Section 9 : Pointers

15 April 2022 16:49

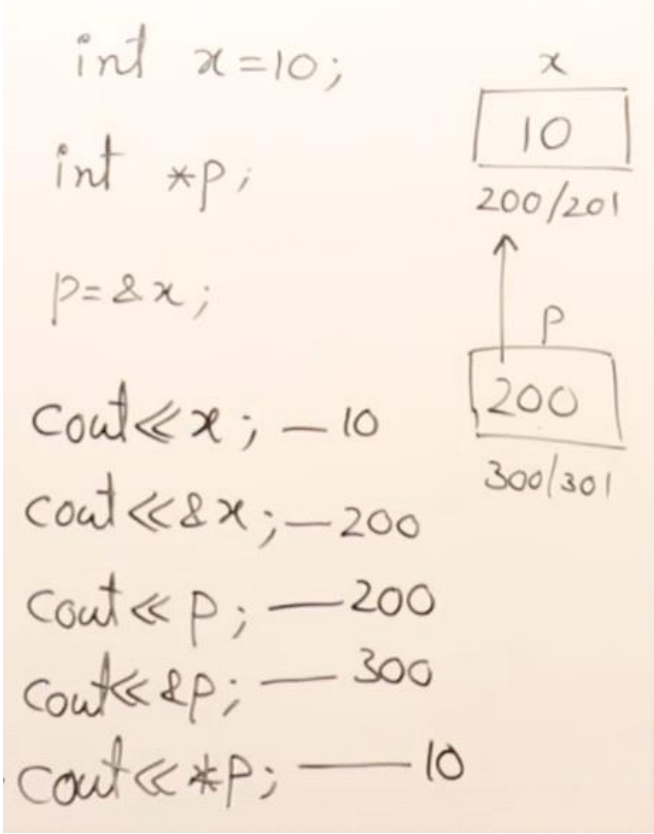
© Rajat Kumar
https://www.linkedin.com/in/imRajat/
https://github.com/im-Rajat

Section 9 : Pointers

Pointers are the types of variables in CPP. It is variable used for storing the address of data.

We have 2 types of variables :

- Data variables (used for storing data) - int x = 10;
- Address variable (used for storing address) - int *p; p = &x;



Int *p // declaration
P = &x // initialization
Cout<<*p; // it's called dereferencing

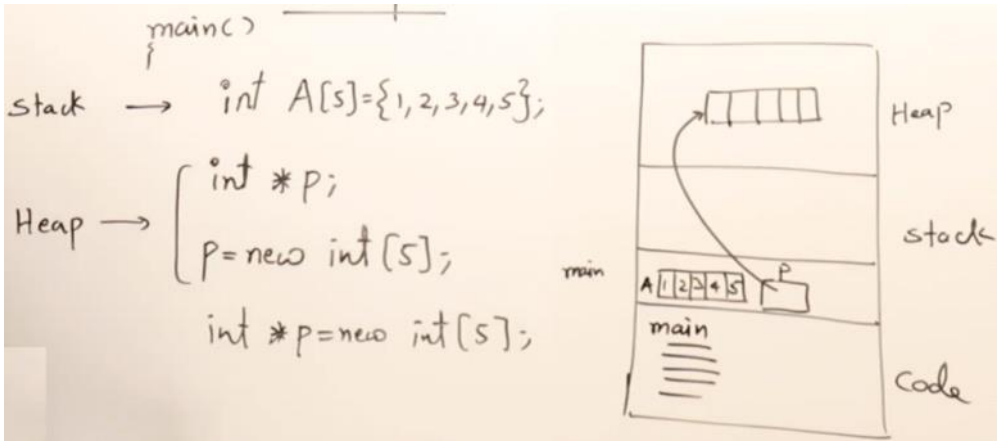
Why Pointers? :
A program can only access code section and stack directly. Pointers help it to access heap section. A pointer is in stack but it's stored address is in heap. So indirectly it helps us to access heap section.

