The Problem- Solving Aspect

A programmer has to go through the following stages to develop a computer program:

1. Defining and Analyzing The Problem
2. Designing The Algorithm
3. Coding or Writing The Program
4. Test Execution
5. Debugging
6. Final Documentation

Defining and Analyzing the Problem

In this step, a programmer studies the problem. He decides the best way to solve these problems. Studying a problem is also necessary because it helps a programmer to decide about the following things:

- The facts and figures which are necessary for developing the program.
- The way in which the program will be designed
- Also, the language in which the program will be most suitable.
- What is the desired output and in which form it is needed, etc.

Designing the Algorithm

An algorithm is a sequence of steps that must be carried out before a programmer starts preparing his program. The programmer designs an algorithm to help visual possible alternatives in a program also.

Coding or Writing the Program

The next step after designing the algorithm is to write the program in a high-level language. This process is known as coding.

Test Execution

The process of executing the program to find out errors or bugs is called test execution. It helps a programmer to check the logic of the program. It also ensures that the program is error-free and workable.

Debugging

Debugging is a process of detecting, locating and correcting the bugs in a program. It is performed by running the program again and again.

Final Documentation

When the program is finalized, its documentation is prepared. Final documentation is provided to the user. It guides the user how to use the program in the most efficient way.

Furthermore, another purpose of documentation is to allow other programmers to modify the code if necessary. Documentation should also be done in each step during the development of the program.

Top-down approach

In this approach, a large project divides into small programs, and these programs are known as modules.

C programming language supports this approach for developing projects. It is always good idea that decomposing solution into modules in a hierarchal manner.

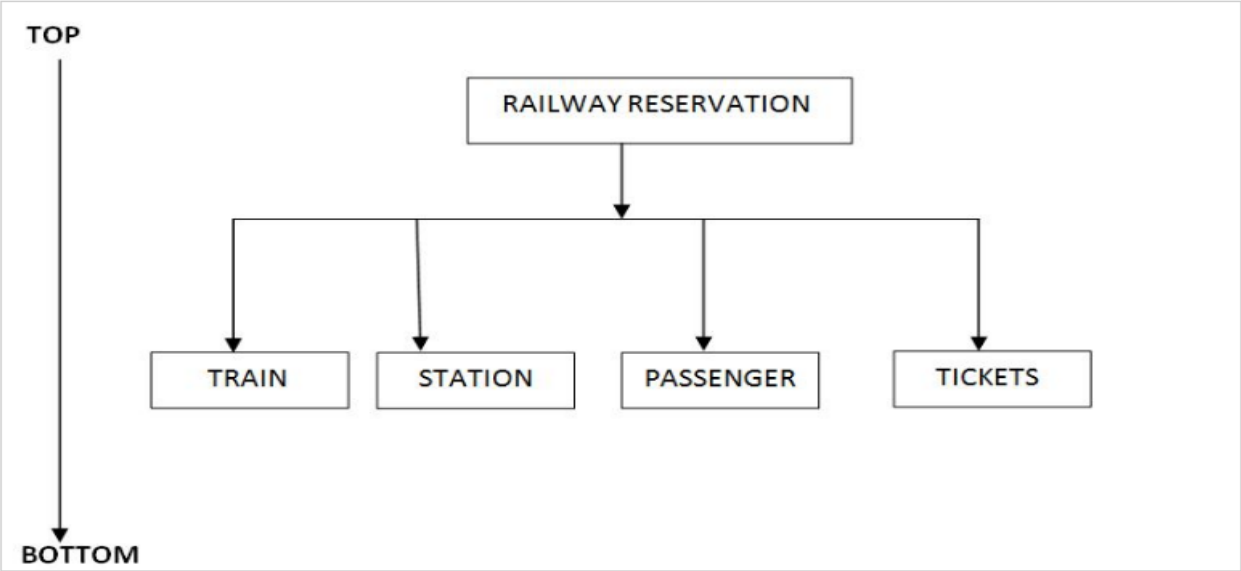
The basic task of a top-down approach is to divide the problem into tasks and then divide tasks into smaller sub-tasks and so on.

