Pointers

↓

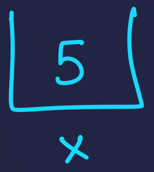
C, C++

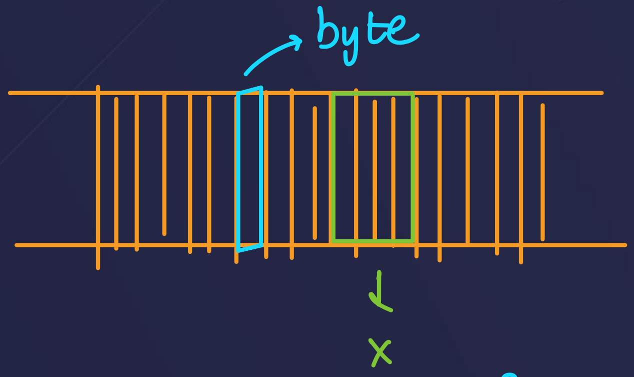
Revisiting variables

What a variable actually is?

int x ;

x = 5;





int → 4 bytes

bool →

char → 1 byte

float →

size of ( )

1byte → 8 bits → 2^8 → 256

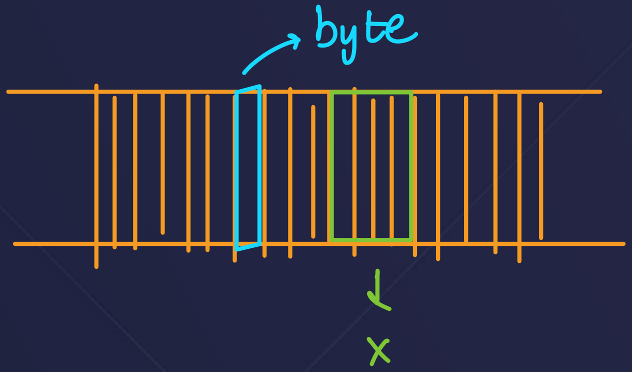
4bytes → 32 bits → 2^{32}

Address of a variable → & operator

Understanding address-of operator

int x ;

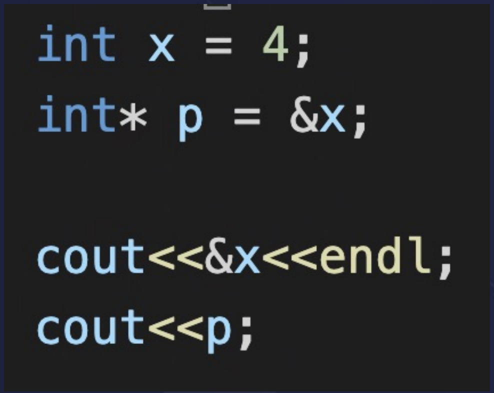
x = 5;



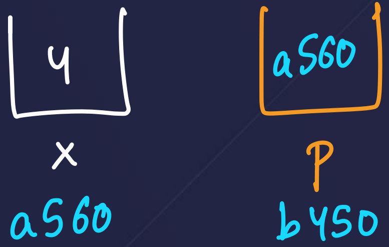
D / Raghav / Movies / Spider Man

A way to store addresses

Introduction to Pointers



```
int x = 4;
int* p = &x;

cout<<&x<<endl;
cout<<p;
```



. a560

. a560

Pointers

```
data_type * pointer_name;
```

Why such syntax?

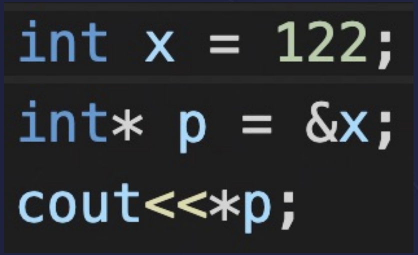
Why not a simple syntax such as:

```
pointer pointer_name;
```