- To access heap memory
- To access to file using file pointer
- To access network connections
- To access devices: keyboard, mouse, printer, etc

- In java there are no pointers in java and c#, so we can't access devices through programs, can access only using JVM or throgh common languages of run time in C$.
- Therefor there is no system programming in java and c#

Heap :

- Dynamic memory
- Memory decide at run time not compile time
- Use new, memory created in heap
- Heap memory don't delete automatically until program is over



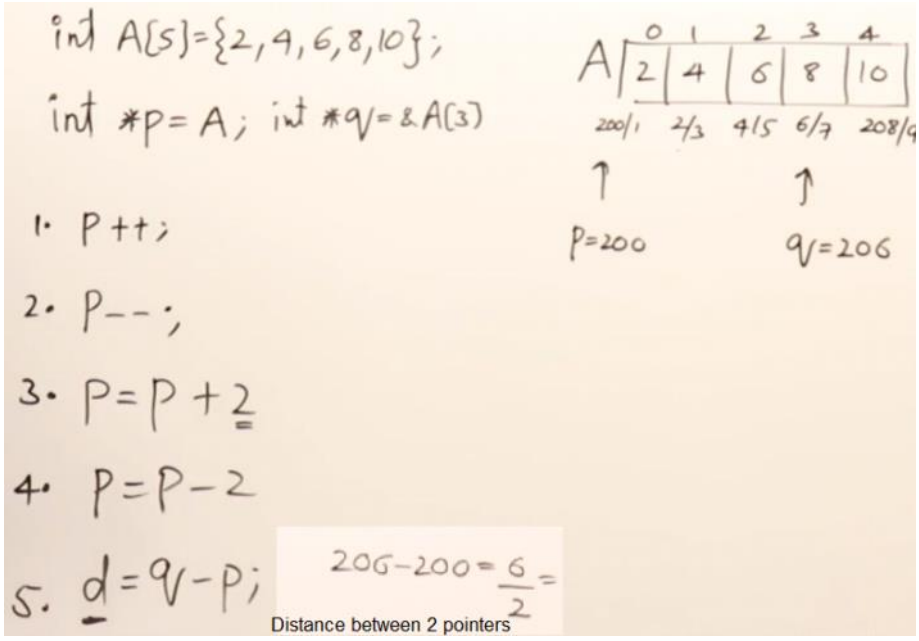
Memory Leak :
We created a memory using new and pointer it to a pointer p, but then we set p to null, after that we don't have access to th at memory. Therefore we need to free the memory (heap) first.

Delete []p;

P = NULL;

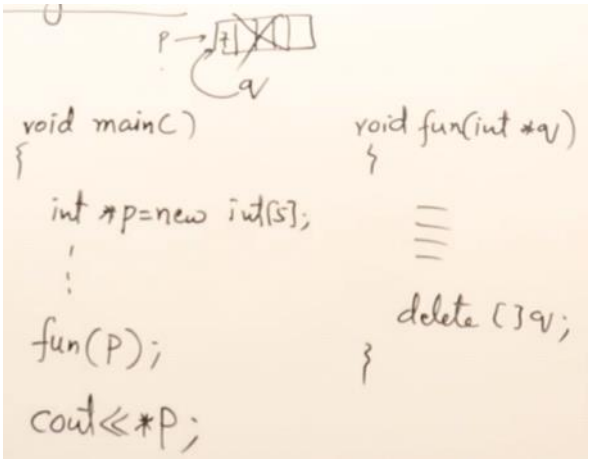
Null Pointer : A pointer pointing to NULL;

Pointer Arithmetic :



Problems with Pointers :

