APPLICATIONS OF COMPUTERS

Word processing : Word processing software enables users to read and write documents. Users can also add images. tables, and graphs for illustrating a concept.

Government : computers used in government organizations to keep records on legislative actions, Internal Revenue Service Records.

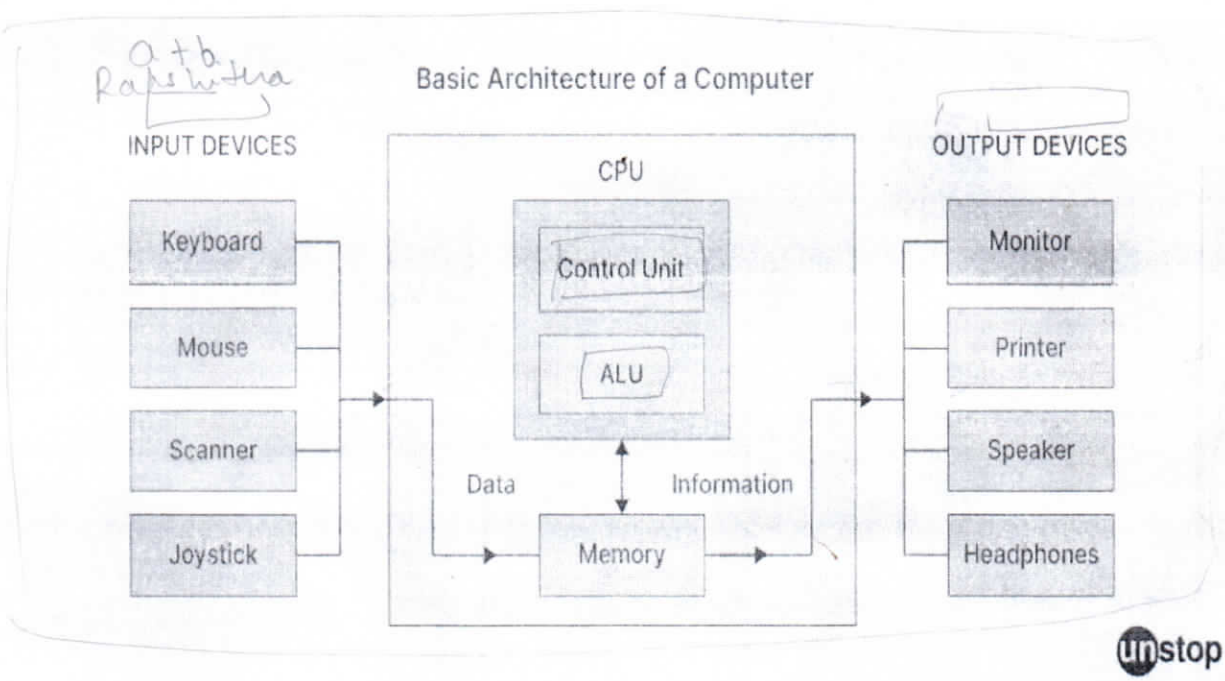
Sports : computers are used to compile statistics, identify weak players and strong players by analysing statistics.

Music : all computers today have a musical instrument digital interface (MIDI) facility, which links musical instruments to PC.

Movies : computers are used to create sets, special effects, animations, cartoons, imaginary characters, videos and commercials.

Hospitals : hospitals use computers to record every information about patients, from time their admission their exit.

BASIC ORGANIZATION OF A COMPUTER



computer is an electronic device that basically performs five major operations:

- **Accepting data or instructions (input)**
- **Storing data.**
- **Processing data.**
- **Displaying results (output).**
- **Controlling and coordinating all operation inside a computer.**

Input This is the process of entering data and instructions (also known as programs) in to the computer system.

- The data and instructions can be entered by using different input devices such as keyboard, mouse, scanner, trackball, etc.
- Note that computers understand binary language, which consists of only two symbols (0 and 1), so it is the responsibility of the input devices to convert the input data into binary codes.

Storage Storage is the process of saving data and instructions permanently in the computer so that they can be used for processing.

A computer has two types of storage areas:

- **Primary storage**
- **Secondary storage**

Primary storage Primary storage also known as the main memory, is the storage area that is directly accessible by the CPU at very high speeds.

- Primary storage space is very expensive and therefore limited in capacity. Another drawback of main memory is that it is volatile in nature, that is, as soon as the computer is switched off, the information stored gets erased.
- it cannot be used as a permanent storage of useful data and programs for future use. An example of primary storage is the RAM.

Secondary storage Secondary storage Also known as the secondary memory or auxiliary memory.

- this is just the opposite of primary memory. It basically overcomes all the drawbacks of the primary storage area. It is cheaper, non-volatile, and used to permanently store data and programs.

Processing The process of performing operations on the data as per the instructions specified by the user (program) is called processing.

- Data and instructions are taken from the primary memory and transferred to the arithmetic and logical unit (ALU), which performs all sorts of calculations.
- The intermediate results of processing may be stored in the main memory, as they might be required again.
- When the processing completes, the final result is then transferred to the main memory. Hence, the data may move from main memory to the ALU multiple times before the processing is over.

Output Output is the process of giving the result of data processing to the outside world (external to the computer system).

The results are given through output devices such as monitor, printer, etc. Since the computer accepts data only in binary form and the result of processing is also in binary form, the result cannot be directly given to the user.

- The output devices, therefore, convert the results available in binary codes into a human-readable language before displaying it to the user.

Control The control unit (CU) is the central nervous system of the entire computer system.

- It manages and controls all the components of the computer system.
- It is the CU that decides the manner in which instructions will be executed and operations performed.
- It takes care of the step-by-step processing of all operations that are performed in the computer.
- ***Note*** *that the CPU is a combination of the ALU and the CU. The CPU is better known as the brain of the computer system because the entire processing of data is done in the ALU, and the CU activates and monitors the operations of other units (such as input, output, and storage) of the computer system.*
- ALU, CU, and CPU are the key functional units of a computer system.

