BASICS MATHS & POINTERS - LEVEL 1

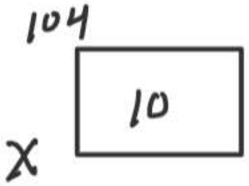
05 October 2023 08:36

04/10/2023

POINTER CLASS-1

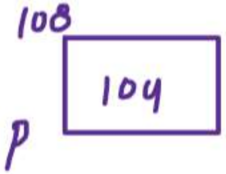
1. What is pointer: *A pointer is a variable that stores the memory address of another variable.*

int x = 10



{Address of x = 104}

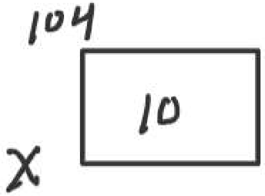
int* p = &10 ;



{value of p = Address of x
= 104}

2. Address operator: *The Address-of operator (&) is a unary operator that returns the memory address of its operand which means it stores the address of the variable*

int x = 10



SYMBOL TABLE

x ⟶ 104

x mapped with 104 address

Address = 104
value = 10
name = x

```
cout << &x ;
cout << x ;
```

3. Creation of pointers: Pointers are created by using the * operator

$$\underbrace{\underbrace{\texttt{int*}}_{1}\ \underbrace{p}_{2}}_{\text{p is a pointer to integer}} = \underbrace{\texttt{------}}_{3};$$

1) Pointing to integer data

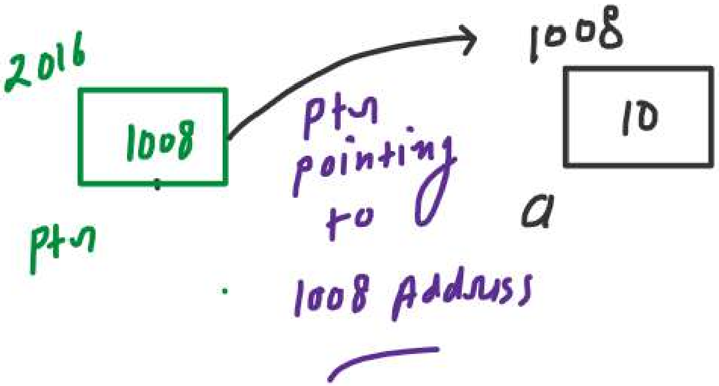
2) pointer name / variable name

3) Memory Address of another var.

4. Access pointer and dereference operator: *The indirection operator (or dereferencing operator) (*) operates on a pointer, and returns the value stored in the address kept in the pointer variable. For example,*

```
int a = 10;

int* ptr = &a;
```

Access: value stored at address stored in ptr



```
cout << *ptr;
    └→ output = 10
```

