

# 19GES01-PROGRAMMING FOR PROBLEM SOLVING USING C



Computer  
Programming  
Language

Department of IT

by

**S.GOPI**

**Assistant Professor**

**Department of Information Technology**

**Muthayammal Engineering College,**

**Rasipuram-637408, Namakkal,**

**Tamilnadu.**



**INTRODUCTION  
TO C  
PROGRAMMING**

**UNIT - I**

**AGENDA**

**Introduction to computer  
software**

**Program Design  
Tools**

**Structure of a C  
program**

**Writing the first C  
program**

**Keywords, Identifiers**



**Computer  
Programming  
Language**

# AGENDA

**INTRODUCTION  
TO C  
PROGRAMMING**

**UNIT - I**

**Basic Data Types in C**

**Variables, Constants**

**Input / Output  
Statements in C**

**Operators in C**

**Type conversion and  
Typecasting**



**Computer  
Programming  
Language**

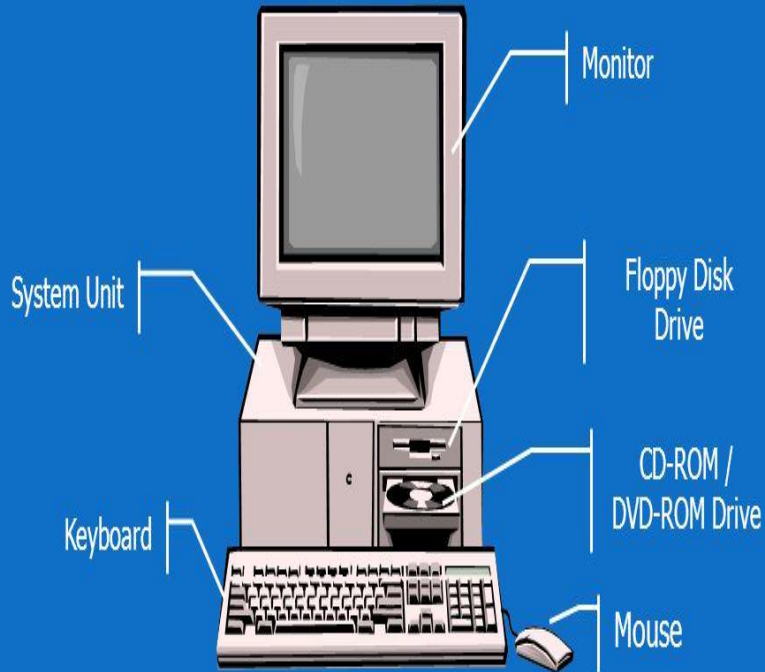
# Expansion of computer

- C - Common
- O - Operating
- M - Machine
- P - Purposely
- U - Used for
- T - Technological and
- E - Educational
- R - Research

# Definition of Computer

A **programmable electronic device** designed to accept data, perform prescribed mathematical and logical operations at **high speed**, and display the **accurate** results of these operations.

# What is a computer?



A computer is an electronic machine that accepts information (**Data**), processes it according to specific instructions, and provides the results as new information.

# Charles Babbage

(December 26, 1791 – October 18, 1871)

Father of Computer

Inventor & Founder  
of Computers



# Father of Computer

Father of Computer

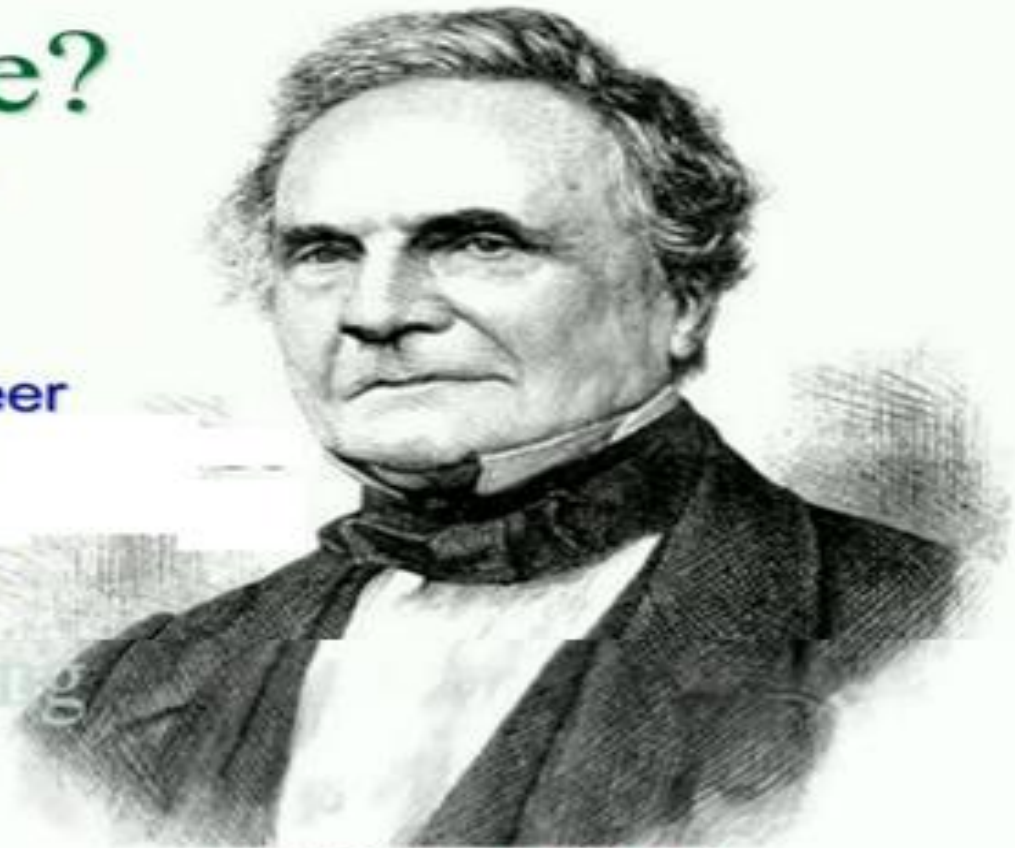
Charles Babbage

## Who was he?

- Mathematician,
- Inventor
- Philosopher
- Mechanical engineer

Inventor  
of the first

mechanical computing  
machine  
in 1821.



(1791 - 1871)

# Hardware



# Software



MS Word



Antivirus

# Charles Babbage

(December 26, 1791 – October 18, 1871)

Father of Computer

Inventor & Founder  
of Computers





# History of Computers

- First counting device
- They used sticks, stones and bones as counting tools.
- As human mind and technology improved with time more computing devices were developed.

## Apple Computer Design Evolution with Base Prices



# History of Computers

## Abacus

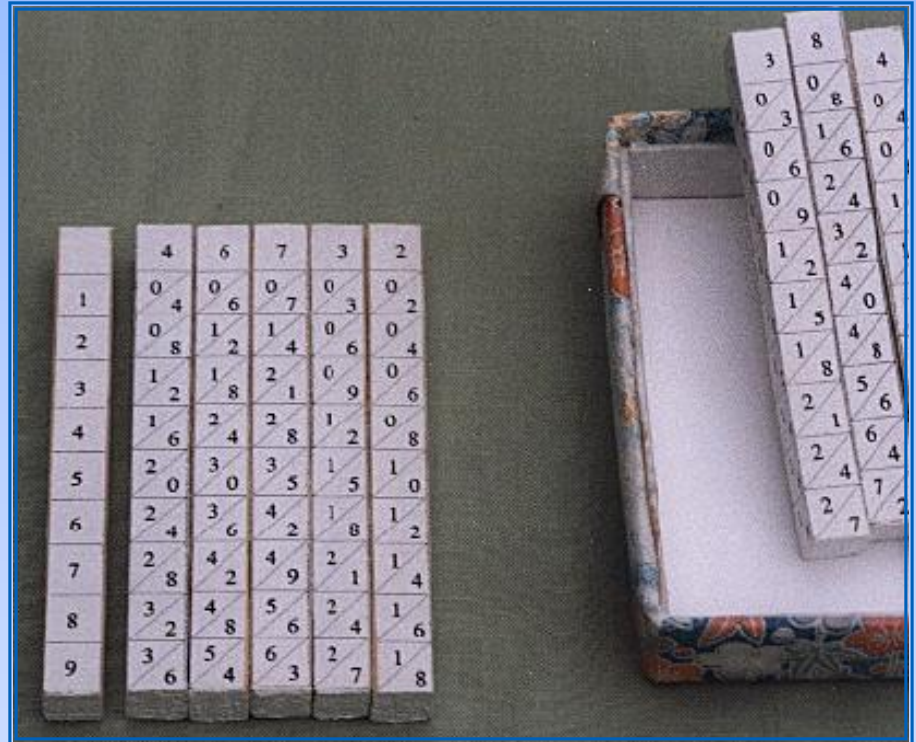
- ❑ Was invented approximately 3000 BC
- ❑ Can do  $\times$ ,  $/$ ,  $+$ ,  $-$
- ❑ Is still in use today in parts of the world



# History of Computers

## Napier's Bones

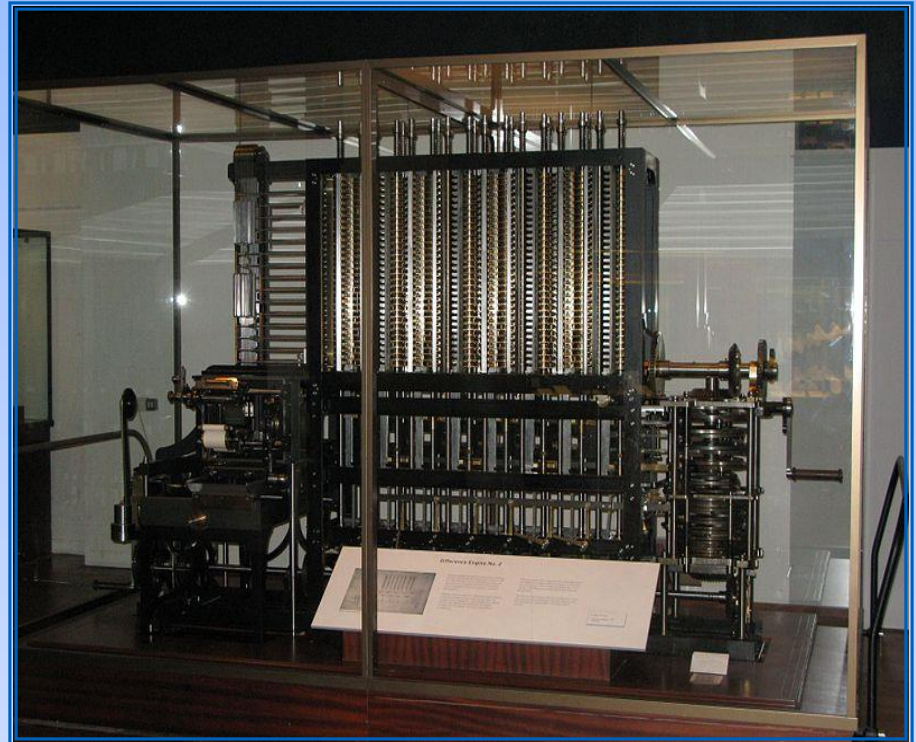
- ❑ Was invented in 1500's by John Napier
- ❑ Can do  $\times$ ,  $/$ ,  $+$ ,  $-$
- ❑ Is able to do multiplication much faster than abacus



