C programming

C is a general-purpose programming language created by *Dennis Ritchie*.
C is a case-sensitive language.

Difference between c and c++

C++ is developed as an extension of C
C++ supports classes and objects but C doesn't.

To start using C,

- A test editor like notepad
- A compiler like GCC,to translate the C code into a language that the computer can understand.

1st Program:

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

Line 1: **#include<stdio.h>** is a header file library that lets us work with input and output functions.Header files add functionality to C programs.**Pre-processor directive**
Line 2:**main()** , this always appears in C program.It is called a function. Any code inside the {} curly braces will be executed.
Line 3:**printf** is a function used to output/print text on the screen.
Line 4:**return 0** ends the main function.Since the main function (int main()) returns a numeric number, we are return 0 .we just use return 0.The primary function will be of data type "void," hence nothing will need to be provided.The return value is the program's exit code.It indicates successful execution.

Escape sequence:

- \t —>creates a horizontal tab
- \\Inserts a backslash character
- \"Inserts a double quote character.

For , single line comment we use— "//"
Multiline comment we use— "*/"

Variables: There are different types of variables in C—

- ➔ Int – stores integers (whole number) without decimals such as 123,-123
- ➔ Float – stores float value with decimals such as 11.23 or -11.23
- ➔ Char –stores single characters like ‘a’,’b’.

Format specifiers

Format specifiers	Data type
%d or %i	int
%f	float
%c	char
%s	Used for string or text

Ascii Value table:

Char	Number
A-Z	65-90
a-z	97-122
0-9	48-57

Memory size:

Datatype	Size
int	2 or 4 bytes
char	1 byte
float	4 bytes
double	8 bytes

Type Conversion:

There are two types of type conversion in C programming–

- **Explicit Type conversion:**Explicit conversion is done manually by placing the type in parenthesis.
- **Implicit Type conversion:**Implicit conversion is automatically done by the compiler when we assign a value of one type to another.

Operators in C:

1. Arithmetic operators**: '+', '*', '/', '%', '++', '--'**
2. Bitwise operators: **'&' ,'|', '^', '~'**
3. Logical operators**: '&&' , '||' ; '!'**
4. Assignment operators: **'='**
5. Comparison operators**: '==', '!=' ,'>', '<', '<=', '>='**

There are **32** keywords in C. Example– int,float,for,double,return etc.

Conditions and If Statements

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be execute

Syntax:

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

switch Statements

Instead of writing many if..else statements, you can use the switch statement.

The switch statement selects one of many code blocks to be executed:

Syntax:

```
switch (expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

- The switch expression is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The break statement breaks out of the switch block and stops the execution
- The default statement is optional, and specifies some code to run if there is no case match

For Loop

Syntax:

```
for (expression 1; expression 2; expression 3) {
  // code block to be executed
}
```

Expression 1 sets a variable before the loop starts (int i = 0).

Expression 2 defines the condition for the loop to run (i must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.

Expression 3 increases a value (i++) each time the code block in the loop has been executed.

Arrays: Collection of different data types stored at contiguous memory locations.

To create an array, define the data type (like int) and specify the name of the array followed by square brackets [ ].

Example:

```
int myNumbers[] = {25, 50, 75, 100};
```

A multidimensional array is basically an array of arrays.

Example:

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
```

Strings: A character array terminated by a '\0'.

```c
#include<stdio.h>
int main(){
    char myname[]={'N','e','h','a',' ','P','a','l','\0'};
    char name[]="Neha Pal";
    printf("%s \n",myname);
    printf("%s \n",name);
}
```

'\0' is known as a "**null terminating character**".

String Functions:

strlen() – to get the length of a string.
sizeof() – it will return the memory size.
strcat() – to compare two strings.
strcpy() – to copy the value of one string to another.
strcmp() – it is used to compare two strings.

gets(str): input a string(even multiword) -> outdated
puts(str): output a string
fgets(str,n,file): stops when n-1 chars input or new line is entered.

C user Inputs

Use the `scanf()` function to get a single word as input, and use `fgets()` for multiple words.

Memory Address

When a variable is created in C, a memory address is assigned to the variable. The memory address is the location of where the variable is stored on the computer. When we assign a value to the variable, it is stored in this memory address.

To print pointer values we use **%p** format specifier**.**

A pointer is a variable that stores the memory address of another variable as its value.A pointer variable points to a data type (like int) of the same type, and is created with the * operator.

Pointer to Pointer:A variable that stores the memory address of another pointer.

Reference & Dereference:
'*' has two different meaning in the code:

- When used in declaration (int* ptr), it creates a pointer variable.
- When not used in declaration, it acts as a dereference operator.

```c
#include<stdio.h>
int main(){
   int num=21; //Variable Declaration
   int* ptr=&num;//Pointer Declaration
   printf("%d \n",num);
   printf("%p \n",ptr);//Reference:output of the address of the number
   printf("%d \n",*ptr);//Dereference:output the value of num with the pointer(21)
}
```

Pointers are one of the things that make C stand out from other programming languages, like Python and Java.

Function

A function is a block of code which only runs when it is called.
We can pass parameters in the function.
Predefined Function – main() which is used to execute the code.**printf()** which is used to print output or print text to the screen.
Create a Function – to create our own function, we have to specify the name of the function, followed by parentheses **( )** and curly braces **{ }**

Syntax:

```
Void myfunc(){
        //code
        }
```

- ➔ **myfinc()** is the name of the function
- ➔ **Void** is the return type of the function,that means the function does not have any return value.
- ➔ We have to write our code inside the curly braces.

To call the function, we have to write the function name followed by the parenthesis.

A function consist of two parts:

- **Declaration:** the function's name, return type, and parameters (if any)
- **Definition:** the body of the function (code to be executed)

Pointers in function:

- Call by value:We pass the value of the variable as an argument.
- Call by reference:We pass the address of the variable as argument.

Structures:

A collection of values of different data types.
Example:
For a Student we have to store this things:

- Name(str)
- roll(int)
- cgpa(float)

File I/O:

File- container in a storage device to store data

- RAM is volatile
- Contents are lost when program terminates
- Files are used to persist the data.

Operations on File:

- Create of File
- Open a file
- Close a file

- Read from a file
- Write in a file

Types of file:

➔ Text Files:textual file(.txt,.c)

➔ Binary Files:binary data(.exe,.jpg)

File Pointer: File is a hidden structure that needs to be created for opening a file. A File ptr that points to the structure & used to access the file.

FILE* fptr

Opening a file

➔ fptr=fopen("filename",mode);

Closing a file

➔ fclose(fptr);

File opening Modes:

➔ "r":open to read

➔ "rb":open to read in binary

➔ "w":open to write

➔ "wb":open to write in binary

➔ "a":open to append

EOF(End of File):fgetc returns EOF to show that the file has ended.

Dynamic Memory Allocation:

It is the way to allocate the memory to a data structure during the runtime.

We need some functions to allocate and free memory.

Functions:

➔ ***malloc():*memory allocation** → takes number of bytes to be allocated & returns a pointer of type void.

Ptr = (int*)malloc(5* sizeof (int));

➔ *calloc():*contiguous allocation → initialize with 0.

Ptr = (int*)malloc(5, sizeof (int));

➔ ***realloc()*** → reallocated (increases or decreases)memory using the same pointer & size.

Ptr = realloc(ptr,newSize);

➔ ***free()*** →It is used to free memory that is allocated using malloc and calloc.

free(ptr);