

C Programming

Unit-1

Introduction To Algorithms and C

Programming Languages:

The language used for communication between humans and machines is called programming language. These programming languages facilitate the users to make their programs by instructing the computer to perform different tasks. User's programs are converted to machine-oriented and the computer does the rest of the work.

Application Programs:

These programs are written by users for specific purposes.

Computer Languages

They are of three types –

1. Low-level language (Machine Language)
2. Middle-level language (Assembly language)
3. Higher-level language (User-Oriented language)

Low-Level Language

The language which is closer to machine architecture is called low-level language. To write programs in low-level language the programmer should know about the detailed internal architecture and instruction set of the computer. These are of two types:

a. Machine language or binary language

The lowest level language in which every program statement is written using a series of 0's and 1's and is directly understandable by the CPU is called machine language.

b. Assembly Language

assembly language makes use of English like words and symbols. With the help of special programs called Assembler, assembly language is converted to machine-oriented language. It is a low-level language in which mnemonics or short codes are used as instruction. It is not understood by the CPU, so needs conversion (to machine-oriented language).

High-Level Language

The language which is closer to human language is called high-level language. They are machine-independent languages and are English-like languages. These are of the following types:

a. Procedure-Oriented Language

Compiled by: Bibek Joshi

These are general-purpose languages that need step-by-step instructions to produce the result. These are easier to understand, modify, and easy to use. eg; C, C++, Java, etc.

b. Problem-Oriented Language

These are specific purpose languages that do not need step-by-step instructions to produce the result. These are also machine-independent and are easier to understand, modify, and find errors. eg; SQL, Oracle, MySQL, etc.

c. Natural Language

The language used by human beings is called natural language. This language is being designed to be used for artificial intelligence, expert systems, etc.

Introduction to C Programming:

C programming language was developed in 1972 by Dennis Ritchie at Bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A. It was initially designed for programming in the UNIX operating system. Now the software tool as well as the C compiler is written in C. Major parts of popular operating systems like Windows, UNIX, and Linux are still written in C.

Historical developments of C(Background)

Year	Language	Developed by	Remarks
1960	ALGOL	International committee	Too general, too abstract
1967	BCPL	Martin Richards at Cambridge university	Could deal with only specific problems
1970	B	Ken Thompson at AT & T	Could deal with only specific problems
1972	C	Dennis Ritche at AT & T	Lost generality of BCPL and B restored

Characteristics (Features) of C

Some of the characteristics of C are as follows:

- C is a general-purpose programming language.
- C is a system-independent programming language i.e. it can run in any computer system.
- C is a compiled language i.e. program written in C uses a compiler as a translator. So the program written in C could be easily used on other computers having a C compiler.
- It supports modular and structured programming concepts i.e. the main problem can be divided into different modules called functions.
- To store values, C provides a facility to create variables. For example, if the program needs two values from the user then we can assign those values to respective memory locations identified by variable names.
- C has a strong and powerful feature to group a number of similar data in an array.
- C has a powerful feature called branching (decision control) using which we can control the flow of program execution according to requirements i.e. C language can perform different sets of actions depending on the circumstances.
- C has another most powerful feature called looping using which a portion of a program can be repeated several times.
- C allows easy and fast manipulation of memory using pointers.
- C is simple, easy to learn and use. It provides a clear set of concepts that can be easily grasped.

Advantages of C

- It is a simple, versatile, general-purpose language and is more user-friendly.
- It is fast, efficient, easy to learn and implement.
- It is a machine-independent language.
- It is easy to debug, test, and maintain.
- It can extend itself. Users can add their own functions to the C library.
- It has 32 keywords so that easy to remember.
- Its compiler is easily available.
- It is a powerful language and use of a pointer has made it unique.

