20CS1101 –PROGRAMMING FOR PROBLEM SOLVING

(Common to all branches)

UNIT – I

INTRODUCTION: Algorithms, Flow charts, Program development steps.

FUNDAMENTALS OF C: History, Structure of a C program, Programming rules and execution. Character set, Delimiters, C keywords, Identifiers, Constants, Variables, Rules for defining Variables, Data types, Declaration and Initialization of Variables.

UNIT – II

OPERATORS AND EXPRESSIONS: Introduction, Operator Precedence and Associativity, Operator Types

INPUT AND OUTPUT IN C: Formatted and Unformatted functions, Commonly used library functions.

UNIT – III

DECISION STATEMENTS: Introduction, Types of If statements, switch statement,

break, continue, goto.

ITERATIVE STATEMENTS: while, do-while and for loops.

UNIT – IV

ARRAYS: Definitions, Initialization, Characteristics of an array, Array Categories.

STRINGS: Declaration and Initialization of strings, String handling functions.

STORAGE CLASSES: Automatic, External, Static and Register Variables.

UNIT – V

POINTERS: Fundamentals, Declaration and initialization of Pointers, Arithmetic Operations, Pointers and Arrays.

FUNCTIONS: Definition, Function Prototypes, Types of functions, Call by Value and Call by Reference, Recursion.

UNIT – VI

STRUCTURES: Definition, Declaration and Initialization of Structures.

UNIONS: Definition, Declaration and Initialization of Union.

FILES: Introduction, File Types, Basic operations on Files, File I/O, Command Line Arguments

TEXT BOOK(S):

1. Programming with ANSI & TURBO C by Ashok N.Kamthane, Pearson Education 2007

REFERENCE BOOKS:

1.A Book on C by Al Kelley/Ira Pohl, Fourth Edition, Addison-Wesley.1999

2.Let Us C byYashvantKanetkar, BPB Publications.

3.Programming in ANSI C by Balaguruswamy 6th Edition, Tata McGraw Hill Education, 2012.

20CS1101 –PROGRAMMING FOR PROBLEM SOLVING

(Common to all branches)

UNIT-I

INTRODUCTION: Algorithms, Flow charts, Program development steps.

Fundamentals of C: History, Structure of a C program, Programming rules and execution. Character set, Delimiters, C keywords, Identifiers, Constants, Variables, Rules for defining Variables, Data types, Declaration and Initialization of Variables.

→1.1.Algorithm

An algorithm is the **step-by-step logical procedure for solving a problem**. Each step is called an instruction. An algorithm can be written in English like sentences or in any standard representation. The algorithm written in **English like language is called "pseudo code**".

[Or]

An algorithm is a collection of well-defined, unambiguous and effectively computable instructions, if execute it will return the proper output.

[Or]

An algorithm is a tool for solving any computational problem. It may be defined as a sequence of finite, precise and unambiguous instructions which are applied either to perform a computation or to solve a computational problem. These instructions are applied on some raw data called the input, and the solution of the problem produced is called the output. It is shown in the diagram given below.

Properties of algorithm: According to D.E Knuth a pioneer (found) in the computer discipline an algorithm must have the following properties.

a) Input b) Output c) Definiteness d) Finiteness e) Effectiveness

a) Input: Every algorithm takes zero or more inputs. Inputs are the numeric or non-numeric quantities that are given to an algorithm.

b) Output: An algorithm produces one or more outputs. If there are no outputs, the algorithm is considered not to have solved any computational problem.

c) Definiteness: Each instruction is an algorithm should be clear and unambiguous (unclear).

d) Finiteness: The algorithm should terminate after a finite number of steps. It should not enter into an infinite loop.

e) Effectiveness: The instructions should be written step by step, which helps computer to understand the control-flow. i.e.; tracing of each step should be possible.

1.1 Three categories of statements for algorithm development:

In general, the steps in an algorithm can be divided into three basic categories as listed below.

1. Sequence: A series of steps that we perform one after the other.

2. Selection: Making a choice from multiple available options.

3. Iteration: Performing repetitive tasks.

1. Sequence

This describes a sequence of actions that a program carries out one after another, unconditionally. Execute a list of statements in order.

Consider an example, Algorithm for Addition of two numbers:

Step1: Start

Step 2: Get two numbers as input and store it in to a and b

Step 3: Add the number a & b and store it into c

Step 4: Print c

Step 5: Stop.

The above example is the algorithm to add the two numbers. It says the sequence of steps to be follow to add two numbers.As,Step2 says that should get two numbers from the user and store it in some variable. In step3, the algorithm start doing the process of adding two numbers and step4 print the result generated by step3.

2. Selection

Selection is the program construct that allows a program to choose between different actions. Choose at most one action from several alternative conditions.

Algorithm to find biggest among 2 numbers:

Step1: Start

Step 2: Get two numbers as input and store it in to a and b

Step 3: If a is greater than b then

Step 4: Print a is big

Step 5: else

Step 6: Print b is big

Step 7: Stop

.

