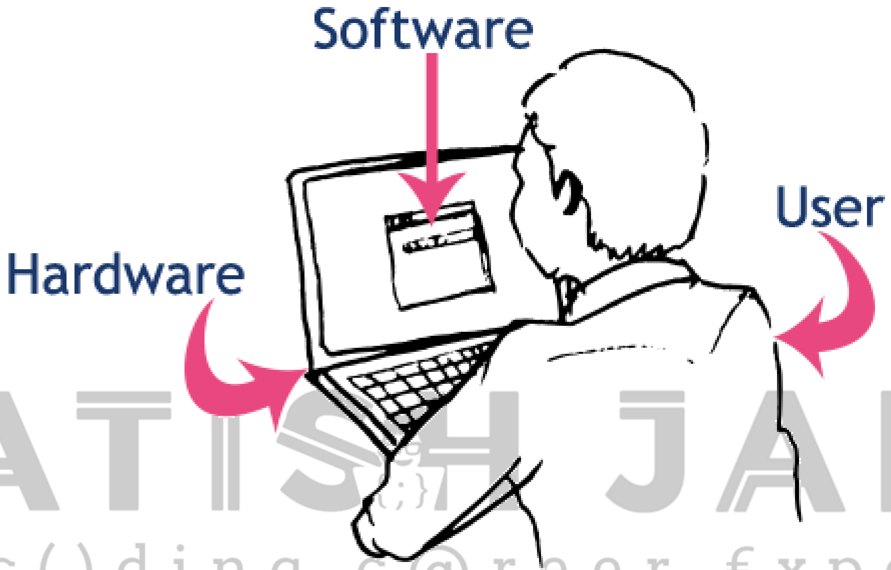
What is Computer?

A computer is a machine that can be programmed to execute sequences of arithmetic or logical operations automatically.

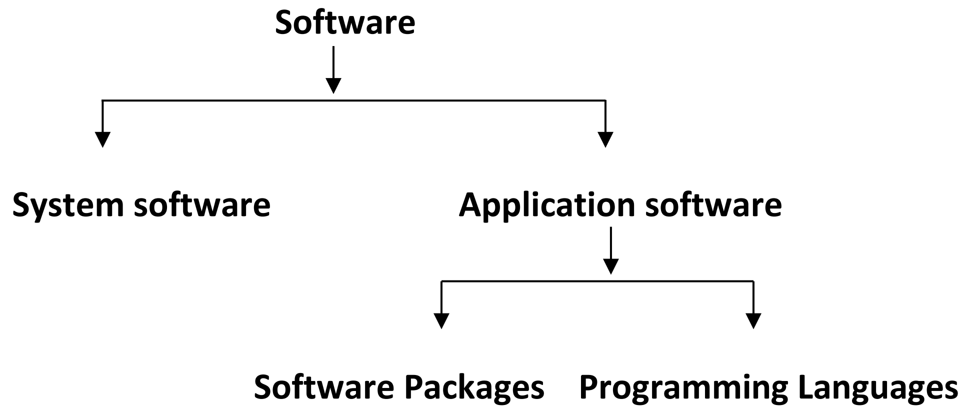
What are the Components of a Computer System?

The components of a computer system are **hardware** (**physical components**), **software** (**programs and applications**), and the **user** (**person interacting with the system**).



Software:

- ✓ Software is the collection of programs and data that instruct a computer on how to perform specific tasks.
- ✓ There are two types of software:
  - System software
  - Application software



System software:
System software, also known as an operating system (OS), initializes and manages the hardware components of a computer system to enable its functionality. Examples include **DOS**, **Windows**, **UNIX**, **Mac OS** and Linux.

Application software:
Application software refers to programs that perform specific tasks for users, such as word processing, web browsing, or gaming.

Application software can be broadly categorized into:

Software Packages: These are pre-written, ready-to-use software applications designed to perform specific tasks or functions, such as word processing (Microsoft Word), spreadsheets (Microsoft Excel), graphic design (Adobe Photoshop), etc.

Programming Languages: Programming languages such as Python, Java, and C++ are essential tools for developers to write code, design software solutions, and customize existing applications to meet specific computational needs or user requirements.

Difference Between Software Packages & Programming Languages:

Software Packages	**Programming Languages**
1. Set of pre-written programs	1. Set of keywords
2. Develops application software	2. Develops system software
3. Programming boundaries	3. No programming boundaries
4. User-friendly	4. Not user-friendly
Examples: MS-Office, Tally etc	Examples: C, Java Python etc

What is Programming?

- ✓ Programming refers to the process of creating and writing sequences of instructions, often using a programming language, to instruct a computer to perform specific tasks or operations.
- ✓ Programming languages are tools that allow programmers to communicate instructions to computers, similar to how natural languages like English and Hindi are used to communicate between humans. Examples include C, Python, and Java etc.

Steps in Programming:

1. **Logic Preparation:** Plan and strategize to solve the problem.
2. **Coding:** Write the program logic in a specific programming language.
3. **Compilation:** Convert the written source code into binary machine code using a compiler or interpreter.
4. **Execution:** Pass the machine code instructions to the CPU to obtain the desired output.

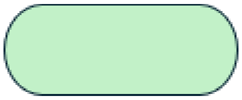
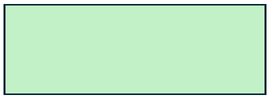
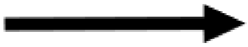
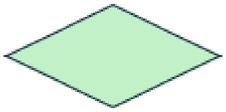
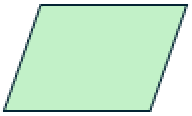
Algorithm:

An algorithm is a sequence of instructions used in computing. It can be expressed in two main ways:

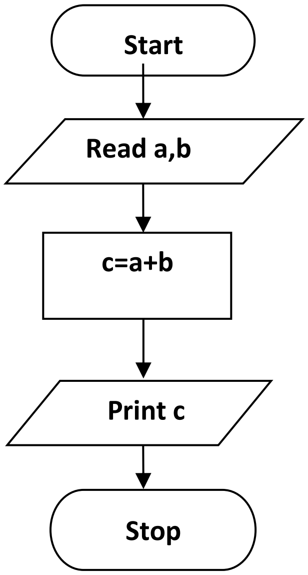
Pseudocode: Simplified language for describing a program's workings, making it easier for non-programmers to understand.

