Graphical user interface

Description automatically generated with medium confidence

NAME:

SYED MUHAMMAD SHAHEER ALI SHAH

REGISTRATION NO.

FA20-BCS-079

SECTION:

2C

SUBJECT:

PROGRAMING FUNDAMENTAL

SUBMITTED TO:

SIR ALI IQBAL

ASSIGNMENT NO.

02

**A picture containing building, outdoor, grass, sky

Description automatically generated**

**QUESTIONS:**

* What are pointers?
* Why they are needed?
* What is static memory and What is dynamic memory?

**QUESTION NO 1:**

What are pointers?

**ANSWER:**

POINTER IS A VARIABLE WHICH STORE THE ADDRESS OF ANOTHER VARIABLE.

Pointer is the powerful feature of C language.

**Types of pointer:**

Pointer can be of type **integer**, **floating points**, **functions** or any other pointer.

**Example:**

If Var is a **Variable** and Ptr is a **Pointer**, then if Var is of integer type then the pointer Ptr which store the address of Var must be of integer type same as the case for floating points.

**Declaration of pointer:**

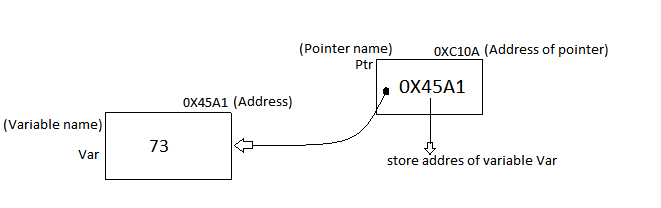
Pointer in C can be declared using **\*(asterik symbol).** Pointer must be declared before they are used.

**Example:**

Pointer are declared as:

**Int \*Ptr;**

**How pointer is initializing and how they work?**

Firstly, we declared a variable like **int Var;**

Then declared a pointer like **int \*Ptr;**

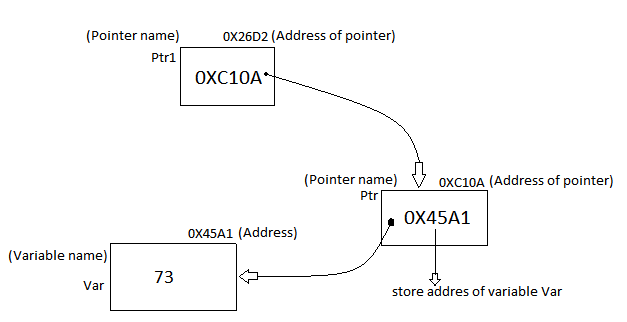
Then store the address of variable Var in the Pointer Ptr like **Ptr = & Var** (we can also declared and intiliaze them in the same line).

Here the pointer **Ptr** store the address of variable **Var**.

**Ptr Points to Var.**

**Pointer to Pointer:**

We can use another pointer which store the address of previous pointer.



**& and \* operators:**

**&** is called the **address operators**. That show the address of a variable.

**i.e** if Var is a variable store a number 73 then **&Var** gives us the address at which the value 73 is stored.

**\*** is called **dereference, in directional and unary operator**. that reference the value store in Var through a pointer.

**i.e.** Ptr is a pointer store the address of variable Var if we write **\*Ptr** then it gives the value store in that address which is 73.

**NULL Pointer:**

A pointer that is not assigned by any value but NULL is NULL pointer.

NULL pointer doesn’t point out any object.

**SOME IMPORTANT PRINTING VALUES:**

Int Var = 73;

Int \*Ptr = &Var;

Printf(“%d”,Var); Print the value that store in variable Var **(73)**.

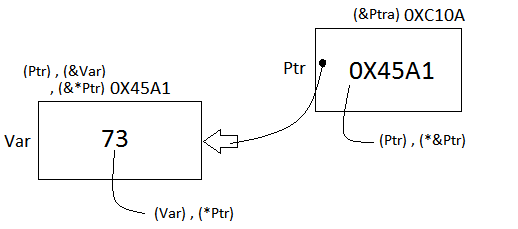
Printf(“%p”,&Var); Print the address of variable Var **(0X45A1)**.

Printf(“%p”,Ptr); Print the address of variable Var **(0X45A1)**.

Printf(“%d”,\*Ptr); Print the value that store in variable Var **(73)**.

Printf(“%p”,&Ptr); Print the address of pointer Ptr **(0X45A1)**.

Printf(“%p , %p”,\*&Ptr,&\*Ptr); Print the address of variable Var **(0X45A1)**. (both statement are same).

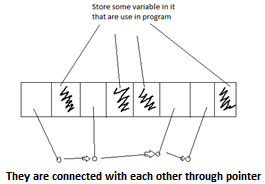


**QUESTION NO 2**

Why they are needed?

**ANSWER:**

* Our memory is a contagious block. If our memory is not free in contagious manner if there is some value store in the mid then we can’t use array Therefor here we use pointer by making a link list between different memory location and point each memory through pointer so, here we traversing and searching between them.



* In dynamic memory allocation pointers are needed.
* By using pointer, we can return multiple values from a functions.
* They are needed because pointer reduced the code and improve the overall performance.

**QUESTION NO 3**

What is static memory and what is dynamic memory?

**ANSWER:**

**STATIC MEMORY DYNAMIC MEMORY**

|  |  |  |
| --- | --- | --- |
| 1 | In the static memory allocation, variables get allocated permanently. | In the Dynamic memory allocation, variables get allocated only if your program unit gets active. |
| 2 | Static Memory Allocation is done before program execution. | Dynamic Memory Allocation is done during program execution. |
| 3 | It is less efficient | It is more efficient |
| 4 | In Static Memory Allocation, there is no memory re-usability | In Dynamic Memory Allocation, there is memory re-usability and memory can be freed when not required |
| 5 | In static memory allocation, once the memory is allocated, the memory size cannot change. | In dynamic memory allocation, when memory is allocated the memory size can be changed. |
| 6 | In this memory allocation scheme, we cannot reuse the unused memory. | This allows reusing the memory. The user can allocate more memory when required. Also, the user can release the memory when the user needs it. |
| 7 | In this memory allocation scheme, execution is faster than dynamic memory allocation. | In this memory allocation scheme, execution is slower than static memory allocation. |
| 8 | In this memory is allocated at compile time. | In this memory is allocated at run time. |
| 9 | In this allocated memory remains from start to end of the program. | In this allocated memory can be released at any time during the program. |
| 10 | **Example:** This static memory allocation is generally used for [array](https://www.geeksforgeeks.org/introduction-to-arrays/). | **Example:** This dynamic memory allocation is generally used for [linked list](http://www.geeksforgeeks.org/data-structures/linked-list/). |