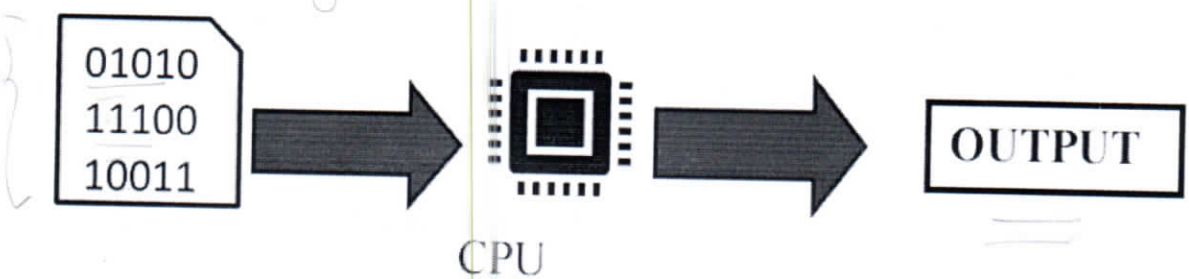
Evolution of Computer Languages

A programming language is a set of instructions and syntax used to create software programs.

Types Of Programming Languages:

- Machine Language
- Assembly Language
- High-Level Language

1. Machine Language



- It is the lowest and most elementary level of Programming language and was the first type of programming language to be Developed. Machine Language is basically the only language which computer Can understand.
- In fact, a manufacturer designs a computer to obey just one Language, its machine code, which is represented inside the computer by a String of binary digits (bits) 0 and 1s.
- The symbol 0 stands for the absence of Electric pulse and 1 for the presence ofan electric pulse . Since a computer is Capable of recognizing electric signals, therefore, it understand machine Language.

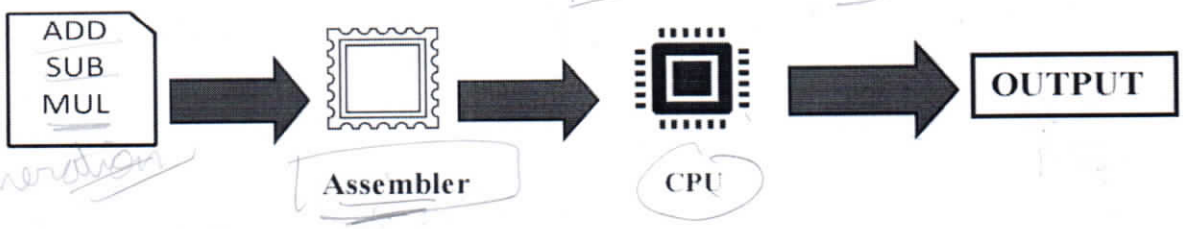
Advantages Of Machine Language :

- ✓ It makes fast and efficient use of the computer.
- ✓ It requires no translator to translate the code i.e. Directly understood by the computer.

Disadvantages Of Machine Language :

- ✓ All operation codes have to be remembered.
- ✓ These languages are machine dependent i.e. a particular Machine language can be used on only one type of computer.

2. Assembly Language



- The second generation programming languages (2GL) include the assembly language.
- They are symbolic programming languages that use symbolic notation to represent machine language instructions.
- low-level language.
- It used symbolic codes, also known as mnemonic codes. which are easy-to-remember abbreviations, rather than numbers.
- Examples of these codes include ADD for addition, CMP for compare, MUL for multiply, etc.
- Assembler(tool) is going to covert assembly language to machine language.

Advantages Of Assembly Language :

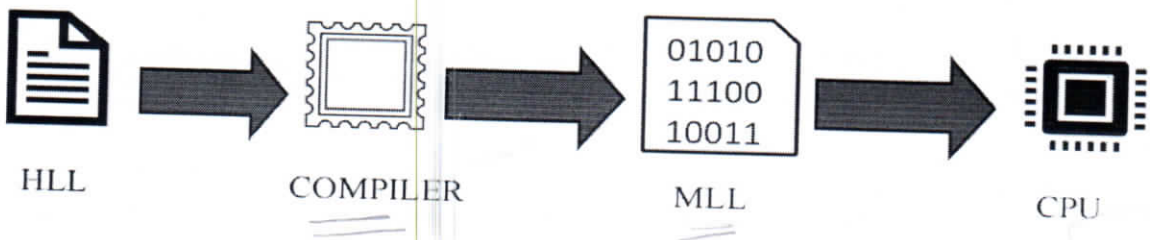
- It is easier to understand and use as compared to machine language.
- It is easy to locate and correct errors.
- It is modified easily.

Disadvantages Of Assembly Language :

- Like machine language it is also machine dependent.
- Since it is machine dependent therefore programmer Should have the knowledge.
- It is easier to understand and use as compared to machine language.
- It is easy to locate and correct errors.
- It is modified easily of the hardware also.

High Level Languages

- High level computer languages give formats close to English language and the purpose of developing high level languages is to enable people to write programs easily and in their own native language environment (English).
- High-level languages are basically symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes.
- Each instruction in the high level language is translated into many machine language instructions thus showing one-to-many translation.



Advantages of High Level Language

- User-friendly
- Similar to English with vocabulary of words and symbols.
- Therefore it is easier to learn.
- They are easier to maintain.

Disadvantages of High Level Language

- A high-level language has to be translated into the machine language by a translator and thus a price in computer time is paid.
- The object code generated by a translator might be inefficient Compared to an equivalent assembly language program.

Design Tools:

1. Algorithms

An algorithm provides a blueprint to writing a program to solve a particular problem.

or

The algorithm is a 'step-by-step procedure' to solve the problem.

- ✓ It is considered to be an effective procedure for solving a problem in a finite

number of steps.