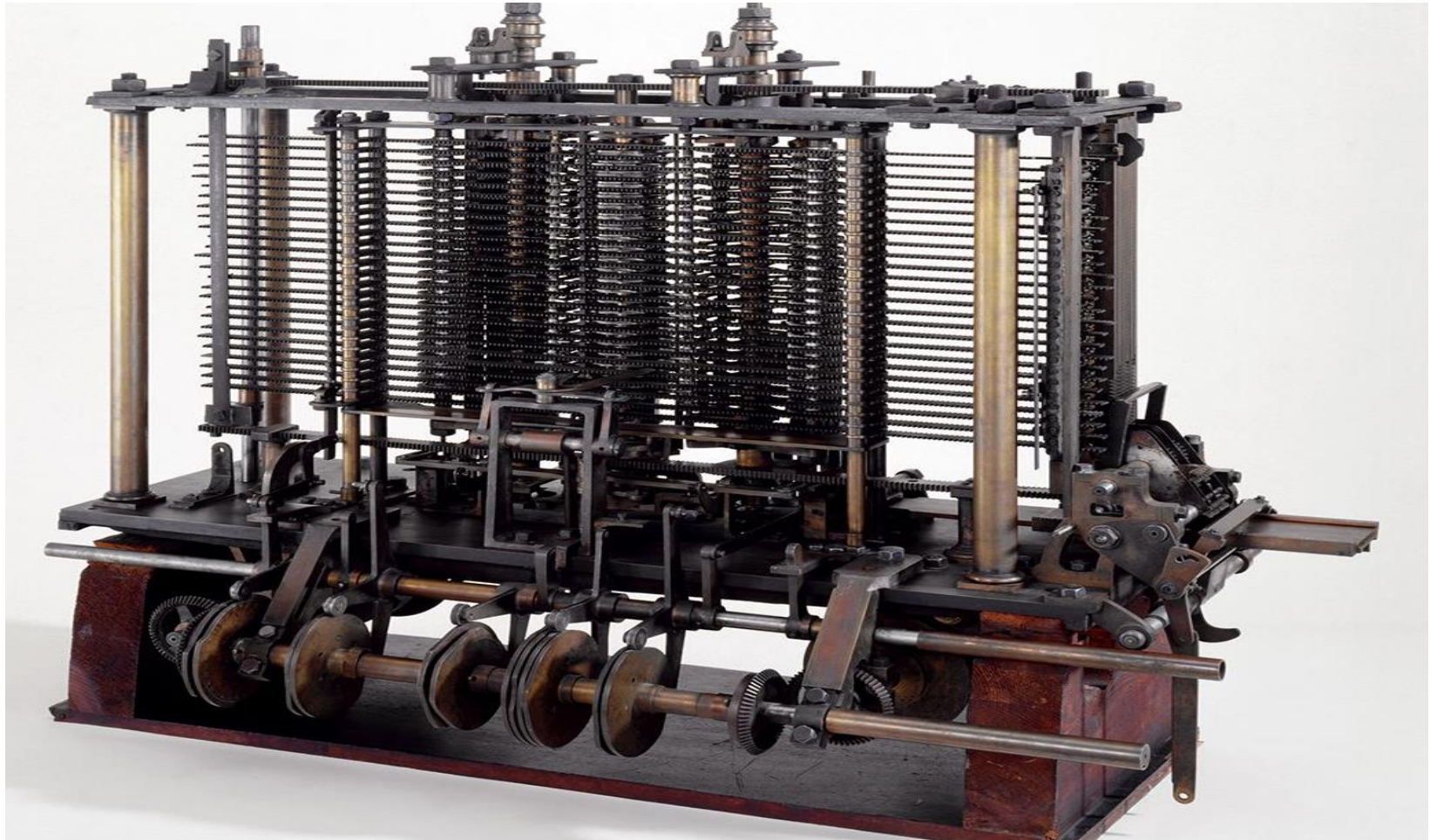
# History of Computers

## Difference Engine

- ❑ Was invented in 1842 by Charles Babbage
- ❑ Would be able to do  $+$ ,  $-$ ,  $\times$ ,  $/$  as well as solve polynomial calculations and logarithms



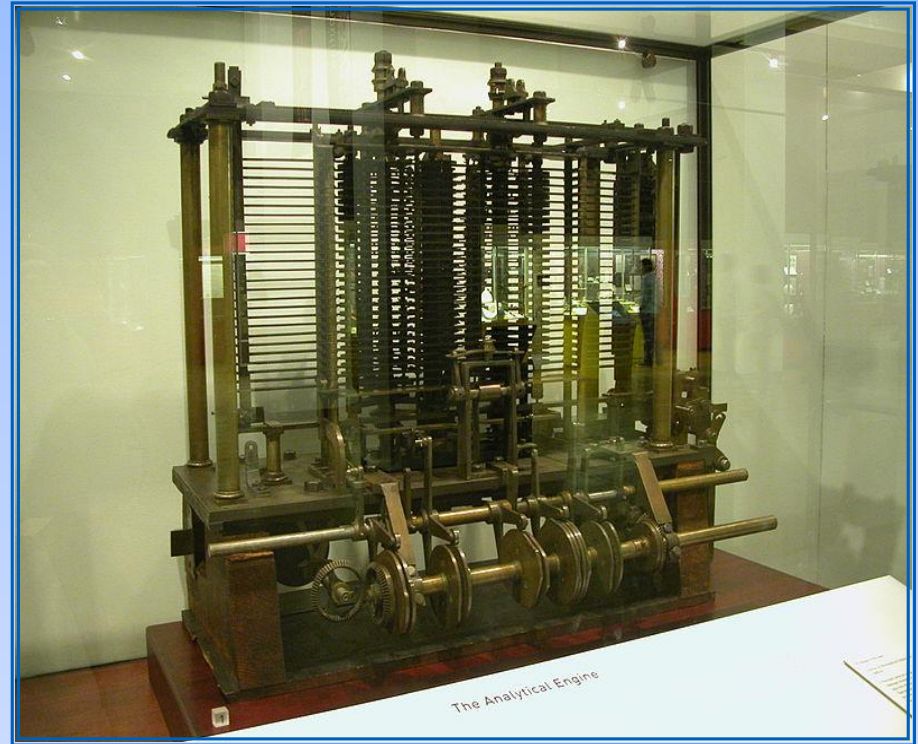
# Analytical Engine



# History of Computers

## Analytical Engine

- ❑ Charles Babbage began working on it in 1848
- ❑ This was the world's first truly programmable device, and therefore the world's first true computer



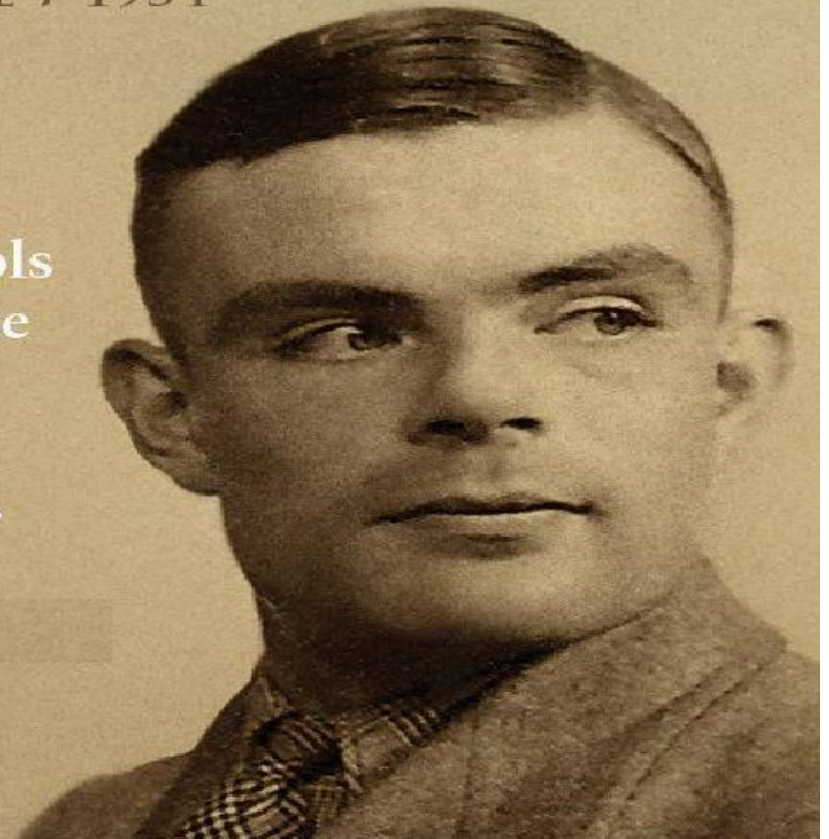
# Father of Modern Digital Computer

## ALAN TURING

FATHER OF COMPUTER SCIENCE

JUNE 23 1912 - JUNE 7 1954

Turing imagined a machine of extreme purity and simplicity. It would be able to compute anything using only two symbols arranged in a potentially infinite one-dimensional sequence. He created this machine in his mind, as a thought experiment. Today we are surrounded by Turing machines.