3. Iteration (Repetition)

Repetition (loop) may be defined as a smaller program the can be executed several times in a main program. Repeat a block of statements while a condition is true. It uses structures called while, do-while , for and repeat , until.

Pseudo code to print first 10 natural numbers

Step 1: Start

Step 2: Initialize a=0

Step 3: while a<10

Step 4: print a

Step 5: end while

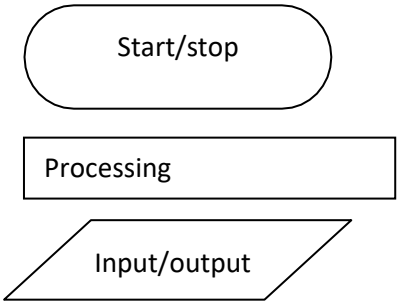
Step 6: stop

→1.2. Flowchart

A flowchart is a pictorial representation of an algorithm. It shows the flow of operations in pictorial form and any error in the logic of the problem can be detected very easily. A flow chart uses different shapes of boxes and symbols to denote different types of instructions. These symbols are connected by solid lines with arrow marks to indicate the operation flow.

Normally an algorithm is represented in the form of a flow chart and the flow chart is then expressed in some programming language to prepare a computer program.

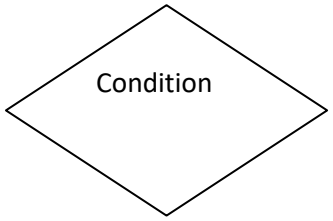
Flow chart symbols: A few symbols are needed to indicate the necessary operations in a flow chart. These symbols have been standardized by the American National Standard Institute (ANSI). These symbols are



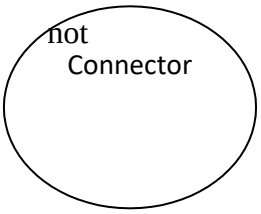
Start and stop commands are written within this symbols.

Operations are written with in these symbols.

Read and writing operations are indicated within this symbol.

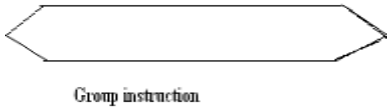


Control operations are indicated with in this symbol.

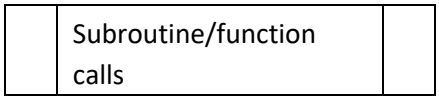


One part of the flow chart is connected to another by using this symbol when it does fit into one sheet.

Flow of control is indicated with this symbol.



A group of steps or instructions or a sub-algorithm is indicated with in this symbol.



An activity carried out as part of a function or subroutine is indicated within this symbol.

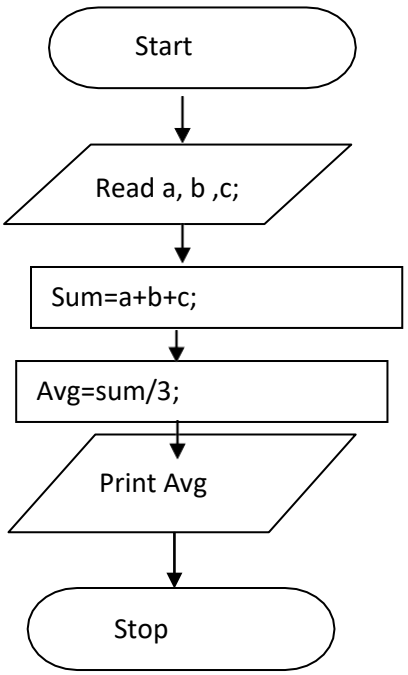
Example: Write an algorithm for finding the average of given three numbers and draw the flow chart for the same.

Sol:

Algorithm

1. Start
2. Read a b and c
3. Compute sum=a+b+c
4. Avg=sum/3;
5. Print avg
6. Stop

Flow chart:



Advantages and draw backs of flow chart:

Advantages:

1. Easy tracking of flow of control
2. Easy comprehension due to the use of loops
3. Easy perception (observation) of solution.
4. Compact (Compressed) representation.

Drawbacks: The major drawback of flow chart is that long flows, leading to fatigue and consequent errors in tracking and problem solving.

