Lesson 01
Programming with C

Pijushkanti Panigrahi

What is C ?

- The C is a programming Language, developed by Dennis Ritchie for creating system applications that directly interact with the hardware devices such as drivers, kernels, etc.
- C programming is considered as the base for other programming languages, that is why it is known as mother language.
- C is famous for its Compactness.
- C language is case sensitive.

Features

It can be defined by the following ways:

- Mother language
- System programming language
- Procedure-oriented programming language
- Structured programming language
- Mid-level programming language

1) C as a mother language ?

- C language is considered as the mother language of all the modern programming languages because **most of the compilers, JVMs, Kernels, etc. are written in C language**, and most of the programming languages follow C syntax, for example, C++, Java, C#, etc.
- It provides the core concepts like the array, strings, functions, file handling, etc. that are being used in many languages like C++, Java, C#, etc.

2) C as a system programming language

- A system programming language is used to create system software.
- C language is a system programming language because it **can be used to do low-level programming (for example driver and kernel)**.
- It is generally used to create hardware devices, OS, drivers, kernels, etc. For example, Linux kernel is written in C.

3) C as a procedural language

- A procedure is known as a function, method, routine, subroutine, etc. A procedural language **specifies a series of steps for the program to solve the problem**.
- A procedural language breaks the program into functions, data structures, etc.
- C is a procedural language. In C, variables and function prototypes must be declared before being used.

4) **C as a structured programming language**

- A structured programming language is a subset of the procedural language. **Structure means to break a program into parts or blocks** so that it may be easy to understand.
- In the C language, we break the program into parts using functions. It makes the program easier to understand and modify.

5) C as a mid-level programming language

- C is considered as a middle-level language because it **supports the feature of both low-level and high-level languages**. C language program is converted into assembly code, it supports pointer arithmetic (low-level), but it is machine independent (a feature of high-level).
- A **Low-level language** is specific to one machine, i.e., machine dependent. It is machine dependent, fast to run. But it is not easy to understand.
- A **High-Level language** is machine independent. It is easy to understand.

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: data_type variable_list;

Data Types in C

- A data type specifies the type of data that a variable can store such as integer, floating, character, etc.
- There are the following data types in C language.

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

- The example of use of data types is given below:

```
int a;          float b;          char 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';
```

Rules for defining variables

- A variable can have alphabets, digits, and underscore.
- A variable name must start with an 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.
- *Valid variable names:*

  int a; **int** _ab; **int** a30;

- *Invalid variable names:*

  int 2; int a b; int long;

C-language program
a general structure

1 Calling header files	#include <stdio.h>
2 Main function	main()
3 Starting brace	{
4 Variable(s) declaration	int a;
5 Executable statement(s)	printf () etc
6 Closing brace	}

```c
/* To Print a message */
#include <stdio.h>
 main()
{
printf ("Hello C Programming\n");
 }
```

Output

```
Hello C Programming
```

printf() function

- The **printf() function** is used for output. It prints the given statement to the console, defined in stdio.h (header file).. The syntax of printf() function is given below:

  printf("format string",   argument_list);

- The **format string** can be

  %d for integer, %f for floating point number, %c for single character, %s for a string etc.

scanf() function

- The **scanf() function** is used for input , defined in stdio.h (header file). It reads the input data from the console.

  scanf("format string", &argument_list);

- The **format string** can be

  %d for integer, %f for floating point number, %c for single character, %s for a string etc.

```
        /* Program to print cube of given number */
#include<stdio.h>
main()
 {
        int number, cube;
        printf("enter a number:");
        scanf("%d",&number);
        cube=number*number*number;
        printf("  cube of number is:%d " cube,);
  }
```

Output

```
enter a number:5 cube of number is:125
```

- The scanf("%d",&number) statement reads integer number from the console and stores the given value in number variable.

- The printf("cube of number is : %d", cube) statement prints the cube of number on the console.