a = 10	ptr = 1008
&a = 1008	&ptr = 2016
	*ptr = 10

Why pointer size was coming 8 while printing ?

```
int a =5 ;
int* p = &a ;
```

↳ sizeof(P)=8

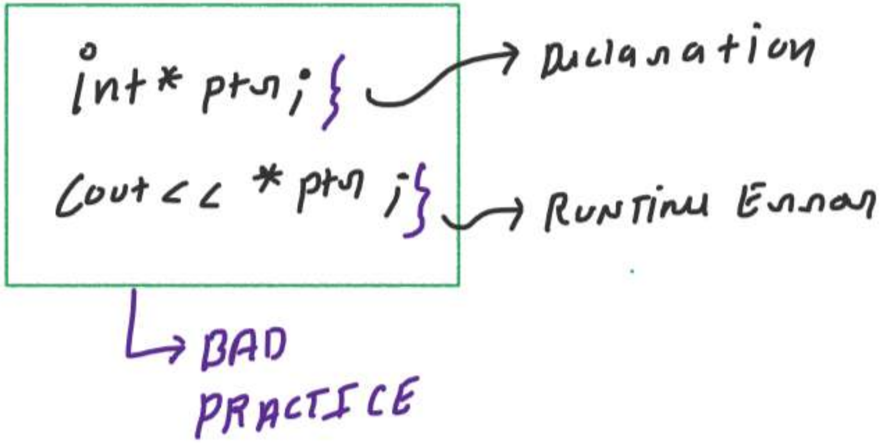
```
char x='A';
char* p = &x;
```

↳ sizeof(P)=8

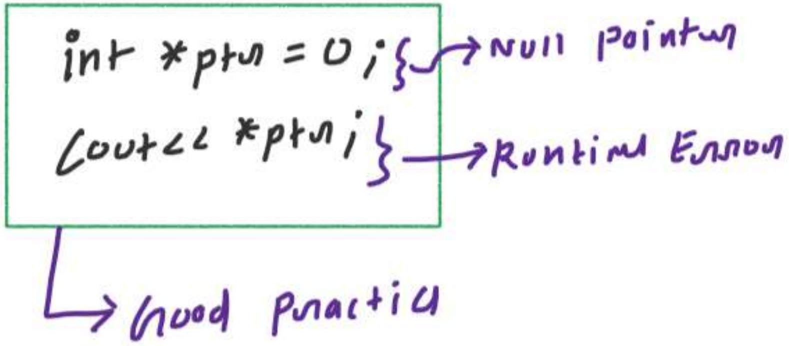
```
long y =10;
long* y = & y;
```

↳ sizeof(P) =8

5. Declaration of pointer: *uninitialized pointer is a bad practice with pointers because of illegal memory access. In short, Anytime a pointer is dereferenced and does not point to valid memory will cause an error.*



```
int* ptr;
cout<< *ptr;
```



```
int *ptr = 0;
cout<< *ptr;
```

5 Practice Questions

1

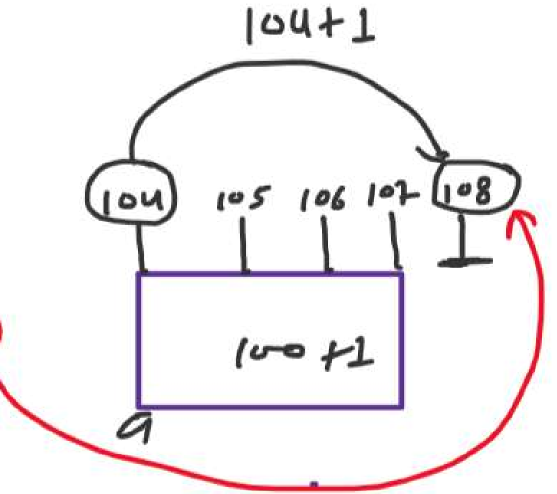
```
int a = 100;
int *ptr = &a;
a = a+1;
ptr = ptr + 1;
```

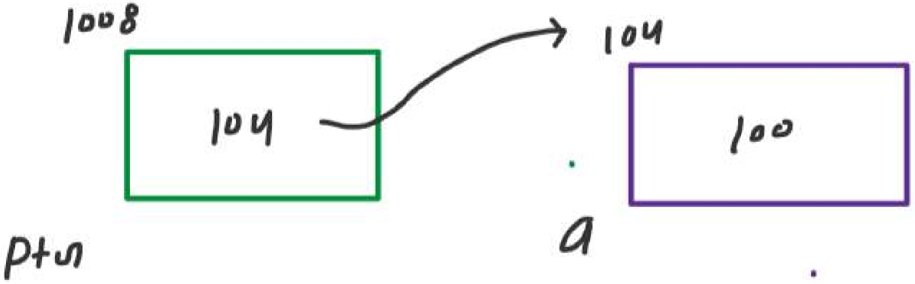
Cout

a = 100+1 = 101

ptr = 104 + 1 = 108

4byty





②

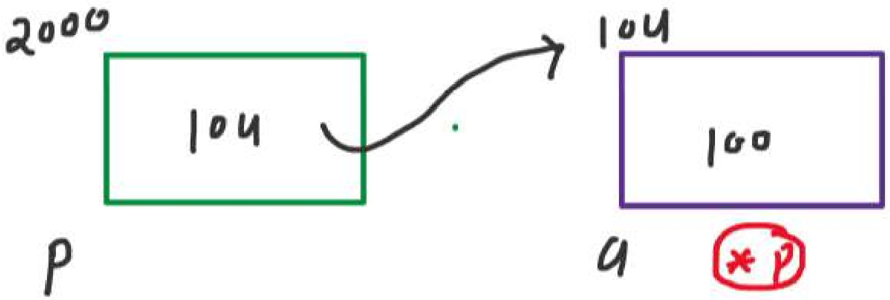
```
int a = 100;
int*p = &a;
 a = a+1;
*p = *p+1;
```