→1.3. Program development steps

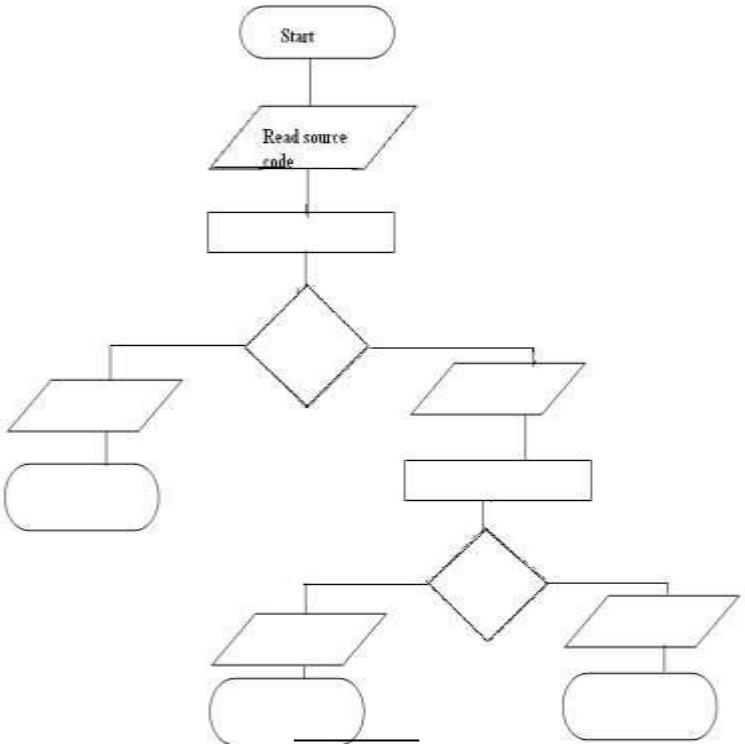


Fig: A flow chart showing the steps in compilation of a high level language.

The program development cycle is completed in four steps

1.Creating the C source code

2.Compiling the source code

3.Linking the compiled code

4. Running the executable file

1.Start.

2.Read the source program.

3.Compile the source program.

4.While compilation is there any syntax errors occurs, if YES, it display error message. If it NO then it read the object code.

5. Execute the objective program.

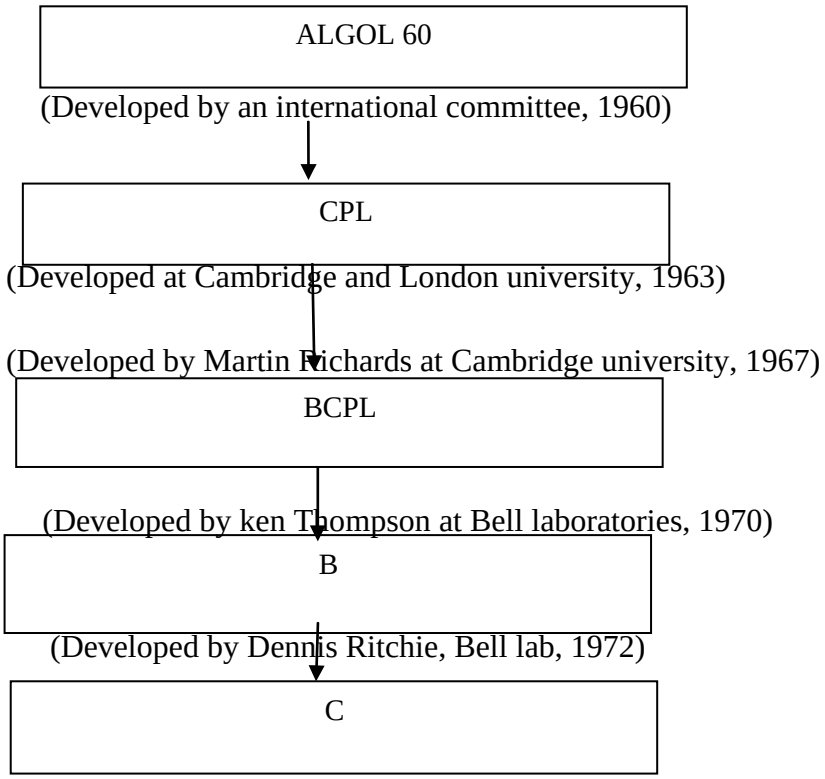
6. While execution is there any logical error, if YES, it display error message. If it NO then it print the results.

7. Stop.

→Fundamentals of C: History

How „C" evolved: The development of C language was a result of the evolution of several languages. This can be called the „ancestors" of C. Those were ALOGOL60, CPL, BCPL, and B.

In 1960 many computer languages, each for a specific purpose, were developed.



1. ALGOL 60: ALGOL 60 was a modular and structured language but it did not succeed because it was found to be too abstract (theoretical) and too general (all-purpose).

2. CPL: The Combined Programming Language was developed at Cambridge University and university of London in 1963. However it was hard to learn and difficult to implement.

3. BCPL: The basic combined programming language was very close to CPL and developed by Martin Richards at Cambridge University in 1967. BCPL was too less powerful and too specific and hence it failed.

4.B: The father of C language was the B language developed by ken Thompson of Bell Laboratories in 1970. It was first designed for UNIX. However, it was machine dependent and a type less language.

5.C: The „C" programming language was invented by Dennis Ritchie in 1972 at Bell Laboratories. The C was closely linked with UNIX for which it was developed.

