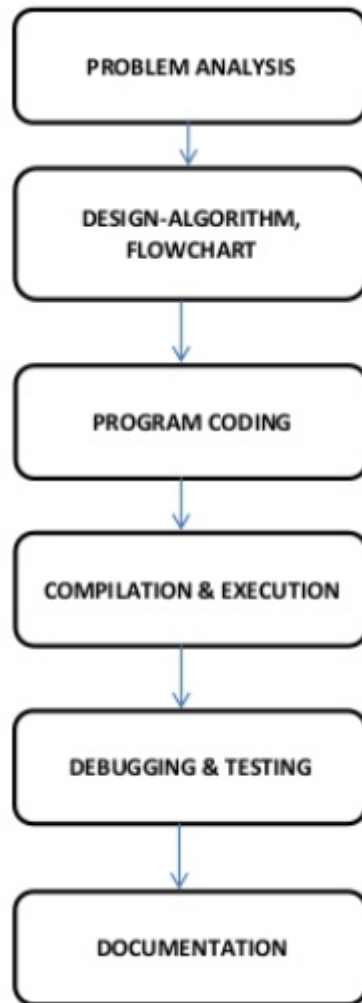


Figure: Problem solving process

Coding

Coding is the translation of an algorithm or flowchart into a suitable computer language like c, c++, java etc. Coding is the real job of programmer. The algorithm to solve a problem which is described by pseudo-code or flow chart is converted into actual programming language code. The code written by programmer by using any programming language like C, C++ etc. is called the source code or source program.

Compilation and Execution

The source code written in any programming language is not directly executed by the computer. It should be translated into to the machine readable format i.e. actual machine language. The process of translation of source code into the target code is called the compilation. Each programming language has its own compiler program that translates the source code into its target code. The converted program in actual machine language is then executed by the computer which is known as program execution.

Debugging and Testing

A written program may have errors, some errors can be detected by the language compilers and some errors cannot be identified by the compiler and occurred during the program run.

Common types of errors are:

Syntax Errors: Identified by compiler at the program compilation time.

Logical Errors: Not identified by the compiler at compile time and identified at the execution time. E.g. misuse of operators

So testing is the process of checking the program for its correct functionality by executing the program with some input data set and observing the output of the program.

Documentation

From the start of the problem solving to the end of the implementation of the program, all the tasks should be documented i.e. kept for future reference. It is also the important part of the problem solving or program development. Documentation may be of two types:

- a. Technical Documentation known as programmer's documentations which includes the problem analysis to implementation details for that program. It is needed for future reference for any modification, update of the program.
- b. User manual is the documentation prepared for the end-user of the program that guides the user how to operate the program.

Introduction to C

C programming,

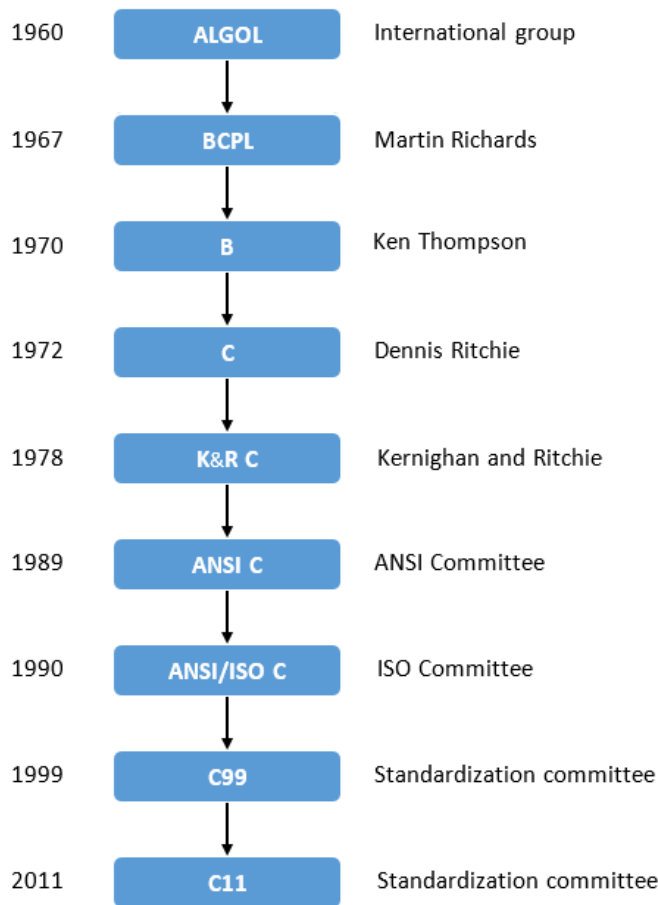
- Was developed by Dennis Ritchie at AT&T Bell Labs, USA in 1972.
- Is a high-level programming language used to develop applications for high-level business and low-level system programs.
- Became popular because of its power, simplicity and ease of use.
- Enables system program writing, using pointers.
- It is reliable, simple and easy to use.

