Data Structures & Algorithms

Pointers

- Introduction to Pointers

```
int i = 10;
```

→ Acc. to Statement System will create Memory of 4 bytes & name it as (i)

```
i = i + 5
```

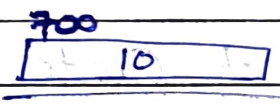
→ Now the question is, where is (i), How System know where the (i) is?

as of now, System Just know, It has alloted 4 bytes to the memory, There is no tag over it to identify that it is (i).

⇒ what System Really do while Executing this Commands??

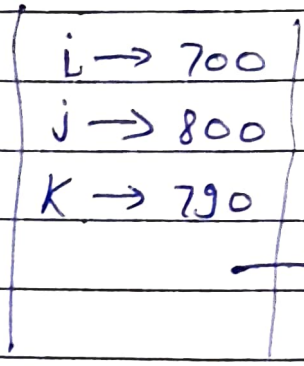
System creates memory of 4 bytes & Provide that 4 bytes a fixed address, so as whenever we call (i), it will go to that address & that address gives access of 4 bytes location.

address Stores two things,
i) where is memory (4 bytes) located
ii) Type [Int, Float, - - -]



overall, 3 Steps involve
i) memory allocation (4 bytes)
ii) addressing of memory (700)
iii) filling up that memory (10)

→ Table in which all the memory got stored is known as Symbol table.



while Compiling such table get Created & make it Easy to access Variables.

⇝ Now, How to find that address where System Stores (i)?

we need to use &i [i.e, address of i]

CODE Ex
```
int i = 10;
Cout << &i << endl;
```

→ output : 0x 61ff0c [Address Stored in Hexadecimal]

→ Storing address of variable into other variable, that is what Pointer is

i.e Storing (&i) in other variable, that Variable is a Pointer to(i)

→ Now, How to Create Pointers?

Lets create Pointer for (&i), (i) is integer, So Pointer will be for Integer.

CODE
```
Int * P = &i;
```
P is an Pointer to an Integer

→ Since, Pointers Store Addresses, So we Put address of (i)

Ex Create float & double Pointers & Print out addresses

Sol"

```
float f = 14;
float * pf = &f;

double d = 122.321;
double * Pd = &d;

Cout << Pf << endl;
Cout << Pd << endl;
```

→ what if we know the address ; but we want the value Stored ?

⇝ using (dereference operator)

CODE

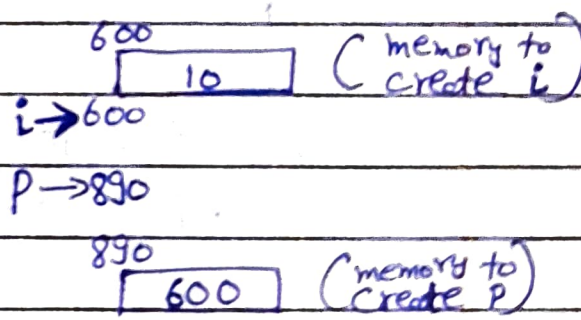
```
int i    ;
int * P =&i ;
Cout << *P << endl;
Cout << i << endl;
```

} Same output

→ In Simple words, {* → Value of} & {& → address of}

when System Executes :

```
int i = 10;
int *p = &i;
```



Now, we know for Integer, Size allotted is 4 bytes

```
Cout <<  Size of (P) << endl;
```

out Put : 8 bytes

→ Changing System Memory / Playing with garbage (dangerous)

```
int i;
Cout << i << endl;

i++;
Cout << i << endl;

int *p;
Cout << p << endl;   {garbage address}
(*p)++;
Cout << *p << endl;  {changed garbage memory} ⊗
```

— Pointer Arthematic

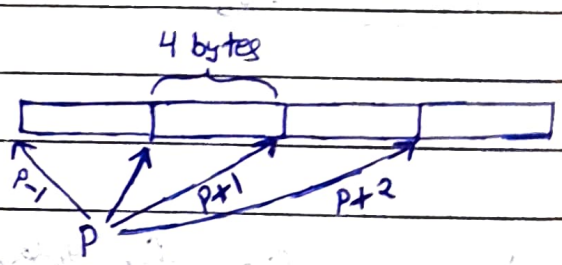
→ Size of Pointer is generally same [8 bytes].

```
int *P;
Char *P;
double *P;
```
} Same Size (8 bytes)

CODE →

```
int i = 10;
int *P = &i;

Cout << P << endl;
P++;
Cout << P << endl;
```