Application areas

1. „C" is a general purpose programming language and not designed for specific application areas like FORTRAN (Scientific) and COBOL (Business).
2. C is well suited for business as well as scientific applications because it has various features requires for these applications. However it is better suited and widely used for system software like operating systems, compilers, interpreters etc.

Why C

There are several reasons why C is a popular programming language.

1. Flexibility: C is a general purpose language. It can be used for diverse applications. The language itself places no constraints on the programmer.

2. Powerful: It provides a variety of data types, control flow instructions for structured programs and other built in features.

3. Small size: C language provides no input/output facilities or file access. These mechanisms are provided by functions. This helps in keeping the language small. C has only 32 keywords which can be described in a small space and learned quickly.

4. Modular design: The C code has to be written in functions which can be linked with or called in other programs or applications. C also allows user defined functions to be stored in library files and linked to other programs.

5. Portability: A C program written for one computer system can be compiled and run on another with little or no modification.

6. High level structured language features: This allows the programmer to concentrate on the logic flow of the code rather than worry about the hardware instructions.

7. Low level features: C has a close relationship with the assembly language making it easier to write assembly language code in C program.

8. Bit engineering: C provides bit manipulation operators which are a great advantage over other languages.

9. Use of pointers: This provides for machine independent address arithmetic.

10. Efficiency: A program written in C has development efficiency as well as machine efficient (i.e., faster to execute).

Limitations of C language

1. It was not suitable for programming of numerical algorithms since it does not provide suitable data structures.

2. C does not perform bound checking on arrays.

3. C is not a strictly type checking language.

4. The order of evaluation of function arguments is not specified by the language.

Example:- In the function call f(i, ++i) is not defined. Whether the evaluation is left to right or right to left.

5. The order in which operators are evaluated is not specified by the language.

→Structure of a C program

A C program consists of functions, one of which is main( ). The program begins executing at main ( ). The basic structure of a C program is as shown:

<table>
<tr><td>Documentation section</td></tr>
<tr><td>Link section</td></tr>
<tr><td>Definition section</td></tr>
<tr><td>Global declaration section</td></tr>
<tr><td>Function section
main( )
{
Declaration part
Executable part
}</td></tr>
<tr><td>Sub program section
<table><tr><td>Function 1</td></tr><tr><td>Function2</td></tr><tr><td>.</td></tr><tr><td>Function n</td></tr></table></td></tr>
</table>

1. Documentation section

This section consists of comments specifies the name of the program, author and other details. These comments beginning with two characters /* and ending with the characters */. No space can be included between these pairs of characters any characters may be included in either upper case or lower case.

2. Link section

This section provides the compiler to link functions from the system library (Ex:- stdio.h , conio.h , math.h).

3. Definition section

This section defines all symbolic constants.(Ex:- # define Max 100 or #define PI 3.14).

4.Global declaration section

Some variables are used in one or more function, such variables are called global variables and are declared in this section. I.e.; outside of the all the functions.

5. Main function section

Every C program must have at least one function ie; main( ) function. This main section has two parts.

1. Declaration part

2.Executable part

The declaration part declares all the variables that are used in the executable part. There is at least one statement in the executable part. These two parts can appear between the opening and closing braces. In all C programs execution begins at this opening and closing braces and ends at this closing brace. All the declaration and executable statements end with a semi colon.

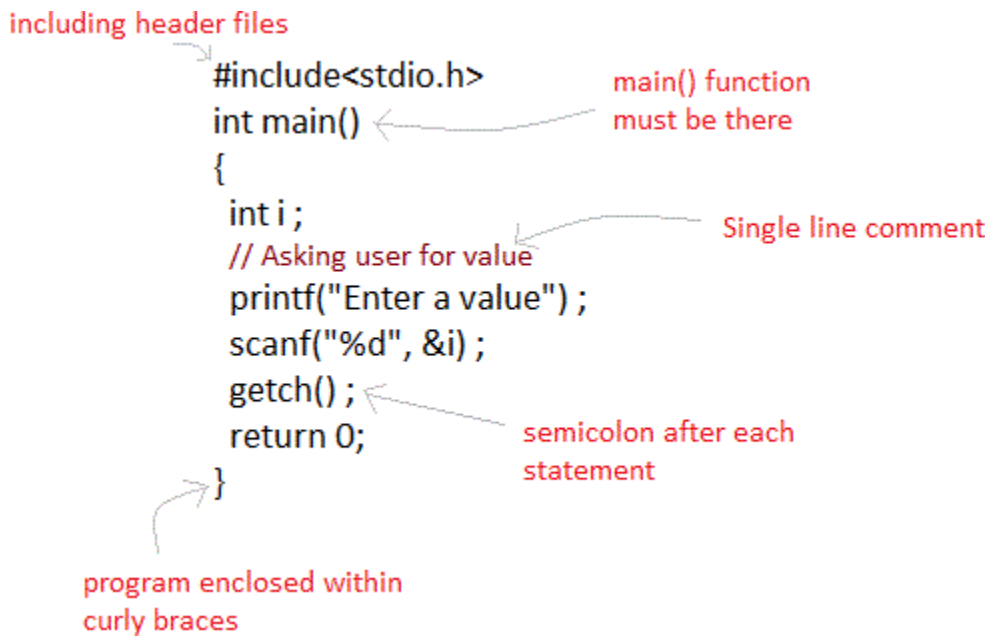
6. Subprogram section

This section contains user defined functions that are mentioned in the main function. User defined functions are generally placed immediately after the main function.

