**DAY 10:**

Nesting function: Ex: func() is calling func1().

{

Func()

Func1()

}

Recursive function: The function call itself.

Int f(int);

Int main(){

Int res =f(5);

Printf(“\nREs=%d\n\n”,res);

Return 0;}

Int f(int v){

If (v==0)

Return 1;

v--;

f(v);

printf(“\n vValues in fun: %d”, v);

return v; }

Pointers:

It is a special variable, and it also have own address. It points to the address. Size of the pointer is 4/8 bytes.

dt \*ptrName;

Poniter does not belong to any type of variable. The address may be any data type.

Ex: int \*prt; => ptr point in to address of int data type

Types:

1. Null pointer: it points towards the nothing. (int \*ptr=null)
2. Void pointer or generic pointer: because it can hold the any type of data type.
3. Wild pointer:
4. Near pointer
5. Far pointer
6. Dangling pointer

A screenshot of a computer

Description automatically generated

Pf(‘%d”,ptr) => gives the address

Pf(“%d”,\*ptr) => give value

Error: invalid use of void expression => we have to do type casting

=> pf(“%d”, \*(int \*)ptr);

So, we have to specify the specific data type like “int ptr’ instead of void ptr.

A screenshot of a computer screen

Description automatically generated

To print array value by using pointer is

* Pf(“%d”,ptr[0]);
* Pf(“%d”,ptr[1]);

To print array value by using pointer notation

* Pf (“%d”, \*(ptr+0));
* Pf (“%d”, \*(ptr+1));

Pointers have 2 thumb rules:

1. &\* = Nullify each other
2. op [ ] => \*op

op\* => op[ ]

1). a=10

ptr=&a; // stores the address of a

\*ptr=101 // assigning the value in address of ptr i.e in a

Then, a =101

2). a=10;

ptr = NULL

\*ptr=101

It gives segmentation fault error because, \*ptr is not storing any address it is a null.

**Malloc:**

* while using malloc we use ‘#include <stdlib.h>’
* Malloc has size of 10 bytes.
* Malloc allocates the value and gives the base address to pointer.
* If malloc size is ‘0’ it returns null.

Malloc (nmem \*size);

* For ‘stdlib.h’ like scanf, printf, malloc. We use ‘perror’ if we want to find the error.

If ptr++ the address increases to 4 bytes.

Similarly ptr—the address decreases to 4 bytes.

* Write a program to find the value or search if it is present.

Static, extern, auto, register are 4 storage elements .

#include <stdio.h>

int \*allocMem();

int main()

{

int a=10;

int \*ptr=NULL;

ptr = allocMem();

printf("\n%d\n",\*ptr);

return 0;

}

int \*allocMem()

{

static int a=10;

return &a;

}

Register: Which stores in the cpu storage.It is a faster.