Flowchart: Graphical representation of an algorithm using standard symbols.

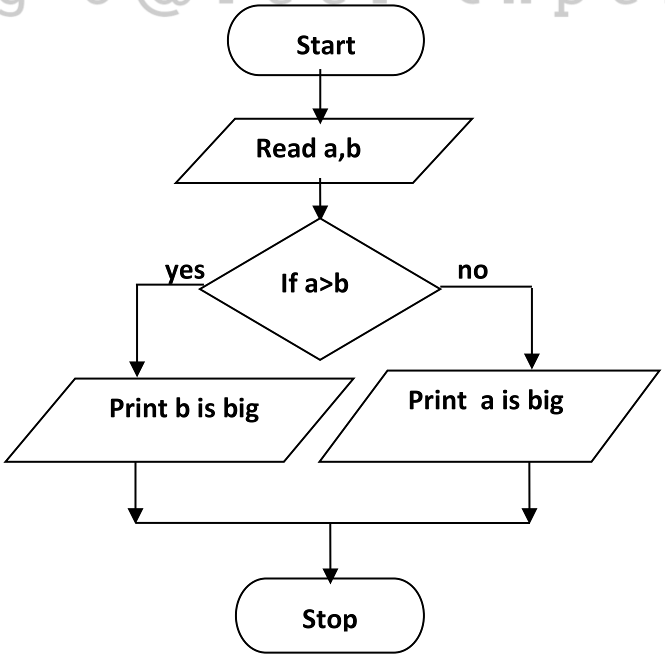
Flowchart Symbos:

Symbol	Name	Function
	Oval	Represents the start or end of a process
	Rectangle	Denotes a process or operation step
	Arrow	Indicates the flow between steps
	Diamond	Signifies a point requiring a yes/no
	Parallelogram	Used for input or output operations

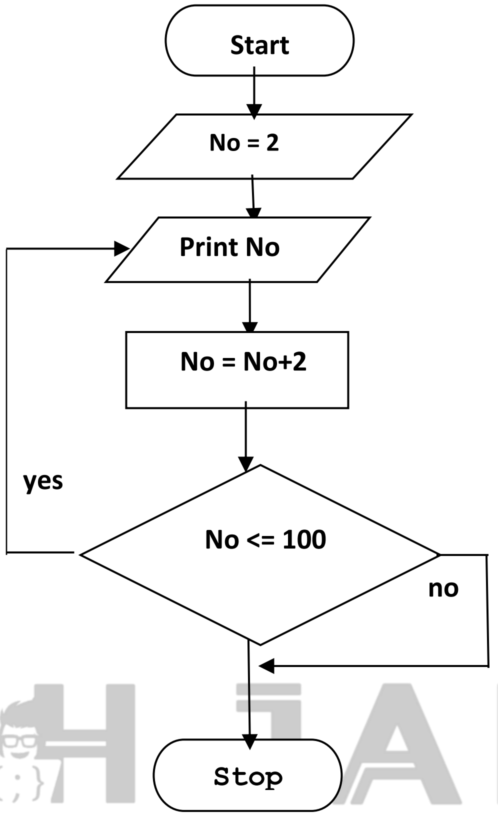
1. Pseudocode and Flowchart to add 2 numbers

Pseudocode	Flowchart
Step1: start Step2: read a,b Step3: c=a+b Step4: print c Step5: stop	

2. Pseudocode and flowchart to find biggest of 2 numbers

Pseudocode	Flowchart
Step1: start Step2: read a,b Step3: if a>b then print a is big else print b is big Step4: stop	 Start Read a,b If a>b yes no Print b is big Print a is big Stop

3. Pseudocode and flowchart to print even nos. till 100

Pseudocode	Flowchart
Step1: start Step2: no=2 Step3: while no<=100 Print no no=no+2 end Step4: stop	 Start No = 2 Print No No = No+2 No <= 100 yes no Stop

Assignments:

1. Write the Pseudocode and draw the corresponding flowchart to calculate Interest for the Input principle amount, time and rate of interest.
2. Write the Pseudocode and draw the corresponding flowchart to check whether the student has passed or failed from the input marks for 3 subjects.
3. Write the Pseudocode and draw the corresponding flowchart to print 5th table till 12.

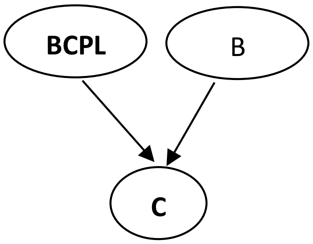
Solutions for Assignments:

1. https://drive.google.com/file/d/1yaxNgvKAIJwB6x-8LVdnquOiKIHWcPmu/view
2. https://drive.google.com/file/d/1o5yiTSo1z_LGvjdFM3CgunjNDUG1BHuq/view
3. https://drive.google.com/file/d/17AUa1SqIDbh-2Z6OoiwwBt8dn_JEPq_Q/view

Introduction to C Language

- ✓ **C** is a **programming language** developed by **"Dennis Ritchie"** at **AT & T Bell Laboratories** of **USA** in **1972**.
- ✓ AT&T Bell Labs, commonly known as Bell Labs, is the research and development division of AT&T
- ✓ C is derived from two other languages **BCPL** and **B**.