→Programming rules and execution

Some basic syntax rule for C program

1. C is a case sensitive language so all C instructions must be written in lower case letter.
2. All C statement must end with a semicolon.
3. Whitespace is used in C to describe blanks and tabs.
4. Whitespace is required between keywords and identifiers



Compile and Execute C Program

Let us see how to save the source code in a file, and how to compile and run it.

Following are the simple steps:

1. Open a text editor and add the above-mentioned code.

2. Save the file as hello.c

3. Open a command prompt and go to the directory where you have savedthe file.

4. Type gcc hello.c and press enter to compile your code.

5. If there are no errors in your code, the command prompt will take you tothe next line and would generate a.out executable file.

6. Now, type a.out to execute your program.

7. You will see the output "Hello World" printed on the screen.

$ gcc hello.c

$ ./a.out

Hello, World!

Make sure the gcc compiler is in your path and that you are running it in thedirectory containing the source file hello.c.

→Character set

The characters that can be used to form words, numbers and expressions depend upon the computer on which the programs run.

The characters in C are classified in the following categories.

1).Letters 2) digits 3) white space 4) special characters.

1. Letters: A to Z and a to z.

2. Digits: All decimal digits 0 to 9.

3. White space: Blank space, horizontal tab, vertical tab, new line and form feed.

4. Special characters: ,(comma) .(period or dot) ;(semicolon) „(Apostrophe) “ (quote marks) !(exclamation mark) | (vertical bar) \ (black slash) ~ (tilde) _ (under score) $ (dollar) ? (question mark) & (Ampersand) ^ (caret) * (asterisk) - (minus) + (plus)< (less than) > (greater than) ( ) (parenthesis) [ ] (bracket) { } (braces) % (percent)

#(hash) = (equal to) @ (at the rate).

→Delimiters

Language pattern of C uses special kind of symbols, which are called as delimiters. They are given as under.

Delimiters	**Use**
: Colon	useful for label.
; Semicolon	Terminates statements.
( ) Parenthesis	used in expression and function.
[ ] Square brackets	used for array declaration.
{ } curly brace	scope of statement.
# hash	preprocessor directive.
, comma	variable separator.

C-Tokens: The smallest individual units in a C program are called tokens.

[**Or]**

A C program is a sequence of characters that will be converted by a C compiler to object code(Machine code). The compiler first collects the characters of the program into tokens.

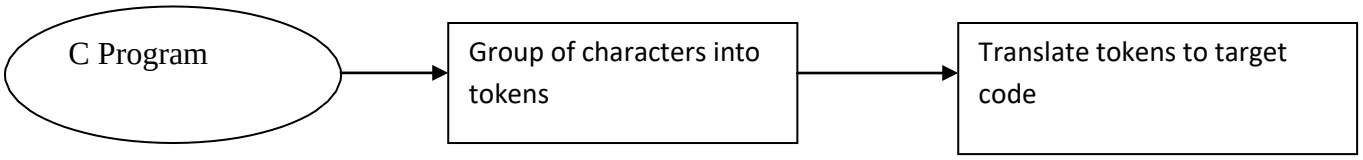


Fig: Compilation process

Consider the following C program

```
void main( void)
{
 int a, b;
a=25;
b= a+100;
printf("values of a and b are=%d%d",a ,b);
}
```

The tokens in the preceding program are illustrate as shown in fig:

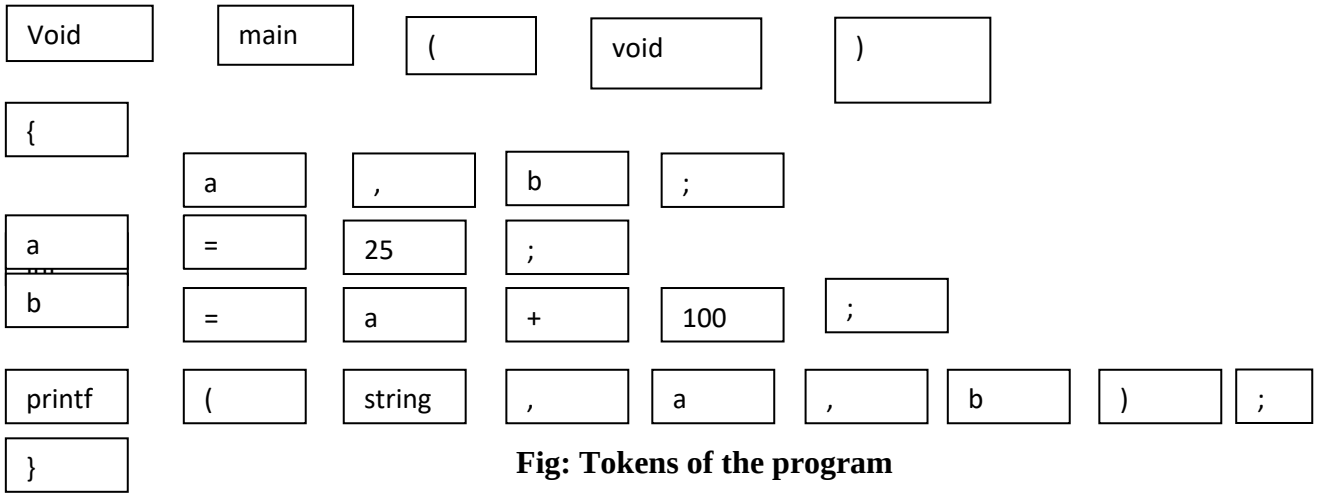
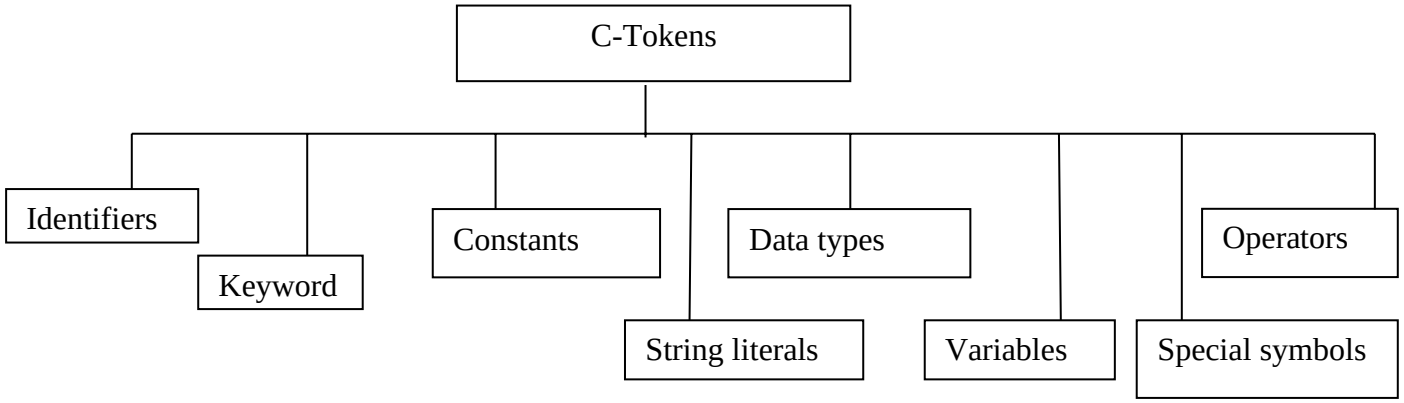


Fig: Tokens of the program



1. Identifiers: Identifiers refers to the names of variables, functions and arrays. They are user defined names, consisting of sequence of letters and digits, with the letter as the first character. Both lower case and upper case letters are also permitted.

Rules for identifiers:

1.First character must be an alphabet.

2.Must consist of only letters, digits or under score.

3.Cannot use a keyword.

4.Must not contain white space.

Example: count test23 high_balance

2. Keywords: keywords have pre-defined meaning and these meanings can not be changed. All the keywords must be written in lower case. Keywords serve as basic building blocks for program statements. ANSI C language has only 32 keywords. They are

ANSI C standard keywords

Auto	double	int	struct	Break	else	longswitch	Case	
enum	register	typedef	Char	extern	return	union		
Const	float	short	unsignedContinue	for	signedvoid			
Default	goto	sizeof	volatileDo	if	static	while		

3. Variables: A variable name is an identifier or symbolic name assigned to the memory location where data is stored. A variable can have only one value assigned to it at any given time during the execution of the program.

Rules regarding naming variables:

1. The variable name is an identifier, the same rules apply.

2. Meaningful names should be given so as to reflect value it is representing.

Syntax:

Data type variable1, variable2.....;

Example:

```
int     I, count;
float price, salary;
char  c;
```

Where variable are declared

Variables will be declared in three basic places.

1.Inside functions.

2.In the definition of function parameters

3.Outside of all functions.

These are

a)Local variables b) Global variables c) Formal variables

a) Local variables: variables that are declared inside a function are called local variables. Local variables exist only while the block of code in which they are declared is executing.

Example:

```
void  fun1(void)
 {
    int    x;        local variables
    x=10;
 }
```

b) Global variable: variables that are declared outside of all functions is called a global variables.

