Introduction to C

History:

- C is a Programming language developed by Dennis Ritchie in 1972.
- It is mainly used in development of Operating Systems, Embedded software, Device drivers, application software etc.

Programming language:

- Programming languages are used to develop applications.
- Computer applications **Store data and perform operations on data**.

For example: Banking application store Accounts data and perform transactions like withdraw, deposit etc.

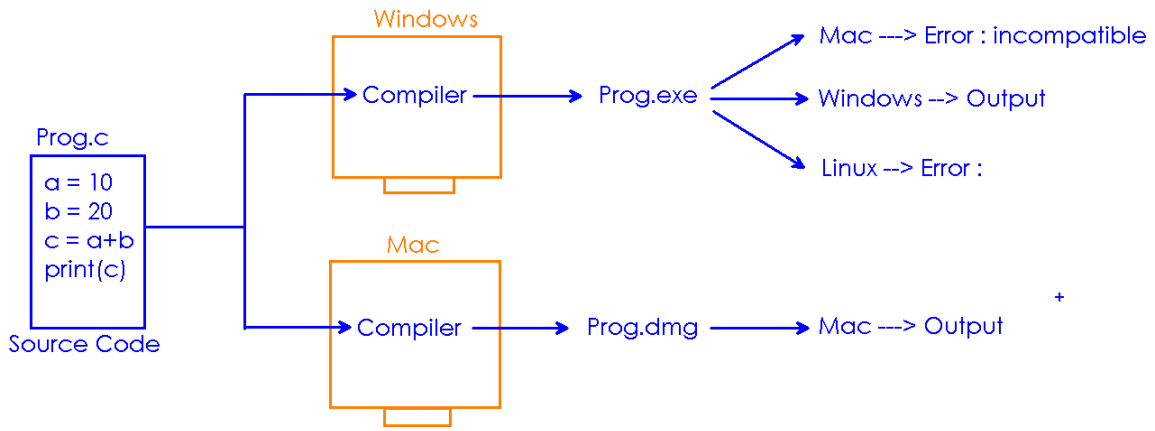
Applications: are mainly two types

1. **Standalone application:**
    - Application runs from single machine.
    - Data connection (internet) is not required to run.
    - Standalone applications are
        - **System software:** Controls Hardware components and other software components. For example, Operating System.
        - **Application software:** is used to perform specific user task. For example, MS-Office, VLC Player, Browser etc.

Note: C is Platform Dependent by which we can develop only Standalone applications

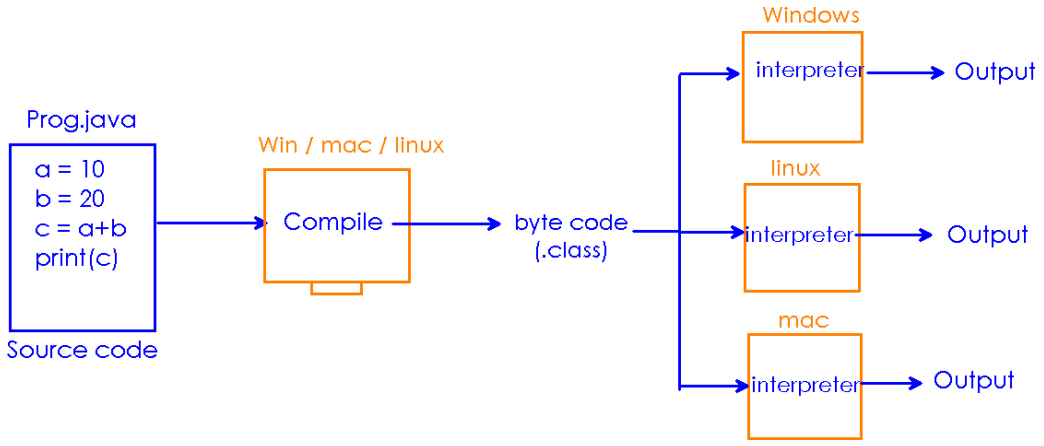
Platform Dependency:

- Platform means "Operating System"
- C and C++ Compilers convert the source program into specific Operating system understandable code. Hence the compiled code is compatible to same Operating system on which it has compiled.



