

Pointers - 3

int a = 5;

int* p = &a;

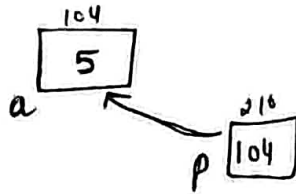
char ch = 'd';

char* cp = &ch;

int** ptr = &p;

→ double pointer.

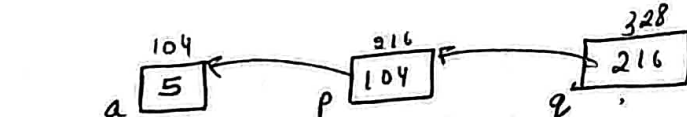
| Symbol Table | |
|--------------|-------|
| a | → 104 |
| p | → 216 |



int a = 5;

int* p = &a;

int** q = &p;



→ p is a pointer to int data.

→ q is a pointer to int* data.

cout → &a → address of a (here - 104)

a → 5

p → 104

&p → 216

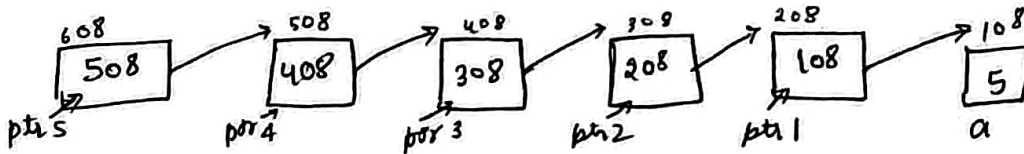
*p → 5

q → 216

&q → 328

*q → 104

**q → 5



a → *ptr1, **ptr2, ***ptr3, ****ptr4, *****ptr5, a

ptr1 → *ptr2, **ptr3, ***ptr4, ****ptr5, ptr1

ptr2 → *ptr3, **ptr4, ***ptr5, ptr2

ptr3 → *ptr4, **ptr5, ptr3

ptr4 → *ptr5

ptr5 → ptr5

① main() {

int a = 5;

int* p = &a;

print a, p, *p ⇒ 5, 104, 5

util(p);

print a, p, *p ⇒ 5, 104, 5

some(p); print a, p, *p;

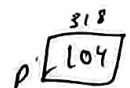
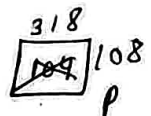


util(int* p) {

p = p + 1;

some(int* p) {

*p = *p + 1;



*p = 5
*p = 5 + 1 = 6

```
main() {
```

```
    int a = 5;
```

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

```
    print a; // o/p = 5
```

```
    util(p);
```

```
    print a;
```

```
    // o/b → 6
```

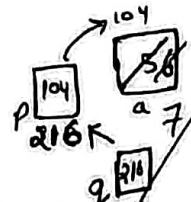
```
    int q = &p;
```

```
    solve(q);
```

```
    print a;
```

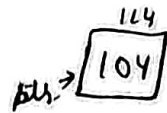
```
    // o/b → 7
```

*q = value present at location stored in q.



```
util(int *ptr) {
```

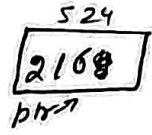
```
    *ptr = *ptr + 1;
```



*ptr = *ptr + 1;
*(104) = 5 + 1 = 6

```
solve(int **ptr) {
```

```
    **ptr = **ptr + 1;
```



*ptr = *ptr + 1
*(216) = *(216) + 1
= 104 + 1 = 108
← address

***ptr → 5

Value present at location stored in ***ptr

**ptr → 104

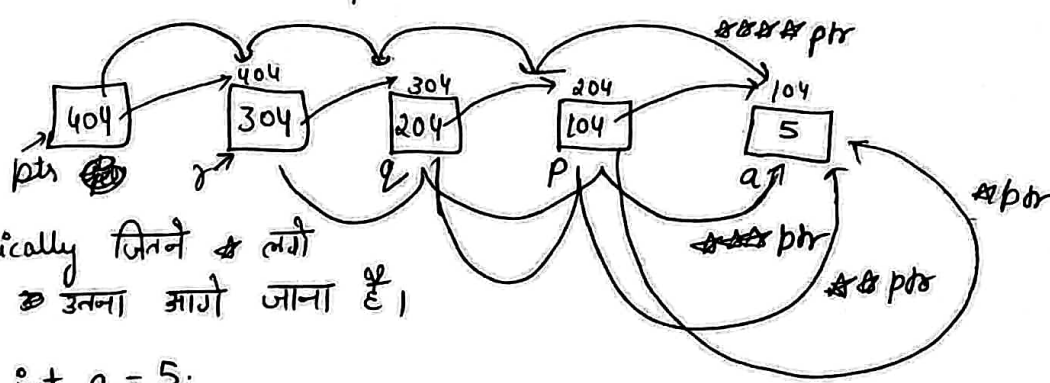
value present at location stored in **ptr