Example:

```
int    x;        global variables
void  fun1(void)
 {
    x=10;
 }
```

c) Formal parameters: If a function is use arguments, it must declare variables that will accept the values of the arguments. These variables are called the formal parameters of a function.

Example:

```
Void add(int y)
 {
    int y=5;
     y=y+5;
         printf ("value of y=%d", y);
       }
```

Constant and volatile variables:

Constant variable: If we want that the value of a certain variable remains the same or remains unchanged during the execution of a program then, it can be done only by declaring the variable as a constant.

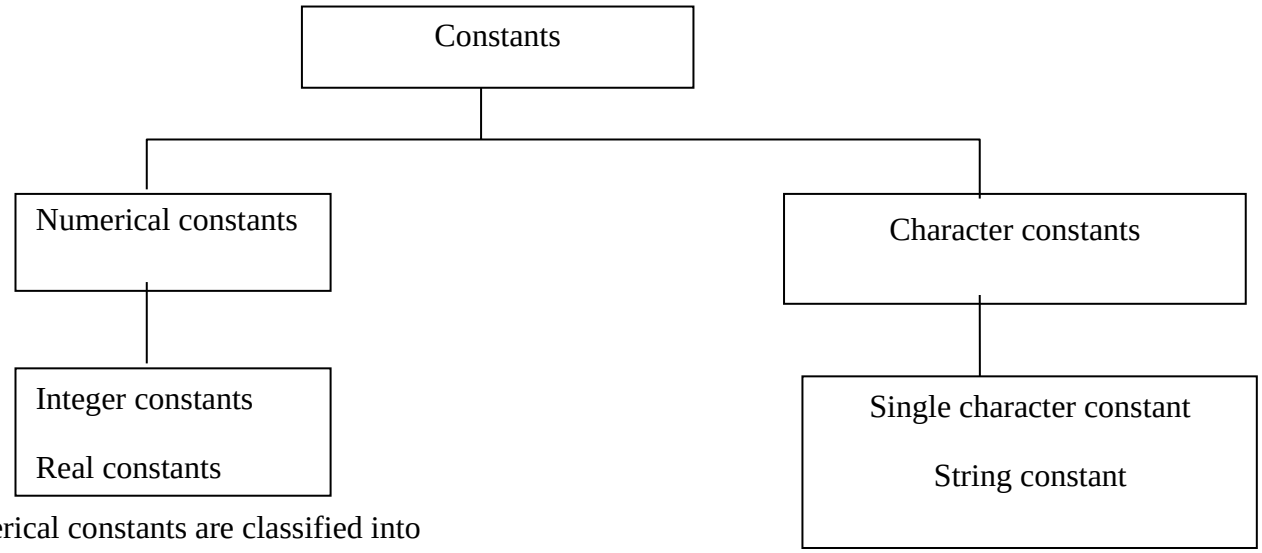
The keyword const is then added before the declaration. It tells the compiler that the variable is a constant.

Example: const int m=10;

Volatile variable: The volatile variables are those variables that are changed at any time.

Example: volatile int d;

4. Constants: Constants refer to fixed values that do not change during program execution. They can be classified as



Numerical constants are classified into

1. Integer constants 2. Real constants

1. **Integer constants:** An integer constant is a sequence of digits without a decimal point. No commas and no blank spaces are allowed. It can be either +ve or - ve. If no sign proceeds it is assumed to be +ve. It requires minimum of two bytes.

Example: 10 20 +30 -15

2. **Real constants:** Real constants often known as floating point constants. These are real numbers having a decimal point or an exponential or both.

Example: 0.246 9.345

Character constants: Character constants are classified into two types.

1.Single character constants 2. String constants

1.Single character constants: Any character written within single quotes is called character constant. Character constants have integer values known as ASCII values.

Example: „a" „8"

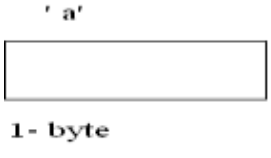
Escape sequence: C supports some special character constants used in output functions. They are also called character constants because they contain a black slash and a character. The complete set of escape sequence is

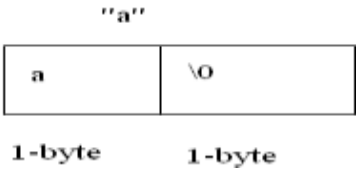
Char	Meaning
\a	alert (bell)
\b	back space
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab
\0	NULL char
\\	back slash
\ ?	question mark

2.String constants or string literals: A string constant is a sequence of zero or more characters enclosed in double quotes. String constants are also known as string literals.