In this approach, first we develop the main module and then the next level modules are developed. This procedure is continued until all the modules are developed.

Advantages of top-down approach:

1. In this approach, first, we develop and test most important module.
2. This approach is easy to see the progress of the project by developer or customer.
3. Using this approach, we can utilize computer resources in a proper manner according to the project.
4. Testing and debugging is easier and efficient.
5. In this approach, project implementation is smoother and shorter.
6. This approach is good for detecting and correcting time delays.

Example :



UNIT I

ALGORITHMS

An algorithm is a step-by-step procedure to perform a given task. Each step in an algorithm is known as instruction.

Characteristics of an algorithm:

1. INPUT: Any algorithm accepts zero or more inputs.
2. OUTPUT: Any algorithm generates one or more outputs.
3. FINITENESS: Algorithm should be terminated in finite number of steps.
4. DEFINITENESS: The steps in the algorithm should be defined clearly i.e. no ambiguity should be raised
5. EFFECTIVENESS: Each step in the algorithm should be precise so that they can be easily traced on a paper.

Instruction can be divided into 3 types:

1. Sequence(also known as process)
2. Decision (also known as Selection)
3. Iteration or Repetition(also known as looping)

Sequence : Sequence means that each step or process in the algorithm is executed in the specified order.

Example:Write an algorithm for

1. To find sum of two numbers.

2. To compute area of triangle.

3.To swap two and three numbers with using temporary variable.

4. To swap two and three numbers without using temporary variable.

Selection : Selection means that in the algorithm the steps will be executed based on the condition. The selection staement is written using **If** (cond) **then** Task1 **otherwise** task2 .

If condition is true then statements after **then** clause will be executed, if condition is false then statement after **otherwise** clause will be executed.

Example:Write an algorithm for the following

1.To check whether entered number is positive or negative.

2.To check whether entered number is even or odd.

3.To check whether person is eligible for voting.

4.To check whether entered year is leap year.

5.To find maximum of two number.

6.To find maximum of three numbers.

Iteration: Iteration process is used when an instruction or set of instructions are to be repeated until some condition is satisfied.

Iteration can be illustrated using 3 ways:

1.It has the general form

```
Repeat
    Step 1
    Step 2
    ……
Until Condition
```

2. Second form

```
while Condition
begin
    Step1
    ……..
end.
```

3. Third form

Step 1

Step 2

….

Step N

if cond then goto Step 1.

Example :

1.Write an algorithm to print your name for 10 times.

2.Write an algorithm to print first n natural numbers where value of n is entered by user.

3.Write an algorithm to find sum of first n natural numbers where value of n is entered by user.

4.Write an algorithm to find sum of n different numbers .

5.Write an algorithm to count number of even numbers between given range.

6.Write an algorithm to sine function

7.write an algorithm to compute factorial of given number.

8.Write an algorithm to generate Fibonacci series.

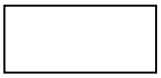
FLOWCHART

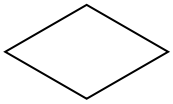
Pictorial representation of algorithm is known as flowchart.

Symbols used in flowcharts are

Start or Stop 

Input or Output 

Computational box 

Decision box 

Flow of direction 

Connector 

Advantages of flowcharts over algorithm

1. Communication: Flow charts are better way of communicating logic of a system to all concern.

2. Effective analysis: With the help of flow charts problems can be analyzed more effectively.

3. Proper documentation: Flow chart serve as good program documentation needed for various purposes.

4. Efficient coding: Flow charts acts as a guide or blue print during the program development phase.

5. Proper debugging: Flow charts help in the debugging process.

Limitation of flowcharts:

1. Complex logic: If the program logic is quite complicated then the flow chart becomes complex and clumsy.
2. Alterations and modifications: If alterations are required the flow chart may need to redrawn completely.