- ✓ A well-defined algorithm always provides an answer, and is guaranteed to terminate.
- ✓ Algorithms are mainly used to achieve software re-use.
- ✓ A good algorithm must have the following characteristics:
  - Be precise.
  - Be unambiguous.
  - Not even a single instruction must be repeated infinitely.
  - After the algorithm gets terminated, the desired result must be obtained.

Control Structures Used In Algorithms

1. Sequence : Sequence means that each step of the algorithm is executed in the specified order.

Example: Algorithm to Add two numbers.

```
Step 1: Input the first number as A
Step 2: Input the second number as B
Step 3: SET SUM = A + B
Step 4: PRINT SUM
Step 5: END
```

2. Decision :. Decision statements are used when **the outcome of the process depends on some condition.** For example, if x=y, then print "EQUAL". Hence, the general form of the if construct can be given as: **if** condition then process.

Example: Algorithm to test Equality of two numbers.

```
Step 1: Input the first number as A
Step 2: Input the second number as B
Step 3: IF A = B
            Then PRINT "EQUAL"
        ELSE
            PRINT "NOT EQUAL"
Step 4: END
```

3. Repetition: Repetition, which involves **executing one or more steps for a number of times**, can be implemented using constructs such as **while, do-while, and for loops**. These loops execute oneor more steps **until some condition is true.**

Example: Algorithm to print first 10 Natural numbers.

```
Step 1: [INITIALIZE] SET I = 0, N = 10
Step 2: Repeat Step while I<=N
Step 3:   PRINT I
Step 4: SET I + 1
Step 5: END
```

Flowcharts :

A flowchart is a graphical representation or symbolic or diagrammatical representation.

Or

The pictorial representation of algorithm is called flowchart.

Uses of flow chart: Flow chart helps to understand the program easily. As different symbols are used specify the type of operation performed, it is easier to understand the complex programs with the help of flowcharts.

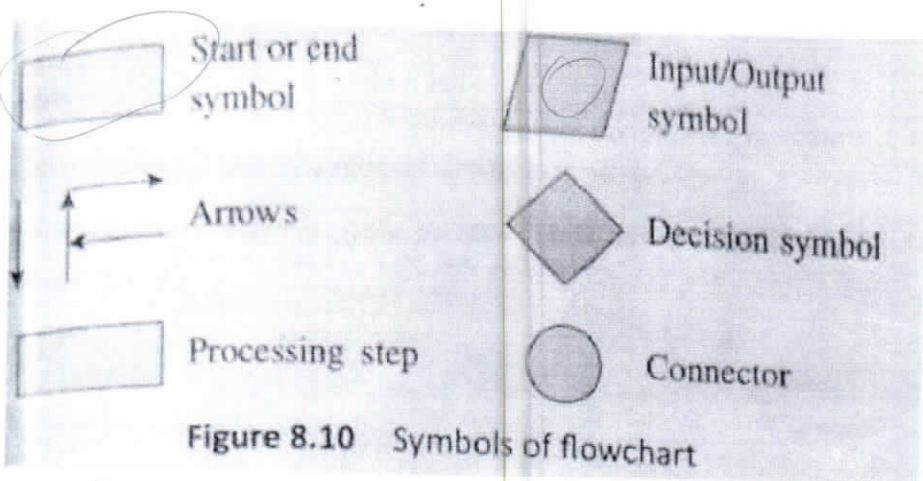
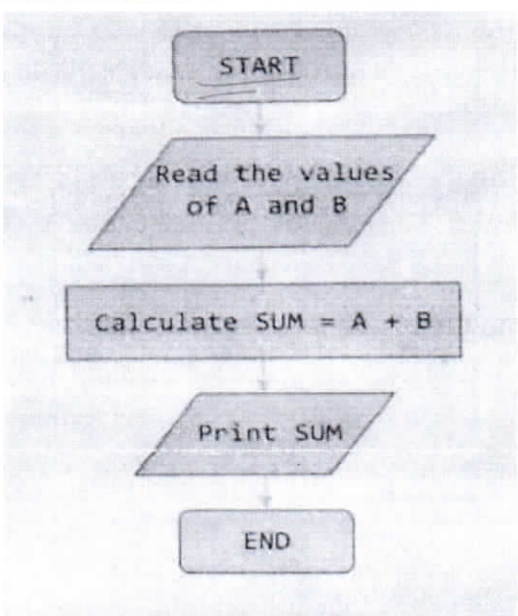


Figure 8.10 Symbols of flowchart

- **Start and end symbols :** are also known as the terminal symbols and are represented as circles, ovals, or rounded rectangles. Terminal symbols are always the first and the last symbols in a flowchart.

- **Arrows :** depict the flow of control of the program. They illustrate the exact sequence in which the instructions are executed. **Input/Output symbols :** are represented using a Parallelogram and are used to get inputs from the users or display the results to them.
- **Generic processing step :** also called as an activity, 1s represented using a rectangle. Activities include instructions such as add a to b, save the result.
- **A conditional or decision symbol :** is represented using a diamond. It is basically used to depict a Yes/No question or a True/False test. The two symbols coming out of it, one from the bottom point and the other from the right point, corresponds to Yes or True, and No or False, respectively. The arrows should always be labelled. A decision symbol in a flowchart can have more than two arrows, which indicates that a complex decision is being taken.
- **Labelled connectors :** are represented by an identifying label inside a circle and are used in complex or multi-sheet diagrams to substitute for arrows. For each label, the 'outflow' connector must have one or more 'inflow' connectors. A pair of identically labelled connectors is used to indicate a continued flow when the use of lines becomes confusing.

EXAMPLES :



Flowchart to ADD two numbers.

Pseudocode:

- Pseudocode is a form of structured English that describes algorithms.