Example: " welcome to hello world"

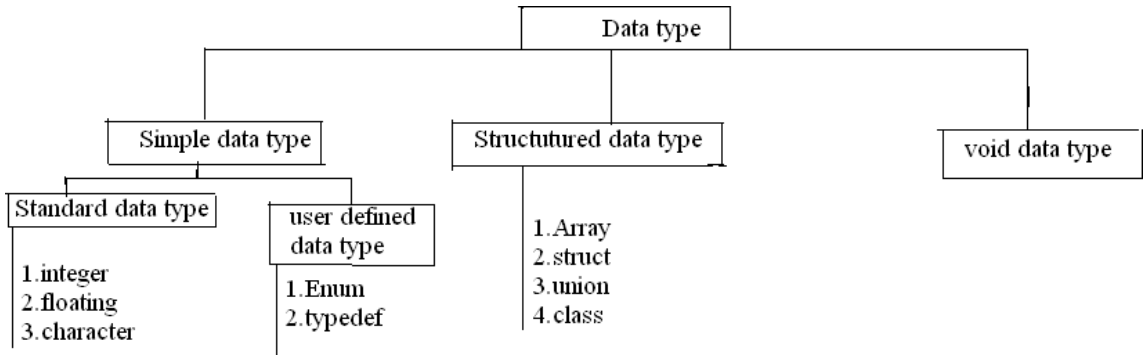
The internal representation of a string has a NULL char (\0) at the end. Therefore the physical storage required is one more than the number of characters in the string.





5.Data types in C : Name given to set of values which have common property is known as data type. [or] A data type defines a set of values and the operations that can be performed on them are called a data type.

C- Data types are classified as



1. Integer data type: Integer data types are basic data type. For example int we can assign any value which has no fractional part to int type.C offers three different integer data types they are

a) Int b) short int c) long int

The difference between these three integers is the number of bytes to occupy and the range of values.

Data type	Format string	Number of Bytes required	Range
int	%d	2	-32768 to 32767(-2^{15} to 2^{15}-1)
Short int	%d or %I	2	-32768 to 32767(-2^{15} to 2^{15}-1)
Long int	%ld	4	-2147483848 to 2147483847 (-2^{31}to2^{31}-1)
Unsigned int	%u	2	0 to 65535(0 to 2^{16}-1)
Unsigned short int	%u	2	0 to 65535(0 to 2^{16}-1)
Unsigned long int	%u	4	0 to 4294967295(o to 2^{32}-1)

2.Float data type: floating data types are the number which consists fractional part also. Like integers floats are divided into three types. They are

a)float b) double c) long double

Data type	Format string	Number of bytes	Range
Float	%f	4	3.4E-38 to 3.4E+38
Double	%lf	8	1.7E-308 to 1.7E+3.8
Long double	%lf	10	3.4E-4932 to 3.4E+4932

3.Character data type: A char data type can store an element of machine‟s character set and will occupy 1 byte. It is of two types. They are

a)Signed char b) unsigned char

Data type	Format string	Number of bytes	Range
Signed char	%c	1	-128 to +127
Unsigned char	%c	1	0 to 255

User defined data type:

1. Enumeration data type: An enumerated data type is a set of values represented by identifiers called enumeration constants.

Syntax: enum data type name { const1, const2, const3,… ..... };

Example: enum dat {mon, tue, wen, thu, fri,sat, sun };

Enum day day1, day2, day3;

Day1, day2, day3 are enumerated data types variables. We can assign any member to day1, day2, day3 variable like

Day1=mon;

Day2=tue;

Day3=wen; …….

An enumerated data type is a user defined data type which provides away for attaching names to numbers. It increasing comprehensibility of the code. This can help in making the program listing more readable. Enumerated variables are usually used to clarify the operation of a program and make the program readable.

2.Typedef: The users can define an identifier that represents an existing data type by a feature known as “typedef”.(Creating a new data type is called typedef) The user defined data type identifier can later be used to declare variables.

Syntax: typedef type identifier;

Here type refers to an existing data type and identifier refers to the new name given to the data type.

Example:

```
#include<stdio.h>
#include<conio.h>
#define H   60
void main( )
 {
   typedet int hours;
hours  hrs;
pritntf(“Enter the hours”);
scanf(“%d”,&hrs);
```

```
printf("minutes=%d",hrs*h);
printf("seconds=%d", hrs*h*h);
getch();
}
```

Void data type: void is an empty data type defined by the keyword void. It is used with functions. Void data type is used in three places.

a)Before the function b) declared the void inside the function c) both

a) Before the function: If we use before the function it does not return any value.

Example: void main( )

b)Declare the void inside the function: If we use the void inside the function, that indicates the function does not takes any argument.

Example: main(void)

c)Both a and b: If we use the void before the function and inside of the function that indicates, the function neither returns a value nor requires any argument.

Example: void main(void).

Structured data type:

The structured data types are a) Array b) Structure c) Union d) Class

a) Array: Array is a collection of elements of similar data types in which each element is located in separate memory location.

b) Structure: The struct is keyword and used to combine variables of different data types into a single record.

c) Union: A union is same as a structure. The only difference is that all the variables will share the same memory space.

d) Class: It is defined as set of data members and member functions in a single unit is called a class.