Draw flowcharts for the above algorithm.

INTRODUCTION TO C LANGUAGE

C was developed by Dennis Ritchie in 1972 at AT &T Bell labs, USA.

C was evolved from ALGOL,BCPL(Basic Combined Programming Language) and B language.

Features of C language (or) Importance of C language

1. Robustness: It is a robust language whose rich set of built-in functions and operators can be used to write any complex program.

2. Efficiency: Programs written in C are efficient and fast. This is due to its variety of data types and operators ,it is many times faster than BASIC.

3. Portability: C is highly portable. This means that C programs written for one computer can be run on another system with little or no modification.

4. Structured Language: C is a general purpose block structured programming language allowing the user to divide the problem in terms of function module or blocks.

5. Extensibility: C has ability to extend itself. A C program is basically a collection of functions that are supported by the C library. It allows us to add our own function to C library.

STRUCTURE OF C LANGUAGE

Include header file section

Global declaration section /*optional*/
Documentation section /*optional*/
main()
{ **Declaration part** **Executable part** **}**
User defined function section /*optional*/ **{** **Declaration part** **Executable part** **}**

Include Header files section

A header file is a file containing C declarations and macro definitions to be shared between source file,

Compiler,pre-processor,C-library, and other header files. These commands tells the compiler to do preprocessing before doing actual compilation. Like *#include <stdio.h>* is a preprocessor command which tells a C compiler to include stdio.h file before going to actual compilation.

There are two ways of including files in C program.

- The file included in angular brackets(< >) : #include<stdio.h>

This method of inclusion tells the preprocessor to look for the file in the pre-defined default location.

- the file included in double quotes : #include “filename”

This method of inclusion tells the preprocessor to look for the file in the current directory first, then look for it in the specified path. It is generally used when non standard header files are used in the program.

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 i.e. outside of all functions.

Documentation section: This section consists of set of comment lines giving the name of program ,author and other details which the programmer would like to use later. Comments are used to give additional useful information inside a C Program. All the comments will be put inside /*...*/ . A comment can span through multiple lines.

main() function section:

- Every **C** program must have one main function section.
- This section contains two parts :declaration part and executable part.
  - The declaration part declares all the variables used in the executable part.
  - There must be at least one statement in the executable part.
  - These two parts must appear between opening brace({) - closing brace (}).
- The program execution begins at the opening brace and ends at the closing brace.
- The closing brace of the main() function section is the logical end of the program.
- All the statements must end with a semi colon(;).

User defined function section: It contains all the definitions of user defined function that are called in the main() function.

SIMPLE C PROGRAM

```
#include<stdio.h>

/*This is a program to print
welcome message*/

main()

{

printf("welcome to C lab");

}
```

TOKENS

CHARACTE ge includes {A..Z,a..z,0..9,+,-.*,/,^,%,……}

In C language, led tokens.

All the progra

C tokens are of six types. They are,

1. Keywords (eg: int, while)
2. Identifiers (eg: main, total),
3. Constants (eg: 10, 20),
4. Strings (eg: “total”, “hello”),
5. Special symbols (eg: (), {}),
6. Operators (eg: +, /,-,*)

1.Keywords

C keywords are the words that convey a special meaning to the c compiler. Keywords have a fixed meaning and meaning cannot be changed by user. Keywords serve as basic building blocks. All keywords should be written in lower case letters. C language supports 32 keywords. The list of C keywords is given below:

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

2.Identifiers

Identifiers refer to the names of variables, functions and arrays. These are user defined names consisting of arbitrarily long sequence of letters and digits with either a letter or the underscore (_) as a first character.

There are certain rules that should be followed while naming identifiers:

1.They must begin with letter and consist of only letters, digits, or underscore.
2. They must begin with a letter or underscore(_).

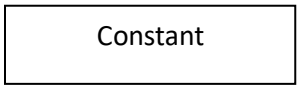
3.No other special character is allowed.
4.It should not be a keyword.
5.It must not contain white space.
6.Maximum length of identifier can be 31 characters but only first 8 characters are significant.
7.Upper case and lower case are significant.