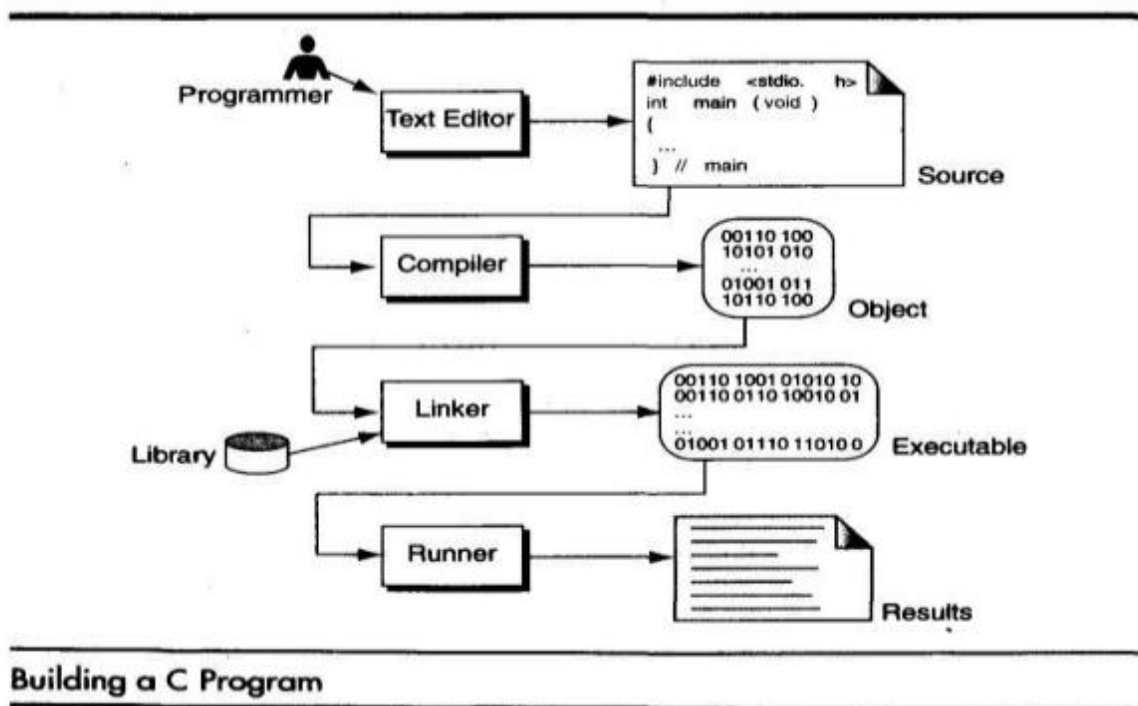
Disadvantages of C

- There is no runtime error checking in C language.
- On large programs, it is hard to fix errors.
- There are not enough library functions for handling modern programming environments.
- It is case-sensitive.

Creating and Running Programs:

Computer hardware understands a program only if it is coded in its machine language. It is the job of the programmer to write and test the program. There are four steps in this process:

1. Writing and Editing the program
2. Compiling the program
3. Linking the program with the required library modules
4. Executing the program.



Sl. No.	Phase	Name of Code	Tools	File Extension
1	TextEditor	Source Code	C Compilers Edit, Notepad Etc..,	.C
2	Compiler	Object Code	C Compiler	.OBJ
3	Linker	Executable Code	C Compiler	.EXE
4	Runner	Executable Code	C Compiler	.EXE

PSEUDOCODE:

- Pseudo code is an English-like language. That helps programmers develop algorithms.
- Pseudo code is a nonprogrammable.
- it is convenient and user friendly.

Structure of a C Program

C program follows some predefined rules called syntax. All C programs consist of one or more module called functions. One function must be present in every C program i.e. main () or void main (). The program always begins by executing the main function, which may access other functions. Although main is not defined in the keyword list, it cannot be used for naming a variable.

The #include is the first line of the program and is called preprocessor directive. It contains the header files that hold the definition of various functions. The next statement is the main () function, which is followed by opening braces „{“ which indicates beginning of function. The closing braces „}“ indicates end of function. The braces { } may contain one or more statements. The statement display information on the screen, read keyboard input, perform mathematical operations, call functions, read disk files, and, any other operations that program needs to perform. Each statement must end with a semicolon (;) and are normally written in lowercase letters.

Basic structure of C programming

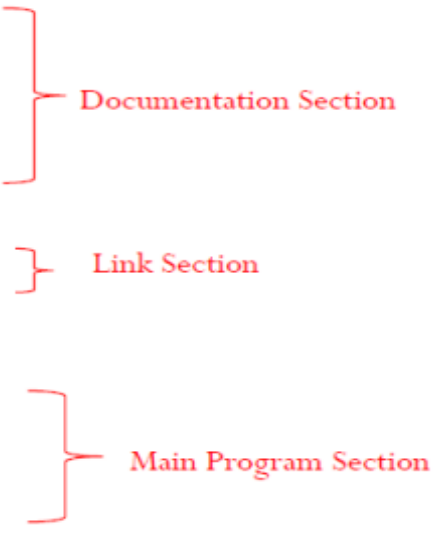
Documentation Section
Link Section
Definition Section
Global Declaration Section
<pre>main() { Main Program Section }</pre>
<pre>Subprogram Section User defined functions</pre>

Basic structure (Contd.)

- Documentation section
 - consists of comment lines giving the name of the program ,the author and other details. These comments beginning with the two Characters * and ending with the characters *\.
- Link section
 - instruct the compiler to include C preprocessors such as header files before compiling the C program
- Definition section
 - defines all symbolic constants
- Global Declaration Section
 - The variables are declared before the main () function as well as user defined functions
- main()
 - means “start here”
- Subprogram section
 - contains all the user defined functions

hello.c

```
/*  
    hello.c is first C program  
*/  
  
#include<stdio.h>  
  
void main()  
{  
    printf("Hello World");  
}
```



Documentation Section

Link Section

Main Program Section

C doesn't care much about spaces

```
/*  
    hello.c is first C program  
*/  
#include<stdio.h>  
void main()  
{  
    printf("Hello World");  
}
```

Question: To write the first c program, open the C console and write the following code:

```
#include <stdio.h>  
#include <conio.h>  
void main(){  
printf("Hello C Learners");  
getch();  
}
```

#include <stdio.h> includes the **standard input output** library functions. The printf() function is defined in stdio.h .

#include <conio.h> includes the **console input output** library functions. The getch() function is defined in conio.h file.

void main() The **main()** function is the **entry point of every program** in c language. The void keyword specifies that it returns no value.

printf() The printf() function is **used to print data** on the console.

getch() The getch() function **asks for a single character**. Until you press any key, it blocks the screen.

PROGRAMMING METHODOLOGY

A computer is used to solve a problem.