*ptr → 204

value present at location stored in *ptr

ptr → 304

value present at location stored in ptr. → 404



Basically जिनने * लगी है. उतना जागे जाना है,

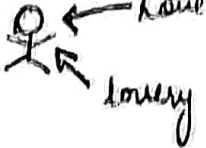
```
int a = 5;
```


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

```
int **ptr = &p;
```

*ptr = *ptr + 1; → Error. → either do this
**ptr = **ptr + 1
or
*ptr = *ptr + 1

① Reference Variable →


eg:  same person two names.

 value

$a++ \rightarrow c$
 $value++ \rightarrow 7$

Ques → Why we need reference variable when we have pointers?

$int\ a = 5;$
 $int\&\ b = a;$ → b is a reference variable & pointing to same memory location as a .

$cout << a; \rightarrow o/p \rightarrow 5$
 $cout << b; \rightarrow o/p \rightarrow 5$  $\leftarrow b$
 $a++;$

| Symbol Table | |
|-----------------|-----|
| $a \rightarrow$ | 104 |
| $b \rightarrow$ | 104 |

$cout << a << b << endl; \rightarrow o/p \rightarrow 5\ 6$
 $b++;$
 $cout << a << b << endl; \rightarrow o/p \rightarrow 7\ 7$


reference variable doesn't take extra memory, just their entries are stored in symbol table.

② Why to use Reference variables?


- We can't set reference variable to null but pointers can point to null, so there is more safety in reference variables. Reference variables always point to a valid memory location.
- Pointers are difficult to understand but reference variables have more readability.

③ Function → Pass by Value and Pass by Reference →

$main() \{$
 $\quad int\ a = 5;$
 $\quad solve(a);$
 $\quad cout << a;$
 $\}$

1000
 $\leftarrow a$

$solve(int\ a) \{$
 $\quad a++;$
 $\quad cout << a;$
 $\}$

2000
 $\leftarrow a$

$o/p \rightarrow 5$

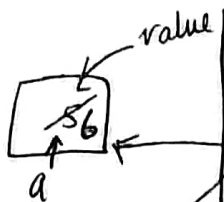
```
main() {
```

```
    int a = 5;
```

```
    solve(a);
```

```
    cout << a;
```

```
}
```



```
solve(int &value) {
```

```
    value++;
```

pass by
Reference

```
}
```

o/p → 6

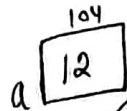
```
int a = 12;
```

```
solve(&a);
```

```
cout << a;
```

o/p → 13

address of a, will be
stored in a pointer.



```
solve(int &val) {
```

```
    &val++;
```



$$\&(104) = \&(104) + 1 \\ = 12 + 1 = 13$$

Pass by value & Reference in pointer →

```
int main() {
```

```
    int a = 5;
```

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

```
    cout << "Before" << p;
```

```
    solve(p);
```

```
    cout << "After" << p;
```

```
}
```

In this func nothing
will be changed

```
solve(int *p) {
```

```
    p = p + 1; ← Pass by value.
```

```
}
```

```
solve(int &p) {
```

```
    p = p + 1; ← Pass by  
reference.
```

Because this is pass by
reference so p (pointer) will be
incremented by 1.

~~Pass by value & Reference~~

① Pass by reference →

```
main() {
```

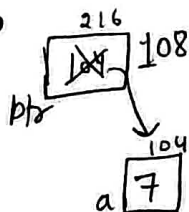
```
    int a = 7;
```

```
    int *ptr = &a;
```

```
    print ptr → 104
```

```
    solve(ptr);
```

```
    print ptr → 108
```



```
solve(int &p) {
```

```
    p = p + 1;
```

```
}
```

H.W. }
j. Explore return by reference.

```
void solve(int *&ptr){
    *ptr=*ptr+1;
    cout<<"ptr+1 "<<*ptr<<endl;
}
int main()
{
```

```
    int a=5;
    int *ptr=&a;
    cout<<"before a : "<<a<<endl;
    solve(ptr);
    cout<<"after a : "<<a<<endl;
    return 0;
}
```

```
void solve(int *ptr){
    *ptr=*ptr+1;
    cout<<"ptr+1 "<<*ptr<<endl;
}
int main()
{
```

```
    int a=5;
    int *ptr=&a;
    cout<<"before a : "<<a<<endl;
    solve(ptr);
    cout<<"after a : "<<a<<endl;
    return 0;
}
```

Same Output

```
PS E:\C++Code> g++ ./f77.cpp
PS E:\C++Code> .\a.exe
before a :5
ptr+1 6
after a :6
```

2 → Output ?
very famous int & solve () {
 int a = 5;
 int &ans = &a;
 return ans;
}