C = Good features of **BCPL** + **B** + some new features.



BCPL - 1967 – Martin Richards at Cambridge University.
B - 1970 – Ken Thompson at AT & T Bells labs.

BCPL – **B**asic Combined **P**rogramming **L**anguage.
AT&T – **A**merican **T**elephone and **T**elegraph

Features of C language:

1. Portability:
C programs are portable, they can be transferred to any other system for execution.

2. Modularization/structured language:
Lengthy program can be divided into modules. It gives maintenance benefits

3. Middle level language:
It is the very important feature of C language. All the programming languages are divided into two categories.

Low level languages:
Direct interaction with the hardware, which gives better Machine efficiency.
Ex: Machine language and Assembly language.

High level language:
Programs are written using simple English language which gives Better programming efficiency.
Ex: BASIC, COBOL, C, CPP, Java etc.

C has the features of both high level and low level language. That’s why C is known as middle level language.

4. General purpose language:

C can be used for system programming as well as application Programming.

5. Case sensitive

It means Capital letters are different from small letters.

6. Function oriented:

Entire program is written as a collection of functions. There are nearly 400+ functions in c language.

7. Weakly typed language:

There is no strict datatype checking. We can convert integer to float and float to integer.

8 . Liberal and Error prone:

C doesn’t check all errors.

The other programming domains are:

- Embedded systems programming.
- Game programming.
- Network programming.

Most of the software in the world were developed using C/C++. world’s software giant’s Microsoft, Borland develop most of their software’s using C and C++.

Anything that can be done on computer can be done using C language.

C language was invented to develop a operating system, i.e UNIX.

History of C language

Year	Language	Developed by	Remarks
1960	ALGOL	International committee	Too general, too abstract
1963	CPL	Cambridge university	Hard to learn, difficult to implement
1967	BCPL	Martin Richards at Cambridge university	Could deal with only specific problems
1970	B	Ken Thompson at AT&T Bell labs	Could deal with only specific problems
1972	C	Dennis Ritchie at AT&T bell labs	Taken the good features of BCPL and B

Structure of C program:

```
/*-----------------*/
# include <header files>
   <global declarations>;
main()
{
  <local declarations>;

   <statement/s>;
}
```

C programs are written by using number of functions and keywords.
There are 32 keywords and 29 header files in C language.(may vary from compiler to compiler)
All the functions are stored in appropriate header files.

/*--*/ - comment lines:

Comment lines are used to give comments to the program. Contents of a comment are ignored by the compiler.

# include – Pre processor directive

Pre-processor means before processing, it tells the compiler to include another program or file along with our source code.

Header file:

The C compiler provides a library of standard pre programmed functions. The developer of the C compiler wrote these functions, each function is declared in a header file having extension as .h

main():

This is a special function. It is the first function that is called during the program execution. It indicates the starting point and ending point of the program. Every 'C' program should have exactly one main() function.

main() function acts like an engine which drives the whole program.

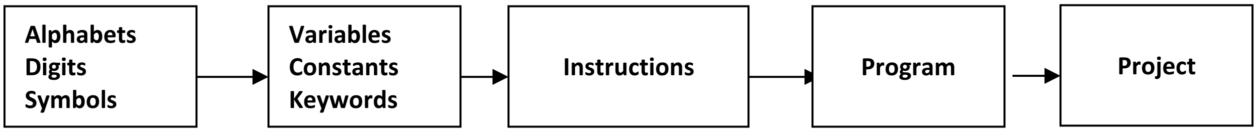
Easy way of learning C language:

Learning of computer programming language is exactly learning a human language.

Steps in learning human language:

Alphabets → words → sentences → Paragraph → essay

Steps in learning C language:



Variable:

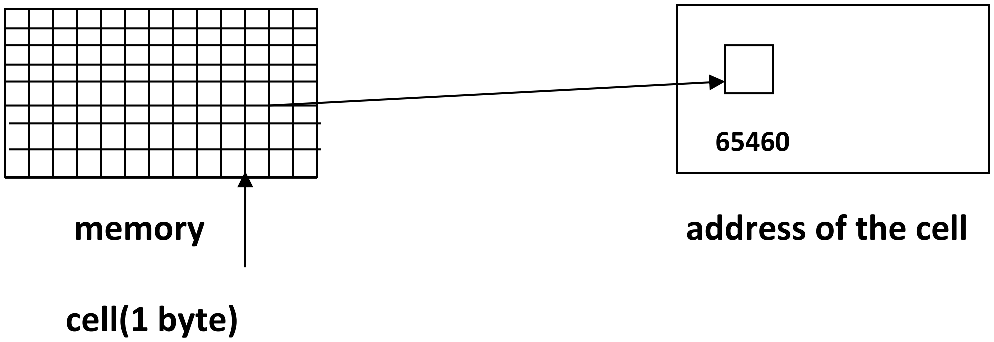
Name of the memory location. Variables are used to store constant values

Need of memory:

- ✓ A program is meant for processing data and producing results.
- ✓ Microprocessor processes data.
- ✓ Microprocessor can process data only if the data is available in RAM.
- ✓ To store the data in RAM, we should reserve required amount of space in the memory.

Memory:

Computer memory is made up of millions of tiny cells, each cell represents a byte, and every byte will have a unique address.



Byte:

It is the term which is used to measure the computer's memory.

Constant:

The value of a constant doesn't change. It is fixed.

Keywords:

Keywords are the words whose meaning is pre-defined to the compiler. Keywords are the reserved words.

Datatype:

It specifies what type of data that a variable will hold.

Basic Datatypes are:

Datatype	Bytes	Format specifier	Range
int	4	%d	
char	1	%c	-128 to +127
float	4	%f	-
double	8	%lf	-

The memory occupied by variables in C can vary across different compilers, operating systems, and hardware architectures due to differences in implementation, alignment, and optimization strategies.

