**DAY 10**

**functions**

1.std lib functions

printf,sqrt,abs,pow

2.user defined

user is defining his /her own task to be performed

return datatype

rdt fName (input args)

{

sts;

return rdt;

}

int add (int,int);

int add(int val1,int val2)

{

int result= val1+val2;

return result;

function definition should be written in .h file

/\*

**2 Demo on recurrsive function**

3 \*/

#include<stdio.h>

int f(int);

int main()

{

int res = f(5);

printf("\nRes=%d\n\n",res);

return 0;

}

int f(int v)

{

int ret;

printf("\nV Value : %d\n",v);

if (v == 1)

return 1;

ret = v \* f(--v);

printf("\nV Value in Func: %d\n",v);

return ret;

}

output :

V Value : 5

V Value : 4

V Value : 3

V Value : 2

V Value : 1

V Value in Func: 1

V Value in Func: 2

V Value in Func: 3

V Value in Func: 4

Res=24

* **pointers declaration**

dt\*ptrname

int \*ptr

* pointer doesnt belong to any kind of data type it just holds the address of the variable
* ptr is a special variable it is pointing to integer datatype.
* **4 types of pointers**

1.null

2.void

3.wild

4.dangling pointer(situation not name of pointer)

4.near and far pointers

void \*ptr is known as generic pointer because it can point to any type of datatypes

int \*ptr=null; is an null pointer

if the pointer is not pointed to anything then it points to anywhere so this is known as wild pointer.

float \*ptr it is not initialised hence it is wild pointer.

invalid use of void expression error \*ptr it is unable to convert to an integer ptr is holding 8 bytes

#include <stdio.h>

int \*allocMem();

int main()

{

int a=10;

int \*ptr=NULL;

// ptr = &a;

ptr = allocMem();

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

return 0;

}

int \*allocMem()

{

static int a=10;

return &a;

}

**Output :**10

#include<stdio.h>

int main()

{

int a=10;

float b=20.2;

void \*ptr = NULL;

int \*ptr1=NULL;

printf("\naddress of a=%u and its value=%d",&a,a);

printf("\naddress of b=%u and its value=%d",&b,b);

printf("\naddress of ptr=%u and its value=%d",&ptr,ptr);

printf("\nsize of a=%d ",sizeof(a));

printf("\nsize of b=%d ",sizeof(b));

printf("\nsize of ptr=%d ",sizeof(ptr));

ptr =&a;

ptr1=&a;

printf("\nprint ptr value %u",ptr);

printf("\nvalue pointed by ptr =%d",\*(int \*)ptr);

printf("\n value pointed by ptr1= %d",\*ptr1);

return 0;

}

**thumb rules:**

if &\* symbol is side by side it will nullify each other.

1.op[]==> \*op

here square bracket is converted to star

2.\*op====>op[]

#include<stdio.h>

int main()

{

int a=10;

int i;

int b[3]={11,18,13};

int \*ptr=NULL;

//ptr =&b[0];//ptr is pointing to the base

//ptr =&\*(b+0);

//ptr=(b+0);

ptr =b;

for(i=0;i<3;i++)

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

\*(ptr+1)=100;

for(i=0;i<3;i++)

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

printf("\n\n");

return 0;

}

segmentation core dumped earlier it is pointing to an address so 101 is stored now it doesnot point to any address

auto int i;

storage classes in c

1.) auto

2.)static

3.)register the value is stored in cpu register so acces is very faster than the variable stored in stack memorystack

4.)extern

ptr to fptr function pointer

type(\*fptr)

**welcome display**

/\*

2

3 =========================================

4 =================Welcome=================

5 =========================================

6

7

8 \*/

9

10 #include <stdio.h>

11 #include <string.h>

12

13 int printChar(int, int);

14

15 int main()

16 {

17 char msg[]="Welcome";

18 int i;

19 printChar('\n',1);

20 printChar('=',50);

21 printChar('\n',1);

22 printChar('=',21);

23 for(i=0;i<strlen(msg);i++)

24 printChar(msg[i],1);

25

26 printChar('=',22);

27 printChar('\n',1);

28 printChar('=',50);

29 printChar('\n',1);

30

31 printf("\n\n");

32

33 return 0;

34 }

35

36 int printChar(int ch, int n)

37 {

38 int i;

39 for(i=0;i<n;i++)

40 putchar(ch);

41 }