1. Analyze the problem
2. Identify the variables involved
3. Design the solution
4. Write the program
5. Enter it into a computer
6. Compile the program and correct errors
7. Correct the logical errors if any
8. Test the program with data
9. Document the program

Algorithm and Flowchart:

The algorithm and flowchart are a roadmap to the solution of the program. The algorithm is the descriptive solution to the problem whereas flowchart is the pictorial solution of the problem.

ALGORITHM —>FLOWCHART —>PSEUDOCODE—> PROGRAM

Algorithm:

An algorithm is a step-by-step procedure to complete a specific task. It is the set of rules that defines how a particular problem can be solved in a finite number of steps. It describes the working of the task. It is used in computer to design computer programs.

Characteristics (Properties) of a good algorithm

Every algorithm should possess some of the properties. All algorithms must have some characteristics. They are explained below:

- **Input/Output:** An algorithm must take any number of inputs and must produce one or more outputs.
- **Termination or finiteness:** An algorithm must terminate after a finite number of steps. It should not enter into to an infinite loop.
- **Generality:** The algorithm for a problem should work for all the values.
- **Effectiveness:** Each step in an algorithm must be carried out in a finite time to produce effective results.

- **Definiteness:** Each instruction written in the algorithm must be simple, clear, and well-defined. It should not be ambiguous.
- **Correctness:** A correct set of output values must be produced from each set of inputs.

Advantages of Algorithm

- Easy to design as each step can be written using the English language and mathematical formulas.
- Easy to understand as the English language is used and no symbols are used.
- Easy to modify and implement.
- They are not dependent on any particular programming language.
- No standard format is required.

Limitations of algorithm

- Normally writing an algorithm is fast. But writing an algorithm for large and complex problem may be time consuming.
- Sometime the algorithm may create confusion.

Example

To make a coffee

Step1: Take the proper quantity of water in a cooking pan
 Step2: Place the pan on a gas stove and light it
 Step3: Add Coffee powder when it boils
 Step4: Put out the light and add sufficient quantity of sugar and milk
 Step5: Pour into cup and have it.

Example 1: Write an algorithm to add any two numbers.

Step 1: start
 Step 2: read x and y
 Step 3: $z = x + y$
 Step 4: print "z"
 Step 5: stop.

Example 2: Write an algorithm to find area and perimeter of a rectangle and area and circumference of the circle.

Step 1: start
 Step 2: read Length (L), Breadth (B), radius (R)
 Step 3: $A1 = L * B$
 $P = 2 * (L + B)$
 $A2 = 3.14 * R * R$
 $C = 2 * 3.14 * R$
 Step 4: display the value of A1, P, A2 and C
 Step 5: stop

Example 3: Finding the average of three numbers

1. Start
2. Let a,b,c are three integers
3. Let d is float
4. Display the message "Enter any three integers:"
5. Read three integers and stores in a,b,c
6. Compute the $d = (a+b+c)/3.0$

7. Display "The avg is:", d
8. End.

Example 4: Write an algorithm for swapping contents of two variables.

Algorithm:

1. Start
2. Read value of A
3. Read value of B
4. $C=A$
5. $A=B$
6. $B=C$
7. Print A
8. Print B
9. End

Example 5: Algorithm To Find how many Positive and Negative numbers:

1. Start
2. Input a number from the user.
3. If number is less than zero, then it is a negative integer, count the negative numbers.
4. Else if number is greater than zero, then it is a positive integer, count the positive numbers.
5. End.

Example 6: Write an algorithm to swap two numbers without using third variables.

STEP 1: START

STEP 2: ENTER x, y

STEP 3: PRINT x, y

STEP 4: $x = x + y$

STEP 5: $y = x - y$

STEP 6: $x = x - y$

STEP 7: PRINT x, y

STEP 8: END

Example 7: Write an algorithm to check whether the entered number is prime or not

Algorithm:

1. Start
2. Read the number N
3. start the loop from $i= 2$ to $i=N-1$
4. If N is divided by "i" in the loop then break the loop

5. If($i=N$) # that means loop has executed till the end of range so N is not divided by any number So it is a prime number
6. Else it is not a prime number
7. End

or

Problem: Compare two number and display larger among two

- 1) START
- 2) Read first number :X
- 3) Read second number :Y
- 4) Is $X > Y$
 If yes, then display X
 If no, then display Y
- 5) STOP

Problem: Find if a number is even or odd

- 1) START
- 2) Read first number :X
- 3) Is X divisible by 2
 If yes, then output "X is even"
 If no, then output "X is odd"
- 4) STOP









Note: Another algorithm is solved accordingly

Flowchart:

A flowchart is a diagrammatic representation of an algorithm. The flowchart is very helpful in writing programs and explaining programs to others.

Symbols Used In Flowchart

Different symbols are used for different states in flowchart, For example: Input/Output and decision making has different symbols. The table below describes all the symbols that are used in making flowchart

Symbol	Purpose	Description
	Flow line	Used to indicate the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Used to represent start and end of flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for airthmetic operations and data-manipulations.
	Desicion	Used to represent the operation in which there are two alternatives, true and false.
	On-page Connector	Used to join different flowline
	Off-page Connector	Used to connect flowchart portion on different page.
	Predefined Process/Function	Used to represent a group of statements performing one processing task.

