Programming Basics : Day 1

LIC
भारतीय जीवन बीमा निगम
LIFE INSURANCE CORPORATION OF INDIA

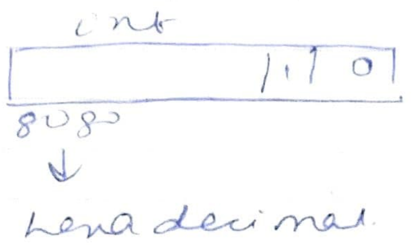
Address of operator

& finds address of variable

x = 10

cout << $\underbrace{\&x}$

hexadecimal numbers



Exception Does not work for char variables

```
char ch = 'A';
cout << & ch; // A
cout << (void*) & ch << endl
```

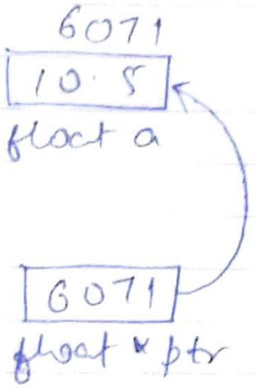
(void*) → explicit typecasting. char* to void*

Pointers

Variable that stores address of another var.

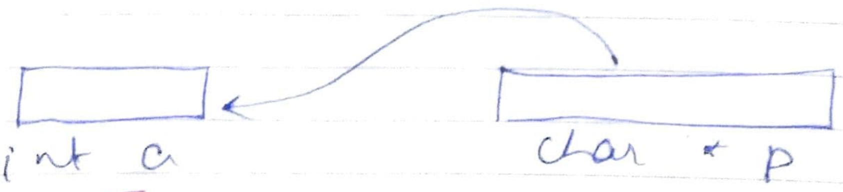
Datatype * var name

int *y = &x   Declaration & initialization



float a = 10.5
float * aptr = & a

[Garbage value if not initialized]

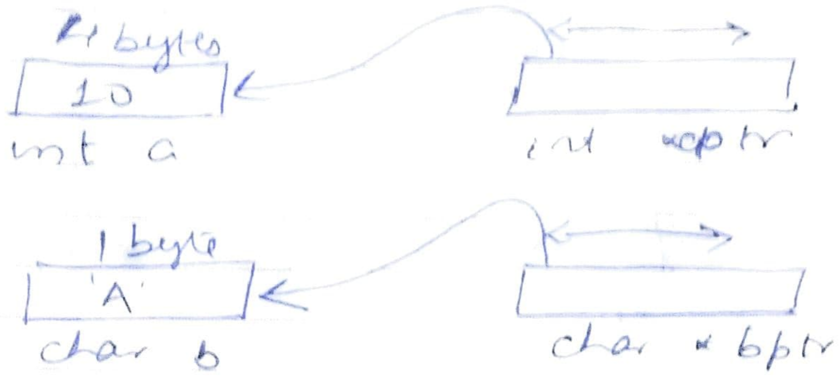


AVOID

[Dereferencing]

If we will read int value using a char pointer, then we get ~~cha~~ char in return instead of integer

Size of a pointer.

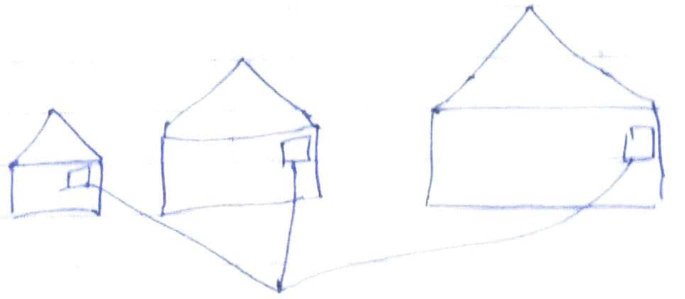


4 / 8 bytes depending on system

4 / 8 bytes

If 64 bit memory

Address of has size $0 \ldots 2^{64}$

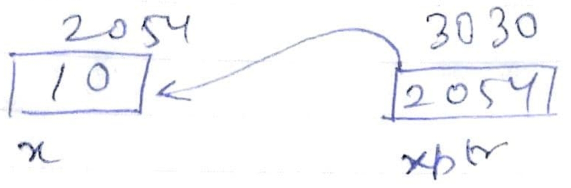


Same address space

We can always reassign address of variable.