- It facilitates designers to focus on the logic of the algorithms without getting bogged down by the details of language syntax.
- It is basically meant for human reading rather than machine reading.
- The purpose of pseudocodes is to enhance human understandability of the solution.
- This helps even non- programmers to understand the logic of the designed solution.
- Keywords used while writing pseudocodes are for looping and selection the designer must include the keywords.
  Do while…..end do;
  If……end if;
  Case…end case;

EXAMPLES: Write a pseudocode for calculating the price of a product after adding the sales tax to its original price.

```
1. Read the price of the product
2. Read the sales tax rate
3. Calculate sales tax = price of the item *;
   sales tax rate
4. Calculate total price = price of the product
   + sales tax
5. Print total price
6. End
variables: price of the item, sales tax rate,
   sales tax, total price
```

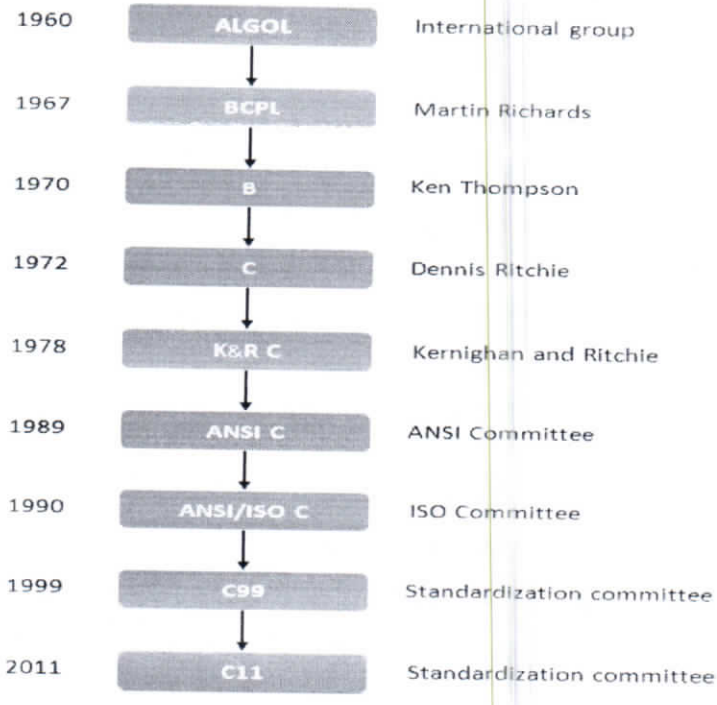
INTRODUCTION TO C

- The programming language C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories.
- It was named 'C' because many of its features were derived from an earlier language called "B".
- It is a good idea to learn C because it has been around for a long time which means there is a lot of information available on it.
- Quite a few other programming languages such as C++ and Java are also based on C which means that one will be able to learn them more easily in

the future.

BACKGROUND

- C is a structured, high-level, machine independent language. It allows software developers to develop programs without worrying about the hardware platforms where they will be implemented.
- The root of all modern languages is ALGOL, introduced in the early 1960s.ALGOL was the first computer language to use a block structure. ALGOL gave the concept of structured programming to the computer science community.
- In 1967, Martin Richards developed a language called BCPL (Basic Combined Programming Language) primarily for writing system software.
- In 1970, Ken Thompson created a language using many features of BCPL and called it simply B. Both BCPL and B were type-less(had no concept of data types) system programming languages.
- C was evolved from ALGOL, BCPL and B by Dennis Ritchie at the Bell Laboratories in 1972. C uses many concepts from these languages and added the concept of data types and other powerful features.
- During 1970s. C had evolved into what is now known as "traditional C. The language became more popular after publication of the book "The C Programming Language by Bnan Kermingham and Dennes Ritchie in 1978. The book was so popular that the language came to be known as K&R C' among the programming community.
- in 1983, American National Standards Institute (ANSI) appointed a technical committee to define a standard for C. The committee approved aversion C in December 1989 which is now known as ANSI C.
- It was then approved by the International of Standards Organization (ISO)in 1990.
- This version of C is also referred to as C89 During 1990's, C++, a language entirely based on C.



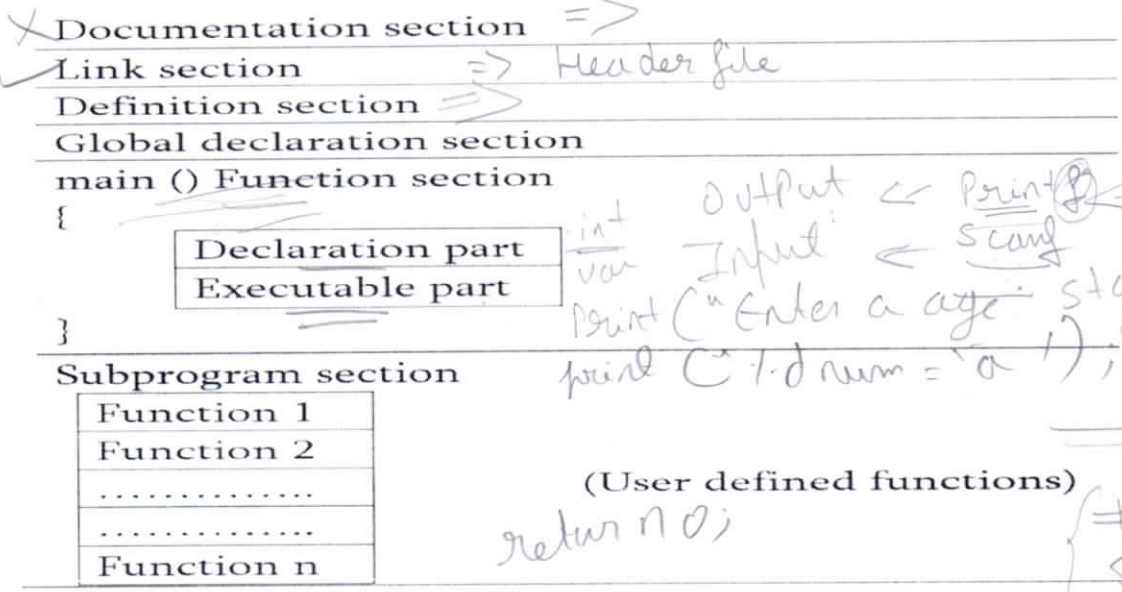
CHARACTERISTICS OF C

- C is a robust language whose rich set of built-in functions and operators can be used to write complex programs.
- A high level programming language enables the programmer to concentrate on the problem at hand and not worry about the machine code.
- Small size C has only 32 keywords. This makes relatively easy to learn as compared to other languages .
- Supports pointers to refer computer memory, array, structures, and functions.
- C is a core language as many other programming languages (such as C++, Java, or Perl) are based on C. If one knows C, learning other computer languages becomes much easier.