➤ **Advantages of Flowcharts**

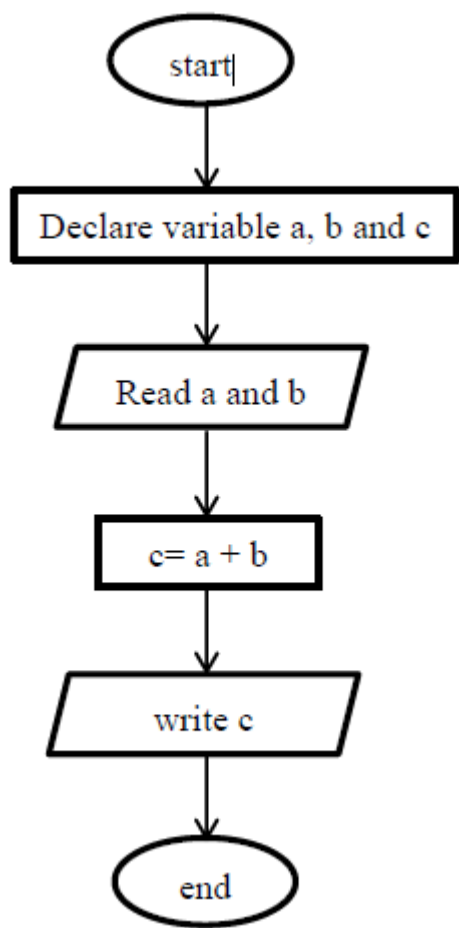
- A flowchart is also easy to understand.
- It acts as a good communication tool.
- They are unambiguous as there can be only one direction of flow at one time.
- It makes easy in error detection and correction.
- It is also independent of any programming language.

➤ **Limitations of flowcharts:**

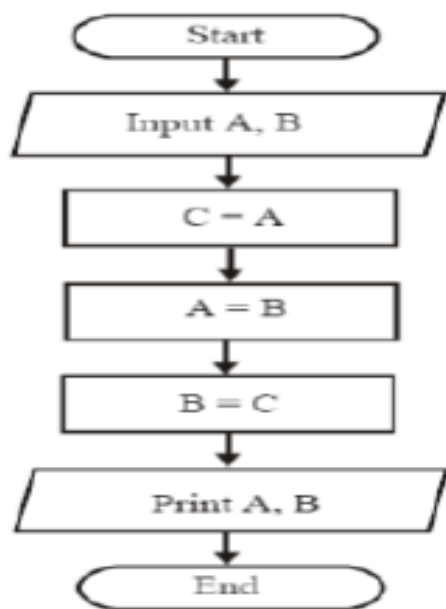
- It can be misunderstood if the proper symbol is not used.
- Drawing a flowchart is a time-consuming task.

- Even for small task modification, the flowchart may require re-drawing completely. So, it is difficult to modify.
- All the logic of very large program cannot be represented by flowchart.

Example: Draw a flowchart to add any two numbers entered by user.