Note:

The difference between float and double is the precision.

float – 6 digits.

double – 15 digits.

Rules for writing programs:

1. Program should be written in lower case only
2. All the statements must be terminated by semicolon(;).
3. All the statements should be in between the curly braces.
4. Every c program must be saved with a file having extension .c
5. Variables must be declared before using in the program.

Programming Elements:

- Identifiers
  - Names of variables/functions
- Standard datatypes
  - To specify the type of data.
- Constants
  - Constant is an entity that doesn’t change.
  - Ex: character, Integer, float, String

Rules for variable names:

1. Variable name should not exceed more than 8 characters.
   (However it is compiler dependent)
2. First character should be an alphabet.
3. Keywords cannot be used as variable names.
4. No special symbols other than underscore.
   a. Ex: net_sal
5. Spaces are not allowed.
6. Variable names must be unique.

Control flow/programming constructs:

Programming constructs, or control flow statements, are essential for controlling a program's execution flow. There are three main types:

- ✓ Sequence
- ✓ Selection
- ✓ Loops

Sequence:

Sequence denotes the execution of all the statements one by one from Top to Bottom. Sequence is the default control flow.

Arithmetic operators -

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (for Remainder)

About IDE's and Compiler's:

IDE - Integrated Development Environment in which one can develop, run, test and debug the application.

IDE's for windows environment:

- Bloodshed **Dev-C++** is a full-featured Integrated Development Environment (IDE) for the C/C++ programming language. It uses Mingw port of **GCC (GNU Compiler Collection)** as it's compiler. F11 – compile and run
- **The codeblocks-13.12mingw-setup.exe** file includes the GCC compiler and GDB debugger from TDM-GCC (version 4.7.1, 32 bit). F9 – compile and run
- **C-Free** is a professional C/C++ integrated development environment (IDE) that support multi-compilers. Use of this software, user can edit, build, run and debug programs.F5 compile and run

An IDE is (**I**ntegrated **D**evelopment **E**nvironment) an integration of the following tools.

- Editor - To write/modify source
- Compiler - Creates .o file
- Linker - Links .o and .lib to create .exe files
- Debugger - To execute line by line

Input/output functions:

The function which is used to display output on the console(screen) is known as Output function.

E.g. printf(); -- It is a readymade function/library function

- Syntax - For displaying a message.
  printf("……………………………");

  E.g. printf("Welcome to C Programming Course");

- Syntax - For displaying a value.
  printf("format specifier",variable);

  E.g. printf("%d",no);

- Syntax - For displaying a message and value.
  printf("……………………………format specifier",variable);

  E.g. printf("The value is: %d",no);

The function which is used to accept data from the user is known as Input function.

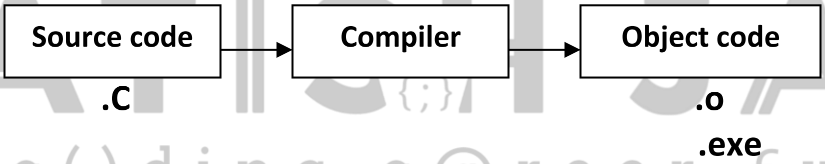
E.g scanf(); -- It is a readymade function/library function

- Syntax - scanf("format specifier",&variable);
  E.g. scanf("%d",&no);

Create a workspace folder in **D:** Drive or **E:** Drive and start your coding journey.

Example: Program to display a Welcome message.

```
/* Program name: welcome.c
  Task: Displays a welcome message
  Author: Atish jain
  Company: AH CAREER ACADEMY
*/

 # include <stdio.h>
 int main()
 {
   printf("Welcome to AH CAREER ACADEMY");
   return 0;
 }
```



- **Compiling** is the process of converting source code into object code. (Alt+F9).
- **Linking** is the process of combining object code with library functions (F9).
- **Execution** is the process of passing the machine instructions to processor.

***** Compiler is a Software.**

For e.g. If the C program is saved with the name Hello, after the compilation it will create following files

Hello.c – contains the source code.
Hello.o – contains the object code.
Hello.exe – it is the executable file. The program can be executed without C software (without source code also).

What is Translator?

Translators are the programs which converts source code into machine code. The Different translators are.

- Assemblers – for Assembly language
- Compilers – for HLL
- Interpreter – for 4GL's

Compiler:

- ✓ The Whole program is compiled at once.
- ✓ A small change in the program requires whole program compilation.
- ✓ So, the Program can be executed only if it is translated.
- ✓ Ex: BASIC, COBOL, C, C++, Pascal, etc...

Interpreter:

- ✓ Code is translated line by line.
- ✓ Interpreter program and source code program both are loaded in the memory at a time.
- ✓ No copy of translation code exists, if the program has to be executed again, it has to be interpreted again.
- ✓ Ex: Javascript, Python etc....

Conclusion:

Compiled code: Program development is slow, Execution is fast.
Interpreted Code: Program development and testing is fast, execution is slow.