2. **Web application:**
    - Application runs from multiple machines connected in a network.
    - Web application installed in server.
    - Data connection is required to run the application.
    - **Examples:** gmail.com, icici.com, irctc.in etc.

Note: We use technologies to develop Web applications which are platform independent



C-software:

- C is a standalone software.
- We have different types of C-software to develop and run C Programs. For example, Turbo-C, C-Free, Code Blocks etc.
- Download and install any C IDE to write and run programs easily.

Download the latest version of C-Free from: https://c-free.soft32.com/

Programming Elements

Program Elements: Program is a set of instructions. Every Program consists,

1. Identity
2. Variables
3. Functions

1. Identity:

- Identity of a program is unique.
- Programs, Classes, Variables and Methods having identities
- Identities are used to access these members.

2. Variable:

- Variable is an identity given to memory location.

  or
- Named Memory Location
- Variables are used to store information of program(class/object)

Syntax	Examples

datatype identity = value;	int age = 23;
	double salary = 35000.00;
	char gender = 'M';
	char* name = "Amar";

3. Function:

- Function is a block of instructions with an identity
- Function performs operations on data(variables)
- Function takes input data, perform operations on data and returns results.

Syntax	Example

returntype identity(arguments)
{
body;
}
	int add(int a, int b)
	{
	int c = a+b;
	return c;
	}

C Application Structure

Program:

- C program consists variables and functions.
- Program execution starts with main() function automatically.

Main.c

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

Application:

- C application contains more than one program.
- We connect these programs by calling one program functions from another program.
- We connect these programs using #include.
- Only one program contain main() function from which application execution starts.

Main.c

```
#include<arithmetic.c>
void main()
{
        // invoking functions
        add();
        subtract();
}
```

Arithmetic.c

```
#include<stdio.h>
void add()
{
        printf("Add");
}
void subtract()
{
        printf("Subtract");
}
```

#include:

- A pre-processor directive is used to connect the programs in C application.

Library:

- Library is a collection of pre-define programs called **Header-files**.
- Header-file contains Variables and Methods.
- Library is used to develop C application quickly.
- Some of the header file names as follows:

stdio.h	math.h	conio.h	string.h	graphics.h
printf() scanf()	sqrt() rand()	clrscr() getch()	strlen() strrev()	circle() line()

Variables in C

Variables:

- Variable is an Identity given to memory location.
- Variable is used to store information.
- We need to specify the datatype of every variable.

Syntax:

datatype identifier = value;

Examples:

char name[] = "amar"
int age = 23
char* mail = "amar786@gmail.com"
char gender = 'M';
double salary = 35000;

Note: Value of a variable will change. For example "age of a person"

Variable Declaration, Assignment, Modify and Initialization:

Declaration:

- Creating a variable without value.
- For example
  - int a ;

Assignment:

- Assigning a value to the variable which is already declared.
- For example,
  - int a ; // declaration
  - a=10 ; // assignment

Modify:

- Increase or decrease the value of a variable
- For example,
  - int a ; // declaration
  - a=10 ; // assignment
  - a=a+20; //modify

Initialization:

- Defining a variable along with value
- For example,
  - int a = 10 ; // initialization
  - a = a+20 ; // modify

Rules to create variable:

- Variable contains alphabets(A-Z , a-z) or Digits (0-9) and only one symbol(_).
- Variable should not starts with digit
    - int 5a = 10; (error)
    - int a5 = 10;
    - int 1stRank; (error)

- Variable should not contain spaces. 8355832471
    - int acc num = 1001; (error)
    - int acc_num = 1001;

- Variable should not allow other symbols except _
    - int acc_num ;
    - int acc-num;

- variable can starts with special symbol _
    - int _acc_num;
    - int _1rank;

- Variable should not match with keywords.
    - int if = 10 ;
    - int else = 20 ;

Variables classified into:

1. **Local variables:**
    a. Defining a variable inside the block or function.
    b. We cannot access local variables outside to the block.
2. **Global variables:**
    a. Defining a variable outside to all functions.
    b. We can access global variable from all functions.

