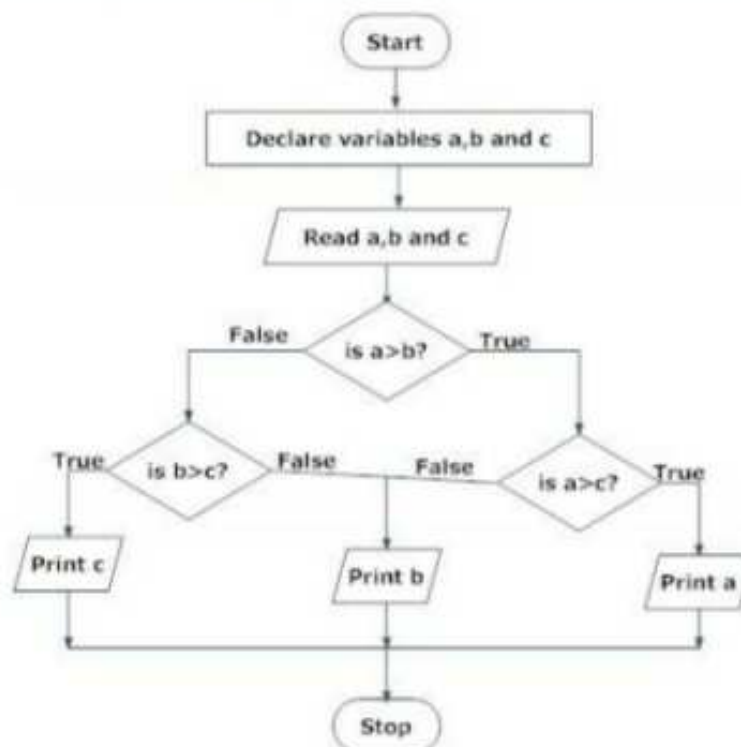


Draw flowchart to find the largest among three different numbers entered by user.



INTRODUCTION TO C LANGUAGE

C is a general-purpose high level language that was originally developed by Dennis Ritchie for the Unix operating system. It was first implemented on the Digital Equipment Corporation PDP-11 computer in 1972.

All the C programs are written into text files with extension ".c" for example *hello.c*. You can use "vi" editor to write your C program into a file.

HISTORY TO C LANGUAGE

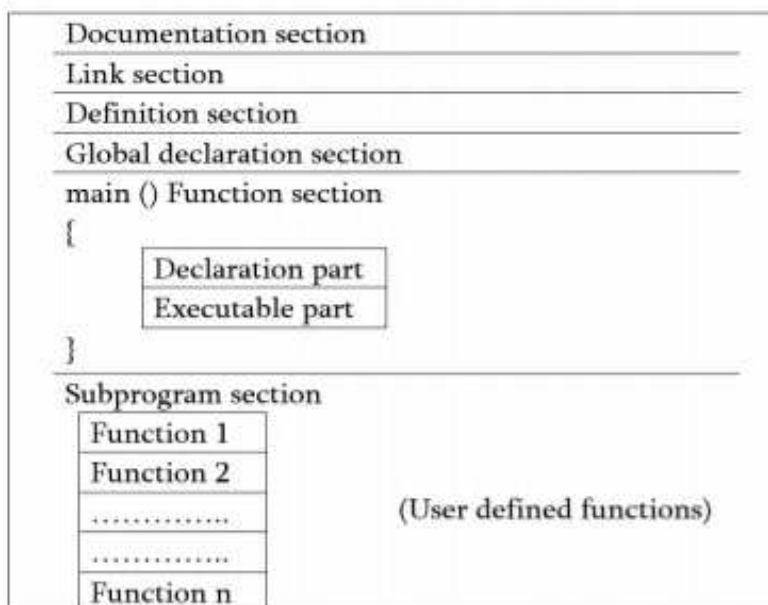
C is a general-purpose language which has been closely associated with the **UNIX** operating system for which it was developed - since the system and most of the programs that run it are written in C.

Many of the important ideas of C stem from the language **BCPL**, developed by Martin Richards. The influence of BCPL on C proceeded indirectly through the language **B**, which was written by Ken Thompson in 1970 at Bell Labs, for the first UNIX system on a **DEC** PDP-7. **BCPL** and **B** are "type less" languages whereas C provides a variety of data types.

In 1972 Dennis Ritchie at Bell Labs writes C and in 1978 the publication of The C Programming Language by Kernighan & Ritchie caused a revolution in the computing world.

In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI standard, or "ANSI C", was completed late 1988.

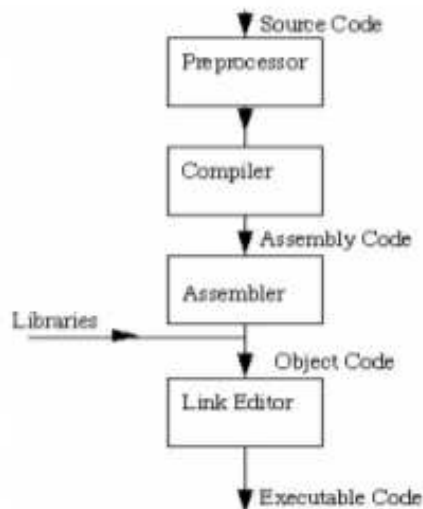
BASIC STRUCTURE OF C PROGRAMMING



1. **Documentation section:** 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.
2. **Link section:** The link section provides instructions to the compiler to link functions from the system library such as using the #include directive.
3. **Definition section:** The definition section defines all symbolic constants such using the #define directive.
4. **Global declaration section:** There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user-defined functions.
5. **main () function section:** Every C program must have one main function section. This section contains two parts; declaration part and executable part
 1. **Declaration part:** The declaration part declares all the variables used in the executable part.
 2. **Executable 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 begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a semicolon.
6. **Subprogram section:** If the program is a multi-function program then the subprogram section contains 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.