Variable Declaration & Initializations:

Variable are used to store the data in the computer memory.

```
//Program to demonstrate variable declaration and initilization
# include <stdio.h>
int main()
{
   int rollno=1001;
   char name[]="Raj";
   float avg=98.99;
   char gender='M';
   double height=5.6;

   printf("Rollno:%d",rollno);
   printf("\nName:%s",name);
   printf("\nAverage:%f",avg);
   printf("\nGender:%c",gender);
   printf("\nHeight:%lf",height);
   return 0;
}
```

- **Program to add two no's**

```
/* Program name: add2.c */

        # include <stdio.h>
        int main()
        {
          int a,b,c;
          a=b=c=0;

          printf("Enter first no:");
          scanf("%d",&a);
          printf("Enter second no:");
          scanf("%d",&b);
          c=a+b;
          printf("The Result is:%d",c);
          printf("\n%d + %d = %d",a,b,c);
          return 0;
         }
```

Program to find total and average for inputted 3 subjects

```
/* Program name: Avg.c */

        # include <stdio.h>
        int main()
        {
         int m,p,c;
         float t,a;

         printf("Enter marks for Maths:");
         scanf("%d",&m);
         printf("Enter marks for Physics:");
         scanf("%d",&p);
         printf("Enter marks for Chemistry:");
         scanf("%d",&c);

         t=m+p+c;
         a=t/3;
         printf("Total is:%.0f",t);
         printf("\nAverage is:%.2f %%",a);
         return 0;
         }
```

Note:

- ✓ %.2f is known as precision specifier, displays two digits after the decimal
- ✓ %.0f displays 0 digits after the decimal (no decimal places)

Program to swap values of two variables A and B

```
/* Program name: swap1.c */
        # include <stdio.h>
        int main()
        {
          int a,b,temp;
          printf("Enter value for A:");
          scanf("%d",&a);
          printf("Enter value for B:");
          scanf("%d",&b);
          temp=a;
          a=b;
          b=temp;
          printf("A is :%d",a);
          printf("\nB is :%d",b);
         return 0;
         }
```

Note: \n is known as escape sequence, it prints the output in new line

WAP to convert the input total minutes into hours and minutes.

```
         /* Program name: hours.c */
        # include <stdio.h>
        int main()
        {
          int tm,h,m;
          tm=h=m=0;
          printf("Enter total minutes:");
          scanf("%d",&tm);

          h=tm/60;
          m=tm%60;

          printf("Time is %d hrs and %d mins",h,m);
          return 0;
         }
```

Hands on-lab:

1. WAP to calculate the amount for the input rate and quantity.
   Note: Rate should be in decimals.
2. WAP to swap the values of two variables without using 3rd variable.
3. WAP to convert the inputted rupees into paise.
4. Write a program to convert the temperature from Fahrenheit to centigrade.
   Formula: c=(f-32)*5/9.
5. WAP to calculate Electricity bill from the input present month and last month reading.
   Note: Rate per unit is 2.50
6. Program to calculate gross salary and net salary from the input basic, allowances and deductions.

Working with characters:

char data type occupies 1 byte of space in the main memory.

WAP to accept a character and display the same.

```
# include <stdio.h>
 int main()
 {
        char ch;

        printf("Enter a character..:");
        scanf("%c",&ch);

        printf("The character is: %c",ch);

        return 0;
        }
```

Note: Only one character is stored in the memory, because char data type allocates only 1 byte.

WAP to accept a string and display the same.

```
# include <stdio.h>
int main()
{
 char name[50];
 printf("Enter u'r Name..:");
 scanf("%s",&name);

 printf("Hello, %s Welcome to World of Coding With Atish....",name);
return 0;
 }
```

Note: %s accepts string without spaces, ie %s terminates string whenever space occurs.

WAP to accept a string with spaces and display the same.

```
# include <stdio.h>
int main()
{
 char name[50];

 printf("Enter u'r Name..:");
 scanf("%[^\n]",&name);

 printf("Hello, %s Welcome to World of Coding With Atish....",name);
return 0;
 }
```

Note: % [^ \n] – keeps on reading the characters from the keyboard buffer till it encounters enter key('\n').

stop

Enter key

Whenever the data we pass it to the program or assign it to the variables is initially stored in keyboard buffer, when we press enter key then the data is passed into RAM.

sizeof operator:

sizeof operator is used to get the size of a variable or a datatype i.e number of bytes allocated.

```
# include <stdio.h>
main()
{
 int x;
 char y;
 double z;

 printf("Size of x is:%d\n",sizeof(x));
 printf("Size of int is:%d\n",sizeof(int));

 printf("Size of y is:%d\n",sizeof(y));
 printf("Size of char is:%d\n",sizeof(char));

 printf("Size of z is:%d\n",sizeof(z));
 printf("Size of double is:%d",sizeof(double));
 return 0;
}
```

ASCII codes:

- ✓ All input data is internally stored as integer (ASCII) values.
- ✓ ASCII – American standard code for information interchange.
- ✓ The data (input) is represented in binary format. Each and every data item has an ASCII code.

There are 256 ASCII codes starting from 0 – 255.

- ✓ A-Z – 65-90
- ✓ a-z – 97-122
- ✓ 0-9 48-57

* Remaining all the codes represents special characters.
(0-47, **48-57,** 58-64**, 65-90,** 91-96, **97-122,** 123-255)
(0-9) (A-Z) (a-z)

- **Write a program to display the ASCII code for the inputted character.**

```
# include <stdio.h>
int main()
{
 char ch;

 printf("Enter an input....:");
 scanf("%c",&ch);
 printf("The ASCII Code for (%c) is %d",ch,ch);
 return 0;
 }
```

- **Write a program to accept a no(0-255) and display its   ASCII character.**

```
# include <stdio.h>
int main()
{
 int code=0;

 printf("Enter an ASCII Code(0-255):");
 scanf("%d",&code);
 printf("The ASCII symbol for %d is %c",code,code);
  return 0;
 }
```

Error:

Error is a bug in the code.

There are three types of errors.

- Compile time errors.
- Logical errors.
- Runtime errors.

Compile time errors:

- These errors occur during the compilation of a program.
- These errors occur if we violate “C” language rules.
- The program can be compiled only after correcting these errors.
- E.g. missing semicolon / forgetting to define a variable.
- Detecting and correcting these errors is very easy.

Logical errors:

- These errors occur if there is any mistake in the logic of the program.
- These errors are not displayed on the screen.
- We can identify the logical error only if we know the result in advance.
- Detecting and correcting these errors is very difficult.