# Alan Turing

- English Mathematician
- Logician
- Cryptanalyst and
- Computer Scientist
- Father of Computer Science and Artificial Intelligence



# Turing machine



- ADA



10.08.2019

**ADA LOVELACE**

**A**

MATHEMATICIAN

**COMPUTER  
ENGINEER**

**FIRST**

*Computer*

**PROGRAMMER**

# Generations of Computers

First, Second, Third, Fourth & Fifth

From 1940 to 2020



First Generation Computers

From 1940 – 1956

Vacuum Tubes



Second Generation Computers

From 1956 – 1963

Transistors



Fifth Generation Computers

From 2010

Artificial Intelligence

Third Generation Computers

From 1964 – 1971

Integrated Circuits

Fourth Generation Computers

From 1972 – 2010

Micro Processors

# Types of Computer

## *Classification of computer based on work method*



**Analog Computer**



**Digital Computer**



**Hybrid Computer**

## *Classification of computer based on size*



**Micro Computer**



**Mini Computer**

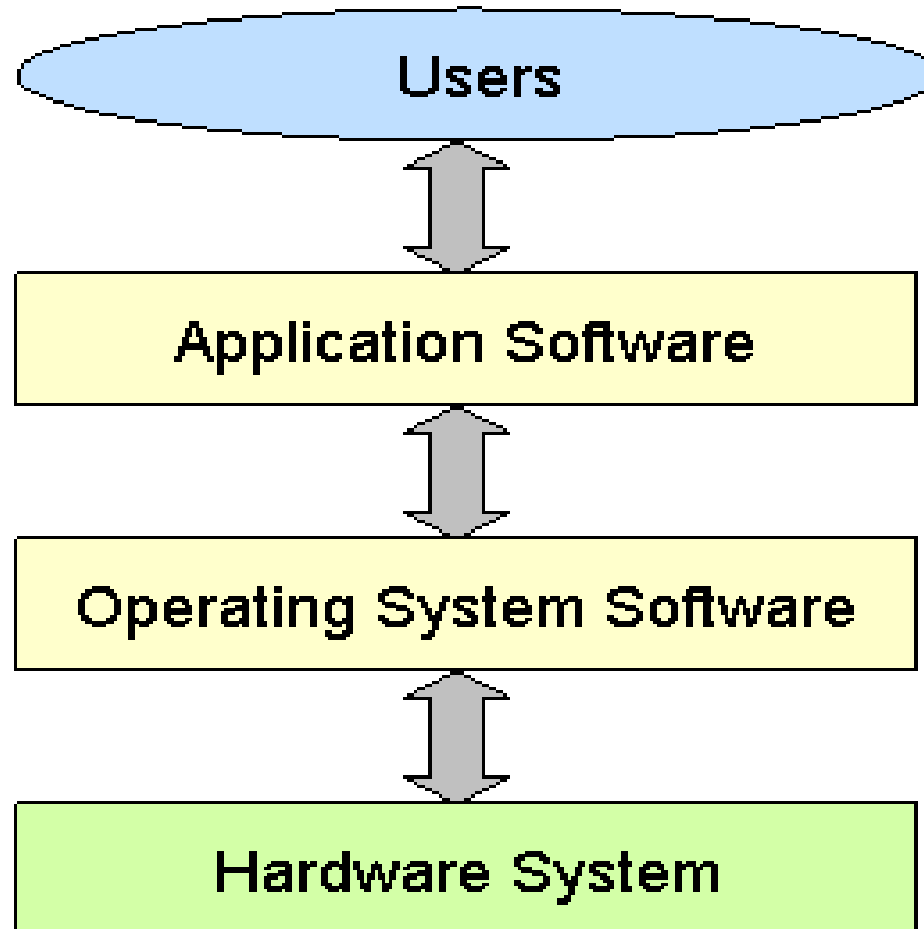


**Mainframe Computer**

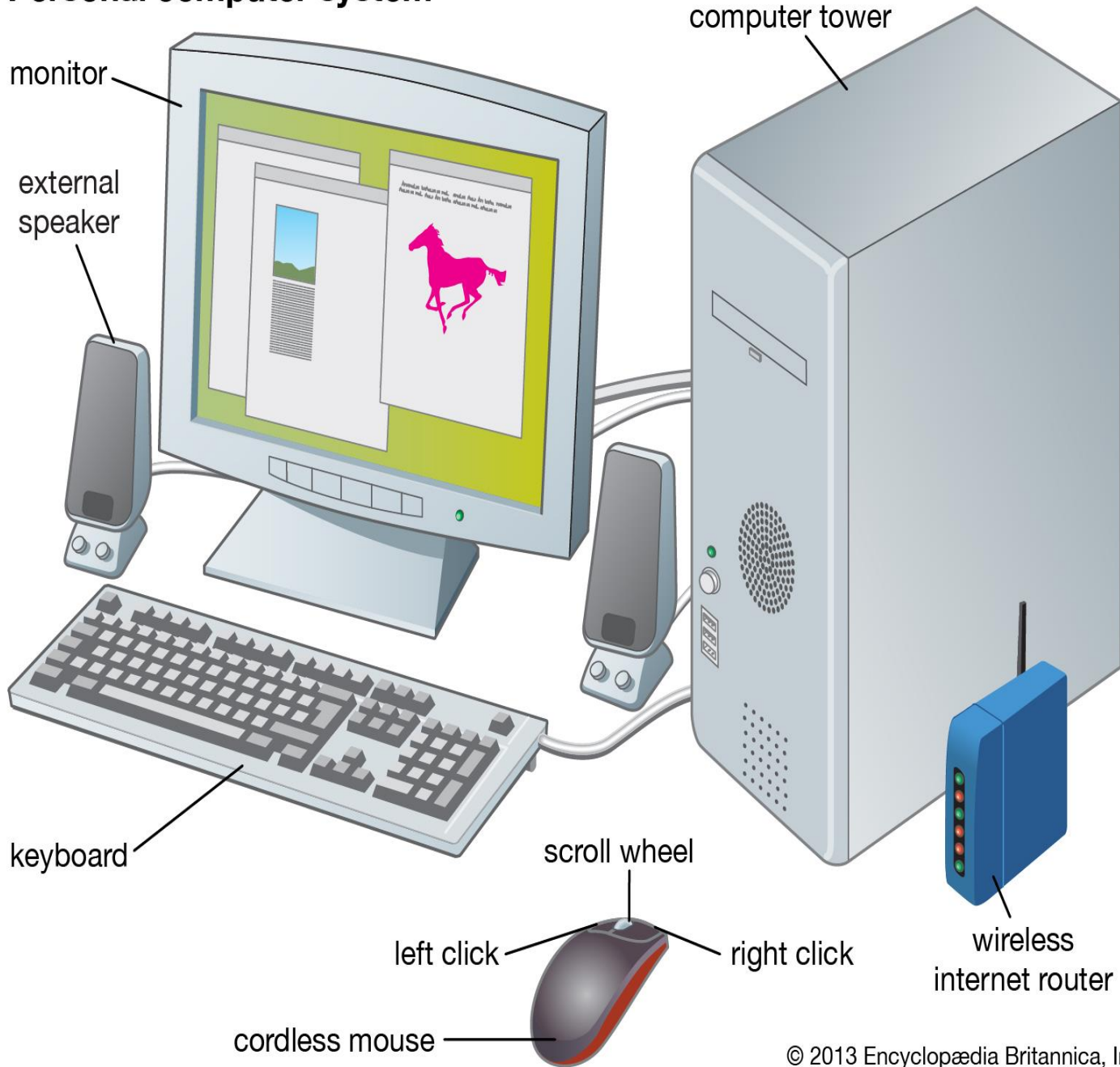


**Super Computer**

# Components of Computer System



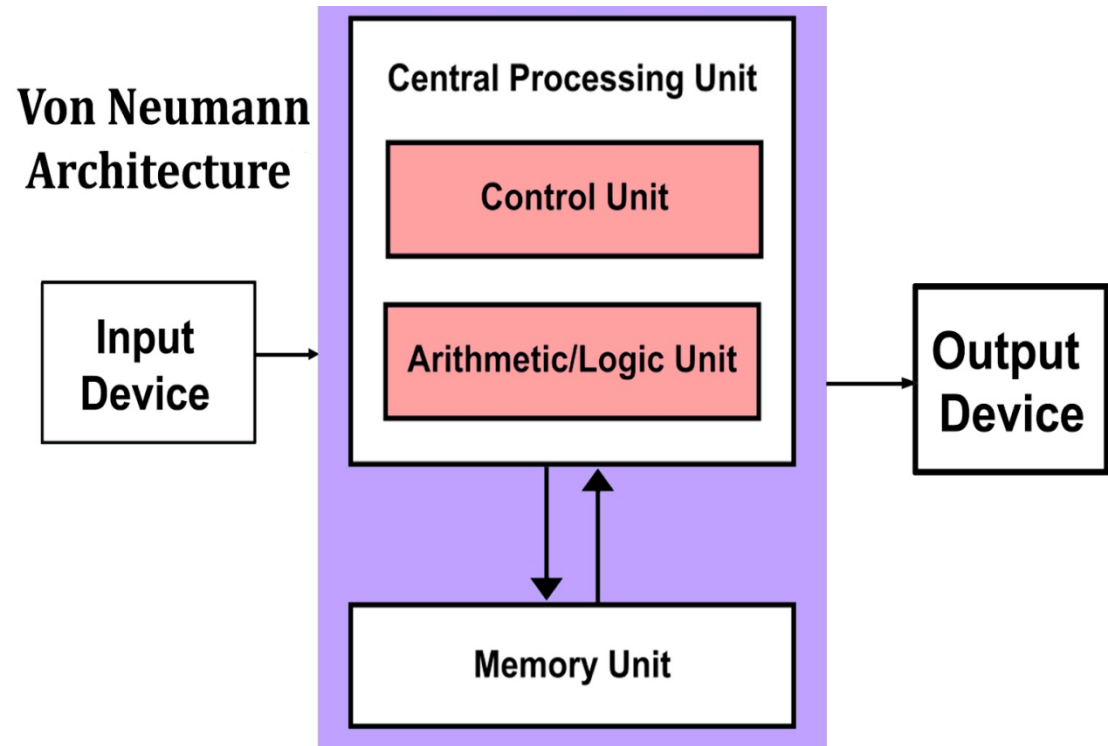
# Personal computer system



# Organization of Computer

Computer organization consist of following parts

- ▶ CPU – central processing unit
- ▶ Memory
- ▶ Input devices
- ▶ Output devices



# Central processing unit

- Alternatively referred to as the **brain of the computer**, processor, central processor, or microprocessor, the CPU
- first developed at Intel in the early 1970's
- The computer CPU is **responsible for handling all instructions** it receives from hardware and software running on the computer
- CPU **performs all types of data processing** operations.
- It **stores data**, intermediate results and instructions
- It **controls the operation** of all parts of computer



# **CPU itself has following three components**

## **1. ALU (Arithmetic Logic Unit)**

All arithmetic calculations and logical operation are performed using the Arithmetic/Logical Unit or ALU

## **2. Memory Unit**

- A memory is just like a human brain.
- It is used to store data and instruction Computer memory is use to Stores information being processed by the CPU

## **3. Control Unit**

unit help to perform operations of input unit, output unit, Memory unit and ALU in a sequence.

