**Arrays**

Conjugative memory location(1 after another)

It should use on one tast like if it has phone number and marks we should not combile both becoz both are different purposes

Syntax :

Dt ArrName[CAP];

Int arr[10] ;

Float arr1[10];

Struct emp

{

Int id;

Char name[20];

};

Struct Emp e[10];

// access elements of arr to store or evaluate

=>index value from 0 to cap-1

Arr[2] = 20;

**BA+(IV \* sizeof(dt))**

Ao(2002+(0\*2))

The index values always in be whole numbers

Thera are different types of arrays

1. static array
2. dynamic
3. stetchable array
4. mutable array

**static array** : the size of the array is known before to the compilation time

ex: int arr[5]

**dynamic arra**y :the size of array is aocated or known at run time (malloc realloc calloc )

**strchable array :**sixe of the array are dexreser or increased for dynamic (malloc,calloc,realloc)

**mutable array :** the size of the array is knownat the time of linking

2d dimensional

Rows and columns

**2d dimensional**

**Rows and columns**

#include <stdio.h>

#define row 2

#define column 3

Have to use macros when u use rows and columns

**FUNCTIONS**

1. Standard libraries – printf,srrt,abs,pow
2. User defines – user defining his or her own task to ne performed

Rdt fname(input args)

{

Sts;

Return rdt;

}

Int add(int,int);🡪 declaration

Int add (inta val1, int val2){

Float result = val1+val2;

Return result;

}

ny function used the \_ it is meant for specifu function to perform the tasks

The scope of the result ialso destroyed

Check the return data type and result data type while u r executing in function

Different types of functions

With out i/p and return type 🡪 void disp()

With i/p and return type 🡪int add(int , int)

No return type and i/p ags 🡪void namechange(\*char)

Calling has address and return address

Calling address is in same page and return address in same or diffr

We pass array in function

#include stdio.h>

#include <stdlib.h>

Int changeValue(int [],int);

Void display(int[],int);

Int main ()

{

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

Disp(a,5);

Return 0;

}

A screen shot of a computer program

Description automatically generated

Recursive function is 🡪 the main is called by itself

V value in fun : 0

V value in fun : 1

V value in fun : 2

V value in fun : 3

V value in fun : 4

Res = 4

The different

Whether the address is same for that called in **recursive function** stack frame **yes the address is same**

**In The function has different address and the address also destroy**