Runtime errors:

- The errors which occur during the execution of a program are called as run time errors.
- E.g. divide a number with zero / modifying illegal memory location.
- Detecting and correcting these errors is very difficult.

A software success depends on how far the runtime errors have been tackled. Most of the program development time is spent on detecting and correcting the runtime errors.

Garbage value (Unknown value):

```
int main()
{
 int n;
 printf("%d",n);
 return 0;
}
```

Output of the program will be unknown value.

Initialization and Assignment

- ✓ Initialization – one time
- ✓ Assignment – many times

- **Program to illustrate Initialization and Assignment**

```
int main()
{
 int n=10;     /* initialization */
 printf("\n%d",n);
 n=100;        /* assignment */
 printf("\n%d",n);
 return 0;
}
```

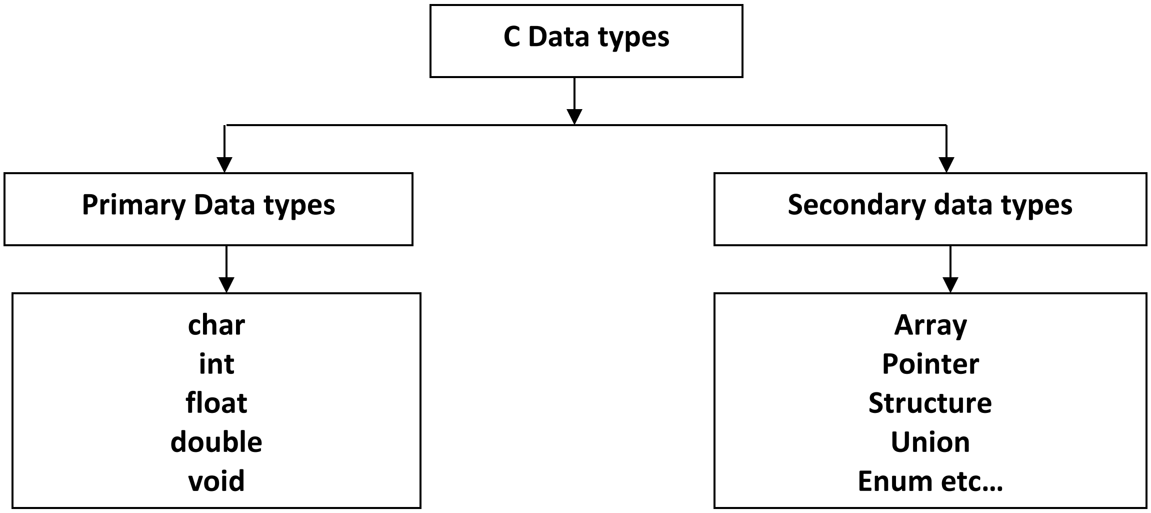
const keyword:
If the value of a variable is fixed and should not be changed even accidentally, then such a variable is defined as **const**.

Program to find area of a circle

```
int main()
{
   float radius,area;
   float const PI=3.14;

   printf("Enter Radius:");
   scanf("%f",&radius);

   area=PI*radius*radius;

   printf("Area of Circle is:%f",area);
   return 0;
}
```

Data types
The data type of a variable tells us the following things.

- Type of value that can be stored.
- Range of values.
- Size of a variable (i.e. number of bytes).



void type means no value. This is usually used to specify when the function return no value
ex: void main()

Syntax for declaring a variable:

[storage class] [modifiers] <data type> <variable> = [value]

Modifiers:

The modifiers signed, unsigned, short, long and long long are applied to get additional data types.

DATATYPE	SIZE	RANGE	FORMAT
Char	1	-128 to +127	%c
signed char	1	-128 to +127	%c
unsigned char	1	0 to 255	%c
int	2	-32768 to +32767	%d
short int	2	-32768 to +32767	%d
signed int	2	-32768 to +32767	%d
signed short int	2	-32768 to +32767	%d
unsigned int	2	0 to 65535	%u
unsigned short int	2	0 to 65535	%u
long	4	-2147483648 to +2147483647	%ld
long int	4	-2147483648 to +2147483647	%ld
long signed int	4	-2147483648 to +2147483647	%ld
long unsigned int	4	0 to 4294967295	%lu
Float	4	-3.4e38 to +3.4e38	%f
Double	8	-1.7e308 to +1.7e308	%lf
long double	10	-3.4e4932 to +1.1e4932	%Lf
long long	8	-	%lld
unsigned long long	8	-	%llu

Character datatypes

- ✓ char/signed char
- ✓ unsigned char

Integer datatypes:

- ✓ int/signed int/short int/short signed int
- ✓ unsigned int/short unsigned int

Long datatypes:

- ✓ long/long int/long signed int
- ✓ long unsigned int
- ✓ long long/long long int

Float datatypes:

- ✓ float

Double datatypes

- ✓ double
- ✓ long double

Escape sequence characters:

- \n – new line
- \t – tab space
- \a – alarm\alert
- \b – back space
- \r – Carriage Return
- \' – to get a single code
- \" – to get a double code
- \\ - to get back slash

Hands on-lab:

- ✓ Write a program to calculate Area of a circle.
- ✓ Write a program to print the Address in 3 lines in center of the screen using single printf() function.

  Output:
  ```
  AH CAREER Academy
      Danavaipeta
      Rajahmundry
  ```

Type casting:

Converting value of one data type to another type during runtime is known as type casting.

Type casting example

```
/* Program name : type1.c*/
 # include <stdio.h>
 int main()
 {
  int a=5,b=2;
  float c=0;
   c=a/b;
  printf("%f",c);
  return 0;
  }
```

Expected output is 2.5 but it will give 2.0, because...