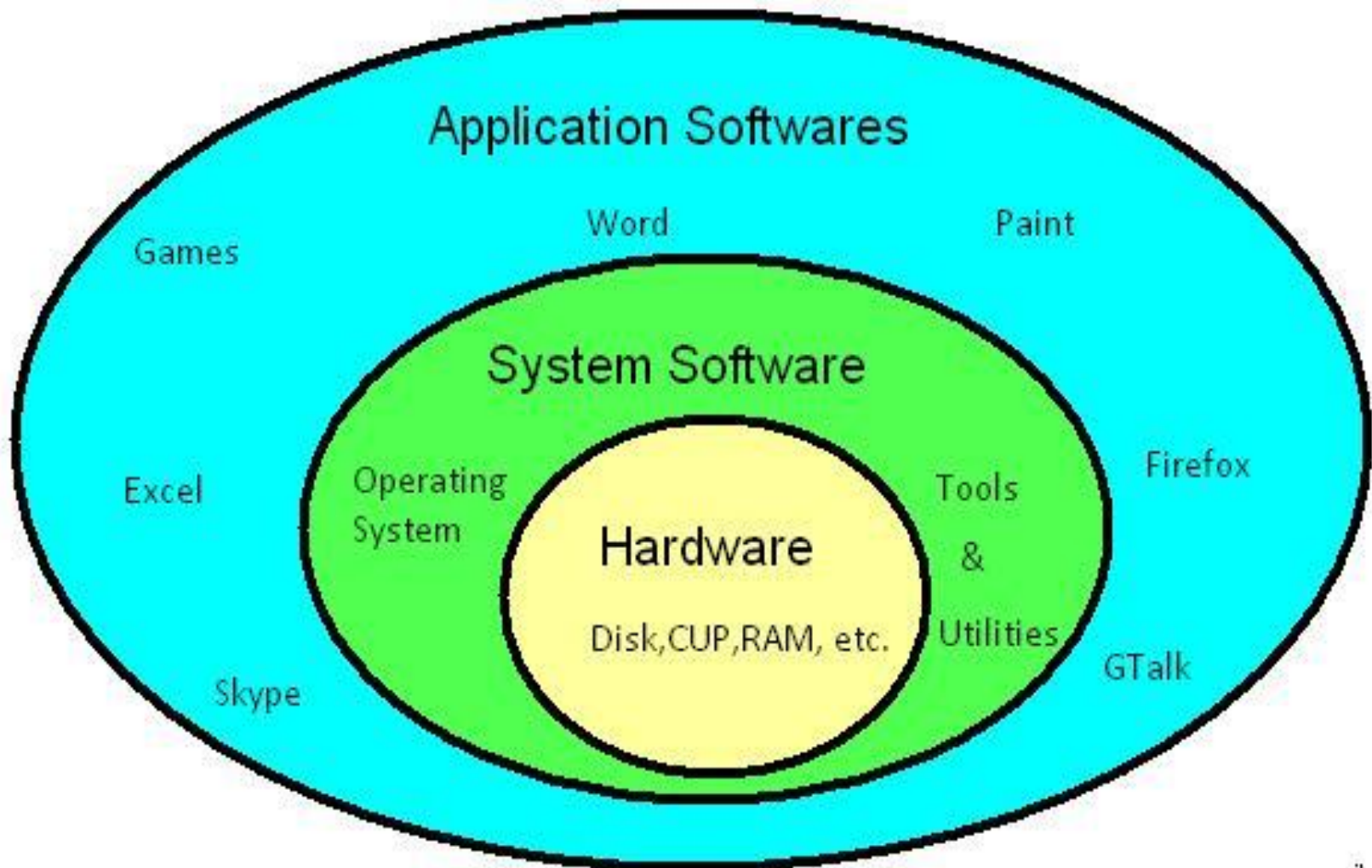
## **Input Devices**

- A device that can be used to insert data into a computer system is called as input device.
- Examples : Keyboards, mouse, scanners and digital cameras

## **Output Devices**

- A device which is used to display result from a computer is called as output device
- Examples: Printer, Scanner, Monitor, etc.

# Examples of software and Hardware



# Hardware

# Software

Hardware is further divided into four main categories:

- Input Devices
- Output Devices
- Secondary Storage Devices
- CPU

Software is further divided into two main categories:

- Application Software
- System Software

Developed using electronic and other materials

Developed by using a programming language

When damaged, it can be replaced with a new component

When damaged it can be installed once more using a backup copy

Hardware is physical in nature and hence one can touch and see hardware

The software cannot be physically touched but still can be used and seen

Hardware cannot be infected by Viruses

The software can be infected by Viruses

An example of Hardware is hard

An example of software is

# Types of Computer Language

- Computer language is defined as **code or syntax** which is used to write programs or any specific applications
- The computer language is used to communicate with computers
- Three categories assembly language, machine language, and high-level language

## 1. Machine Language

- The Machine language is considered a low-level language
- Other name -machine code or object code
- Which is set of binary digits 0 and 1
- These binary digits are understood and read by a computer system
- Example of machine language for the text “Hello World”.

```
01001000 0110101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100
```

## 2. Assembly Language

- Intermediate-level language for microprocessors
- It is second-generation language

## 3. High-Level Language

- The high-level language is easy to understand and
- human-readable program
- Examples: C++, C, JAVA, FORTRAN, etc..

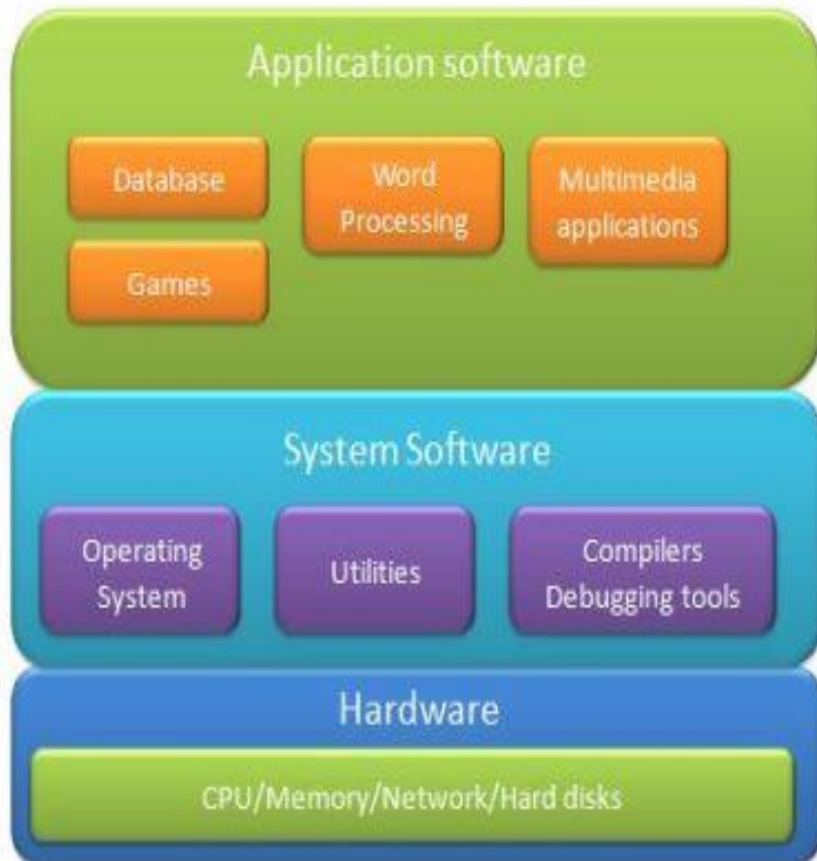
# Computer – Software

- Software is a set of programs, which is designed to perform a well-defined function.
- A program is a sequence of instructions written to solve a particular problem.



# There are two types of software

- System Software
- Application Software



Computer  
Programming  
Language



# System Software

- The system software is a **collection of programs**
- designed to **operate, control, and supports the process of computer**
- System software –**Inbuilt in System**
- System software written in **low-level languages**
- Use to **Interact** with the hardware and software
- It Serves as the **interface between** the hardware and end users
- Examples: **Operating System, Compilers, Interpreter, Assemblers, etc..**

# System Software

Some examples of system software are Operating System, Compilers, Interpreter, Assemblers, etc.

- Close to the system
- Fast in speed
- Difficult to design
- Difficult to understand
- Less interactive
- Smaller in size
- Difficult to manipulate
- Generally written in low-level language



# Application Software

- Application software products are designed to satisfy a particular need of a particular environment
- It is a collection of programs, often called a software package, which work together to **accomplish user task**, such as a spreadsheet package
- Some examples: Payroll Software , Student Record Software , Income Tax Software and Railways Reservation Software

# Application Software

- Application software products are designed to satisfy a particular need of a particular environment.
- All software applications prepared in the computer lab can come under the category of Application software.
- Application software may consist of a single program, such as Microsoft's notepad for writing and editing a simple text.



Application  
Software

# Application Software

Examples of Application software are the following –

- Payroll Software
- Student Record Software
- Inventory Management Software
- Income Tax Software
- Railways Reservation Software
- Microsoft Office Suite Software
- Microsoft Word
- Microsoft Excel
- Microsoft PowerPoint



## Application Software



Key	System Software	Application Software
Definition	System Software is the type of software which is the interface between application software and system	Application Software is the type of software which runs as per user request. It runs on the platform which is provide by system software
Development Language	low level language	high level language
Usage	System software is used for operating computer hardware	Application software is used by user to perform specific task
Installation	Installed on the computer when operating system is installed	Application software are installed according to user's requirements
Dependency	System software can run independently, It provides platform for running application software	Application software can't run independently. They can't run without the presence of system software

# Program Design Tools

- **Program Design tools** are the tools used to develop a program. A program is the expression of an algorithm in a programming language.



Algorithms



Flowcharts



Pseudo codes



# Algorithm

- ▶ An algorithm in general is a sequence of steps to solve a particular problem. Algorithms are universal.
- ▶ The algorithm you use in C programming language is also the same algorithm you use in every other language.





# Qualities of a good algorithm

1. Input and output should be defined precisely.
2. Each steps in algorithm should be clear and unambiguous.
3. Algorithm should be most effective among many different ways to solve a problem.
4. An algorithm shouldn't have computer code. Instead, the algorithm should be written in such a way that, it can be used in similar programming languages



Computer  
Programming  
Language

# Examples Of Algorithms In Programming

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum

$sum \leftarrow num1 + num2$

Step 5: Display sum

Step 6: Stop



Computer  
Programming  
Language

# Examples Of Algorithms In Programming

Write an algorithm to find the largest among three different numbers entered by user.

Step 1: Start

Step 2: Declare variables a,b and c.

Step 3: Read variables a,b and c.

Step 4: If  $a > b$

    If  $a > c$

        Display a is the largest number.

    Else

        Display c is the largest number.

Else

    If  $b > c$

        Display b is the largest number.

    Else

        Display c is the greatest number.

Step 5: Stop








Computer  
Programming  
Language

# Flowchart

- A flowchart is a type of diagram that represents a workflow or process.
- A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.
- The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.





# Symbols Used In Flowchart

Symbol	Purpose	Description
	Flow line	Indicates the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Represents the start and the end of a flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for arithmetic operations and data-manipulations.
	Decision	Used for decision making between two or more alternatives.

Computer  
Programming  
Language

# Symbols Used In Flowchart

	Decision	Used for decision making between two or more alternatives.
	On-page Connector	Used to join different flowline.
	Off-page Connector	Used to connect the flowchart portion on a different page.
	Predefined Process/Function	Represents a group of statements performing one processing task.