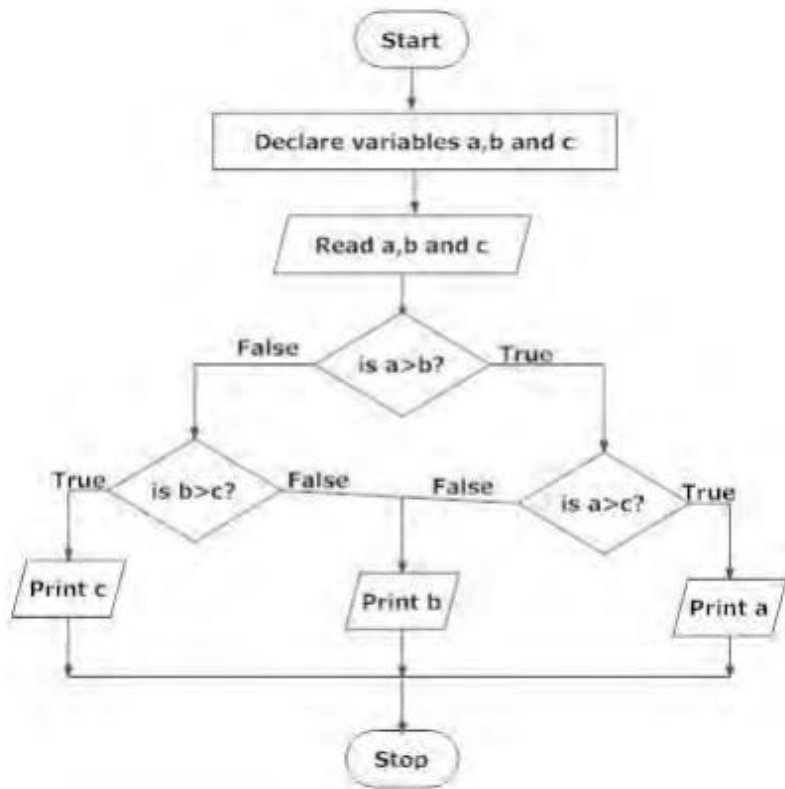


Example 2: Draw a flowchart for swapping two variables

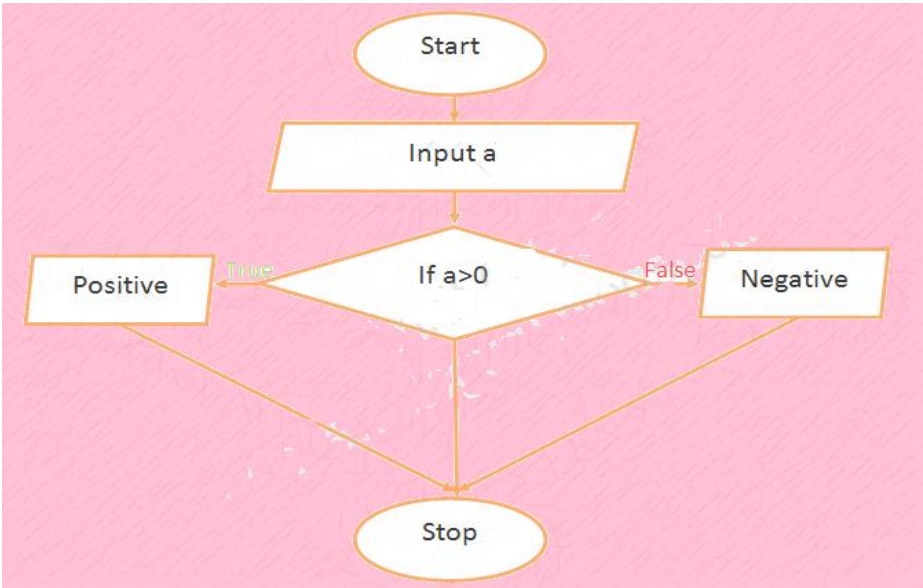


Example 3:

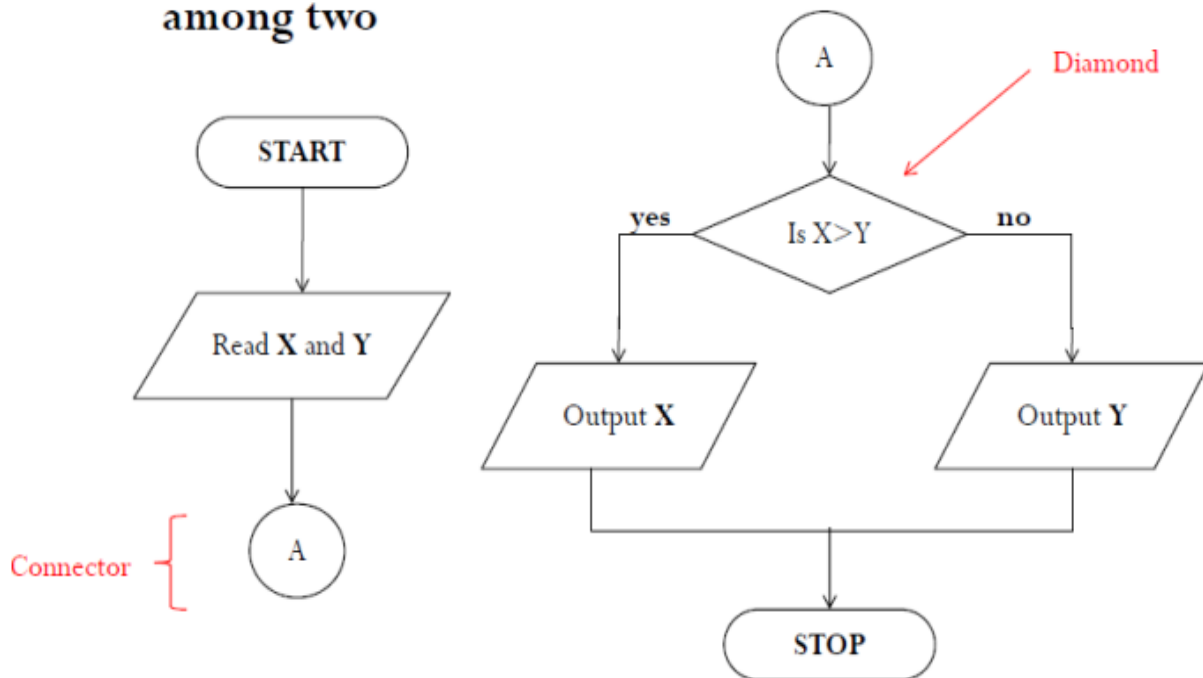
Draw flowchart to find the largest among three different numbers entered by user.