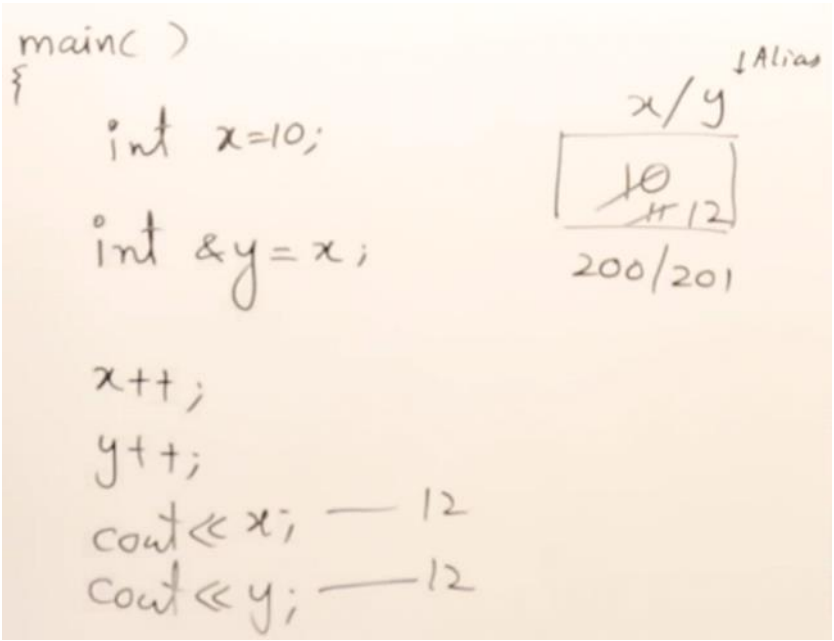
- Uninitialized pointer
  - Created a pointer (int *p) and then assign (*p = 2.5) but pointer p is not pointing to any address
- Memory leak
  - Delete memory before setting pointer to null
  - NULL = 0 = 0 nullptr (nullptr used in modern c++, recommended, it's an address).
- Dangling pointer
  - A pointer pointing to a location which is not exist (deleted/deallocated).

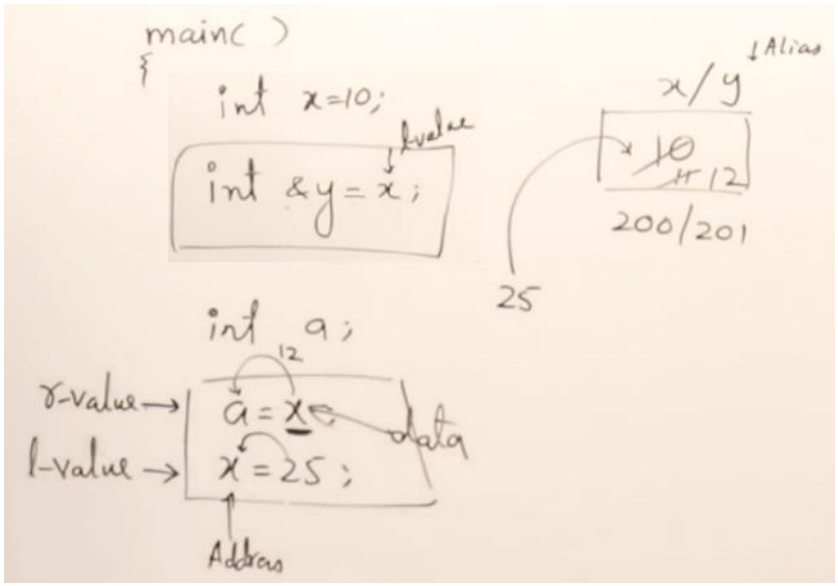


References :

Int x = 10;
Int &y = x; (it's a reference, must initialize at that time)

- Reference is nothing but a alias/nickname of a variable. Reference doesn't consume any memory.
- Declaration of reference variable requires an initializer.
    - Int &y = x; (correct)
    - Int &y; (not possible)
- We can't change reference again.
- Int &y = a; this is not possible. (because it's already initialize as int &y = x)
    - The reference y can't reference any other value at all now.





- size of a pointer is independent of its data type. int *p1; or float *p2; or char *p3; all takes 8 bytes in latest compilers. (Note: I am assuming pointer takes 2 bytes to make explanation simple)
- pointer increment will move the pointer depending on the data type of pointer. int is 4 bytes so pointer will move by 4 bytes. if pointer is char type then it will move by 1 byte

```
int A[]={2,4,6,8,10,12};
int *p=&A[3];
cout<<p[-2];
```

- int *p=&A[3]; p will be pointing on 8 at index 3. p[-2] means 2 index backward. cout<<p[-2]; will print 4.

```
int x=10;
int &y=x;
y=x+y;
cout<<x;
```

- y is a reference to x. it means x and y are 2 names of same variable. y=x+y; is y=10+10=20. y becomes 20, it means x also becomes 20. so 20 is printed

```
int x=10;
int *y=&x;
int * &z=y;
```

- x is a variable. y is a pointer variable, pointing to x. z is a reference to a pointer variable. int *&z=y; means z is another name of y. now y and z are 2 names of same pointer.