**DAY – 09**

**ARRAYS- --- []**

It contain only homogenous datatype of elements.

It is stored in consecutive memory location.

syntax: dt ArrName[size];

e.g.; int Arr[10];

all 10 elements are used for one purpose.

Struct emp

{

int id;

char name[20];

};

Struct emp e[10];

ArrName[indexvalue] = value;

it starts from lowest

BA+(indexvalue+size of (datatype)]

the index values always be in whole number and can't be float.

4 types of arrays:

1.static

\*\* the size of the array known before to the compile time --- stack memory

2.dynamic

\*\*the size of the array is allocated or known at run time -- heap memory

can be achieved by calloc, malloc, realloc --- stdlib.h

3.strechable array

\*\*renaming the dynamic array --- size of the array or increased or decreased depending on the need.

4.mutable array

\*\*\* the size of the array is known at the time of the linking & before execution

write a program to store odd numbers in an array between n & m

**to know the array capacity**

**CAP = sizeof(a)/sizeof(a[0])**

**2D Array:**

**Syntax: dt arrange [Row][col] ;**

#include <stdio.h>

#define ROW 2

#define COL 3

int main(){

int a[ROW][COL]= {{1, 2, 3},{4, 5, 6}}

printf("\n%d",a[1][2]);

printf("\n\n");

return 0;

}

.

#include <stdio.h>

#define ROW 2

#define COL 3

int main(){

int a1[ROW][COL]= {{1, 2, 3},{4, 5, 6}};

int a2[ROW][COL]= {{1, 2, 3,4, 5, 6}};

int riv,civ;

printf("\n%d\n",a2[0][2]);

for(riv=0;riv<ROW;riv++){

for(civ=0;civ<COL;civ++)

scanf("%d",a2[riv][civ]);

}

for(riv=0;riv<ROW;riv++){

for(civ=0;civ<COL;civ++)

printf("%d",a2[riv][civ]);

printf("\n");

}

printf("\n\n");

return 0;

}

#include <stdio.h>

#define CAP 100

int main(){

int a[CAP],i;

int mid , countOdd, countEven;

int m =51, n=103;

mid = (n+m)/2;

for(i=m,countOdd = 0; countEven =mid;i<n;i++)

{

if(i%2!=0)

{

a[countOdd] = i;

countOdd++;

}

else

{

a[countEven] = i;

countEven++;

}

}

}

**Array rule**:

y[2.5] access y[2]

y[10.9] access y[10]

truncaton is done here

print the reverse array:

#include <stdio.h>

#define CAP 10

int main()

{

int i=0,t,mid;

int a[CAP]={1,2,3,4,5,6,7,8,9,10};

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

printf(" %d ",a[i]);

for(i=0,mid=CAP/2;i<mid;i++){

t = a[i];

a[i]=a[CAP-1-i];

a[CAP-1-i]=t;

}

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

printf(" %d ",a[i]);

return 0;

}

**FUNCTIONS:**

* A function is a self-contained block that performs a particular task.

1.standard libraries

printf,sqrt,abs,pow

2.user defined

user is defining their own task to be performed

**SYNTAX: rdt fName(input args)**

{

sts;

return rdt;

}

* **No need to declare the name of the variable when you are declaring the prototype**.

**Function defines ---- implimentation in .c file**

**interface .h files**

**Recursion Function:**

A function calling itself.

It causes stack overflow because we call the same function multiple times.

I recursive it does not destroy the address it only destroy the value.

The address is same for all the values.

Ex:

#include <stdio.h>

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("\nV value i Func: %d\n",v);

return v;

Eg: change Value

#include <stdio.h>

#include <stdlib.h>

int changeValue(int[],int);

void disp(int[],int);

int main(){

int a[5] = {1,2,3,4,5};

printf("\nBA of array(main): %u\n",&a[0]);

disp(a,5);

return 0;

}

void disp(int arr[],int n){

int i;

printf("\nBA of arr (disp): %u\n",&arr[0]);

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

printf("%d",arr[i]);

printf("\n\n");

}

* **pointer size is --- 4**

E.g: Replace the 3rd element with 40.

#include <stdio.h>

#include <stdlib.h>

int changeValue(int\*,int);

void disp(int[],int);

int main(){

int a[5] = {1,2,3,4,5};

disp(a,5);

//int CAP = sizeof(a)/sizeof(a[0]);

changeValue(a,2);

disp(a,5);

// printf("\nBA of array(main): %u\n",&a[0]);

// printf("\nCAP = %d\n",CAP);

//disp(a,CAP);

return 0;

}

void disp(int arr[],int n){

int i;

//printf("\nBA of arr (disp): %u\n",&arr[0]);

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

printf("%d ",arr[i]);

//printf("\n\n");

}

int changeValue(int arr[],int n){

arr[2] = 40;

}