Operand 1	Operand 2	Result
Integer	Integer	Integer
Integer	Float	Float
Float	Integer	Float
Float	Float	Float

So the above problem can be solved by declaring any of the input variables to float, or type casting....

```
/* program name : type2.c*/
 # include <stdio.h>
 int main()
 {
  int a=5,b=2;
  float c=0;
  c=(float)a/b;
  printf("%f",c);
  return 0;
  }
```

Datetype of **a** is temporarily Promoted to float.

Converting int variable to char type.

```
# include <stdio.h>
int main()
{
  int a;
  char b='a';
  a=b;
  printf("a=%d",a);
  return 0;
 }
```

Note:
C automatically does the conversion on its own. Still there are cases where we have to do it explicitly.

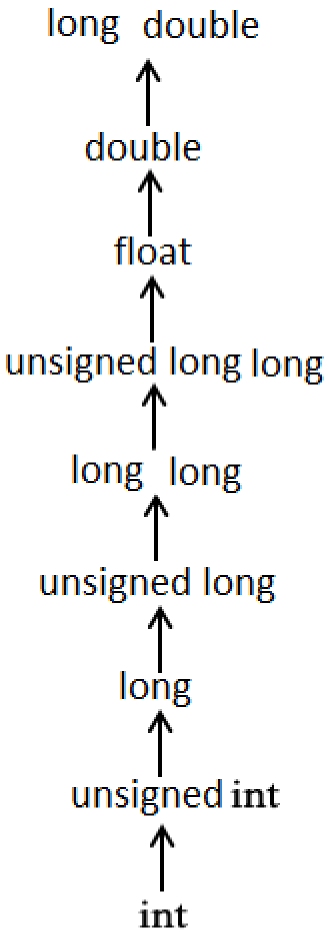
Integer Promotion:
Integer promotion is the process by which values of integer type "smaller" than int or unsigned int are converted either to int or unsigned int. Consider an example of adding a character in an int:

```
#include <stdio.h>
int main()
{
  int x=100;
  char c = 'c'; /* ascii value is 99 */
  int sum;

  sum = x + c;
  printf("Value of sum : %d\n", sum );
  return 0;
}
```

- ✓ When the above code is compiled and executed, it produces the following result:
- ✓ Value of sum : 199
- ✓ Here value of sum is coming as 199 because compiler is doing integer promotion and converting the value of 'c' to ascii before performing actual addition operation.

Usual Arithmetic Conversion:

The usual arithmetic conversions are implicitly performed to cast their values in a common type. Compiler first performs integer promotion, if operands still have different types then they are converted to the type that appears highest in the following hierarchy:



long double
↑
double
↑
float
↑
unsigned long long
↑
long long
↑
unsigned long
↑
long
↑
unsigned int
↑
int

Control structures:

Till now the style of programming we have used is sequence, where

- All the statements are executed only once.
- All the statements are executed one after the other.

Whenever we want to control the sequence of the program execution, then we have to use Control structures.

Control structures are divided in to 2 types.

- Selection statements.
- Repetition statements.

Selection statements:

Whenever we want to execute a set of statements based on a condition, then we have to use Selection statements.

Comparison operators:

>	Greater than
>=	Greater than equal to
<	Less than
<=	Less than equal to
==	Equal to
!=	Not equal to

Note:

= is used for Initialization whereas == is used for Comparison.

Logical operators:

Logical operators are used to combine the conditions.

&&	And
||	Or
!	Not

if and **else** are the two keywords which are used to construct selection type of programming.

Selection statements are divided into 4 types

- Simple if
- If with else
- Multiple if's
- Nested if

Simple if

Executes the statements if the condition is true.

Syntax:

```
        if <condition>
          {
            stmt1;
            stmt2;
            stmtN;
          }
```

If the condition is satisfied (i.e. holds true) then if block statements (stmt1,stm2,..) are executed.

Program to check whether a person is eligible for vote or not.

```
# include <stdio.h>
int main()
{
  int age=0;

   printf("Enter your age:");
   scanf("%d",&age);

    if(age>=18)
      {
        printf("Eligible for vote...");
      }
   return 0;
 }
```

If with else:

Executes if block statements if the conditions is true otherwise else block statements are executed

Syntax:

```
          if <condition>
             {
                     stmt1;                  }             IF block
                     stmt2;
             }
          else
             {
                      stmt3;                 }             ELSE block
                      stmt4;
             }
```

Note:

Curly braces are optional if there is a single statement in **if block** or **else block.**

Program to check whether the student has passed or failed in an examination.

```
  # include <stdio.h>

int main()
{
        float avg=0;

        printf("Enter average marks:");
        scanf("%f",&avg);

        if(avg>=40)
          printf("Student passed...");
        else
          printf("Student failed...");

        return 0;
}
```

Program to demonstrate curly braces importance

```
# include <stdio.h>
int main()
{
   float avg=0;
   printf("Enter average marks:");
   scanf("%f",&avg);

    if(avg>=40)
      {
        printf("Ur passed...");
        printf("\nWell done...");
      }
    else
     {
       printf("Ur failed...");
       printf("\nBetter Luck Next time...");
     }
     printf("\nThank you!");
     return 0;
 }
```

Hands on-lab:

1. WAP to check whether the inputted no is even or odd.
2. WAP to find the biggest of 2 no's.
3. WAP to calculate the discount and final amount based upon the data given below :-
    a. If sales>= 25000 then 10% is allowed other wise 5% discount is allowed.
4. WAP to accept present month reading and previous month reading from the user and calculate the amount based upon the data given below:
    a. Units Consumed Rate
    b. >100 1.50
    c. <100 0.80
5. WAP to check whether the inputted year is a leap year or not .
6. WAP to accept names and ages of two persons and display the elder person.

