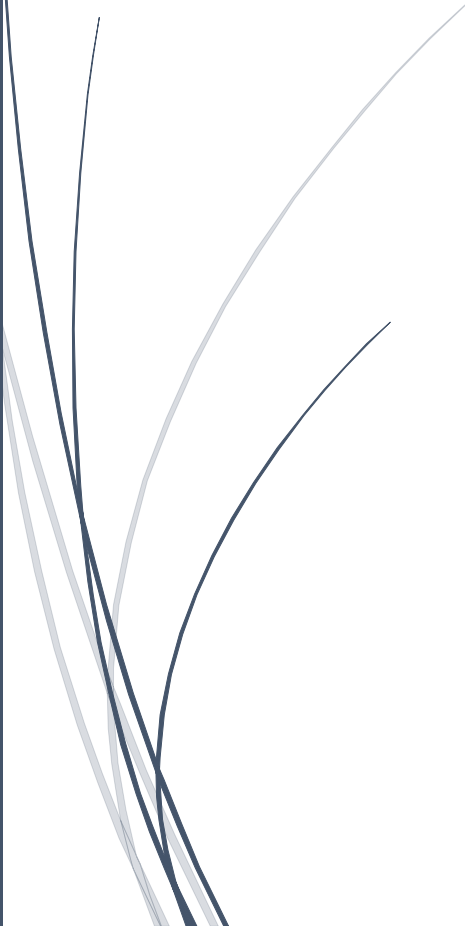


Course No. : EE207



EE207-Computer Programming

Module 1 note



Module 1 : Introduction to Programming

Syllabus :-

Introduction to Programming:

- ❖ Machine language
- ❖ assembly language
- ❖ high level language
- ❖ Compilers and assemblers
- ❖ Flow chart and algorithm
- ❖ Development of algorithms for simple problems.

Basic elements of C

- ❖ Structure of C program
 - ❖ Keywords, Identifiers
 - ❖ data types, Operators
 - ❖ expressions
 - ❖ Input and Output functions
-
- ✓ 5 Hours for this module(for 1st teaching)
 - ✓ In semester exam 15% marks from this module
 - ✓ Simple and theoretical module compared to other module
 - ✓ In this module we studying basics of C program
 - ✓ Reference ANSI C- Second edition

Introduction

C is a general purpose programming language. It has been closely associated with the UNIX system where it was developed, since both the system and most of the programs that run on it are written in C. C provides the fundamental control flow constructions required for well-structured programs: statement grouping, decision making (if-else), selecting one of possible cases (switch), looping with the termination test at the top (while, for) or at the bottom (do), and early loop exit (break).

Programming language

Computer language or programming language is a coded syntax used by computer programmer communicate with the computer. Computer languages establishes a flow of communication between programmer and computer. Programming languages can be classified into following categories

1. Machine language
2. Assemble language
3. High level language

1. Machine language

Machine language or machine code is the native language directly understood by computer's central processing unit. This type of computer language is not easy to understand, as it only uses a binary system, an element of notations containing only a series of numbers consisting of "1 and 0" to produce commands. The machine language is system independent because there are different sets of binary instructions for different types of computer systems.

Limitations of machine language:

The machine language is very tedious

Error prone process of writing programs in machine language

2. Assembly language

Assembly language is a low-level programming language in which the sequence of zeros and ones are replaced by some codes. Typical instructions for addition and subtraction

Eg:- ADD for addition

SUB for subtraction


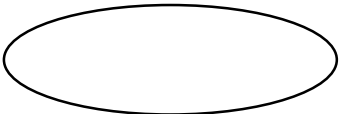
Since our system only understands this language of zeros and ones, so a system program is known as **assembler**. Which is designed to translate an assembly language program into machine language program. i.e., assembly language programs are translated into machine language by a program called assembler.

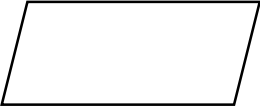

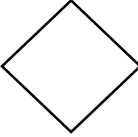

3. High level language

High-level languages are user-friendly languages which are similar to English with a vocabulary of words and symbols. These are easier to learn and require less time to write. They are problem-oriented rather than machine-based. Programs written in a high-level language can be translated into any machine language and therefore can run on any computer for which there exists an appropriate translator. Programs written in high-level languages are translated into assembly language or machine language by **compiler**.

Flow chart

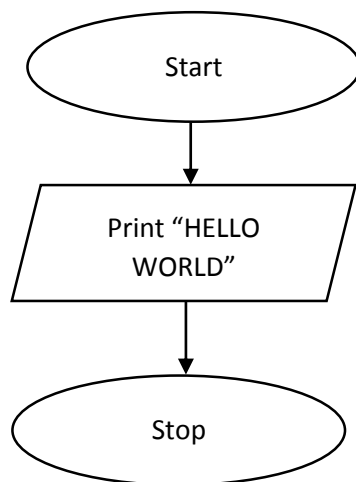
Symbols used in the flowchart :-

| | |
|-------------------------------------------------------------------------------------|---------------------------------------------|
|  | Flow lines : direction of program execution |
|  | Starting / ending of program |