Note: We will discuss briefly about Local and Global variables after Functions concept

Functions and Calling

Main() function:

- Main() function must be defined in every C program
- C program execution starts from main() function.
- Main() function automatically invoke by Operating system.

By default main() returns integer

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

Write a return statement with integer value to avoid errors.

User function:

- Defining other functions along with main() function.
- We must declare(prototype) of user function.

```
#include<stdio.h>
void abc();  ->Prototype of user function
int main()
{
    printf("Main \n");
}
void abc()  ->Definition of user function
{
   printf("ABC \n");
}
```

Calling the function:

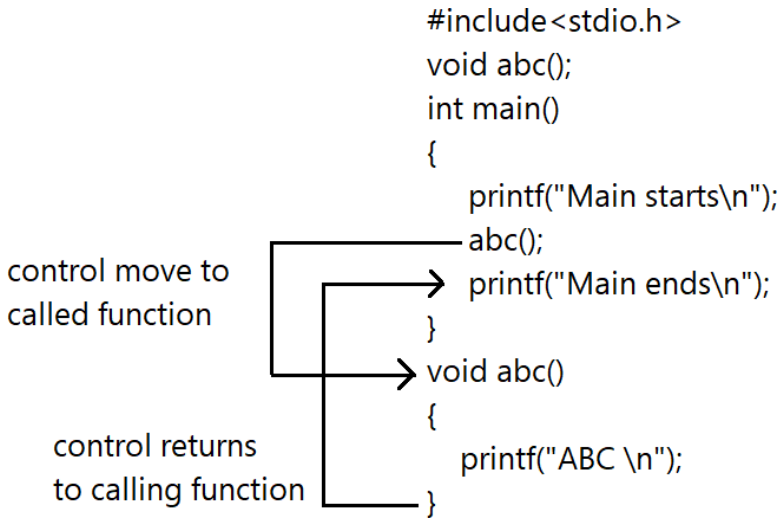
- Only main() function invokes automatically by OS
- All other functions in C program must be called manually.
- User function logic executes only when we call.

```
#include<stdio.h>
void abc();
int main()
{
    printf("Main \n");
    abc();  ->User function executes only on call
}
void abc()
{
   printf("ABC \n");
}
```

The following program clearly explains how the control back to calling function after execution of called function.

Calling Function: the function from which other function called.

Called Function: the function which is called from other function.



We can invoke the function any number of times once it has defined.

```
#include<stdio.h>
void abc();
int main()
{
    printf("Starts\n");
    abc();
    abc();
    abc();
    printf("Ends\n")
}
void abc()
{
   printf("ABC \n");
}
```

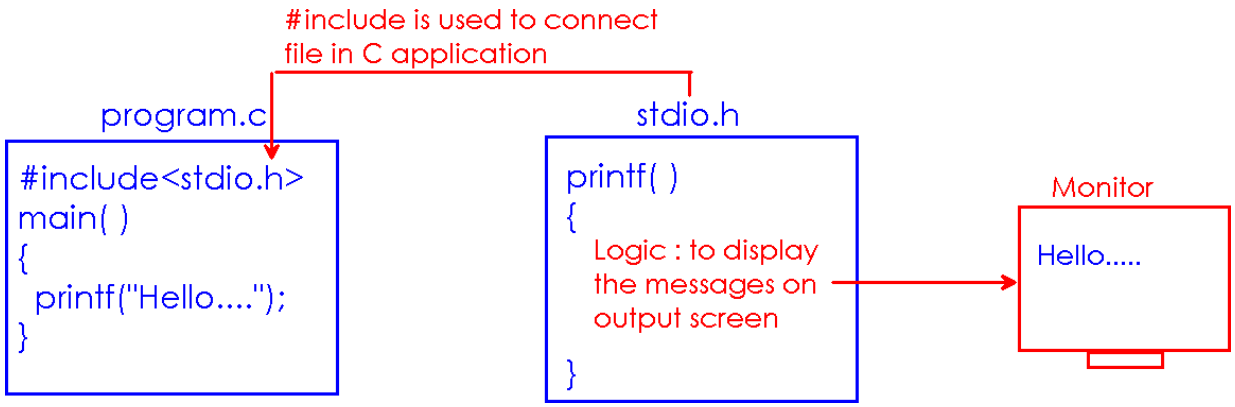
function can invoke any number of times once it has defined.