Multiple If's:

checks for multiple conditions and executes the statements.

Syntax:

```
    if <condition 1>
      {
            stmt1;                          IF block
            stmt2;
      }
    else
      if <condition 2>
            {
                  stmt3;                    ELSE block
                  stmt4;
               }
        else
            if <condition N>
                  {
                        stmt5;
                        stmt6;
                   }
            else
                  {
                         stmt7;
                         stmt8;
                   }
```

- **Program to print the final result of a student depending upon his average marks, using the data given below.**

<u>**Average marks**</u>	<u>**Result**</u>
Above 60%	**First class**
50 – 60	**Second class**
40 – 50	**Third class**
Below 40	**Fail**

```
# include <stdio.h>

int main()
{
 float avg=0;

 printf("Enter average marks:");
 scanf("%f",&avg);

  if(avg>=60)
    printf("First class...");
  else
            if(avg>=50)
            printf("Second class...");
        else
              if(avg>=40)
                  printf("Third class...");
               else
                      printf("FAIL...");
return 0;
}
```

Hands on-lab:

1. WAP to check whether the inputted no is Positive or Negative or Neutral.
2. WAP to accept marks of 3 subjects and perform the following:
   Calculate Total and Average and display the Grade based upon the data given below.

Average	**Grade**
Above 90%	A+
80 – 90	A
70 – 80	B+
60 – 70	B
50 – 60	C
Below 50	F

3. WAP to print the discount and the actual amount from the given details.

Sales	Discount
>=25000	25%
20000 – 25000	20%
10000 – 20000	10%
5000 – 10000	5%
Below 5000	no discount

4. WAP to accept basic salary, allowances, deductions and experience of an Employee and perform the following -
   i. Calculate gross and net salary.
   ii. Calculate bonus from experience
      a. If experience is > 5 years then 3 months net salary is given as bonus
      b. If experience is >3 years then 2 months net salary is given as bonus
      c. Otherwise 1 month of net salary is given as bonus.

Nested if's

if with in if is known as nested if

Syntax-

```
if <condition>
      if <condition>
              {
                      stmt1;
                      stmt2;
              }
      else
              {
                      stmt3;
                      stmt4;
                      }
else
      {
      stmt5;
      stmt6;
      }
```

Program to find biggest of 3 no’s

```
# include <stdio.h>

int main()
{
 int a,b,c;

 printf("Enter 3 no's:");
 scanf("%d%d%d",&a,&b,&c);

  if(a>b)
           if(a>c)
                      printf("Biggest no is:%d",a);
           else
                      printf("Biggest no is:%d",c);
  else
            if(b>c)
                      printf("Biggest no is:%d",b);
      else
                      printf("Biggest no is:%d",c);

 printf("\n");
 return 0;
 }
```

Hands on-lab:

1. WAP to find the Biggest of 3 no’s using logical operators.
2. WAP to check whether the inputted character is capital letter or small letter or digit or special character.
3. WAP to check whether the inputted character is a vowel or a consonant.
   Use: toupper() or tolower() function
4. WAP to check whether the person is eligible for concession in Railways or not (using ! operator).
5. WAP to check whether the person is eligible for bonus or not based upon the data given below . Bonus is given only to the following persons.-
   i. Persons who are married.
   ii. Unmarried Male above 30 years.
   iii. Unmarried Female above 25 years.
6. Solve the above program using logical operators

7. South central railway gives 50% concession to its passengers those who fulfill the following eligibility criteria.
   i. Male above 60 years of age
   ii. Female above 50 years of age
   iii. Full concession for below 5 years (no ticket is required).

WAP to check if a passenger is eligible for concession based on the above data.

8. ABC ltd company sells TV’s to their customers on cash and credit basis. If the customer pays cash, then 25% of discount is allowed. If a customer buys on credit and paid within 7 days then 15% of discount is allowed else 10% extra is charged. WAP to generate final bill of a customer.

9. WAP to prepare electricity bill from the following data

Units consumed	**Domestic**	**Commercial**
<100	Rs. 100	Rs.150
100 – 200	Rs. 1.50 PU	Rs.2.50 PU
200 – 300	Rs. 3.00 PU	Rs.4.50 PU
>=300	Rs. 5.00 PU	Rs.7.50 PU

10. WAP to accept marks of c, cpp, java and calculate total, average, class only if the student passes in all the 3 subjects otherwise display the subject in which he is failed and message as “no grade”.

Ternary operators/Conditional operators

? and : are called ternary operators because it accepts three arguments.
Ternary operators are also known as conditional operators.

Syntax -

```
( expr1 ? expr2 : expr3);
   true           false
    or              or
non-zero       zero
```

If **expr1** is true (non zero) then **expr2** is executed otherwise **expr3** is executed.

Note:
Ternary operators are used instead of if-else statements.

Program to find biggest of 2 no's using ternary operators

```
# include <stdio.h>
int main()
{
 int a,b;
 a=b=0;

 printf("Enter Values for A and B:");
 scanf("%d%d",&a,&b);

 (a>b? printf("Big is:%d",a) : printf("Big is:%d",b) );

 printf("\n");
 return 0;
 }
```

Hands on-lab:
WAP to check whether the input number is even or odd using Ternary operators.

Switch case

- ✓ The control statement which allows us to make a decision from the number of choices is called a switch-case statement.
- ✓ Switch case is replacement for multiple if’s.

Syntax -

```
switch( <expr> )
 {
           case <val 1> : <stmt 1>;
                  break;
              case <val 2> : <stmt 2>;
                  break;
            .
            .
              case <val N> : <stmt N>;
                  break;

           default: < default statement>;
 }
```

Hands on-lab:

WAP to implement the below using Switch case.

```
Enter a no..:
Enter another no..:

Enter ur option(+,-,*,/,E)..:
```