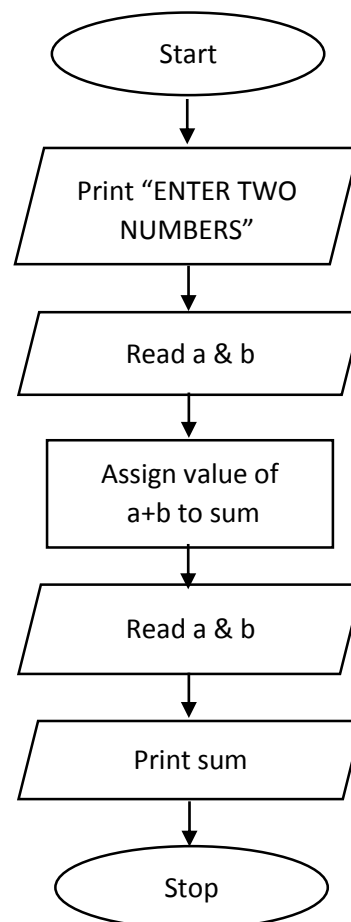
| | |
|-----------------------------------------------------------------------------------|--------------------------------------------------|
|  | Give input / Get output at this stage of program |
|  | Processing at this stage |
|  | Decision making at this stage |
|  | Connector |

Eg:-

Flow chart to print "HELLO WORLD"



Flow chart for the addition of 2 numbers



Algorithm

Algorithm is the simplest representation of a program in our own language. In this algorithm numbering each stage of the program and we also numbering the sub stages as sub numbers of the previous main stage.

Eg:-

Algorithm for addition of 2 no's

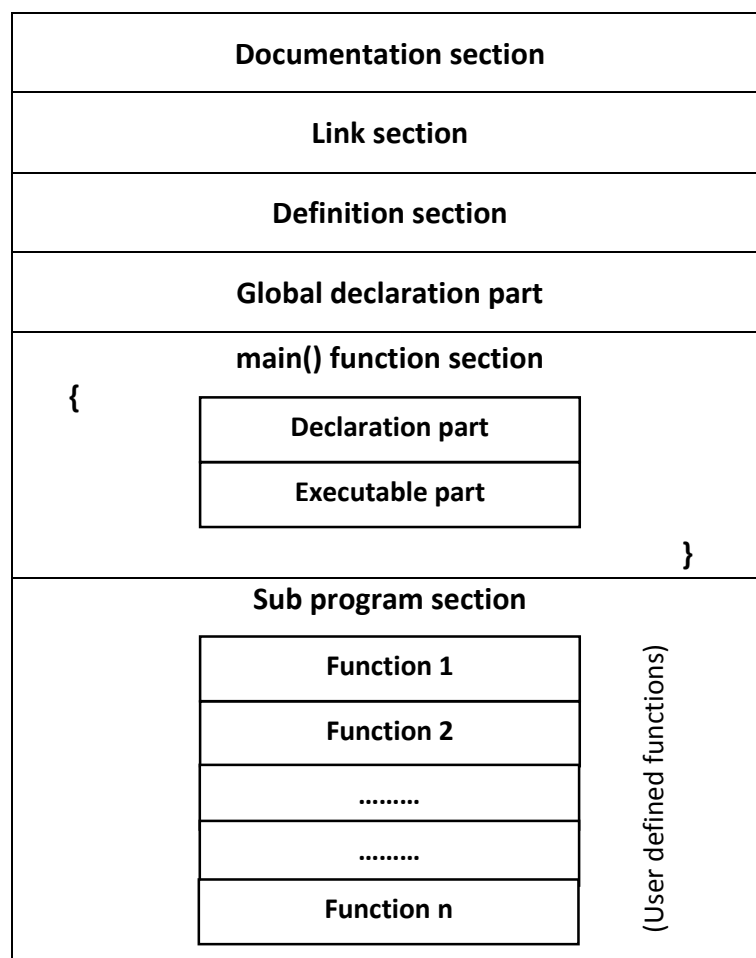
Program :-

```
#include<stdio.h>
Main()
{ int a,b,sum;
  printf("Enter 2 numbers");
  scanf("%d%d",&a,&b);
  sum=a+b;
  printf("Sum of numbers=%d",sum);
}
```

Algorithm :-

- (1) Start
- (2) Declare integer variables a,b,sum
- (3) Print "Enter 2 numbers"
- (4) Read a,b
- (5) Add a & b to sum
- (6) Print "Sum of numbers="sum
- (7) Stop

Basic structure of C program



A C program can be viewed as a group of building blocks called **functions**. A function is a sub routine that may include one or more statements designed to perform a specific task. To write C program, we first create functions and then put them together. A C program may contain one or more sections as shown in figure above

The documentation section consist 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 computer 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 function also declares all user defined functions

Every C program must have one main() function section. This section contains two parts, declaration part and execution part. The declaration part declares all the variable used in the execution part. There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. The program execution will start 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 part end with a semicolon (;)

The subprogram section contain all the user-defined functions that are called in the main function. User-defined functions 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

Character set of “C”