- We can define any number of functions in C-Program.
- We can define the functions in any order.
- We invoke the defined functions in any order.

```c
#include<stdio.h>
void abc();
void xyz();
int main()
{
    printf("Main Starts \n");
    xyz();
    printf("Back to Main from xyz \n");
    abc();
    printf("Back to Main from abc \n");
}
void abc()
{
    printf("abc \n");
}
void xyz()
{
    printf("xyz \n");
}
```

Execution flow of HelloWorld program briefly:

- Every C Program execution starts with main() function.
- OS invokes main() function automatically.
- #include is a pre-processor directive which is used to connect programs in C application.



Local and Global variables

Local variables:

- A variable inside the function or block.
- Local variable cannot access outside of that block.

```
int main()
{
        int a ; -> local variable
}
```

Local variables automatically initialize with garbage values. We cannot guess garbage value.

```
#include<stdio.h>
int main()
{
        int a;
        printf("%d \n", a); -> Prints unknown value.
        return 0;
}
```

Format specifiers:

- Formatting the result is very important before display to user.
- We use following format specifiers to read and print different types of data values.

Data type	Format specifier
int	%d
char	%c
float	%f
string	%s

Display information of different data types:

```
#include<stdio.h>
int main (){
        char* name = "Amar";
        int age = 23;
        float salary = 35000;
        char gender = 'M';
        printf("Name is : %s \n", name);
        printf("Age is : %d \n", age);
        printf("Salary is : %f \n", salary);
        printf("Gender is : %c \n", gender);
        return 0;
}
```

We can access local variables only from the same function in which they have defined.

```
#include<stdio.h>
void test();
int main (){
        int a=10;
        printf("a value in main : %d \n", a);
        test();
        return 0;
}
void test(){
        printf("a value in test  :%d \n", a); // Error :
}
```

Global Variables:

- Defining a variable outside to all functions.
- Global variables can be accessed from all the functions.

```
#include<stdio.h>
void test();
int a=10;
int main (){
        printf("a value in main : %d \n", a);
        test();
        return 0;
}
void test(){
        printf("a value in test  :%d \n", a);
}
```

Global variables automatically initialize with default values:

Datatype	Default Value
int	0
float	0.00000
char	Blank
string	Null

```
#include<stdio.h>
int a;
float b;
int main (){
        printf("int : %d \n", a);
        printf("float : %f \n", b);
        return 0;
}
```

Escape Sequences: Performs specific action when we use inside the String.

\b	Backspace
\n	New line
\r	Carriage return
\t	Horizantal tab
\v	Vertical tab
\\	Backslash
\'	Single quote
\"	Double quote
\0	Null

Display String in multiple lines:

```
#include
<stdio.h
> int
main (){
        printf("String \ndisplayed \nin
        \nMultiple \nLines"); return 0;
}
```

Display information with tab spaces:

```
#include
<stdio.h
> int
main (){
        int a=10, b=20,
        c=30;
        printf("%d\t%d\t%
        d", a, b, c); return 0;
}
```

Display Message with Single Quotations:

```
#include<stdio.h>
int main ()
{
        printf("'C' Tutorial\n");
        return 0;
}
```

```
#include<stdio.h>
int main ()
{
        printf("\'C\' Tutorial\n");
        return 0;
}
```