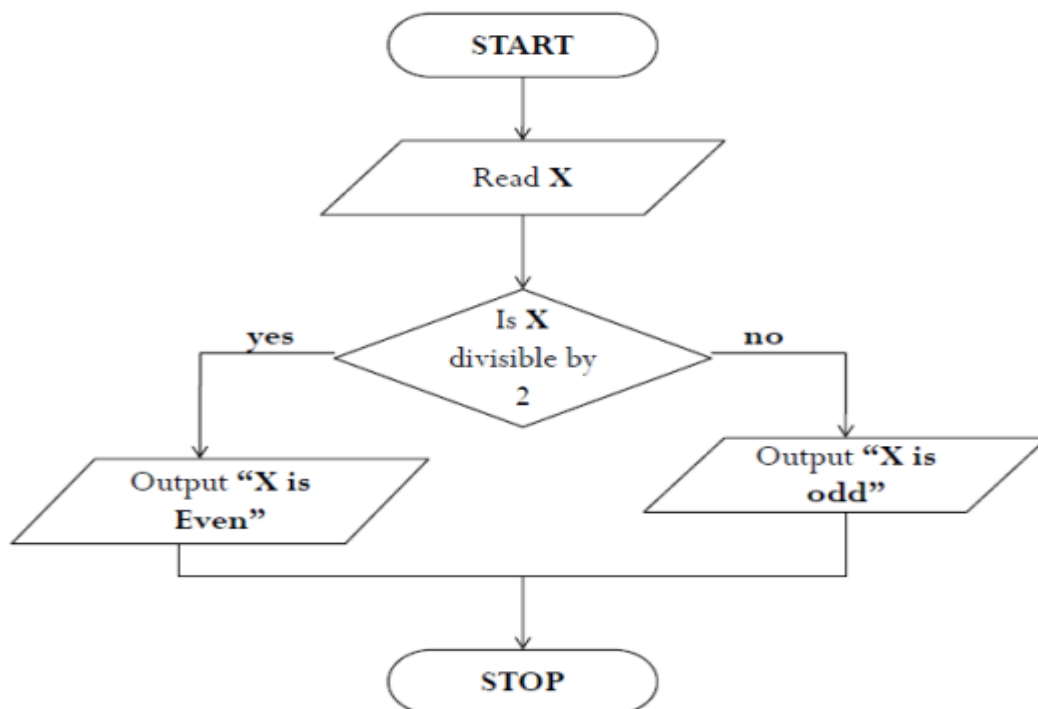
Example 4:



Problem: Compare two number and display larger among two



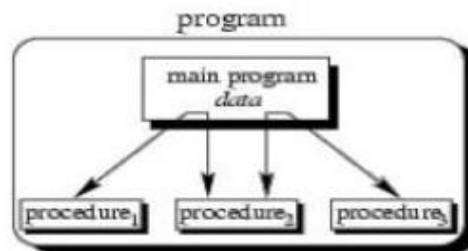
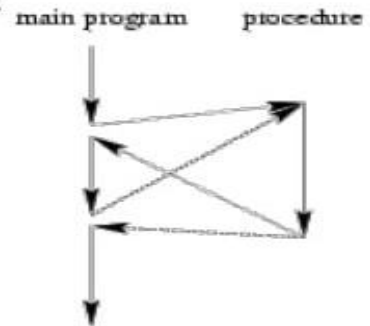
Problem: Find if a number is even or odd



Procedural approach:

- **Procedural Programming:**

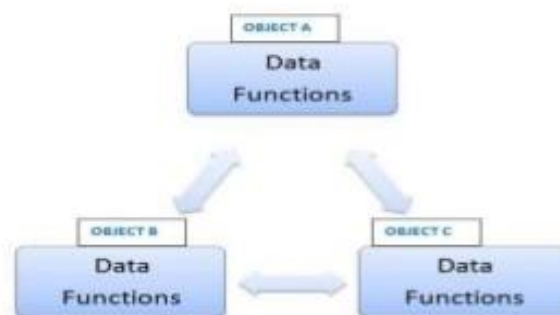
- Combine returning sequences of statements into one single place
- A **procedure call** is used to invoke the procedure
- After the sequence is processed, flow of control proceeds right after the position where the call was made



- Now we have a single program which is divided into small pieces called procedures
- To enable usage of general procedures or groups of procedures also in other programs, they must be separately available
- For that reason, modular programming allows grouping of procedures into **modules**

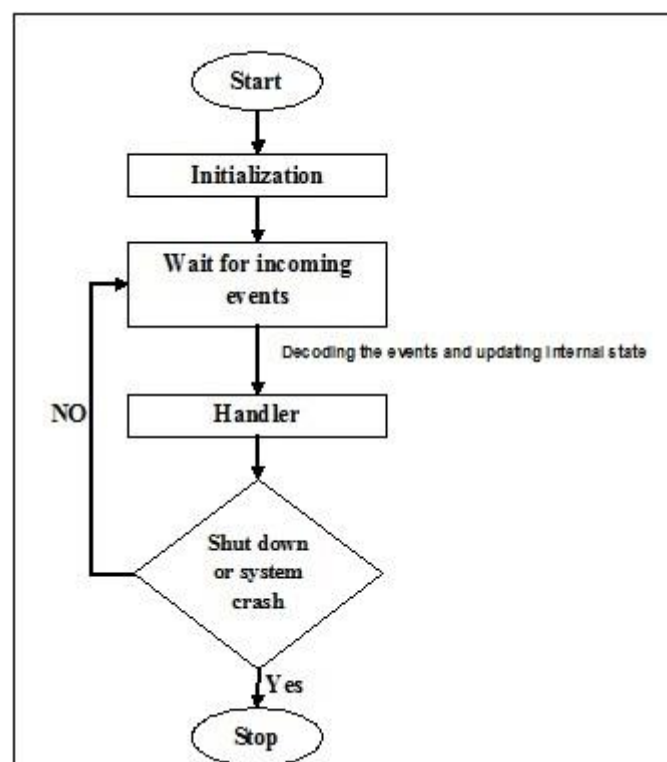
- **Object Oriented Programming(OOP) Paradigm:**

- In the OOP approach, data and the functions, which are supposed to have the access to the data, are packed together into a single unit known as an object
- An objected-oriented program may thus consists of number of objects .
- Each Objects of the program is capable of receiving message, processing data , and sending messages to other objects thus, can be viewed as an independent machine with a distinct role and responsibility.