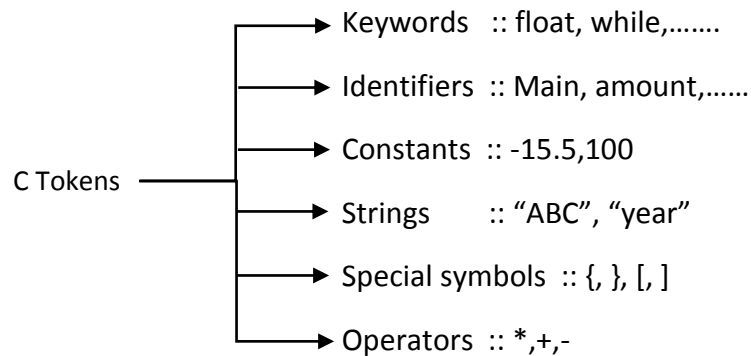
The characters in C are grouped into the following categories

- Letters :: Upper case A.....Z
Lower case a.....z
- Digits :: All decimal digits 0.....9
- Special characters :: !@#\$%^&*()_+|\",:~{}[]<>,. Etc.....
- White spaces :: Blank space, horizontal space, carriage return, new line, from feed

******The compiler ignores white spaces unless they are a part of string constant. White spaces may be used to separate word, but are prohibited between the characters of keywords and identifiers

C Tokens

The smallest individual units in a C program are known as a C tokens. C has six types of tokens as shown below:-



Keywords

Every C word is classified as either a keyword or an identifier. All keywords have fixed meaning and these meaning cannot be changed. There are 32 keywords in C. they are written to be in lower case letters

Eg:- int, void, char, float, do, if, for, while,.....

Identifier

Identifier refer to the names of variables, functions and arrays. These are user-defined names and consists of a sequence of letters and digits, with a letter as a first character. Both upper case and lower case letters are permitted. The underscore character is also permitted in identifiers to use as a link between two words in long identifiers

Rules for identifiers

- First character must be an alphabet
- Must consist of only letters, digits, or underscore
- Only first 31 characters are significant
- Cannot use a keyword
- Must not contain while space

Data types

Different data types in c program

1. Primer/Fundamental data type :: int, char, float, void
2. Derived data types :: arrays, pointers
3. User defined data types :: structures, unions

Primary data types

Data type "int" again classified into (Based on the size of data types)

Short int
int
Long int

Data type "float" again classified into (Based on the size of data types)

Float (upto 6 digits after pointer)
Double (upto 10 digits after pointer)
Long double (more than 10 digits after pointer)

Operators and Expressions

An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables. They usually form a part of the mathematical or logical expressions

C operators can be classified into a number of categories

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment & decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators

1. Arithmetic operators

The arithmetic operators are used in order to perform the arithmetic operators such as addition, subtraction, multiplication and division. It includes

+ :: Addition

- :: Subtraction

* :: Multiplication

/ :: Division

% :: Modulo division : it gives the remainder as the result

Arithmetic operators again classified

- Integer arithmetic (in this operations between integers)
- Real arithmetic (in this operations between real numbers)
- Mixed-mode arithmetic (in this operations between real numbers and integers)

2. Real operators

Relational operators can be used for the comparison of two quantities relational operators includes

< :: less than

<= :: less than or equal to

> :: greater than

>= :: greater than or equal to

== :: is equal to

!= :: is not equal to

3. Logical operators

C has logical operators :- && Logical AND

|| Logical OR

! Logical NOT

4. Assignment operators

Assignment operators are used to assign the result of an expression to a variable. The usual assignment operator is “=”. It can also be used to assign the result of an expression

Eg:- a=5 (Assigning the value 5 to a)

x=a+b+c (assigning the value of expression a+b+c to x)

5. Increment/Decrement operators

The increment and decrement operators in C are, ++ and – (Pre increment: ++c, Post increment: c++)

Eg:- a=5

b=a++ (the value 5 is assigned to b, and then a is increment to 6)

b=++a (The value of a is incremented to 6 and then 6 is assigned to b)

6. Conditional operators

exp 1 ? exp 2 : exp 3

Where exp1,exp2,exp3 are expressions. The operators ? : works as follows :- exp1 is evaluated first. If it is true, then the expression exp2 is evaluated and becomes the value of the expressions. If exp1 is false, exp3 is evaluated and it's value becomes the value of the parameters

7. Bitwise operators

Bitwise operators are used for the manipulation of data at bit level. These operators are used for testing the bits, or shifting them right or left. Bitwise operators may not be applied to float or double

& -bitwise AND

| -bitwise OR

^ -bitwise XOR

<< -shift left

>> -shift right

8. Special operators

Special operators in C includes,

- Comma operator
- Size of operator
- Pointer operators (& and *)
- Member selection operator (. and →)

Comma operator is used to link the related expressions together

Eg :- `value=(x=10,y=5,x+y);`

`m=sizeof(sum);`

Note prepared by
VISHNU CV KTU live

For more notes and question papers visit KTU live

All module Notes of Computer programming now available at our site



Now we available at WhatsApp 81-57097-880 (Send ur name reg. no & name to this to connect with us)