Features of C:

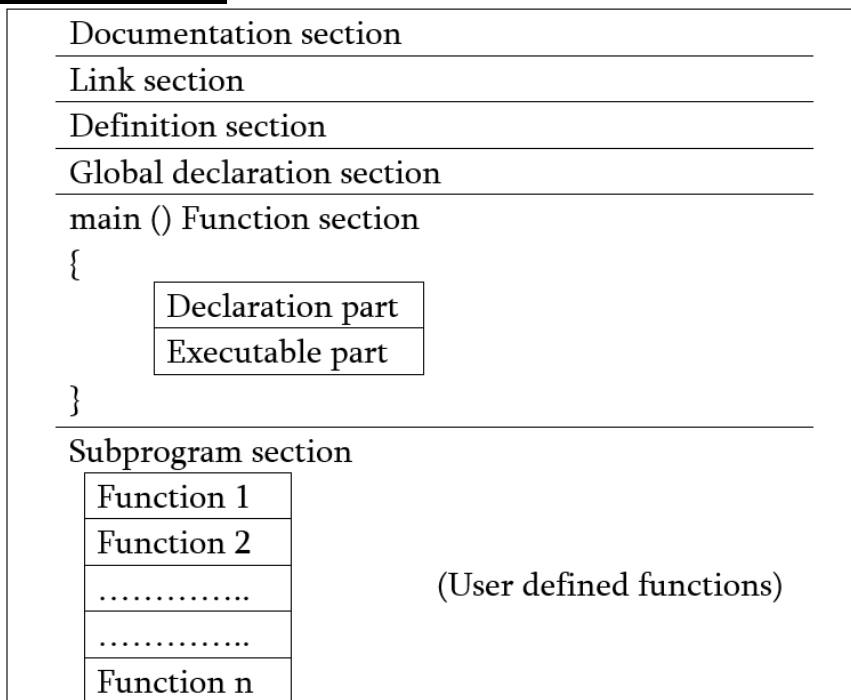
- Robust language, which can be used to write any complex program.
- Has rich set of built-in functions and operators.
- Well-suited for writing both system software and business applications.
- Efficient and faster in execution.
- Highly portable.
- Well-suited for structured programming.
- Dynamic memory allocation.

History of C

C language has evolved from three different structured language ALGOL, BCPL and B language. It uses many concepts from these languages and introduced many new concepts such as data types, struct, pointer.



Structure of C Program



1. Documentation Section

The documentation section is the part of the program where the programmer gives the details associated with the program. He usually gives the name of the program, the details of the author and other details like the time of coding and description. It gives anyone reading the code the overview of the code.

E.g.

```
/*
```

```
    C programming basics & structure of C programs
```

```
    Author: Jayanta Poudel
```

```
    Date : 28/09/2019
```

```
    Update:29/09/2020
```

```
*/
```

2. Link Section

The link section provides instructions to the compiler to link functions from the system library such as using the include directive.

E.g. #include<stdio.h>

stdio.h is an input output header file which contains the input/output and file handling functions which are used in C programs. The required header file should be included using #include preprocessor directive.

3. Definition Section

The definition section defines all symbolic constants such using the #define directive.

E.g. #define PI 3.14

4. Global Declaration Section

There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user-defined functions.

E.g. int a=10;

```
    int sum(int, int);
```

5. Main() function Section

Every C-program needs to have the main function. Each main function contains 2 parts.

Declaration part: The declaration part declares all the variables used in the executable part.

Executable part: There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. The program execution begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program.

E.g.

```
int main( )
{
    int a=10;
    printf("%d", a);
    return 0;
}
```

6. Subprogram Section

User can define their own functions in this section which perform particular task as per the user requirement. So user create this according their needs.

E.g.

```
int sum(int a, int b)
{
    return a+b;
}
```

Simple C program

```
/* C-program to display "Hello, World" */
#include <stdio.h>
int main()
{
    printf("Hello, World!");
    return 0;
}
```

Output:

```
Hello, World!
```

printf():

printf is an output function which has been defined in stdio.h file. Whatever is put inside the function printf between two double quotes is printed on the screen.

Program to add two integers

```
#include <stdio.h>
int main()
{
    int number1, number2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    // calculating sum
    sum = number1 + number2;

    printf("%d + %d = %d", number1, number2, sum);
    return 0;
}
```

Output:

```
Enter two integers: 12
11
12 + 11 = 23
```

The scanf() function:

scanf() is an input function defined in stdio.h header file. scanf accepts the input from keyboard.

The & is an address operator in scanf function . **&variablename** specifies the memory address for the variable and the value entered from keyboard is stored in that specified location.

For more notes visit:

<https://collegenote.pythonanywhere.com/>