Computer  
Programming  
Language

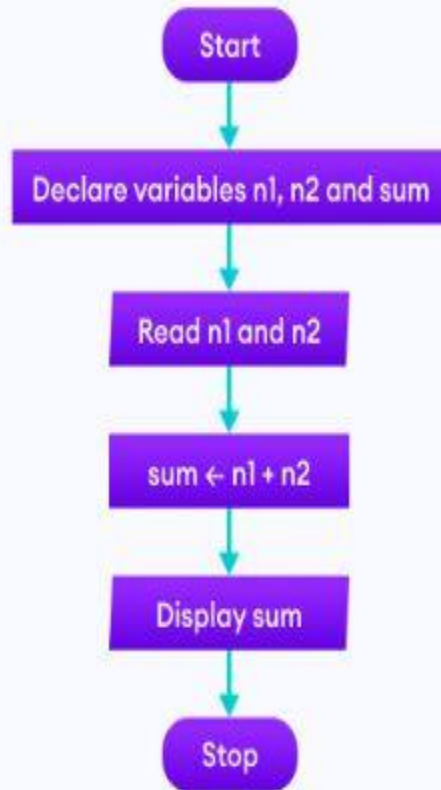
# Guidelines for Developing Flowcharts

## Guidelines for Developing Flowcharts

- These are some points to keep in mind while developing a flowchart –
- Flowchart can have only one start and one stop symbol
- On-page connectors are referenced using numbers
- Off-page connectors are referenced using alphabets
- General flow of processes is top to bottom or left to right
- Arrows should not cross each other



## 1. Add two numbers entered by the user.

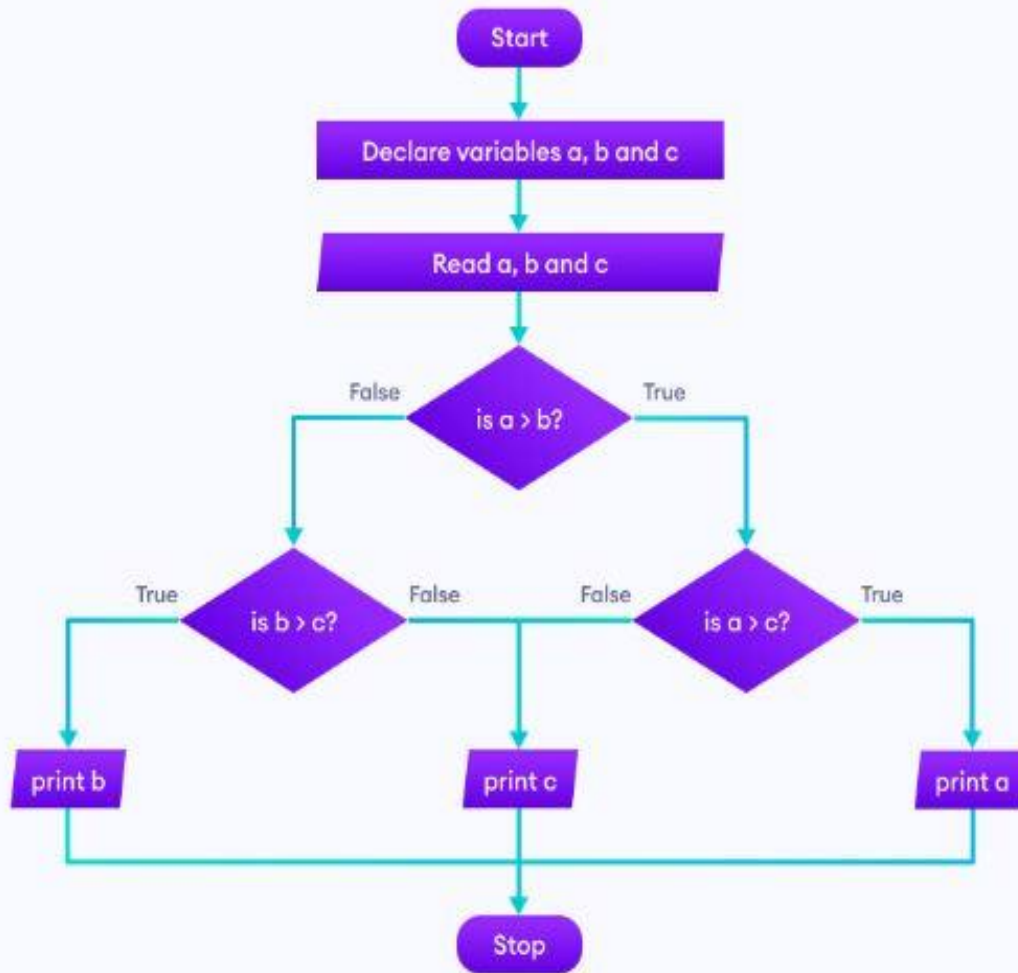


Flowchart to add two numbers

Computer  
Programming  
Language



2. Find the largest among three different numbers entered by the user.



Computer  
Programming  
Language

# Guidelines for Developing Flowcharts

## Guidelines for Developing Flowcharts

- These are some points to keep in mind while developing a flowchart –
- Flowchart can have only one start and one stop symbol
- On-page connectors are referenced using numbers
- Off-page connectors are referenced using alphabets
- General flow of processes is top to bottom or left to right
- Arrows should not cross each other



# Pseudocode

- **pseudocode** is a method of writing an **algorithm**.
- It cannot be compiled or run like a regular program.
- Pseudocode can be written how you want. But some companies use specific pseudocode syntax to keep everyone in the company on the same page.
- Syntax is a set of rules on how to use and organize statements in a programming language



Computer  
Programming  
Language

# Advantages of Pseudocode

- Improves the readability of any approach.
- It's one of the best approaches to start implementation of an algorithm.
- Acts as a bridge between the program and the algorithm or flowchart.
- The main goal of a pseudo code is to explain what exactly each line of a program should do,
- Hence making the code construction phase easier for the programmer.



# Advantages of Pseudocode

How to write a Pseudo-code?

Example:

```
if "1"  
    print response  
    "I am case 1"  
  
if "2"  
    print response  
    "I am case 2"
```

## Do's :

- . Use control structures
- . Use proper naming convention
- . Indentation and white spaces are the key
- . Keep it simple.
- . Keep it concise.

## Don'ts :

- . Don't make the pseudo code abstract.
- . Don't be too generalized.
- .



Computer  
Programming  
Language

# Example

SumOfTwoNumbers()

Begin

Set sum =0;

  Read: a, b;

  Set sum = a + b;

  Print sum;

End

# ALGORITHM VERSUS PSEUDOCODE

## ALGORITHM

An unambiguous specification of how to solve a problem

Helps to simplify and understand the problem

## PSEUDOCODE

An informal high-level description of the operating principle of a computer program or other algorithm

A method of developing an algorithm

Visit [www.PEDIAA.com](http://www.PEDIAA.com)

# PSEUDOCODE VERSUS FLOWCHART

## PSEUDOCODE

An informal high-level description of the operating principle of an algorithm

Written in natural language and mathematical notations help to write pseudocode

## FLOWCHART

A diagrammatic representation that illustrates a solution model to a given problem

Written using various symbols

Visit [www.PEDIAA.com](http://www.PEDIAA.com)



# History of C Language

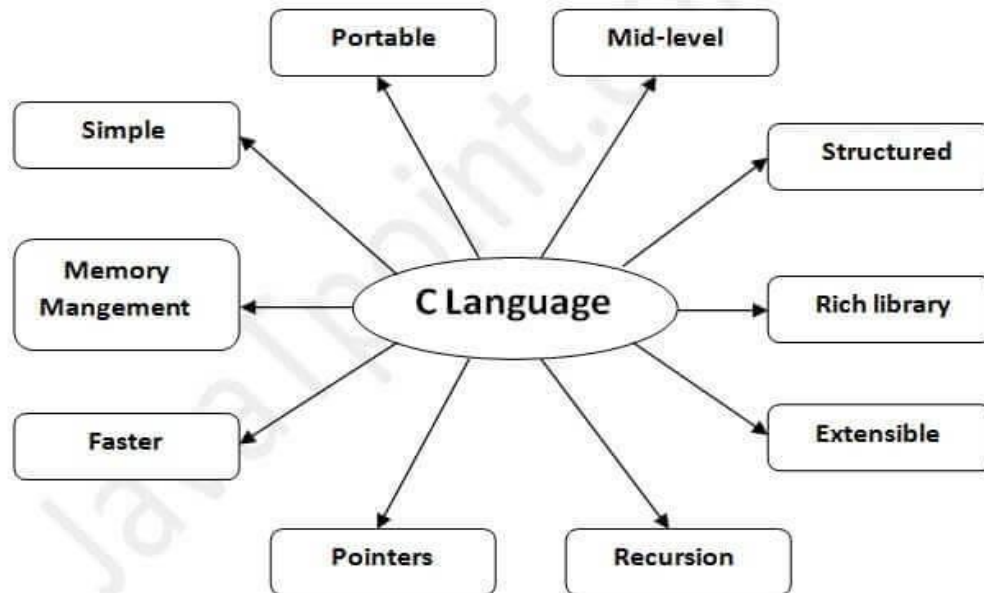
- **C programming language** was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.
- **Dennis Ritchie** is known as the **founder of the c language**.
- It was developed to overcome the problems of previous languages such as B, BCPL, etc.



Computer  
Programming  
Language

# Features of C Language

- C is the widely used language. It provides many **features** that are given below.

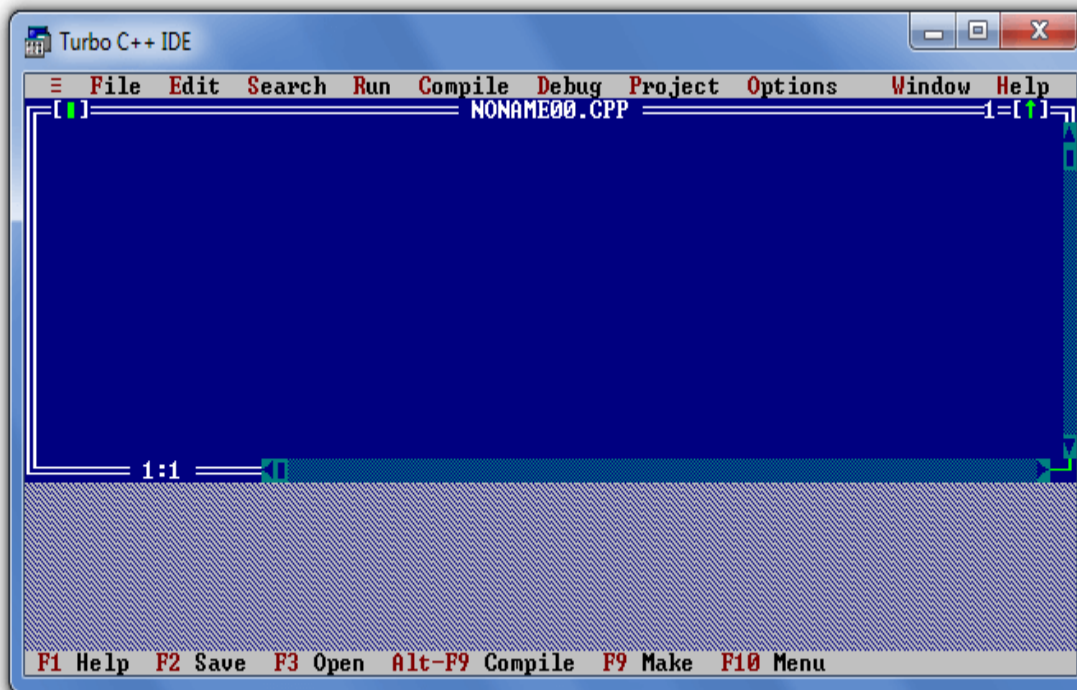


JavaTpoint.com

Computer  
Programming  
Language

# C- compiler

- There are many compilers available for c and c++. You need to download any one. Here, we are going to use [Turbo C++](#).



