**Introduction to C**

C language is a programming language which has it’s origin dated back to 19’s

C language is a low level language nearly equivalent to assembly language

C language has some keywords,identifiers,datatypes etc:

Variable : A variable is a memory allocation which stores a value or a character in it.(identifiers are called as

variables)

There are different types of variables in C

Signed: variables which can hold positive,negative values in it.

Unsigned: variables which can hold only positive values in it. Format specifier of it is %u.

Note: After incrementing the unsigned after zero then the variable goes to the highest number which it can hold

Keywords : A keywords are pre-defined words in the c library these keywords have a pre-defined meaning to it.

Datatypes : C language has some different types of datatypes like char, int, float ( decimal values ) etc.

Operators : C language has some basic operators as well as some preliminary operators

Header files : In C language there are header files in which some functions are predefined in the header files

I/O functions : I/O functions in the C language are two types

1. **Formatted**: in the formatted function we need not specify the format specifier
2. printf(“%d”,var);

**Note**: in C language while printing the evaluation may be done from any side that is from left or right making note of precedence also.

**Note:** printf function returns no. of characters it is printing (this is mostly used when we are printing printf function [ printf(printf(“Hello”) ] )

1. scanf(“%d”,&var);
2. **Unformatted:** in the formatted function we need not specify the format specifier
3. Ch= gets(),getchar();
4. puts(ch),putchar(ch);

**If else control statements**

If statements in C language ( control statements ) : C language consists some control statements

Syntax:

if(condition)

{

……

}

If-else control statements : similarly like if control statement this control statement checks if a condition is true (if true then the code in the if statement will be executed ) if condition fails to be true then the condition in the else statement executes

Syntax :

if(condition)

{

…..

}

else

{

……

}

If-else ladder : If in some scenario there are many conditions to be checked then if else control statement can be used

Syntax :

if(condition)

{

…..

}

elseif(condition)

{

……

}

else

{

….

}

Switch statements : if at all in C program you have to chose only one choice in the given choices then switch can be used

Syntax :

switch ( variable )

{

case value :

// code

case value2 :

// code

}

**Loops, Break, continue and goto statements**

Loops : if you want to execute a given code again and again then you can automate the code with the help of loops in c

There are 3 loops in c language they are :

1. For loop
2. While
3. Do…while

For loop : A flexible loop which can be used almose anywhere

Syntax :

for(initialization;condition;expression)

{

// code

}

While loop :

while ( condition )

{

// code

}

Do…While loop :

do

{

// code

}

while (condition );

**Difference between do… while and while loops:**

|  |  |
| --- | --- |
| While loop | Do while loop |
| Condition is checked first then statement(s) is executed.  It might occur statement(s) is executed zero times, If condition is false.  No semicolon at the end of while. while(condition)  If there is a single statement, brackets are not required.  Variable in condition is initialized before the execution of loop.  while loop is entry controlled loop. | |  | | --- | | Statement(s) is executed atleast once, thereafter condition is checked. | | At least once the statement(s) is executed. | | Semicolon at the end of while. while(condition); | | Brackets are always required. | | variable may be initialized before or within the loop. | | do-while loop is exit controlled loop. | |  | |

Break,continue and goto statements : Break,continue and goto statements are used in the loops to either stop the loop ( break ) or to skip the ( iteration ) of the loop or to send the control to random places ( goto )

**Type casting**

Type casting : In C language you cannot apply operations to different data types you need to convert them before you do your operation

Syntax :

(data\_type)var;

**Preprocessor and Macros**

Preprocessor directives : In C language there are pre processor directives there are codes which start before compilation of the code these preprocessor directives are used to link up the header files which are pre defined in the compiler they are also used to make some macros

There are two types of preprocessor directives :

1. #include
2. #define

#include is used to link up the files in the C compiler

#define is used to make some macros which can be used in the code

Macro substitution : macro substitution in C is nothing but making some global variables which are defined outside the main function with the help of the #define statement

Syntax :

#define var1 = initialization

**Arrays and strings**

Array : Array is a derived datatype which can hold homogenous data elements in it in a contiguous memory allcation

Note : an array is an internal pointer which stores the base address of the array

Syntax :

data\_type array\_name[];

Strings : Strings are nothing but the array of the strings

Note: If you initialize numbers in strings as (“8”) then they will be taken as a char and need to be converted into integer this is done with the help of atoi function.

Strings also have some functions in the string.h header file

1. strlen
2. strcmp
3. strcat

**Functions**

Function: Functions in C language allow users to make their own user defined functions which enable users to split up the code into small chunks which make program easy to read

To make function there are 3 steps to follow :

1. Function declaration
2. Function definition
3. Function call

Function declaration : Function declaration is nothing but creating a new user defined function it is also called function prototype

Syntax :

return\_type function\_name( data\_type par1….parn);

Function definition : Function definition is nothing but giving meaning to the function which you have created

Syntax :

return\_type function\_name(data\_type par1….parn)

{

// code

}

Function call : Function call is nothing but calling the function you’ve just created ( literally using it )

Syntax :

function\_name(var1….varn);

**Note**: Function call is done in main function

**Note**: Name of the function is actually a pointer variable to the function and prints the address of the function

Recursion : Calling a function in its own definition is called recursion

Passing an array to a function : Arrays can be passed to the function just like any other variables

Syntax :

return\_type function\_name(data\_type array\_name[]);

**Structures and Unions**

Structure: Structure is a user defines datatype which user can use it to make his own datatype which consists of heterogenous data elements in it

Syntax:

struct tagname

{

data\_type element\_name;

.

