What is the Size of a Pointer?

A Pointer are declared using data type. But its size is not dependent on its data type.

Purpose of data type is for pointer arithmetic. Not for size.

Note: in whiteboard lecture I am assuming pointer takes 2 bytes, to make explanation easy.

In Present day compilers **every pointer takes 8 bytes**.

int *p1;

float *p2;

double *p3;

char *p4;

struct Test *p5; // pointer to structure. (You will learn it later)

Rectangle *p6; // pointer to an object. (You will learn it later)

All pointers from p1 to p6, they take 8 bytes.

Declaring and Initialising Pointer.

Example:

int x=10; is a data variable.

int *p=&x; p is a pointer and will be pointing on x. p is declared and initialised.

Example:

int x=10; is a data variable.

int *p; p is declared.

p=&x; p is initialised. It will be pointing to x. * is used for declaration.

What Is Meant By Dereferencing

Dereferencing: Accessing the data present at the address where pointer is pointing.

int x=10; x is a data variable

int *p=&x; p is a pointer pointing to x

cout<<*p; this will print the value of x. *p is dereferencing.

Dynamic memory in Heap.

What is Dynamic memory in Heap?

Dynamic memory is created in Heap using pointers.

We can use Dynamic memory for creating Array, Linked List, Trees, Map etc. (These are called as Data Structures)

Heap memory can be accessed from anywhere in the program, if its address is available.

Heap Memory can be used for storing data of entire application.

Application of Dynamic memory?

We take a example of Chrome browser and discuss from Chrome side..

When we open chrome browser, it doesn’t know how many tabs user will open.

It will keep the first tab ready with Google page open.

Pages are downloaded from internet and kept in the memory of Chrome program.

Downloaded page is kept in heap memory.

When we open new tab then memory is created in heap for that page.

When we close a tab the memory should be deleted, as it is not in use.

Conclusion: we may be opening and closing many tabs.

Chrome doesn't know how many we will open.

Chrome should allocate memory for the pages at runtime.

It will allocate pages Dynamically in Heap memory and delete them when not required.

Memory Leak: If a program like Chrome is allocating memory for tab but not deallocating it when tab is closed then it is called as **Memory Leak**

Memory Leak

If a program requires memory at runtime, it will allocate in Heap using pointer.

If Heap memory is not in use, it should be deallocated.

Memory Leak: If a program like Chrome is allocating memory for tab but not deallocating it when tab is closed, then it is called as **Memory Leak.**

If a program is not deallocating then the memory reserved for Heap may become empty.

Dangling Pointer

If a pointer is having an address of a memory location which is already deallocated.

example:

```
int *p=new int[5];

delete []p;
```

Now p is a **Dangling pointer**.

Stack vs Heap memory

Stack:

- all the variables we declare in a function are created in stack.
- All variables are accessible only within that function.
- Compiler can finalise the amount of memory required for all the variables.
- When a function is called, memory is created in stack
- When a function terminates, memory is deleted from stack.

Heap:

- pointer is used for allocating memory at runtime.
- new operator is used for allocating memory in heap.
- Heap memory should be allocated when required and deallocated when not in use.
- Heap memory can be accessed by entire program if pointer is available.

NULL vs nullptr

NULL:

- it is a constant whose value is 0.
- NULL means, pointer is not pointing on any valid location.
- In place of NULL, 0 can be used.
- Using 0 I place of NULL may create confusion for programmer.

nullptr

- It is a keyword in C++.
- nullptr means, pointer is not pointing on any valid location.(same as NULL)
- Nullptr doesn’t mean 0.
- 0 cannot be used in its place.

Address of Array

Name of an array is the base address of an array

Name itself acts as a pointer. Though it is a name, not a pointer. As it represents address, it act as a pointer.

Example:

int A[5]={2,4,6,8,10};

int *p;

p=A; // this will store the base address in p. It will be address of 2.

p=&A;// this is invalid.

p=&A[3]; // this will store the address of 8. & should be used if index is given.

r-value and l-value

a variable will have attributes. 2 of them are address and value. I will explain thru example.

int x=10,y;

x=20; // 20 is stored in x, it needs address attribute of x. x is on left side of = , so it is l-value

y=x; // value of x is stored in y, x is on right side, its value attribute is used. it is r-value.

y takes l-value and x takes r-value.