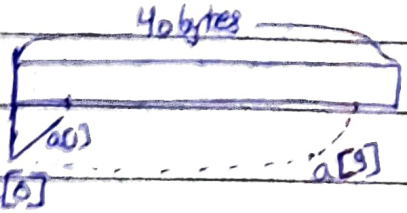
i.e, adding (1) in address will Result in adding (4 bytes) in Hexadecimal.

output: 0x 61ff04
0x 61ff08

{Sensible only in Case of Array}

— Arrays & Pointers

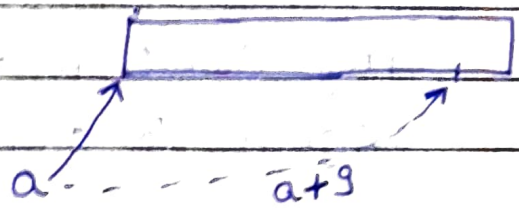
CODE
```
int a[10];
Cout << a << endl;
Cout << &a[0] << endl;
```
} Same output : 0x61fedc



i.e a works as a Pointer

Kind of ↓

CODE
```
a[0] = 5; a[1] = 10;
Cout << *a << endl;
Cout << *(a+1) << endl;
```



output: 5
10

Interal Calculations

*(a+1) ⟹ a[1]

*(a+i) ⟹ a[i]

It is that dumb, that →

a[1] = 1[a] } for System there
a[i] = i[a] } is no difference.

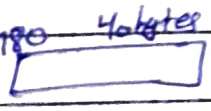
CODE
```
Cout << 1[a] << endl;
Cout << 0[a] << endl;
```

→ Difference In Pointers & Arrays

1) Size Of {for Pointers SizeOf always → 8}

→

```
int a[10];
```



Symbol table

a → 780

but, No 8 bytes alloted for 'a'.
which in Case of Pointers are there.

2) & operator

CODE

```
Cout << P << endl;
Cout << &P << endl;
Cout << a << endl;
Cout << &a << endl;
```

output:
0x 61ff10
0x 61ff08
0x 61feeO
0x 61feeO

→ because 'P' has its own different memory but 'a' don't have

3) Array Cannot be assigned to Pointer, but Pointer can be assigned to a Array

```
int * p = & a[10];
```
→ This will Create 8 bytes for 'P' but, vise-verse not possible as no bytes will Created for 'a'.

i.e, $a \neq P$

$P = a$ ✓

CODE

```
int* R = &a[0];

Cout << a << endl;
Cout << R << endl;
Cout << &R << endl;
Cout << &a << endl;
```

output:
0x 61feeO
0x 61feeO
0x 61fedc
0x 61feeO

i.e, `P = P+1;` ✓

`a = a+1;` (error): array type 'int[10]' is not assignable.

— Characters & Pointers

→ except & Printing addresses, character array & character Pointer Prints out actual Content;

CODE
```
int a[] = {1,2,3}
char b[] = "abC";
Cout << a << endl;
Cout << b << endl;

char* C = & b[0];
Cout << C << endl;

Char C1 = 'a';
Char* Pc = & C1;
Cout << C1 << endl;
Cout << PC << endl; → This Leads to keeP on Printing till System finds 'null' character.
```

— Pointers & Functions

→ we mostly used functions with variables & character, Lets use functions with Pointers.

CODE
```
Void Print (int* p) {
    Cout << *P << endl;
}
int main() {
    int i = 10;
    int *P = & i;
    Print (P);
```

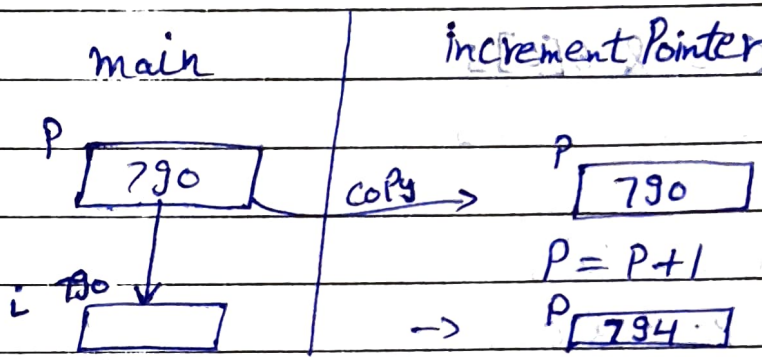
OutPut: 10

CODE

```
Void incrementPointer(int* p) {
    P = P+1;
}
int main() {
    Cout << P << endl;
    incrementPointer(P);
    Cout << P << endl;
```

output: 0x 61ff08
0x 61ff08

→ we got same address as output, why?



i.e, P=P+1, effect the 'P' of incrementPointer only, will not effect main 'P'.