.

}var1,….varn;

Union : Union is also similar to the structure but the memory allocation of the union variable is the the memory allocation of the largest element in the union

Syntax :

union tagname

{

data\_type element\_name;

.

.

}var1,….varn;

**Files**

**Files:** In C language the programs are stored in the RAM while executing it and the data which program generates will be destroyed after the execution of the program if the data must be there even after the program gets executed then the data must be stored in the file

There are 3 attributions of a file:

1. Read
2. Write
3. Append

There are mainly six modes of file handling:

1. r (read only)
2. w (write only )
3. a (appending only)
4. r+ (read and write)
5. w+ (write and read)
6. a+ (append and read)

**File also has some i/o functions:**

fprintf ( output ) syntax : fprintf( fp,”%d”,ch);

fputs ( output ) syntax : fputs (ch,fp);

fputc ( output ) syntax : fputc (ch,fp);

fscanf ( input ) syntax : fprintf( fp,”%d”,ch);

fgets ( input ) syntax : fgets (ch,fp);

fgetc ( input ) syntax : fgetc (ch,fp);

Note: File is a structured datatype which is predefined in the <stdlib.h> header file

Syntax:

FILE \*file\_pointer;

file\_pointer fopen(”file\_name.txt”,”attribute”); // to open the file firstly

fclose(file\_pointer); // to close the file

**Random access files:**

Functions in Files: there are some functions in the C language to randomly edit the file

1. fseek()
2. ftell()
3. rewind()

fseek: fseek() is used to move file pointer associated with a given file to a specific position.

Syntax:

int fseek(FILE \*pointer, long int offset, int position)

pointer: pointer to a FILE object that identifies the stream.

offset: number of bytes to offset from position

position: position from where offset is added.

Ftell(): tells the position of the cursor.

Rewind(): Brings back the pointer to the starting location

**Pointers**

**Pointer**: A pointer in C language is nothing but a special type of variable which stores the address of another variable

Syntax:

datatype\_\*pointer\_name;

A pointer can be of any type (int, char, float) etc;

A pointer in C programming language will be declared with help of \* symbol

‘\*’ is a dereference operator

Uses of pointer:

1. Pointers are extensively used in the arrays
2. Pointers are used in dynamic memory allocations
3. Pointers are used in the call by reference in the function

Pointer arithmetic :

There 4 types of arithmetic operators :

1. ++
2. –
3. +
4. –

Note : if these operators are used for the pointers then the address of the pointers won’t will be incremented but the sizeof the address will be incremented in terms of the datatype (ex if pointer is an int pointer then if the pointer has some x value then if ++ is used on the pointer then the answer will be x+ sizeof(int))

**Relation of pointer and array:**

Pointer and array are bit similar to each other

An array (address of array ) can be assigned to the pointer then the base address of the array will be assigned to the pointer then the values of the array can be printed with the help of the dereference operator

**Types of pointer:**

1. Void pointer: Void pointer is a pointer which has no valid datatype it can be type casted into any other data type easily Void pointer will be used when dynamically allocating the memory. Void pointer cannot be dereferenced in C language
2. NULL pointer: NULL pointer is a pointer which has nothing in it. It is not having a valid value in it. It should not be dereferenced. A NULL pointer can be used by the programmer to debug the code by initializing the waste pointers to NULL.
3. Dangling pointer: Dangling pointer is a pointer which points to some address which has been extincted or deleted
4. Wild pointer: Wild pointer is an uninitialized pointer which stores garbage address in it the address of the pointer which changes all the time

**Dynamic Memory Allocation**

**Dynamic memory allocation**: While making the array we don’t exactly allocate memory we keep some buffer memory

**Memory Layout**: In C memory has four segments

1. Code: code is the memory which is allocated by the source code
2. Static variables: the memory consumed by the variables
3. Stack: memory which grows as the program continues and new functions are called
4. Heap: Heap is the memory in which we are going to do memory allocation

**For dynamically allocating memory there are three functions:**

1. Malloc
2. Calloc
3. Realloc

Malloc is memory allocation it reserves block of memory in given amount of bytes the return value is a void pointer all the values are initialized to garbage values once allocated

Syntax:

ptr = (ptr-type\*) malloc(size\_in\_bytes);

Calloc is memory allocation it reserves block of memory in given amount of bytes the return value is a void pointer all the values are initialized to zero once allocated

Syntax:

ptr = (ptr-type\*) calloc(n,size\_in\_bytes);

Realloc is memory allocation it reserves block of memory in given amount of bytes again if the memory is not sufficient after allocating with malloc/calloc the return value is a void pointer all the values are initialized to zero once allocated

Syntax:

ptr = (ptr-type\*) realloc(ptr,new\_size\_in\_bytes);

Free: free() function is used to free the memory if the work is done

Syntax: free(ptr);