Deferencing Operator (*)

& Bucket → Address
* Address → Bucket



int x = 10
int *xptr = &x

→ ~~int~~ cout << *(&x) ⇒ x
→ cout << *(xptr) ⇒ x

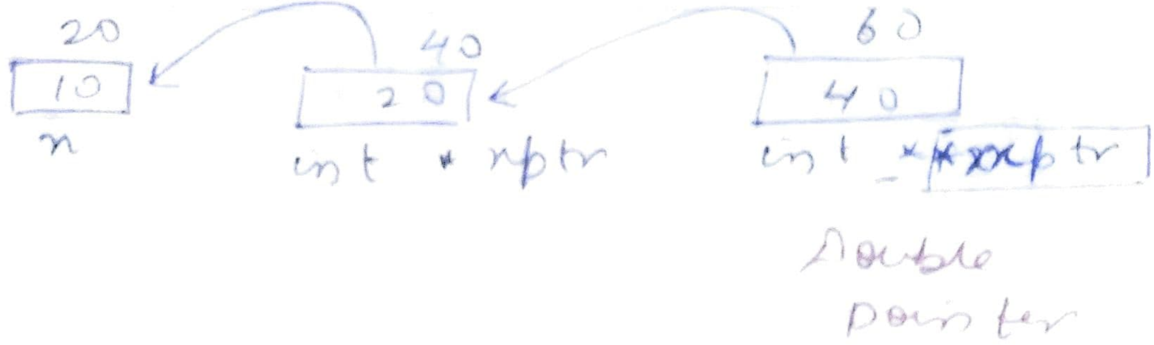
→ cout << *(xptr) + 1 ⇒ x+1 = 11

→ cout << *(xptr+1) ⇒ add $\left(\begin{matrix}2054 + 4 \\ 2058\end{matrix}\right)$
⇓
Some garbage value

→ cout << $\underbrace{*(\&xptr)}$ << endl;
*(3030)
⇓
$\boxed{2054}$ ⇒ xptr

→ cout << &(*xptr) << endl;
&([illegible])
⇓
&(10) ⇒ $\boxed{2054}$ ⇒ xptr

LIC
भारतीय जीवन बीमा निगम
LIFE INSURANCE CORPORATION OF INDIA



~~Call Stack~~

Null pointer
If we make out pointer, point to nothing.
double *p = 0;

Arrays & pointers

→ Linked in a very complicated manner

→ An array is actually a pointer that points to the first element of the array.
Array variable is a pointer, referring to element 0.

→ a[i] is same as *(a + i)

LIC
भारतीय जीवन बीमा निगम
LIFE INSURANCE CORPORATION OF INDIA

Difference - Arrays & Pointers.

1. The size of operator

⇒ size of (array) ⇒ Returns amount of memory taken by array.

int array [3] = { 1 , 2, 3 }

cout << size of (array) ; ⇒ 12
3 x 4

⇒ size of (pointer) ⇒ Return amount of memory used by pointer variable itself.

size of (nptr) ⇒ $\boxed{4}$

2. The & operator

⇒ &(array) ⇒ & array [0]

&(pointer) ⇒ address of pointer

3. <u>String literal initialization of char array</u>

⇒ char array[ ] = "abc"

```
array[0] = 'a'
  "   [1] = 'b'
  "   [2] = 'c'
  "   [3] = '\0'
```

⇒ char *ptr = "abc"

It sets pointer to the address of the "abc" string (Immutable)

4 <u>Assignment / Re-assignment</u>

```
int  a[10];
int  *p;
~~int~~  p = a;  //allowed
         a = p;  // Not allowed
```

5. Arithmetic

```
int   a[10];
int   *p;

a++   // Not allowed.

P++   // Allowed
```