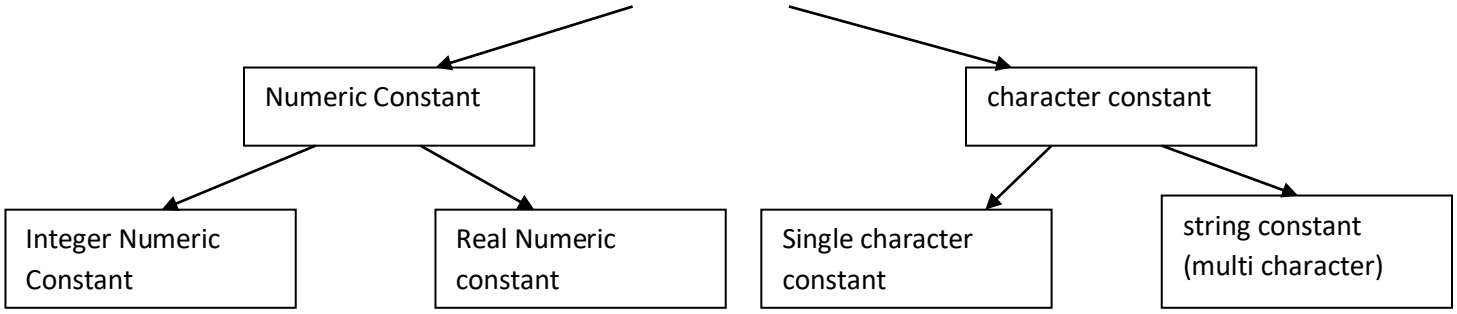
Some examples of c identifiers:

Name	**Remark**
abc	Valid
X12	Valid
A$b	Invalid as it contains special character other than the underscore
int	Invalid as it is a keyword
X yz	Invalid as it contains white space
_ab	Valid

3.Constants

C constants refer to the data items that do not change their value during the program execution. Several types of C constants that are allowed in C are:





Integer Constants

Integer constants are whole numbers without any fractional part. It must have at least one digit and may contain either + or – sign. A number with no sign is assumed to be positive.

There are three types of integer constants:

Decimal Integer Constants

Integer constants consisting of a set of digits, 0 through 9, preceded by an optional – or + sign.

Example of valid decimal integer constants
341, -341, 0, 8972

Octal Integer Constants

Integer constants consisting of sequence of digits from the set 0 through 7 starting with 0 is said to be octal integer constants.

Example of valid octal integer constants
010, 0424, 0, 0540

Hexadecimal Integer Constants

Hexadecimal integer constants are integer constants having sequence of digits preceded by 0x or 0X. They may also include alphabets from A to F representing numbers 10 to 15.

Example of valid hexadecimal integer constants
0xD, 0X8d, 0X, 0xbD

It should be noted that, octal and hexadecimal integer constants are rarely used in programming.

Real Constants

The numbers having fractional parts are called real or floating point constants. These may be represented in one of the two forms called *fractional form* or the *exponent form* and may also have either + or – sign preceding it.

Example of valid real constants in fractional form or decimal notation
0.05, -0.905, 562.05, 0.015

Representing a real constant in exponent form

The general format in which a real number may be represented in exponential or scientific form is

mantissa e exponent

The mantissa must be either an integer or a real number expressed in decimal notation.
The letter e separating the mantissa and the exponent can also be written in uppercase i.e. E
And, the exponent must be an integer.

Examples of valid real constants in exponent form are:
252E85, 0.15E-10, -3e+8

Character Constants

A character constant contains one single character enclosed within single quotes.

Examples of valid character constants

'a' , 'Z', '5'

It should be noted that character constants have numerical values known as ASCII values, for example, the value of 'A' is 65 which is its ASCII value.