USES OF C

- C is a very simple language that is widely used by software professionals around the globe.
- C language is primarily used for system programming.
- C is widely used to implement end-User applications.
- Programming language is the basic foundation of the other programming language; therefore, it is called the 'mother language' for many other programming languages.

Structure of C program



- The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.
- The link section provides instruction to the compiler to link functions from the system library.
- The definition section defines all symbolic constants. 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.
- Every C program must have one main() function. This section contains two parts, declaration part and executable part.
- The declaration part declares all the variables used in the executable part. There is atleast one statements in the executable part. These two parts must appear between the opening and the closing braces.
- 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 statements in the declaration and executable parts end with the semi colon(;).
- The sub-program section contains all the user-define functions that are

called in the main function.

- User-define function are generally placed immediately after the main function, although they may appear in any order.

All sections, except the main function section may be absent when they are not required.

WRITING THE FIRST C PROGRAM

To Write a C program, we first need to write the code. For this, open a text editor. If you are a Windows user you may use C Notepad and if you prefer working on UNIX/Linux you can use emacs or vi. Once the text editor is opened on your screen, type the following statements:

```
#include <stdio.h>
int main( )
{
 printf("\n Welcome to the world of C Program");
return 0;
}
```

Output :

Welcome to the world of C Program

#include <stdio.h> :

- This is a pre-processor command that comes as the first statement in our code.
- All pre-processor commands start with symbol hash (#).
- The #include statement tells the compiler to include the standard input/output library or header file (stdio.h) in the program.

int main () :

- Every C program contains a main() function which is the starting point of the program. int is the return value of the main() function.
- After all the statements in the program have been written, the last statement of the program will return an integer value to the operating system.
- { } The two curly brackets are used to group all the related statements of the main function. All the statements between the braces form the function body.
- The function body contains a set of instructions to perform the given task.

printf("\n Welcome to the Note world of C ") :

- The printf function is defined in the stdio.h file and is used to print text on

the screen.

- The message that has to be displayed on the screen is enclosed within double quotes and put inside brackets.
- '\n' is an escape sequence and represents a newline character. It is used to print the message on a new line on the screen.

return 0; :

This is a return command that is used to return the value 0 to the operating system to give an indication that there were no errors during the execution of the program.

Steps required to create and execute a C program

To create and execute a C program in Turbo C (an older IDE often used for C/C++ programming, especially in educational setups), follow these steps:

1. **Download and Install Turbo C**

If you don't already have Turbo C installed, follow these steps:

- **Windows**:
- Download Turbo C or Turbo C++ from sites offering older software archives.
- Install it by running the installer.

If you're using a modern Windows OS (Windows 7 or later), you might need to use a DOSBox emulator to run Turbo C, as it's a 16-bit application.

2. **Open Turbo C**
   - Open the Turbo C application.
   - If using DOSBox, open Turbo C within the DOSBox environment.

3. **Create a New Program :** From the **File** menu, select **New**. This will open the editor window where you can type your C code.

Example C program:

```
int main() {
    int localVar = 10;  // Local variable
    printf("Hello, World!");  // Statement to print a message
    return 0;
}
```

4. **Save the program**
   - Go to **File > Save** or press **F2.**
   - Save the file with a **.c** extension(e.g..., hello.c).

5. **Compile the Program**
   - To compile the program, go to compile > compile or press **Alt + F9.**

- If there are no errors, the program will be compiled successfully. If there are any errors, they will be displayed in the message window at the bottom. Fix errors andrecompile.

6. Run the Program.

- To run the program, go to Run > Run or press **Ctrl + F9.**
- The program will execute, and the output will be shown in the output window.
- For the example above, you should see the output.

7. View the Output.

After the program runs, you can view the output by pressing **Alt + F5**. This brings up the output screen where you can see the program's results.

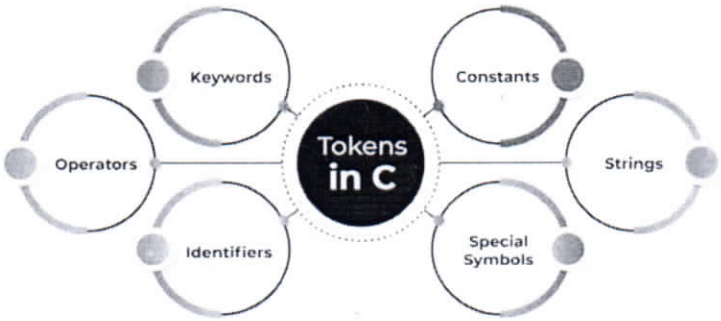
8. Exit Turbo C : Once done, you can exit Turbo C by going to **File > Quit.**

Tokens

- A token in C can be defined as the smallest individual element of the C programming language that is meaningful to the compiler.
- It is the basic component of a C program.

Types of Tokens in C

The tokens of C language can be classified into six types based on the functions they are used to perform. The types of C tokens are as follows:



1. **Keywords**
2. **Identifiers**
3. **Constants**
4. **Strings**
5. **Special Symbols**
6. **Operators**

Keywords and identifiers

- Every c words is classified as either keyword or identifiers.
- The keywords are pre-defined **or** reserved words in a programming language.
- It has some pre-defined meaning and this meaning cannot be changed.
- All keywords must be written in a lowercase.
- C language supports 32 keywords.

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

Identifiers

- It refers to the name of the objects.
- These are the names given to variables, functions, macros etc.