$$*p = \underbrace{*p}_{101}+1;$$

Cout

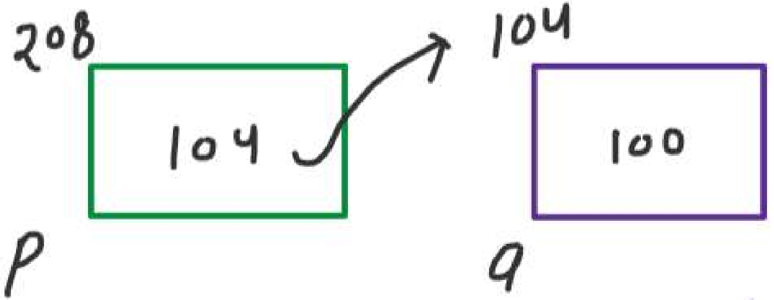
a = 100 + 1 = 101

*p = 101 + 1 = 102



③

```
int a = 100;
int* p = &a;
```



Print

↳ a = 100

↳ &a = 104

↳ *a = ERROR

↳ p = 104

↳ *p = 100

↳ &p = 208

↳ (*p)++ = (100)++ = 101

↳ ++(*p) = ++(101) = 102

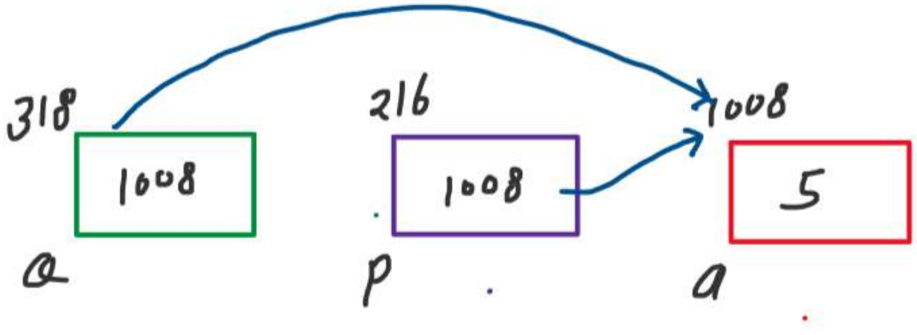
↳ $*p = \frac{*p}{2} = \frac{102}{2} = 51$

↳ *p = *p - 2 = 51 - 2 = 49

(4)

```
int a = 5;
int* p = &a;
int* q = p;
```

Pointer Copy



Print

a = 5

&a = 1008

*a = ERROR

p = 1008

&p = 216

*p = a → 5

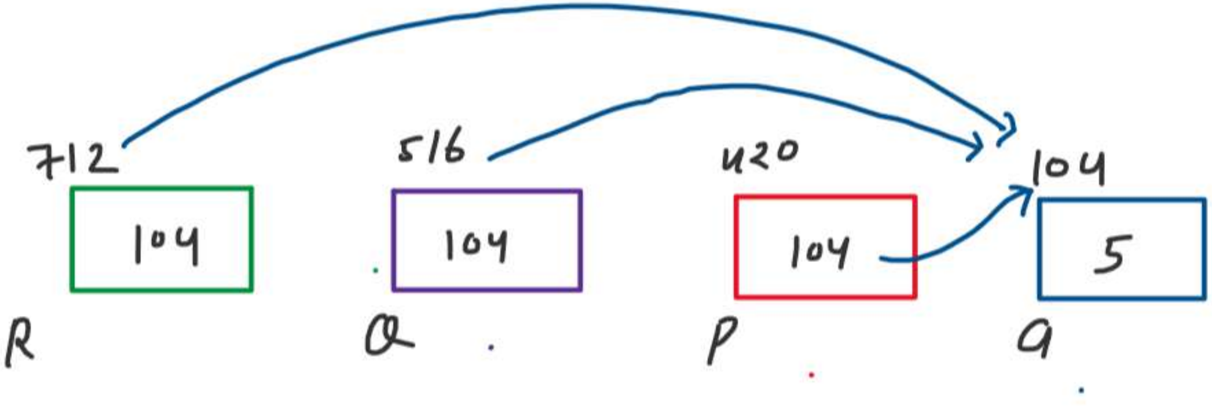
q = 1008

&q = 318

*q = a → 5

5

```
int a = 5;
int* p = & a ;
int* Q = P;
int* R = Q;
```



Print

a = 5	p = 104	Q = 104	R = 104
&a = 104	&P = 420	&Q = 516	&R = 712
*a = ERROR	*P = 5	*Q = 5	*R = 5