Escape Characters/ Escape Sequences

C allows us to have certain non graphic characters in character constants. Non graphic characters are those characters that cannot be typed directly from keyboard, for example, tabs, carriage return, etc.These non graphic characters can be represented by using escape sequences represented by a backslash(\) followed by one or more characters.

NOTE: An escape sequence consumes only one byte of space as it represents a single character.

Escape Sequence	Description
\a	Audible alert(bell)
\b	Backspace

\f \n \r	Form feed New line Carriage return
\t	Horizontal tab
\v	Vertical tab
\ “ ‘	Backslash Double quotation mark Single quotation mark
?	Question mark
	Null

String Constants

String constants are sequence of characters enclosed within double quotes. For example, “hello”,“abc”,“hello911”,”a”,”1”.

4. Special Symbols

The following special symbols are used in C having some special meaning and thus, cannot be used for some other purpose.

[] () {} , ;, :, *, … #

Braces{}: These opening and ending curly braces marks the start and end of a block of code containing more than one executable statement.

Parentheses(): These special symbols are used to indicate function calls and function parameters.

Brackets[]: Opening and closing brackets are used as array element reference. These indicate single and multidimensional subscripts.

5. Strings

String constants are sequence of characters enclosed within double quotes. For example, “hello”,“abc”,“hello911”,”a”,”1”.

DATATYPES

Data types in c refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted. It also specifies the valid operation on the type.

The types in C can be classified as follows –

1. Primary or fundamental data type 2. Derived data type 3. User defined data type

Primary or fundamental data type

The various primary data types supported by C language are

1. Integers

2. Real

3. Character

4. Void

Integer Types

The keyword for integers is **int.** The integer data type requires 2 bytes of storage in windows and 4 bytes of storage in Linux environment.

Integer data types are of two types signed (both positive and negative numbers)and unsigned(only positive numbers).

The following table provides the details of standard integer types with their storage sizes and value ranges –

Type	Storage size	Value range
signed int or int	2 bytes(windows) or 4 bytes(Linux)	-32,768 to 32,767 **or** -2,147,483,648 to 2,147,483,647
unsigned int	2 bytes(windows) or 4 bytes(Linux)	0 to 65,535 **or** 0 to 4,294,967,295
short int or signed short int	1 byte(windows) or 2 bytes(Linux)	-128 to 127 or -32,768 to 32,767
unsigned short	1 byte(windows) or 2 bytes(Linux)	0 to 255 (or) 0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

If a data type needs n-bits of storage then

Signed	-2^{n-1} to $2^{n-1}-1$
Unsigned	0 to $2^{n}-1$

Floating-Point Types

The keyword for real numbers is **float**. The memory required for float data type is 4 bytes.

The following table provide the details of standard floating-point types with storage sizes and value ranges and their precision −

Type	Storage size	Value range	Precision
float	4 byte	-3.4E-38 to +3.4E+38	6 decimal places
double	8 byte	-1.7E-308 to +1.7E+308	15 decimal places
long double	10 byte	-3.4E-4932 to 1.1E+4932	19 decimal places

The header file float.h defines macros that allow you to use these values and other details about the binary representation of real numbers in your programs.

Character data types

The keyword for character data type is **char**. The memory required for char data type is 1 byte.

Type	Storage size	Value range
unsigned char	1 byte	0 to 255
signed char or char	1 byte	-128 to 127

void data type :void type means no value. This is usually used to specify the type of functions.

Derived Data types: The data types which have been derived from fundamental data types is known as derived data type.

Ex: arrays, functions ,strings, structures, unions ,pointers.

User defined data type: Those data types which are defined by the user as per his/her will are called user-defined data types.

C supports the features “typedef” that allows users to define the identifier which would represent an existing data type. This defined data type can then be used to declare variables:

Syntax: typedef int numbers;

numbers num1,num2;

in this example, num1 and num2 are declared as int variables. The main advantage of user defined data type is that it increases the program’s readability.