Event Driven Approach

Event-driven programming focuses on events. Eventually, the flow of the program depends upon events. Until now, we were dealing with either sequential or parallel execution models but the model having the concept of event-driven programming is called the asynchronous model. Event-driven programming depends upon an event loop that is always listening for new incoming events. The working of event-driven programming is dependent upon events. Once an event loops, then events decide what to execute and in what order. Following flowchart will help you understand how this works –



Character set of C

Character set is a package of valid characters recognized by compiler/interpreter. Each natural language has its own alphabet and characters set as the basic building. We combine different alphabets to make a word, sentence, paragraph and a meaningful passage. Similarly each computer languages have their own character set. Particularly, C also has its own character set, shows in following table:

Uppercase alphabets	A, B, C..... Z
Lowercase alphabets	a, b, c.....z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	+ - * / ! ? \ < > & , ; # % * () [] . _ \$ ^
White space characters	Blank space, new line, horizontal tab, vertical tab, etc

Keywords

Keywords are the predefined reserved (built-in) words whose meaning has already been defined to the c compiler. A programmer can ‘t uses the keywords for other task than the task for which it has been defined. For example: a programmer can ‘t uses the keyword for variable names.

There are 32 keywords in c, which are listed in figure:

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

SYMBOLIC CONSTANTS

A symbolic constant is a name that substitutes for a sequence of characters, which represent a numeric, character or string constant. A symbolic constant is defined in the beginning of a program by using #define, without: at the end.

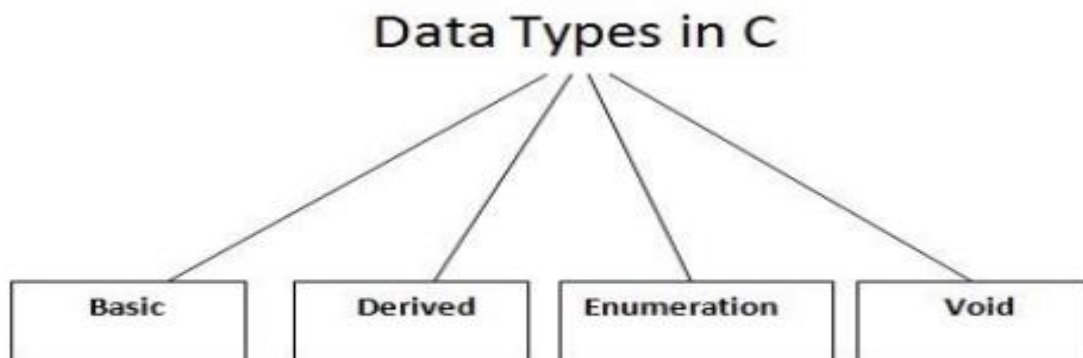
Example :

```
#define pi 3.1459
#define INTEREST P*N*R/100
```

With this definition it is a program the values of p, n ,r are assigned the value of INTEREST is computed.
Note : symbolic constants are not necessary in a C program.

Data Types/Types:

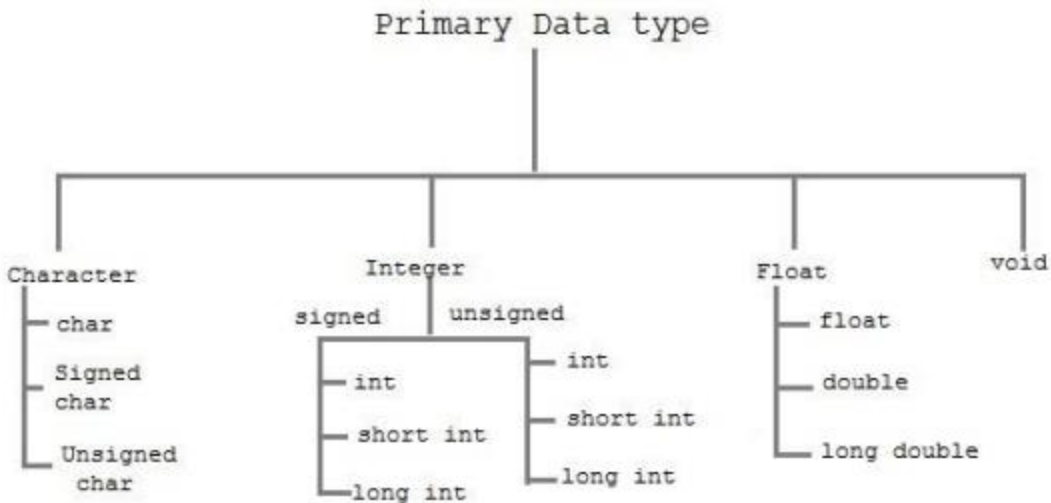
To store data the program must reserve space which is done using datatype. A datatype is a keyword/predefined instruction used for allocating memory for data. A data type specifies the type of data that a variable can store such as integer, floating, character etc. It used for declaring/defining variables or functions of different types before to use in a program.



