**DAY -9 (6-11-2024)**

**Nesting of for loops:**

* convert for loop from entry controlled to exit controlled ?
* Goto statement : The goto statement in C is used to transfer control to a specified label within the same function. It is an unconditional jump statement that can be used to jump from anywhere to anywhere within a function

label: is not a executable statement.

A screen shot of a computer code

Description automatically generated

**Arrays:**

* It is used for a particular task or purpose
* And also have the same data type
* [] is used to declare an array.
* Memory is stored in contiguous manner(sequence form)
* Size is fixed
* Declaration for an array:

Syntax: Datatype ArrName[capacity or size]

Eg: int arr[10];

For structure :

Struct emp

{

Int id;

Char name[20];

};

Struct emp e[10];

* Access elements of array to store or evaluate

Syntax: ArrName[indexValue]=value;

* indexValue from 0 to size-1

eg: arr[2]=20;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010

This is arr[5] if arr[2]=10; for accessing and updating we use address of(2004)=10;

While printing it deferencing the address and takes the contents of that address i.e contentsof[2004]

* allocation of address is starts from bottom to top
* int a=10;

int b[2]={1,2};

int c=20;

printf(“%d %d “,b[0],b[1]); 🡺 1,2

printf(“%d %d “,b[2],b[-1]); => 10,20 which is a,c is stores

we consider here base address=> 2002 for that array but the values are start at base address of 2000 the allocation of memory is done by

2000 => c =20

2002=>b[0]

2004=>b[1]

2006=> a=10

It stored from bottom to top

So the answer for this is printf(“%d %d “,b[2],b[-1]); => 10,20

* b[-1] ==20
* b[2]==10
* BASE ADDRESS+(INDEX VALUE \*Sizeof(datatype));
* For Storing :

Address of( BASE ADDRESS+(INDEX VALUE \*Sizeof(datatype)));

For accessing :

Contents of( BASE ADDRESS+(INDEX VALUE \*Sizeof(datatype)));

* Index values cannot be a float value it have to be whole numbers(integer).
* We access the elements ay using scanf

For(int i=0;i<n;i++)

{

Scanf(“%d”,&n);

}

* Different types of an array:

1. **Static array** : the size of the array is known before to the compilation time.it is in stack memory

Eg: int arr[5];

1. **Dynamic array** :the size of the array is allocated or known at the runtime. It is inheap memory

Eg: calloc,malloc,realloc

1. **Stretchable array**: the size of the array is stretchable (decrease or increase) depending on the need for dynamic arrays

Eg : malloc, calloc,realloc

1. **Mutable array**: the size of the array is known or allocated at the time of linking and before execution.

Question 1) Write a program to store the odd numbers in an array between n and m

**For 2-D array:**

* Declaration:

datatype arrName[row][col];

Eg: int a[2][3] =>6 elements are occupied

* Initialization:

int arr[2][3]={{1,2,3},{4,5,6}};

(or) j=0 j=1 j=2

int arr1[2][3]={1,2,3,4,5,6}; => 1 2 3 ---i=0

4 5 6 ----i=1

it will try to arrange first in columns and next rows

**Rules:**

* Y[2.5] access y[2]
* Y[10.5] access y[10]
* Y[x], x is an int
* Arr[2] and 2[Arr] it is same it gives the value that present in the index 2.

**Functions:**

* It is a block of code it performs certain task or particular purpose

1. std lib function

printf,sqrt,abs,pow

1. user defined – user is defining his/her own task to be performed

* Syntax :

returndatatype filename(input arguments)

{

Statements;

Return returndatatype;

}

* Any function starting with the underscore it refers the special purpose
* Definition:

int add(int val1,int val2)

{

int result=val1+val2;

return result;

}

* The scope of the “result “ here local to that function and also val1 and val2

These are in the stack

Prototyping or declaring : int add(int,int); [no need of using variables]

* Function declaring is used in .h files and definition is write within the .c file
* First write the declaration and then definition
* 1) functions without return type and without arguments

Void display();

2) functions with return type and with arguments

Int add(int ,int);

3)functions without return type and with arguments

void changeName(char\*);

* int changeValue(int [] ,int) ; =>declaration for the array here second int is for the size
* **pointers are same as a arrays**

int changeValue(int\*,int);

to known the size of an array is size =sizeof(arr)/sizeof(arr[0]);

program :

#include <stdio.h>

#include <stdlib.h>

int changeValue(int \*, int);

void disp(int [], int);

int main()

{

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

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

printf("\nBA of arr (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");

}