The rules to write an identifier are as follows:

1. It must contain only alphabets (A to Z, a to z), numbers (0 to 9) and underscore (_).
2. It must start with alphabet or underscore but not numeric character.
3. It should not contain any special symbols apart from underscore.
4. It should not have any space.
5. Keywords cannot be used as identifiers.
6. Identifiers are case sensitive.
7. Maximum length of an identifier is 31 characters.

Examples of valid identifiers include the following :

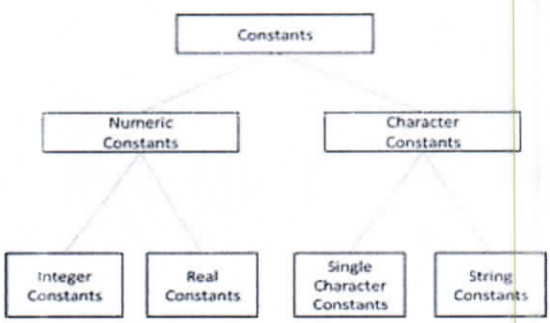
Roll_number, RollNo, marks, name, dept

Examples of invalid identifiers include :

23_student, %marks, @name, #emp_number, _#dept.

Constants

- Constants in c refers to the fixed values that do not change during the execution of a c program.
- Variables can change their value at any time but constants can never change their value.
- Constants are used to define fixed values such as pi.
- The **const** keyword is used to define the constants.



Integer constants

An integer constant refers to a sequence of digits.

There are three types of integers:

1. **Decimal integers :**
   - consist of a set of digits, 0 through 9.
   - Preceded by an optional – or + sign.
   - valid examples: 123, -321, 0 , 976535, +18.
   - Invalid examples: 15 789, 20,000, $1000.
2. **Octal integers :**
   - Consist of any combination of digits from set 0 through 7 with leading 0.
   - Examples: 037, 0, 0456, 0334
3. **Hexadecimal integers.**
   - A sequence of digits preceded by 0x or 0X.
   - They may also include alphabets A through F or a through f.
   - The letter A through F represents the number 10 through 15.
   - Examples: 0X2, 0x9F, 0Xbcd, 0x.

Real constants :

- Integer numbers are inadequate to represent quantities that vary continuously, such as distances, heights, temperatures, prices, and so on.
- These quantities are represented by numbers containing fractional parts like 17.548. Such numbers are called real (or floating point) constants.
- Examples of real constants are: 0.0083, -0.75, 435.36, +247.0.
- These numbers are shown in decimal notation, having a whole number followed by a decimal point and the fractional part.
- It is possible to omit digits before the decimal point, or digits after the decimal point.
- Examples 215. 95-71 +5.

Character Constants :

- A single character constant (or simply character constant) contains a single

character enclosed within a pair of single quote marks.

- Example of character constants is as follows: 'c' '8' ';' ' '
- Note that the character constant '5' is not the same as the number 5.
- The last constant is a blank space.
- Character constants have integer values known as ASCII values.
- For example, the statement: printf("%d", 'a'); would print the number 97, the ASCII value of the letter a. Similarly, the statement printf("%c", 97'); would output the letter 'a'.

String constants:

- A string constant is a sequence of characters enclosed in double quotes. The characters may be letters, numbers, special characters and blank space.
- Examples are: "Hello!", "1987", "WELL DONE" ,"?", "5+3" ,"X"

Variables:

- A variable is a data name that may be used to store a data value.
- A variable name can be chosen by the programmer in a meaningful way so as to reflect its function or nature in the program.
- Examples: average, height, total, class_strength.
- When using a variable , we actually refer to an address of the memory where the data is stored.
- A variable name contains maximum of 30 characters.
- A variable name includes alphabets and numbers, but it must start with an alphabet.
- It cannot accept any special characters, blank spaces except under score(_).
- It should not be a reserved word.

C Variable Syntax

```
data_type variable_name = value;
```

data_type: Type of data that a variable can store.
variable_name: Name of the variable given by the user.
value: value assigned to the variable by the user.

Example:

int var; // integer variable

char a; // character variable

float fff; // float variables

Note: *C is a strongly typed language so all the variables types must be specified before using them.*

aspects of defining a variable:

1. **Variable Declaration**: Each variable to be used in the program must be declared. To declare a variable, specify the data type of the variable followed by its name.

Ex: int marks, char grade, float salary,

2. **Variable Initialization**: While declaring the variables, we can also initialize them with some value.

Ex: int marks=100; char grade='A';

Data Types :

A **data type** specifies the type of data that a variable can store. It defines the size, layout of memory, and the range of values that can be stored, as well as the set of operations that can be performed on the variable.

ANSI C supports three classes of data types:

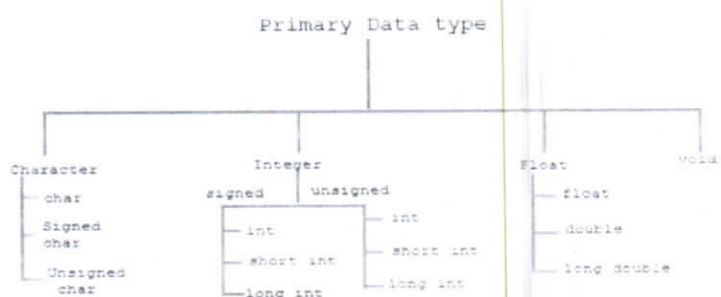
1. **Primary (or fundamental) data types**
2. **Derived data types**
3. **User-defined data types**

All C compilers support five fundamental data types, namely

- integer (int),
- character (char),
- floating point (float),
- double-precision floating point (double)
- void.
- Many of them also offer extended data types such as long int and long double.