PROCESS OF COMPILING AND RUNNING C PROGRAM

We will briefly highlight key features of the C Compilation model here.



The C Compilation Model

The Preprocessor

The Preprocessor accepts source code as input and is responsible for

- removing comments
- Interpreting special *preprocessor directives* denoted by #.

For example

- `#include` -- includes contents of a named file. Files usually called *header* files. *e.g*
 - `#include <math.h>` -- standard library maths file.
 - `#include <stdio.h>` -- standard library I/O file
- `#define` -- defines a symbolic name or constant. Macro substitution.
 - `#define MAX_ARRAY_SIZE 100`

C Compiler

The C compiler translates source to assembly code. The source code is received from the preprocessor.

Assembler

The assembler creates object code. On a UNIX system you may see files with a .o suffix (.OBJ on MSDOS) to indicate object code files.

Link Editor

If a source file references library functions or functions defined in other source files the *link editor* combines these functions (with main()) to create an executable file.

C TOKENS

C tokens are the basic building blocks in C language which are constructed together to write a C program.

Each and every smallest individual unit in a C program is known as C tokens.

C tokens are of six types. They are

Keywords	(eg: int, while),
Identifiers	(eg: main, total),
Constants	(eg: 10, 20),
Strings	(eg: "total", "hello"),
Special symbols	(eg: (), {}),
Operators	(eg: +, /, -, *)

C KEYWORDS

C keywords are the words that convey a special meaning to the c compiler. The keywords cannot be used as variable names.

The list of C keywords is given below:

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

C IDENTIFIERS

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

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

They must begin with a letter or underscore (_).

They must consist of only letters, digits, or underscore. No other special character is allowed.

It should not be a keyword.

It must not contain white space.

It should be up to 31 characters long as only first 31 characters are significant.

Some examples of c identifiers:

Name	Remark
_A9	Valid
Temp.var	Invalid as it contains special character other than the underscore
void	Invalid as it is a keyword

C CONSTANTS

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

Integer Constants

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

There are three types of integer constants:

Decimal Integer Constants

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

Example of valid decimal integer constants

341, -341, 0, 8972

Octal Integer Constants

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

Example of valid octal integer constants

010, 0424, 0, 0540

Hexadecimal Integer Constants

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

Example of valid hexadecimal integer constants

0xD, 0X8d, 0X, 0xbD

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

Real Constants

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

Example of valid real constants in fractional form or decimal notation

0.05, -0.905, 562.05, 0.015

Representing a real constant in exponent form

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

mantissa e exponent

The mantissa must be either an integer or a real number expressed in decimal notation.

The letter e separating the mantissa and the exponent can also be written in uppercase i.e. E

And, the exponent must be an integer.

Examples of valid real constants in exponent form are:

252E85, 0.15E-10, -3e+8

Character Constants

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

Examples of valid character constants

'a', 'Z', '5'

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

Escape Characters/ Escape Sequences

C allows us to have certain non graphic characters in character constants. Non graphic characters are those characters that cannot be typed directly from keyboard, for example, tabs, carriage return, etc.

These non graphic characters can be represented by using escape sequences represented by a backslash() followed by one or more characters.

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

Escape Sequence	Description
a	Audible alert(bell)
b	Backspace
f	Form feed
n	New line
r	Carriage return
t	Horizontal tab
v	Vertical tab
\	Backslash
"	Double quotation mark
'	Single quotation mark
?	Question mark
	Null

STRING CONSTANTS

String constants are sequence of characters enclosed within double quotes. For example,

"hello"

"abc"

"hello911"

Every sting constant is automatically terminated with a special character "" called the **null character** which represents the end of the string.

For example, "hello" will represent "hello" in the memory.

Thus, the size of the string is the total number of characters plus one for the null character.

Special Symbols

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

`[] () {} , ; : * ... = #`

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

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

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

VARIABLES

A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C is case-sensitive. Based on the basic types explained in the previous chapter, there will be the following basic variable types –

Type	Description
char	Typically a single octet(one byte). This is an integer type.
int	The most natural size of integer for the machine.
float	A single-precision floating point value.
double	A double-precision floating point value.
void	Represents the absence of type.

C programming language also allows defining various other types of variables like Enumeration, Pointer, Array, Structure, Union, etc.

Variable Definition in C

A variable definition tells the compiler where and how much storage to create for the variable. A variable definition specifies a data type and contains a list of one or more variables of that type as follows –

```
type variable_list;
```

Here, **type** must be a valid C data type including char, w_char, int, float, double, bool, or any user-defined object; and **variable_list** may consist of one or more identifier names separated by commas. Some valid declarations are shown here –

```
int i, j, k;  
char c, ch;  
float f, salary;  
double d;
```

The line **int i, j, k;** declares and defines the variables i, j, and k; which instruct the compiler to create variables named i, j and k of type int.

Variables can be initialized (assigned an initial value) in their declaration. The initializer consists of an equal sign followed by a constant expression as follows –

```
type variable_name = value;
```

Some examples are –

```
extern int d = 3, f = 5; // declaration of d and f.  
int d = 3, f = 5;      // definition and initializing d and f.  
byte z = 22;           // definition and initializes z.  
char x = 'x';          // the variable x has the value 'x'.
```

For definition without an initializer: variables with static storage duration are implicitly initialized with NULL (all bytes have the value 0); the initial value of all other variables are undefined.