6. Pointer with array:

int arr[5] = { 10, 20, 30, 40, 50 };

	104	108	112	116	120
	10	20	30	40	50
arr	0	1	2	3	4

Print

arr = 104

arr[0] = 10

&arr[0] = 104

&arr = 104

Important point

arr
&arr[0]
&arr

} Base Address = 104

4 Practice Questions

1.

```
int arr[5] = {10, 20, 30, 40, 50}
```

	104	108	112	116	120
arr	10	20	30	40	50
	0	1	2	3	4

Print
↳ arr = 104
↳ &arr = 104
↳ arr[0] = 10
↳ &arr[0] = 104
↳ *arr = 10
↳ *arr + 1 = 11

↳ *(arr)+1 = *(104)+1 = 10+1 = 11
↳ *(arr+1) = *(104+1) = *(108) = 20 = arr[1]
↳ *(arr+2) = *(104+2) = *(112) = 30 = arr[2]
↳ *(arr+3) = *(104+3) = *(116) = 40 = arr[3]

* => value stored at address (116)

Note:1

```
*(arr+0) = arr[0]
*(arr+1) = arr[1]
*(arr+2) = arr[2]
   :          :
*(arr+i) = arr[i]
```

OR

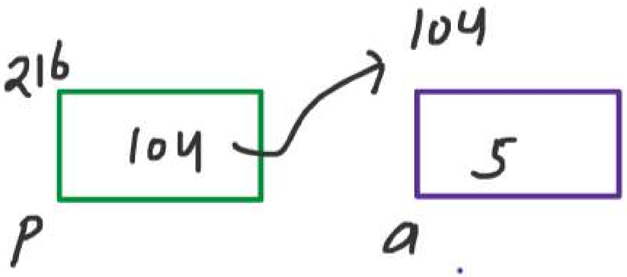
Remember

```
{ *(i+arr) = i[arr] OR
             arr[i] }
```

Note:02

(2)

```
int a = 5
int* p = &a
```



Print

```
p = p+1
cout << p ;
```

} output = Garbage value

```
int arr[5] = {10, 20, 30, 40, 50}
```

Print

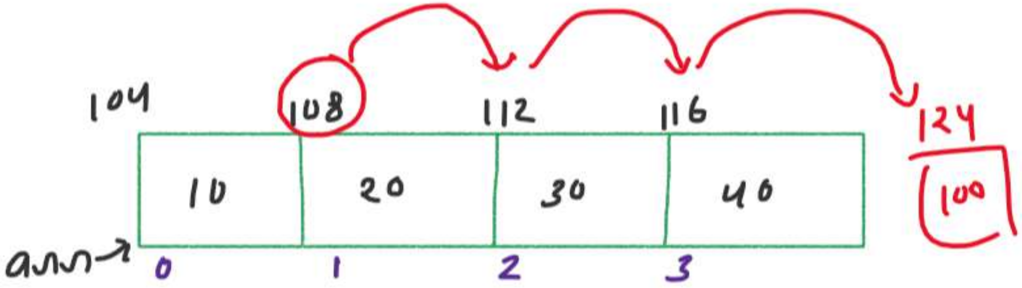
```
arr = arr + 1 ;
cout << arr ;
```

} Compile Time ERROR

HW 02: why we can not do [arr = arr + 1;] in C++

③

```
int arr [4] = { 10,20,30,40};
  int* p = arr;
  int* Q = arr+1;
```



Print

arr = 104

&arr = 104

arr [0] = 10

&arr [0] = 104

p = 104

&p = 104

*p = 10

Q = 108

&Q = 108

*Q = 20

$*p+1 = *(104)+1 = 10+1=11$

$*(p)+2 = *(104)+2 = 12$

$*(Q)+2 = *(108)+2 = 22$

$*(Q+4) = *(108+4\times4)$

$= *(124)$

$= 100$

4

```
int arr[4]={10,20,30,40}
cout << sizeOf(arr);
```

Output = 16

```
int *p = arr;

sizeOf(P);
```

Output ⇒ 8

arr [4]