There are 4 types of data types in C language.

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	void

Note: We call Basic or Primary data type.



The basic data types are integer-based and floating-point-based. C language supports both signed and unsigned literals. The memory size of basic data types may change according to 32 or 64 bit operating system. Let's see the basic data types. Its size is given according to 32 bit architecture.

There are 2 types of qualifiers

Sign qualifier- signed & unsigned

Size qualifier- short & long

Size and Ranges of Data Types with Type Qualifiers

Type	Size (bytes)	Range	Control String
char or signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
int or signed int	2	-32768 to 32767	%d or %i
unsigned int	2	0 to 65535	%u
short int or signed short int	1	-128 to 127	%d or %i
unsigned short int	1	0 to 255	%d or %i
long int or signed long int	4	-2147483648 to 2147483647	%ld
unsigned long int	4	0 to 4294967295	%lu
float	4	3.4E-38 to 3.4E+38	%f or %g
double	8	1.7E-308 to 1.7E+308	%lf
long double	10	3.4E-4932 to 1.1E+4932	%Lf

Derived (or Secondary) Data Types

The secondary (derived) data types are a collection of primary (primitive) data types. These are derived from the primary data types. Example: array, union, structure, etc

User Defined Data Types

The data types defined by the user is known as user defined data types. Example: enum(enumerated data type) and typedef (type definition data type).

Identifiers

Identifiers are the user defined names and consists of a sequence of letters and digits with letter or underscore as a first character. It may be name of the variables, functions, arrays, etc. In identifiers we can use both upper and lower cases.

Rules for Identifiers

- First characters must be an alphabet or underscore.
- Must consist of only alphabets, digits or underscore.
- Cannot use a keyword.
- Must not contain white spaces. To connect two words underscore can be used.

Delimiters

Delimiters are small system components that separate the various elements (variables, constants, statements) of a program. They are also known as separators. Some of the delimiters are:

- Comma: It is used to separate two or more variables, or constants.
- Semicolon: It is used to indicate the end of a statement.
- Apostrophes (single quote): It is used to indicate character constant.
- Double quotes: It is used to indicate a string.
- White space: space, tab, new line etc.

Escape Characters/ Escape Sequences

An escape sequence is used to express non printing character like a new line, tab etc. it begin with the backslash (\) followed by letter like a, n, b, t, v, r, etc. the commonly used escape sequence are

\a : for alert

\n : new line

\0 : null

\b : backspace

\f : form feed

\? : question mark

\f : horizontal tab

\r : carriage return

\' : single quote

\v : vertical tab

\” : quotation mark

Variables

A variable is a data name that may be used to store a data value. Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during execution. Constant supplied by the user is stored in block of memory by the help of variable.

Consider an expression: $2*x + 3*y = 5$. In this expression the quantities x and y are variable. A variable is a named location in memory that is used to hold certain values.

Rules for declaring variable name

1. Every variable name in C must start with a letter or underscore.
2. A valid variable name should not contain commas or blanks.
3. Valid variable name should not be keyword.
4. A variable name must be declared before using it.

Declaring variable

After defining suitable variable name, we must declare them before used in our program. The declaration deals with two things:

- It tells the compiler what the variable name is.
- It specifies what type of data the variable can hold.

Syntax:

Data_type variable_names;

For example:

```
int account_number;  
float a, b, c;  
char name[10];
```

Constants

Constant in C refers to fixed values that do not change during the execution of a program. C supports several types of constants, such as numeric constants and character constants.

Numeric constants:

It consists of:

Integer Constants:

It refers to the sequence of digits with no decimal points. The three kinds of integer constants are:

- **Decimals:** Decimal numbers are set of digits from 0 to 9 with leading +ve or -ve sign. For example: 121, -512 etc.
- **Octals:** Octal numbers are any combination of digits from set 0 to 7 with leading 0. For example: 0121, 0375 etc.
- **Hexadecimal:** Hexadecimal numbers are the sequence of digits 0-9 and alphabets a-f (A-F) with preceding 0X or 0x. For example: 0xAB23, 0X787 etc.

5.1.2 Real or Floating-Point Constants:

They are the numeric constants with decimal points. The real constants are further categorized as:

- **Fractional Real constant:** They are the set of digits from 0 to 9 with decimal points. For example: 394.7867, 0.78676 etc.
- **Exponential Real constants:** In the exponential form, the constants are represented in two forms: Mantissa E exponent; Where, the Mantissa is either integer or real constant but the exponent is always in integer form. For example: $21565.32 = 2.156532 \text{ E}4$, where $\text{E}4 = 10^4$.

Character constant: Character constant is the set of alphabets. Every character has some integer value known as the American Standard Code for Information Interchange (ASCII). The character constants are further categorized as:

Single character constant:

It contains a single character enclosed with in a pair of single quote mark (''). Thus, 'X', '5' are characters but X, 5 are not.

String constant:

String constants are a sequence of characters enclosed in double quotes marks (" "). They may be letters, numbers, special symbols or blank spaces. For example: "Hello", "X+Y". Note: 'x' \neq "x".

Characters ASCII Values

A - Z (65 – 90)

a - z (97 – 122)

0 - 9 (48 – 57)

COMPLIATION & EXECUTION OF A C PROGRAM