Another type is enumerated type. This is also a user defined data type

Syntax:enum identifier {value1,value2, value 3,…};

“Enum” is the keyword and “identifier” is the user defined data type that is used to declare the variables. It can have any value enclosed within the curly braces. For example:

enum day {January, February,March,April,..};

enum day month_st,month_end;

The compiler automatically assigns integer digits beginning from 0 to all the enumeration constants. For example, “January” will have value 0 assigned, “February” value 2 assigned and so on. You can also explicitly assign the enumeration constants.

VARIABLES.

Variables are user defined entity to hold values and values may vary in course of execution. These are user defined names consisting of arbitrarily long sequence of letters and digits with either a letter or the underscore (_) as a first character.

There are certain rules that should be followed while naming variable:

- They must begin with a letter or underscore(_).
- They must consist of only letters, digits, or underscore.
- No other special character is allowed.
- It should not be a keyword.
- It must not contain white space.
- Maximum length of variable can be 31 characters but only first 8 characters are significant.

- Upper case and lower case are significant.

Some examples of C variables:

Name	Remark
abc	Valid
X12	Valid
A$b	Invalid as it contains special character other than the underscore
int	Invalid as it is a keyword
X yz	Invalid as it contains white space

DECLARATION OF VARIABLE

A declaration statement begins with a type ,followed by the name of one or more variables.

The general form is

data_type var1, var2,var3,…….varn;

ex: int a;

float x,y;

char m,n,p;

- Declaration introduces variables in the program.
- Definition directs the compiler to allocate memory for the variable.

Input and Output functions

C language supports two types of input-output functions

1. Unformatted input-output functions

2. Formatted input-output functions

Unformatted I/O in C

Unformatted I/O functions works only with character data type (char).

The unformatted Input functions used in C are getch(), getche(), getchar(), gets().

Syntax for getch () in C:

variable_name=getch();

getch() accepts only single character from keyboard. The character entered through getch() is not displayed in the screen (monitor).

Syntax for getche() in C :

variable_name=getche();

Like getch(), getche() also accepts only single character, but unlike getch(), getche() displays the entered character in the screen.

Syntax for getchar() in C :

variable_name=getchar();

getchar() accepts one character type data from the keyboard.

Syntax for gets() in C :

gets(variable_name);

gets() accepts any line of string including spaces from the standard Input device (keyboard). gets() stops reading character from keyboard only when the enter key is pressed.

The unformatted output statements in C are **putch, putchar** and **puts**.

Syntax for putch in C :

putch(variable_name);

putch displays any alphanumeric characters to the standard output device. It displays only one character at a time.

Syntax for putchar in C :

putchar(variable_name);

putchar displays one character at a time to the Monitor.

Syntax for puts in C :

puts(variable_name);

puts displays a single / paragraph of text to the standard output device.

Formatted Input and Output

printf()

The function printf() is used for formatted output to standard output based on a format specification. The format specification string, along with the data to be output, are the parameters to the printf() function.

Syntax:

printf ("control string", data1, data2,……..);

In this syntax control string can be the format specification string or simple text message or line feed character or tab space. The format specification begins with the symbol % followed by a character called the conversion character because it allows one data type to be converted to another type and printed.

The following table shows conversion character and their meanings.

Conversion Character	Meaning
d	The data is converted to decimal (integer)
c	The data is taken as a character.
s	The data is a string and character from the string , are printed until a NULL, character is reached.
f	The data is output as float or double with a default Precision 6.
Symbols	**Meaning**

\n	For new line (linefeed return)
\t	For tab space (equivalent of 8 spaces)

Example:
printf ("%c", data1); prints a character value on the monitor.
Formatted Input

The function scanf() is used for formatted input from standard input and provides many of the conversion facilities of the function printf().

Syntax

scanf ("control string", var1, var2,……);
The function scnaf() reads and converts characters from the standards input depending on the format specification string and stores the input in memory locations represented by the arguments (var1, numvar2,….).

For Example:

scanf("%c%d",&Name,&Roll No);

reads a character and integer value from keyboard and stores at address locations of name and roll No.