/* Program to print sum of 2 integer numbers */

```
#include<stdio.h>
main()
{
    int x=0,y=0,result;
     printf("enter first number:");
     scanf("%d",&x);
     printf("enter second number:");
     scanf("%d",&y);
         result=x+y;
         printf("sum of 2 numbers:%d ",result);
  }
```

Output

```
    enter first number:19 enter second number:11 sum of 2 numbers:30
```

If-else Statement

- The if-else statement in C is used to perform the operations based on some specific condition. The operations specified in if block are executed if and only if the given condition is true.
- There are the following variants of if statement in C language.
    - ➢if statement
    - ➢if-else statement
    - ➢If-else-if ladder
    - ➢Nested if

if Statement

The *if* statement is used to check some given condition and perform some operations depending upon the correctness of that condition. It is mostly used in the scenario where we need to perform the different operations for the different conditions. The syntax of the if statement is given below.

```
if(expression)                       if (a>b)
  {                                    {
   executable statement(s);             printf (“a is large”);
    }                                    }
```

```
#include<stdio.h>
main()
  {
    int a, b;
    printf("enter a:");
    scanf("%d",&a);
    printf("enter b:");
    scanf("%d",&b);
        if(a>b)
            {
              printf(“a is large”);
             }
    }
```

Output

- enter a:14 enter b: 12   a is large
- enter a:19 enter b: 25

if-else Statement

- The *if-else* statement is used to perform two operations for a single condition. The *if-else* statement is an extension to the *if* statement using which, we can perform two different operations, i.e., one is for the correctness of that condition, and the other is for the incorrectness of the condition. The syntax of the *if-else* statement is given below.

```
if(expression)
    {  executable statement 1;
     }
else
    { executable statement 2;
     }
```

```
#include<stdio.h>
main()
  { int a, b;
    printf("enter a:");        scanf("%d",&a);
    printf("enter b:");        scanf("%d",&b);
        if(a>b)
            {
              printf(“a is large”);
             }
          else
           {
              printf(“b is large”);
             }
    }
-------------
```

Output   enter a:14 enter b: 12   a is large

                  enter a:19 enter b: 25   b is large

```
#include <stdio.h>
main()
   {  int age;
       printf("Enter your age?");        scanf("%d",&age);
          if(age>=18)
               {    printf("You are eligible to vote...");      }
          else
               {     printf(“You are not eligible to vote...");    }
   }
```

Output

- Enter your age?28 You are eligible to vote...
- Enter your age?13 You are not eligible to vote....

if-else-if ladder Statement

The *if-else-if* statement is an extension to the *if-else* statement. It is used in the scenario where there are multiple cases to be performed for different conditions. In *if-else-if* ladder statement, if a condition is true then the statements defined in the *if* block will be executed, otherwise if some other condition is true then the statements defined in the *else-if* block will be executed, at the last if none of the condition is true then the statements defined in the *else* block will be executed.

Syntax of if-else-if statement

```
if(condition1)
      {
         executable statements 1;
       }
else if(condition2)
        {
          executable statements 2;
          }
 else if(condition3)
         {
            executable statements 3 ;
          }
          ...
 else
     {
           executable statements last;

       }
```

Program to prepare the result of the student

```
#include <stdio.h>
 main()
{   int marks;
    printf("Enter your marks?");           scanf("%d",&marks);
    if(marks >=600)
      {   printf("Congrats ! you are place in the FIRST CLASS");      }
    else if (marks >=500)
       {   printf("You are placed in the SECOND CLASS");                }
    else if (marks >= 400 )
         {  printf("You are placed in the THIRD CLASS");          }
    else
         {       printf("Sorry you FAILED");     }
}
```

Output

```
Enter your marks?31     Sorry You FAILED
 Enter your marks?478  You are placed in THIRD CLASS
Enter your marks?545    You are placed in SECOND CLASS
Enter your marks?726     Congrats ! you are place in the FIRST CLASS
```