Data type	Keyword used	Size in bytes	Range	Use
Character	char	1	–128 to 127	To store characters
Integer	int	2	–32768 to 32767	To store integer numbers
Floating point	float	4	3.4E–38 to 3.4E+38	To store floating point numbers
Double	double	8	1.7E–308 to 1.7E+308	To store big floating point numbers
Valueless	void	0	Valueless	—

1. **Primary (or fundamental) data types :**



Integer Data Type :

Integer data type is used to store whole numbers, both positive and negative, without any decimal points. These numbers can be as simple as 5, -20, or 1000. C provides several types of integers based on the size and range of values they can store.

Here are the most common **integer data types** in C:

1. **int**: Stores a standard whole number (typically 4 bytes, so it can hold numbers in the range -2,147,483,648 to 2,147,483,647).
2. **short int** or **short**: Stores smaller whole numbers (typically 2 bytes, with a smaller range).
3. **long int** or **long**: Stores larger whole numbers (typically 4 or 8 bytes, for storing larger values).

4. **unsigned int**: Stores only positive whole numbers (it doubles the positive range by removing negative values).

Examples in C :

```
#include <stdio.h>

int main() {
   int num1 = 10;       // Regular int
   short num2 = -50;    // Short int
   long num3 = 1000000; // Long int
   unsigned int num4 = 30; // Unsigned int (only positive)
   printf("num1: %d\n", num1); // %d for printing int
   printf("num2: %d\n", num2); // %d for short
   printf("num3: %ld\n", num3); // %ld for long int
   printf("num4: %u\n", num4); // %u for unsigned int
   return 0;
}
```

Key points :

- int is the most commonly used integer type and can store both positive and negative whole numbers.
- short and long adjust the size of the number you can store.
- unsigned int is used when you only need positive numbers, allowing you to store larger positive numbers.

So, integer data types in C allow you to store whole numbers of different sizes, depending on how large or small the number needs to be.

Character Data Type :

Character data type is used to store a single character, such as a letter, number, or symbol. The character data type is represented by char and takes up 1 byte of memory (which can hold 256 different values, including letters, numbers, and special symbols).

- char: It stores a single character. For example, 'A', '5', or '#'.
- Characters are stored as numbers based on the ASCII value of the character. For instance, the character 'A' is stored as the number 65 in memory.

Example in C:

```
#include <stdio.h>
int main() {
   char letter = 'A';  // declaring a char variable and assigning 'A'
   printf("The character is: %c\n", letter);  // %c is used to print a character
   return 0;
}
```

- We declare a char variable named letter and store the character 'A' in it.
- The %c format specifier is used in printf() to print the character.

So, the character data type (char) in C is a simple way to store and use individual characters.

Float Data Type :

The float data type is used to store numbers that have decimal points, such as 3.14, -0.5, or 10.75. These are also called "floating-point numbers" because the decimal point can "float" or be placed anywhere in the number. They are useful when you need to store more precise values, like measurements or percentages.

Here are the main types of floating-point numbers in C:

1. **float**: Used to store single-precision decimal numbers. It typically takes 4 bytes of memory and can handle numbers with up to 6-7 decimal digits of precision.
2. **double**: Used to store double-precision decimal numbers. It takes 8 bytes of memory and can handle numbers with up to 15 decimal digits of precision.
3. **long double**: Used for even more precision (more digits after the decimal point), though it's less commonly used.

Example in C:

```
#include <stdio.h>
int main()
 {
   float num1 = 3.14f;        // Single precision float (with 'f' to specify float)
```

```
    double num2 = 3.1415926535; // Double precision float
    long double num3 = 3.14159265358979323L; // Long double (with 'L' to specify long double)
    printf("num1 (float): %f\n", num1);    // %f is used to print a float
    printf("num2 (double): %lf\n", num2);  // %lf for double
    printf("num3 (long double): %Lf\n", num3); // %Lf for long double
    return 0;
}
```

Key Points:

- float is used when you need to store decimal numbers with moderate precision (about 6-7 decimal places).
- double provides higher precision, useful when you need more accuracy with decimal values.
- long double is for even higher precision, though it's rarely used.

In summary, float data types in C are used when you need to work with numbers that aren't whole (i.e., they have decimal points) and depending on how much precision you need, you can choose between float, double, or long double.

Void Data Type :

The void data type is a special type used to indicate "no type" or "no value." It is mainly used in two situations:

1. void as a function return type: When a function doesn't return any value, it is declared with void. This tells the program that the function does something, but doesn't give back a result.
2. void pointers: A void* (void pointer) is a type of pointer that can point to any type of data. It is used when the exact data type isn't known, and it can later be cast (converted) to a specific type.

Example in C :

```
#include <stdio.h>
// A function that returns nothing, so we use 'void'
void greet()
{
    printf("Hello, World!\n");
}
int main()
```

```
{
    greet();  // Calling the function, but it doesn't return anything
    return 0;
}
```

In this example, the function greet() is defined with void, meaning it performs an action (printing "Hello, World!") but does not return a value.

Detailed list of data types:

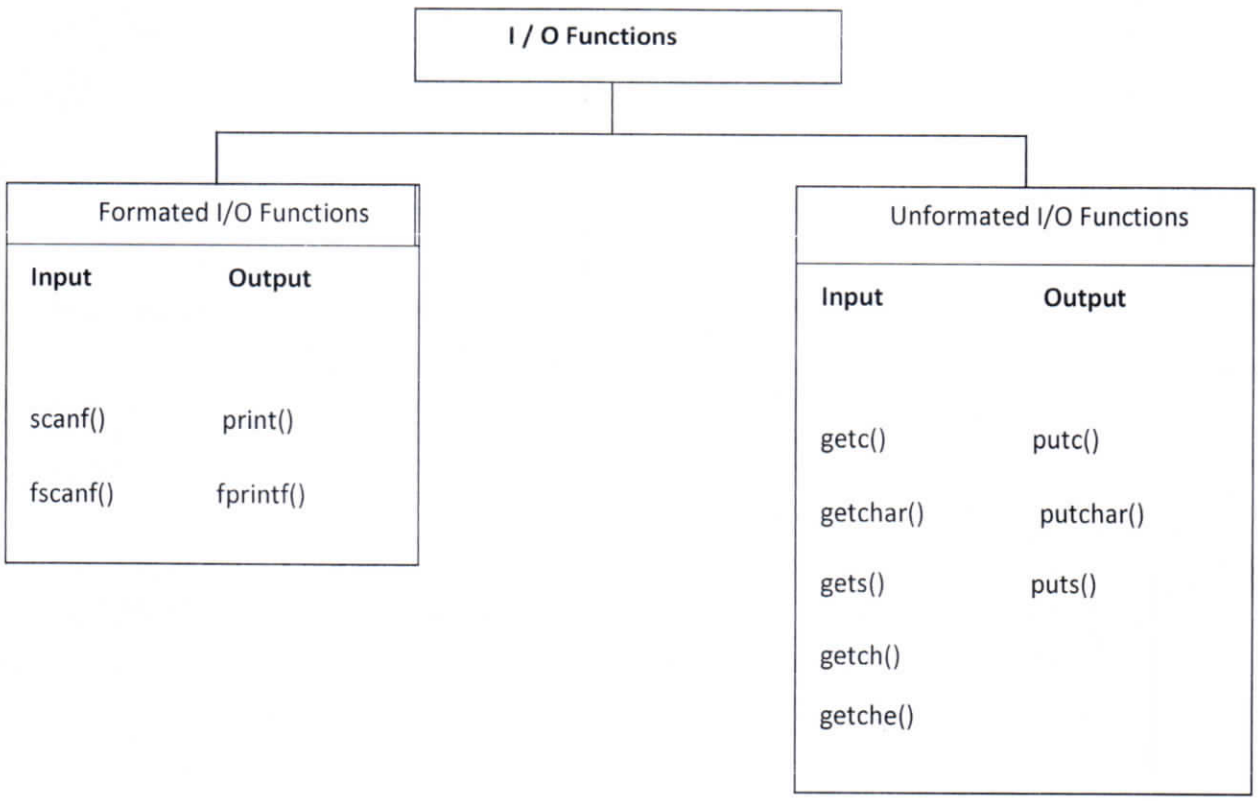
Table 8.4 Detailed list of data types

Data type	Size in bytes	Range
char	1	-128 to 127
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2	-32768 to 32767
unsigned int	2	0 to 65535
signed short int	2	-32768 to 32767
signed int	2	-32768 to 32767
short int	2	-32768 to 32767
unsigned short int	2	0 to 65535
long int	4	-2147483648 to 2147483647
unsigned long int	4	0 to 4294967295
signed long int	4	-2147483648 to 2147483647
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

Assignment questions

1) What are keywords in C? List at least 10 keywords and explain their significance.
2) Define identifiers in C. What are the rules for naming identifiers? Provide examples of valid and invalid identifiers.
3) Compare keywords and identifiers in C. How are they different?
4) What is a variable in C? How are variables declared and initialized?
5) Explain the concept of constants in C.
6) Write the difference between variables and constants.
7) Create a variable called myNum of type int and assign the value 15 to it.
8) Declare a variable of type float, char, int.
9) List the basic data types available in C and explain their purpose.

INPUT/OUTPUT STATEMENT IN C



Formated I/O Functions :

Formatted input/output functions are used to read and print data in a specific format. They allow you to control how information is displayed on the screen or read from the user, making it easier to work with different types of data, like integers, floats, and strings.

Here's a simple explanation of two of the most common formatted input/output functions:

1. **printf() – Formatted Output:**

- **printf()** is used to display (print) information on the screen.
- You can use format specifiers (like %d, %f, etc.) to tell printf() how to format different types of data.

Example:

```
#include <stdio.h>
int main()
 {
   int age = 25;
   float height = 5.9;
   printf("I am %d years old and my height is %.1f feet.\n", age, height);
   return 0;
}
```

Explanation:

- %d is a format specifier for integers (whole numbers), used to print age.
- %.1f is a format specifier for floating-point numbers, used to print height with one decimal place.
- The output will be: I am 25 years old and my height is 5.9 feet.

2. scanf() – Formatted Input:

- scanf() is used to get (input) information from the user.
- You use format specifiers (like %d, %f, etc.) to tell scanf() what type of data you expect the user to enter.

Example:

```
#include <stdio.h>
int main()
{
   int age;
   float height;
   printf("Enter your age: ");
   scanf("%d", &age);  // %d for integer input
   printf("Enter your height in feet: ");
   scanf("%f", &height);  // %f for float input
   printf("You are %d years old and %.1f feet tall.\n", age, height);
   return 0;
 }
```

Explanation:

- %d tells scanf() to expect an integer and store it in the variable age.
- %f tells scanf() to expect a floating-point number and store it in the variable height.
- The & symbol is used before variable names to give scanf() the memory address where the input should be stored.

Common Format Specifiers:

- %d – Integer (whole numbers)
- %f – Float (decimal numbers)
- %c – Character (single letters or symbols)
- %s – String (a series of characters, like a word or sentence)

Key Points:

- printf() is for printing or displaying formatted data on the screen.
- scanf() is for reading (getting) formatted input from the user.
- Both functions use format specifiers (like %d for integers or %f for floats) to handle different types of data.

In simple words, these functions help you control how information is displayed or input in your program, making it easier to work with different types of data, such as numbers or text.

Unformatted I/O functions:

Character I/O:

getchar(): Used to read a character from the standard input.

putchar(): Used to display a character to standard output.

getch() and getche(): These are used to take the any alpha numeric.

Characters from the standard input:

getche() read and display the character.

getch() only read the single character but not display.

putch(): Used to display any alpha numeric characters to standard output.

String I/O:

gets(): Used for accepting any string from the standard input(stdin) Eg: gets()

puts(): Used to display a string or character array Eg: puts()

Cgets():read a string from the console Eg: cgets(char *st)

Cputs():display the string to the console Eg: cputs(char *st)