Example Mere Pass ek Paper hai, usPar 790 likha hai, ab maine uski Copy banakar friend ko di, ab friend agar 790 ko 794 bana bhi deta hai, to 790 jo mera hai usPar koi effect nahi aane wala

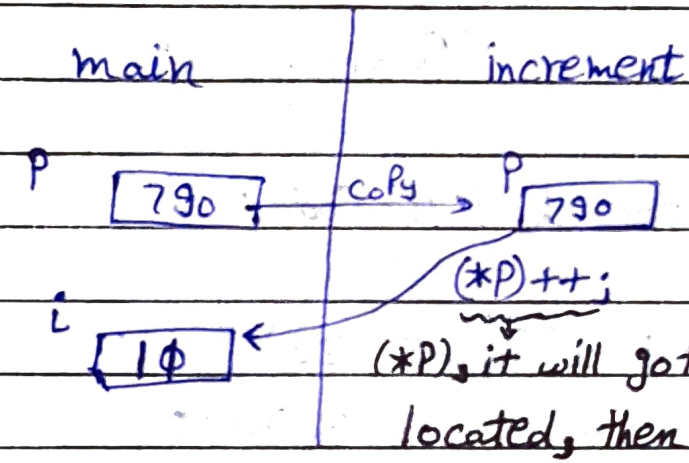
CODE

```
Void increment (int* p) {
    (*p)++;
}

Cout << *P << endl;
 increment (P);
Cout << *P << endl;
```

OutPut : 10
11

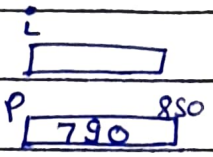
→ This time outPut is not Same, Elaborate?



Example Mere Pass Ek location tak Pauchan ki Pahali hai, now i copy & give Same Pahali to my friend, mera friend usko Pahale Solve kar leta & location Par Jakar changes kar deta, ab Jab mai wapa location Par Jata tab tak '10' change hokar '11' ban chuka tha.

Double Pointer

`int* P = &i;` ⟶



i ⟶ 790

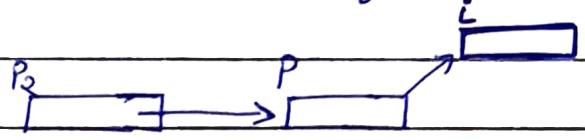
P ⟶ 850

→ Now, Can we store address of P (&P) in some other Pointer?

⇒ Pointers are variables which store address of other variables, i.e, we Can store (&P) in some other variable

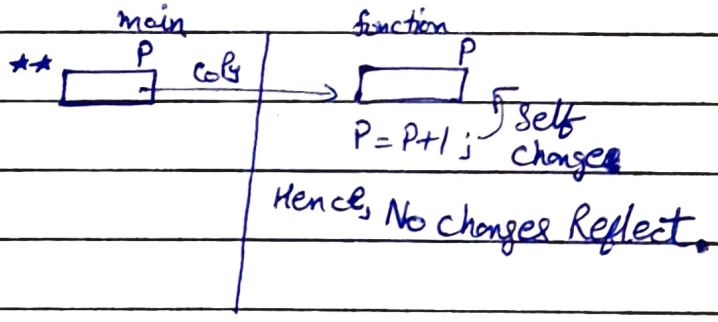
`int** P2 = &P;`

→ double Pointers are Pointers which store address of other Pointers.

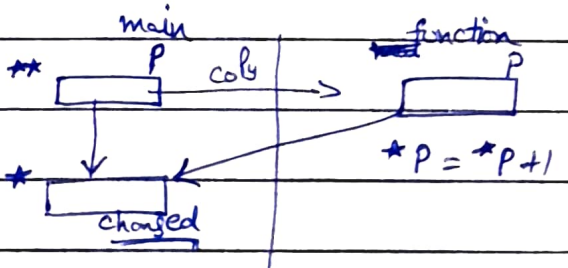


CODE Refer → Pointers_4_DoublePointers.cpp

→ increment 1



→ increment 2



Hence, changes will Reflect.

→ increment 3

Similar to increment 2, changes will Reflect.