Computer  
Programming  
Language

# Compilation process in c

- The compilation is a process of converting the source code into object code. It is done with the help of the compiler.
- The compiler checks the source code for the syntactical or structural errors, and if the source code is error-free, then it generates the object code.

```
#include<stdio.h>
main()
{
printf("Hello java Tpoint");
return 0;
}
```



```
01000000000000
0111111111111111
01010101101010
0000001111111111
0000011111111111
00000010101011
```

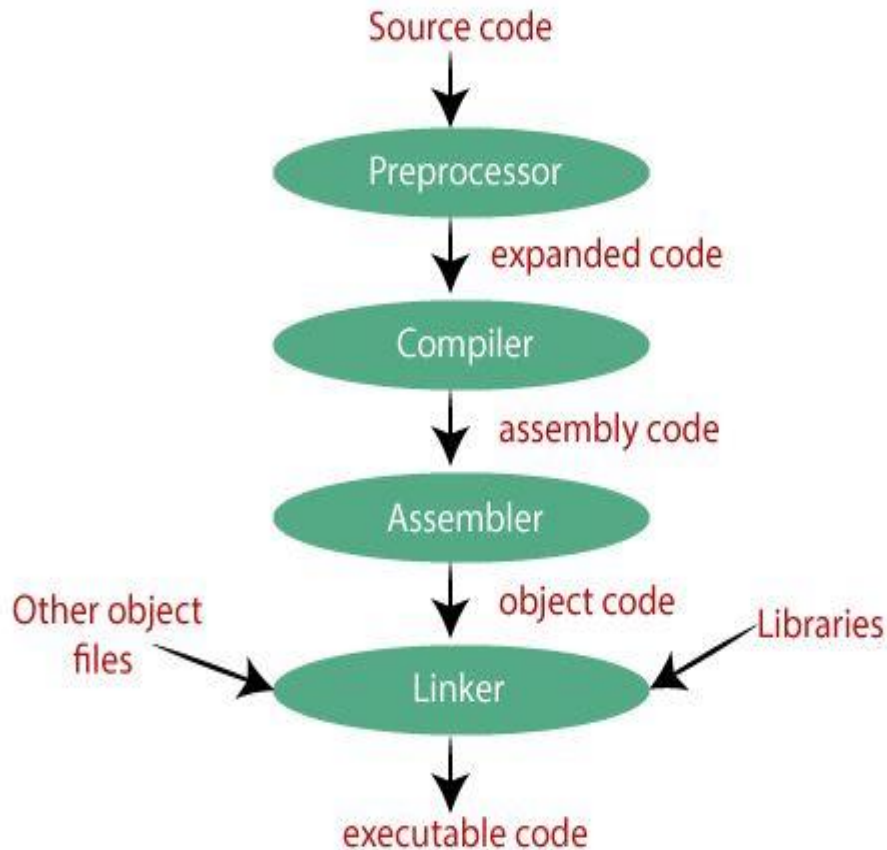
Computer  
Programming  
Language

# Compilation process in c

- The compilation process can be divided into four steps, i.e., Pre-processing, Compiling, Assembling, and Linking.
- The following are the phases through which our program passes before being transformed into an executable form:
  1. **Preprocessor**
  2. **Compiler**
  3. **Assembler**
  4. **Linker**



# Compilation process in c



Computer  
Programming  
Language

# Compilation process in c

## Preprocessor

- The source code is the code which is written in a text editor and the source code file is given an extension ".c".
- This source code is first passed to the preprocessor, and then the preprocessor expands this code.
- After expanding the code, the expanded code is passed to the compiler.



# Compilation process in c

## Compiler

- The code which is expanded by the preprocessor is passed to the compiler. The compiler converts this code into assembly code.
- Or we can say that the C compiler converts the pre-processed code into assembly code.





# Compilation process in c

## Assembler

- The assembly code is converted into object code by using an assembler. The name of the object file generated by the assembler is the same as the source file.
- The extension of the object file in DOS is '.obj,' and in UNIX, the extension is 'o'. If the name of the source file is '**hello.c**', then the name of the object file would be 'hello.obj'.



# Compilation process in c

## Linker

- Mainly, all the programs written in C use library functions.
- These library functions are pre-compiled, and the object code of these library files is stored with '.lib' (or '.a') extension.
- Therefore, we conclude that the job of the linker is to link the object code of our program with the object code of the library files and other files.



# Compilation process in c

## Linker

- The output of the linker is the executable file. The name of the executable file is the same as the source file but differs only in their extensions.
- In DOS, the extension of the executable file is '.exe',
- For example, if we are using printf() function in a program, then the linker adds its associated code in an output file.



# Compilation process in c

Let's understand through an example.

**hello.c**

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Hello javaTpoint");
```

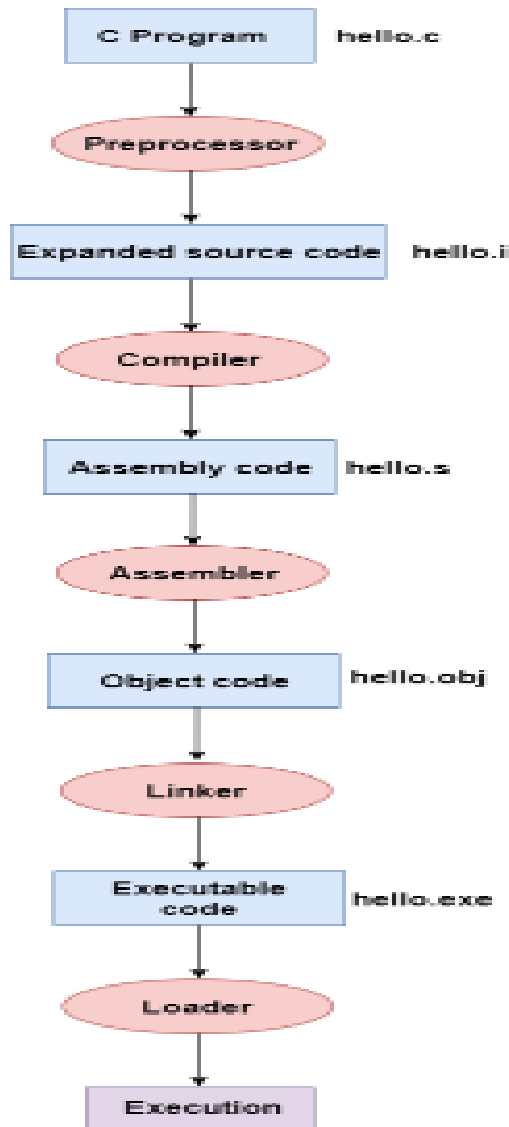
```
    return 0;
```

```
}
```



Computer  
Programming  
Language

# Compilation process in c



Computer  
Programming  
Language

# Structure of c program

A C program involves the following sections:

- Documentations (Documentation Section)
- Preprocessor Statements (Link Section)
- Global Declarations (Definition Section)
- The main() function
- Local Declarations
- Program Statements & Expressions
- User Defined Functions



Computer  
Programming  
Language

# First C Program

```
/*My first c program*/  
  
#include<stdio.h>  
  
Void main()  
{  
printf("Hello, World!\n");  
}
```



Computer  
Programming  
Language

**/\* Comments \*/** - Comments are a way of explaining what makes a program. The compiler ignores comments and used by others to understand the code.

**#include<stdio.h>** - This is a preprocessor command. That notifies the compiler to include the header file `stdio.h` in the program before compiling the source-code.

**Void** – the function returns null

**main()** - The `main()` is the main function where program execution begins. Every C program must contain only one main function.

**Braces** - Two curly brackets "{...}" are used to group all statements.

**printf()** - It is a function in C, which prints text on the screen.





```
// Name of Program
```

 → Documentation section

```
#include<stdio.h> }  
#include<conio.h> }
```

 → Preprocessor Directives

```
#define max 100
```

 → Definition section

```
void add(); }  
int x=100; }
```

 → Global declaration section

```
int main()
```

 → main () Function section / Entry Point

```
{ int a=100;
```

 → Variable declaration

```
printf("Hello Main"); }  
return 0; }  
}
```

 → Body of Main function

```
void add(){  
printf("Hello add"); }  
}
```

 → Function Definition

# Computer Programming Language

# Compile and Execute C Program

- Open a text editor and add the above-mentioned code.
- Save the file as *hello.c*
- *Alt + F9 – Compile*
- *Ctrl + F9 – Compile & Run*



# Input and output statements

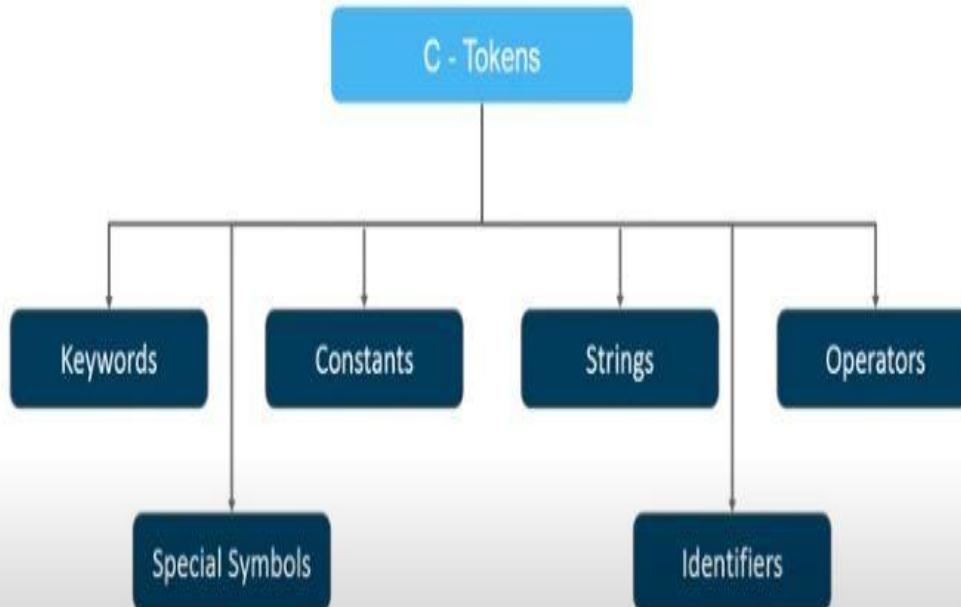
- **printf()** is used to display the output and **scanf()** is used to read the inputs.
- **printf()** and **scanf()** functions are declared in “stdio.h” header file in **C** library.
- All syntax in **C** language including **printf()** and **scanf()** functions are case sensitive.



Computer  
Programming  
Language

# Tokens – small individual unit

## C TOKENS



Computer  
Programming  
Language

# Keywords in C

A keyword is a reserved word. You cannot use it as a variable name, constant name, etc. There are only 32 reserved words (keywords) in the C language.

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



Computer  
Programming  
Language

# C Identifiers

- Identifier refers to name given to entities such as variables, functions, structures etc.
- Identifiers must be unique. They are created to give a unique name to an entity to identify it during the execution of the program.
- For example:
  - **1. int money;**
  - **2. double accountBalance;**
  - **Here, money and accountBalance are identifiers.**
- Also remember, **identifier names must be different from keywords. You cannot use int as an identifier because int is a keyword.**



Computer  
Programming  
Language

# Rules for naming identifiers

- 1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
- 2. The first letter of an identifier should be either a letter or an underscore.
- 3. You cannot use keywords as identifiers.
- 4. There is no rule on how long an identifier can be. However, you may run into problems in some compilers if the identifier is longer than 31 characters.



Computer  
Programming  
Language

# Variables in C

- A **variable** is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times.
- It is a way to represent memory location through symbol so that it can be easily identified.
- Let's see the syntax to declare a variable:

**type variable\_list;**

- The example of declaring the variable is given below:
- **int a;**
- **float b;**
- **char c;**



Computer  
Programming  
Language



# Variables in C

- Here, a, b, c are variables. The int, float, char are the data types.
- We can also provide values while declaring the variables as given below:
- **int a=10,b=20;//declaring 2 variable of integer type**
- **float f=20.8;**
- **char c='A';**



Computer  
Programming  
Language

# Variables in C

## Rules for defining variables

- A variable can have alphabets, digits, and underscore.
- A variable name can start with the alphabet, and underscore only. It can't start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword, e.g. int, float, etc.



Computer  
Programming  
Language

# Variables in C

## Local Variable

- A variable that is declared inside the function or block is called a local variable.
- It must be declared at the start of the block.