Arr length

$$\text{arr}[4] \rightarrow \underset{\text{Arr length}}{4} \times \underset{1\ \text{int} = 4\ \text{bytes}}{4}$$

= 16

4 bytes	4 bytes	4 by	4bytes

Total size = 16

7. Char array and pointer

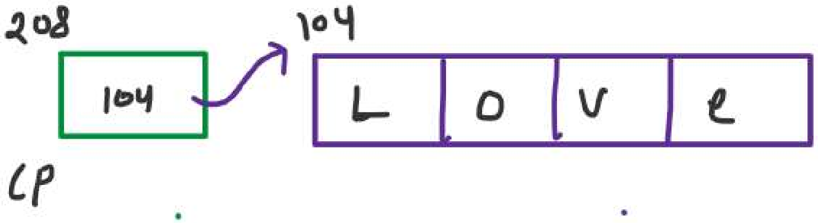
```
Char ch[50] = "Love";

char* cp = ch OR &ch ;

cout<< cP ;  → Love
cout<< ch;  → Love
```

4 Practice Questions

①

char ch[50] = "Love";

char* cP = ch ✓ or ~~&ch~~; ERROR



Print

ch = Love

&ch = 104

ch[0] = L

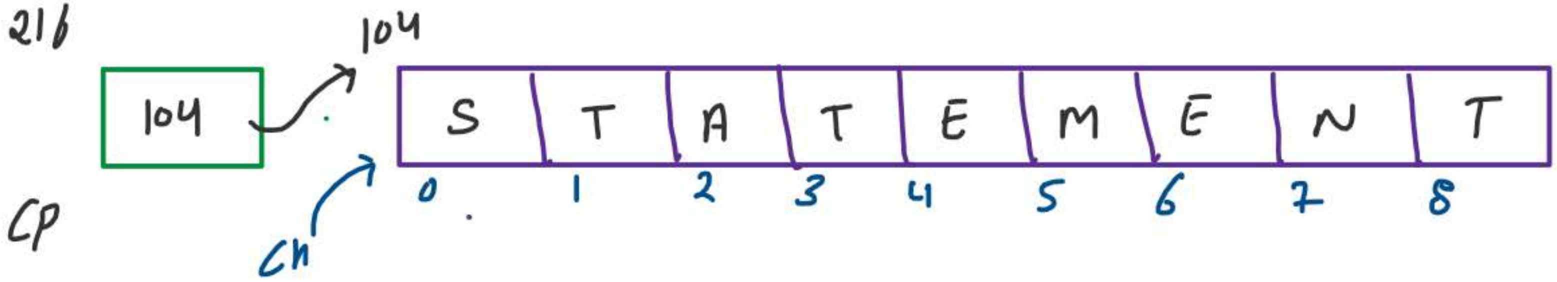
cP = Love

&cP = 208

*cP = *(cP+0) = ch[0] = L

2

```
char ch[50] = "STATEMENT"
char* cp = &ch[0];
           OR ch
```



Point

ch = STATEMENT
&ch = 104
*(ch+3) = ch[3]
= T

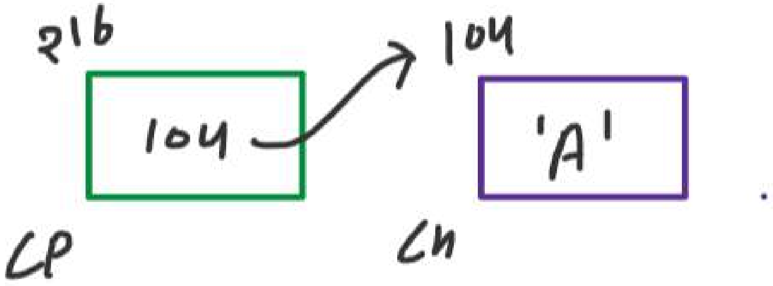
cp = STATEMENT
&cp = 216
*(cp+3) = ch[3]
= T

cp+2 = ATEMENT
*cp = S
*ch = S

3

```
char ch = 'A';
char* cp = &ch;
cout << cp;
```

Output: $A\underbrace{\square--\square---}_{\text{Random Character}}$



④

```
Char* cP = " BABBAR"
```

```
cout << CP ;
```

→ BAD PRACTIC

why?

→ "BABBAR" is stared in Temp storage.