```
void main(){  
int x=10;//local variable  
}
```



Computer  
Programming  
Language

# Variables in C

## Global Variable

- A variable that is declared outside the function or block is called a global variable.
- Any function can change the value of the global variable. It is available to all the functions.

```
int value=20;//global variable
void main(){
int x=10;//local variable
}
```



Computer  
Programming  
Language

# Constants in C

A constant is a value or variable that can't be changed in the program, for example: 10, 20, 'a', 3.4, "c programming" etc.

There are two simple ways in C to define constants

Using **#define** preprocessor.

## Syntax:

```
#define identifier value
```

## Example

```
#define LENGTH 10
```

```
#define WIDTH 5
```

Using **const** keyword.

## Syntax:

```
const type var
```

```
const int LENGTH = 10;
```

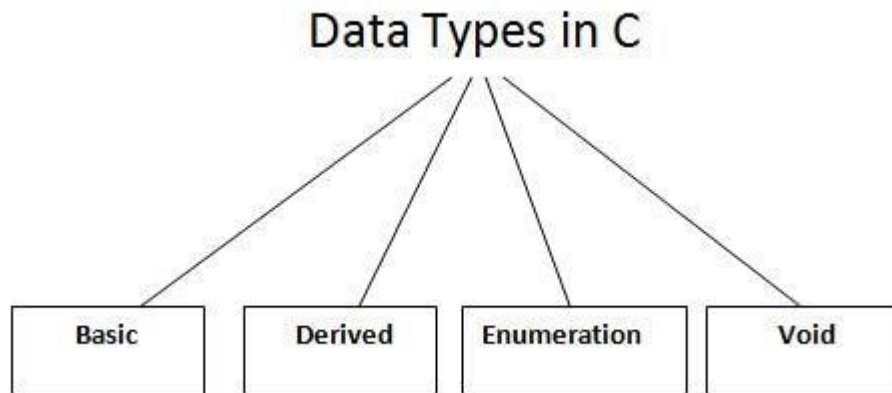
```
const int WIDTH = 5;
```



Computer  
Programming  
Language

# Data Types in C

A data type specifies the **type of data** that a variable can store such as integer, floating, character



Computer  
Programming  
Language

# Data Types in C

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



Computer  
Programming  
Language

# Format specifier

- The **Format specifier** is a string used in the formatted input and output functions.
- The **format string** determines the format of the input and output.
- The format string always starts with a '%' character.

## EXAMPLE:

`%d` - int

`%f` - float

`%c` - char



Computer  
Programming  
Language



Data type	Size(bytes)	Range	Format String
char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
short	2	-32,768 to 32,767	%d
unsigned short	2	0 to 65535	%u
int	2	32,768 to 32,767	%d
unsigned int	2	0 to 65535	%u
long	4	-2147483648 to +2147483647	%ld
Unsigned long	4	0 to 4294967295	%lu
float	4	-3.4e-38 to +3.4e-38	%f
double	8	1.7 e-308 to 1.7 e+308	%lf
long double	10	3.4 e-4932 to 1.1 e+4932	%lf

# INPUT OUTPUT (I/O) STATEMENTS

- `scanf()` function to take input from the user,
- `printf()` function to display output to the user

## EXAMPLE:

```
#include <stdio.h>
int main()
{
// Displays the string inside quotations
printf("C Programming");
return 0;
}
```



Computer  
Programming  
Language

# Output statement in C

- printf() function to display output to the user

**Syntax: printf(“%X”, variableOfXType);**

where %X is the format specifier in c

- In C programming, printf() is one of the main output function. The function sends formatted output to the screen.
- The printf() is a library function to send formatted output to the screen. The function prints the string inside the quotations.
- To use printf() in our program, we need to include stdio.h header file using #include <stdio.h> statement.



Computer  
Programming  
Language

# EXAMPLE

```
#include <stdio.h>
void main()
{
int testInteger = 5;
printf("Number = %d", testInteger);
}
```

- **Output: Number = 5**
- We use %d format specifier to print int types.
- Here, the %d inside the quotations will be replaced by the value of testInteger



Computer  
Programming  
Language

# Example 3: float and double Output

```
#include <stdio.h>
void main()
{
float number1 = 13.5;
double number2 = 12.4;
printf("number1 = %f\n", number1);
printf("number2 = %lf", number2);
}
```



Computer  
Programming  
Language

# Print Characters

```
#include <stdio.h>
void main()
{
char chr = 'a';
printf("character = %c", chr);
}
```

## Output

character = a

To print char, we use %c format specifier



Computer  
Programming  
Language

# INPUT OUTPUT (I/O) STATEMENTS

➤ scanf() function - to take input from the user,

**Syntax:** `scanf("%X", &variableOfXType);`

where **%X** is the format specifier in c

**&** is the address operator in C.

## EXAMPLE:

```
#include <stdio.h>
void main()
{
int a;
// Displays the string inside quotations
printf("C Programming");
//getting input from the user
scanf("%d",&a);
printf("Given data is %d",a);
}
```



Computer  
Programming  
Language

# C Input

- In C programming, scanf() is one of the commonly used function to take input from the user.
- The scanf() function reads formatted input from the standard input such as keyboard

## Integer Input/Output

```
#include <stdio.h>
void main()
{
int testInteger;
printf("Enter an integer: ");
scanf("%d", &testInteger);
printf("Number = %d",testInteger);
}
```

## Output

```
Enter an integer: 4
Number = 4
```



Computer  
Programming  
Language



# C Input

- Here, we have used %d format specifier inside the scanf() function to take input from the user. When the user enters an integer, it is stored in the testInteger variable.
- Notice, that we have used &testInteger inside scanf(). It is because &testInteger gets the address of testInteger, and the value entered by the user is stored in that address.



Computer  
Programming  
Language

# C Input

## Example 6: Float and Double Input/Output

```
#include <stdio.h>

int main()
{
float num1;
double num2;
printf("Enter a number: ");
scanf("%f", &num1);
printf("Enter another number: ");
scanf("%lf", &num2);
printf("num1 = %f\n", num1);
printf("num2 = %lf",
num2);
return 0;}
```



Computer  
Programming  
Language

# C Input

## C Character I/O

```
#include <stdio.h>
void main()
{
char chr;
printf("Enter a character: ");
scanf("%c",&chr);
printf("You entered %c", chr);
}
```

## Output

Enter a character: g  
You entered g.



Computer  
Programming  
Language

The Syntax for input and output for these are:

- **Integer:**

```
Input: scanf("%d", &intVariable);  
Output: printf("%d", intVariable);
```

- **Float:**

```
Input: scanf("%f", &floatVariable);  
Output: printf("%f", floatVariable);
```

- **Character:**

```
Input: scanf("%c", &charVariable);  
Output: printf("%c", charVariable);
```



# C Input

- When a character is entered by the user in the above program, the character itself is not stored. Instead, an integer value (ASCII value) is stored.
- And when we display that value using %c text format, the entered character is displayed. If we use %d to display the character, it's ASCII value is printed.



# ASCII Value

```
#include <stdio.h>
int main()
{
char chr;
printf("Enter a character: ");
scanf("%c", &chr);
// When %c is used, a character is displayed
printf("You entered %c.\n",chr);
/*When %d is used, ASCII value is displayed */
printf("ASCII value is % d.", chr);
return 0;
}
```

Output

Enter a character: g

You entered g.

ASCII value is 103.



Computer  
Programming  
Language

# I/O Multiple Values

Here's how you can take multiple inputs from the user and display them.

```
#include <stdio.h>
int main()
{
int a;
float b;
printf("Enter integer and then a float: ");
// Taking multiple inputs Page13
scanf("%d%f", &a, &b);
printf("You entered %d and %f", a, b);
return 0;
}
```

## Output

Enter integer and then a float: -3

3.4

You entered -3 and 3.400000



Computer  
Programming  
Language

# Format Specifiers for I/O

Data Type	Format Specifier
int	%d
char	%c
float	%f
double	%lf
short int	%hd
unsigned int	%u
long int	%li
long long int	%lli
unsigned long int	%lu
unsigned long long int	%llu
signed char	%c
unsigned char	%c



Computer  
Programming  
Language



# C Operators

- An operator is simply a symbol that is used to perform operations.
- There can be many types of operations like arithmetic, logical, bitwise, etc.
- There are following types of operators to perform different types of operations in C language.
  - Arithmetic Operators
  - Relational Operators
  - Shift Operators
  - Logical Operators
  - Bitwise Operators
  - Ternary or Conditional Operators
  - Assignment Operator
  - Misc Operator



# Arithmetic Operators

- The following table shows all the arithmetic operators supported by the C language.
- Assume variable **A** holds **10** and variable **B** holds **20** then

Operator	Description	Example
+	Adds two operands.	$A + B = 30$
-	Subtracts second operand from the first.	$A - B = -10$
*	Multiplies both operands.	$A * B = 200$
/	Divides numerator by de-numerator.	$B / A = 2$
%	Modulus Operator and remainder of after an integer division.	$B \% A = 0$
++	Increment operator increases the integer value by one.	$A++ = 11$
--	Decrement operator decreases the integer value by one.	



Computer  
Programming  
Language

# Arithmetic Operators

## EXAMPLE:

```
1. #include <stdio.h>
2. int main()
3. {
4.     int a = 9,b = 4, c;
5.     c = a+b;
6.     printf("a+b = %d \n",c);
7.     c = a-b;
8.     printf("a-b = %d \n",c);
9.     c = a*b;
10.    printf("a*b = %d \n",c);
11.    c = a/b;
12.    printf("a/b = %d \n",c);
13.    c = a%b;
14.    printf("Remainder when a divided by b = %d \n",c);
15.    return 0;
16. }
```

## Output

```
a+b = 13
a-b = 5
a*b = 36
a/b = 2
Remainder when a divided by b=1
```



Computer  
Programming  
Language

# Relational Operators

Operator	Description	Example
<code>==</code>	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	<code>(A == B)</code> is not true.
<code>!=</code>	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	<code>(A != B)</code> is true.
<code>&gt;</code>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	<code>(A &gt; B)</code> is not true.
<code>&lt;</code>	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	<code>(A &lt; B)</code> is true.
<code>&gt;=</code>	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	<code>(A &gt;= B)</code> is not true.
<code>&lt;=</code>	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	



Computer  
Programming  
Language

# Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

```
1. #include <stdio.h>
2. int main()
3. {
4.     int a = 5, b = 5, c = 10;
5.     printf("%d == %d is %d \n", a, b, a == b);
6.     printf("%d == %d is %d \n", a, c, a == c);
7.     printf("%d > %d is %d \n", a, b, a > b);
8.     printf("%d > %d is %d \n", a, c, a > c);
9.     printf("%d < %d is %d \n", a, b, a < b);
10.    printf("%d < %d is %d \n", a, c, a < c);
11.    printf("%d != %d is %d \n", a, b, a != b);
12.    printf("%d != %d is %d \n", a, c, a != c);
13.    printf("%d >= %d is %d \n", a, b, a >= b);
14.    printf("%d >= %d is %d \n", a, c, a >= c);
15.    printf("%d <= %d is %d \n", a, b, a <= b);
16.    printf("%d <= %d is %d \n", a, c, a <= c);
17.    return 0;
18. }
```



Computer  
Programming  
Language

# Relational Operators

## Output

```
5 == 5 is 1
5 == 10 is 0
5 > 5 is 0
5 > 10 is 0
5 < 5 is 0
5 < 10 is 1
5 != 5 is 0
5 != 10 is 1
5 >= 5 is 1
5 >= 10 is 0
5 <= 5 is 1
5 <= 10 is 1
```



Computer  
Programming  
Language

# Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
&&	Logical AND. True only if all operands are true	If c = 5 and d = 2 then, expression <code>((c==5) &amp;&amp; (d&gt;5))</code> equals to 0.
	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression <code>((c==5)    (d&gt;5))</code> equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression <code>!(c==5)</code> equals to 0.



Computer  
Programming  
Language

```
1. #include <stdio.h>
2. int main()
3. {
4.     int a = 5, b = 5, c = 10, result;
5.     result = (a == b) && (c > b);
6.     printf("(a == b) && (c > b) is %d \n", result);
7.     result = (a == b) && (c < b);
8.     printf("(a == b) && (c < b) is %d \n", result);
9.     result = (a == b) || (c < b);
10.    printf("(a == b) || (c < b) is %d \n", result);
11.    result = (a != b) || (c < b);
12.    printf("(a != b) || (c < b) is %d \n", result);
13.    result = !(a != b);
14.    printf("!(a != b) is %d \n", result);
15.    result = !(a == b);
16.    printf("!(a == b) is %d \n", result);
17.    return 0;
18. }
```

## Output

```
(a == b) && (c > b) is 1
(a == b) && (c < b) is 0
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a != b) is 1
!(a == b) is 0
```



# Computer Programming Language



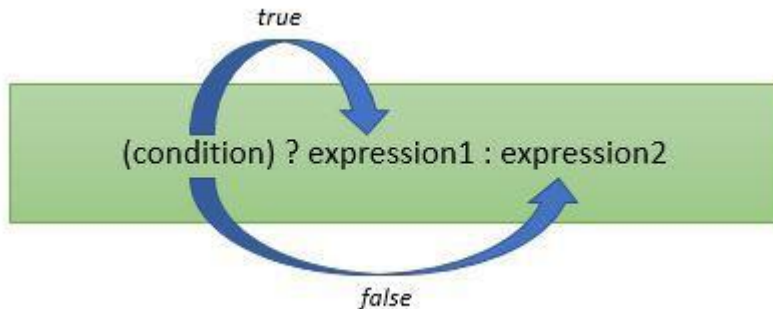
- $(a == b) \ \&\& \ (c > 5)$  evaluates to 1 because both operands  $(a == b)$  and  $(c > b)$  is 1 (true).
- $(a == b) \ \&\& \ (c < b)$  evaluates to 0 because operand  $(c < b)$  is 0 (false).
- $(a == b) \ || \ (c < b)$  evaluates to 1 because  $(a = b)$  is 1 (true).
- $(a != b) \ || \ (c < b)$  evaluates to 0 because both operand  $(a != b)$  and  $(c < b)$  are 0 (false).
- $!(a != b)$  evaluates to 1 because operand  $(a != b)$  is 0 (false). Hence,  $!(a != b)$  is 1 (true).
- $!(a == b)$  evaluates to 0 because  $(a == b)$  is 1 (true). Hence,  $!(a == b)$  is 0 (false).



# conditional operator

**(condition) ? expression1 : expression2**

If the **condition** is true then **expression1** is executed  
else **expression2** is executed



**For example:**

```
( x > y ? "x is greater" : "y is greater");
```

Computer  
Programming  
Language

# Example

```
#include <stdio.h>
int main()
{
int mark;
printf("Enter mark: ");
scanf("%d", &mark);
puts(mark >= 40 ? "Passed" : "Failed");
return 0;
}
```

## Output

```
Enter mark: 39
Failed
```



Computer  
Programming  
Language

# Type Conversion in C

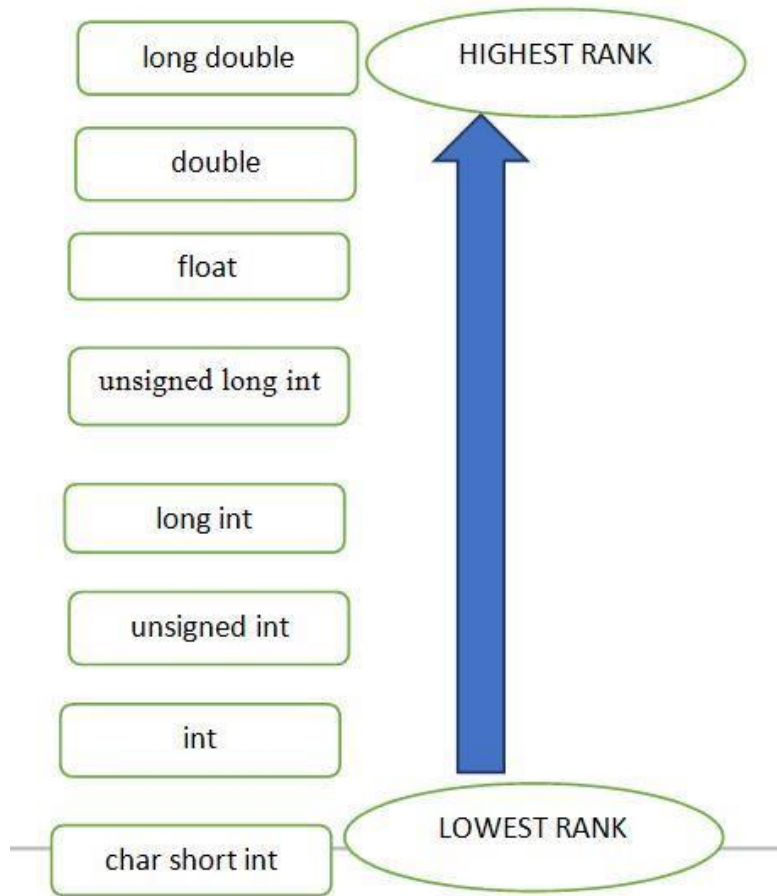
- The type conversion process in C is basically converting one type of data type to other to perform some operation.
- The conversion is done only between those datatypes wherein the conversion is possible
- EX – char to int and vice versa.
- There are two types of type conversion
  - 1) **Implicit Type Conversion**
  - 2) **Explicit Type Conversion**



# 1. Implicit Type Conversion

- This type of conversion is usually performed by the compiler when necessary without any commands by the user. Thus it is also called "**Automatic Type Conversion**".
- The compiler usually performs this type of conversion when a particular expression contains more than one data type.
- In such cases either type promotion or demotion takes place.





Whenever the compiler deals with different data types in an expression, the operand which is present at the lower rank will be converted to the corresponding datatype of the operand with the higher rank.



# Rules...

1. char or short type operands will be converted to int during an operation and the outcome data type will also be int.
2. If an operand of type long double is present in the expression, then the corresponding operand will also be converted to long double same for the double data type.
3. If an operand of float type is present then the corresponding operand in the expression will also be converted to float type and the final result will also be float type.
4. If an operand of unsigned long int is present then the other operand will be converted to unsigned long int and the final result will also be unsigned long int.
5. If an operand of long int is present then the other operand will be converted to long int and the final result will also be long int.
6. If an operand of unsigned int is present then the other operand will be converted to unsigned int and the final result will also be unsigned int.



Computer  
Programming  
Language

# Example

```
int a = 20;
```

```
double b = 20.5;
```

```
a + b;
```

- Here, first operand is int type and other is of type double.
- So, as per rule 2, the variable a will be converted to double.
- Therefore, the final answer is double  $a + b = 40.500000$ .



Computer  
Programming  
Language



## 2) Explicit Type Conversion

- Explicit type conversion rules out the use of compiler for converting one data type to another instead the user explicitly defines within the program the datatype of the operands in the expression.



## 2) Explicit Type Conversion

### Example:

```
double da = 4.5;
double db = 4.6;
double dc = 4.9;

//explicitly defined by user
int result = (int)da + (int)db + (int)dc;
printf("result = %d", result);
```

### Output:

1. Thus, in the above example we find that the output result is 12 because in the result expression the user has explicitly defined the operands (variables) as integer data type.
2. Hence, there is no implicit conversion of data type by the compiler. If in case implicit conversion was used the result would be 13.



Computer  
Programming  
Language

# C - Type Casting

- Type casting is a way to convert a variable from one data type to another data type.
- For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int'.
- You can convert the values from one type to another explicitly using the **cast operator** as follows –
- (type\_name) expression



# Example:

```
#include <stdio.h>
main() {
int sum = 17, count = 5;
double mean;
mean = (double) sum / count;
printf("Value of mean : %f\n", mean );
}
```

- When the above code is compiled and executed, it produces the following result
- Value of mean : 3.400000



Computer  
Programming  
Language

# Integer Promotion

Integer promotion is the process by which values of integer type "smaller" than **int** or **unsigned int** are converted either to **int** or **unsigned int**.

```
#include <stdio.h>
main() {
int i = 17;
char c = 'c'; /* ascii value is 99 */
int sum;
sum = i + c;
printf("Value of sum : %d\n", sum );
}
```

Value of sum : 116



Computer  
Programming  
Language



THANK  
YOU



Computer  
Programming  
Language