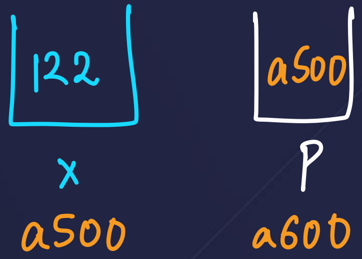
Playing with pointers

Dereference operator

* Star Operator



```
int x = 122;
int* p = &x;
cout<<*p;
```

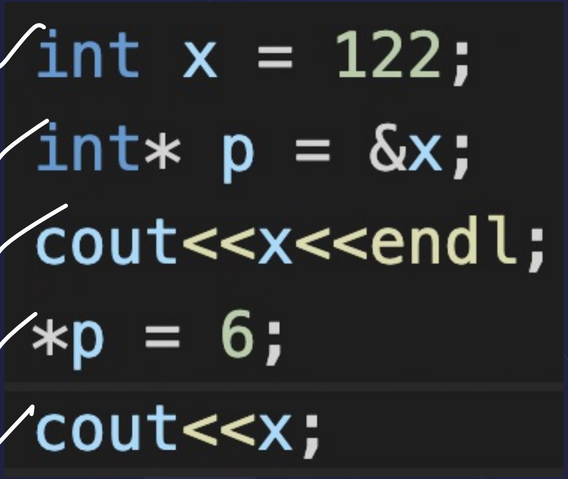


Accessing

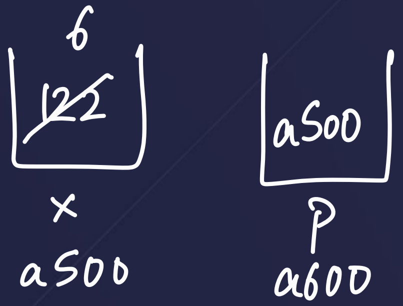
p ke andar jo address rakha hai,
us address pe jao, and vahan ki
value print kar do.

Playing with pointers

Dereference operator



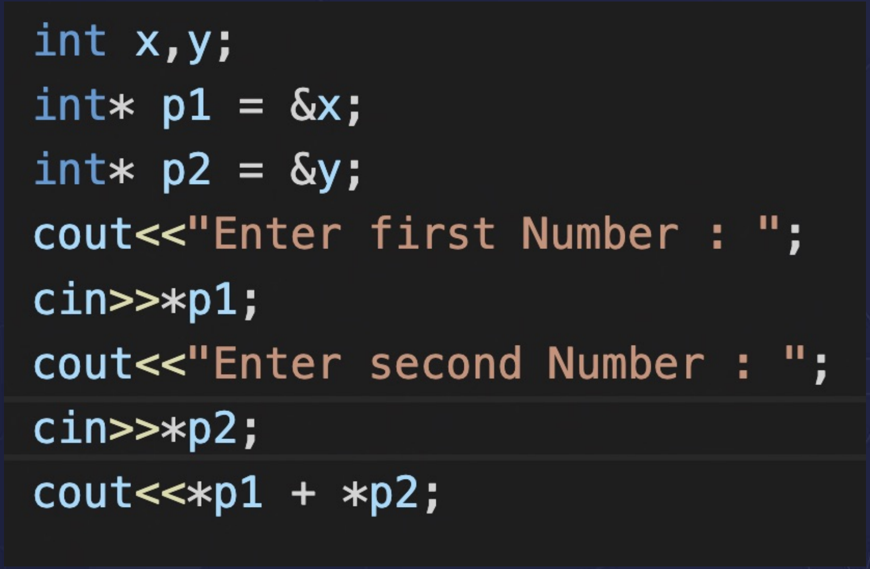
```
✓int x = 122;
✓int* p = &x;
✓cout<<x<<endl;
✓*p = 6;
✓cout<<x;
```



- 122
- 6

Practice

Write a program to calculate sum of two numbers using pointers.

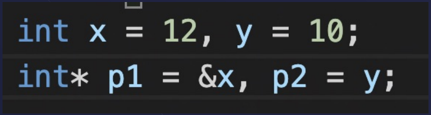


```
int x,y;
int* p1 = &x;
int* p2 = &y;
cout<<"Enter first Number : ";
cin>>*p1;
cout<<"Enter second Number : ";
cin>>*p2;
cout<<*p1 + *p2;
```

Pointers syntax problem :

```
int x;
int y;      →      int x, y;

int x = 5;
int y = 6;  ] →    int x=5, y=6;

int* p1 = &x;
int* p2 = &y;   →   int* p1 = &x, p2 = &y;
                              ↑
                            wrong
```

Pointers syntax problem :



```
int x = 12, y = 10;
int* p1 = &x, p2 = y;
```

int x,y; → x & y are both int

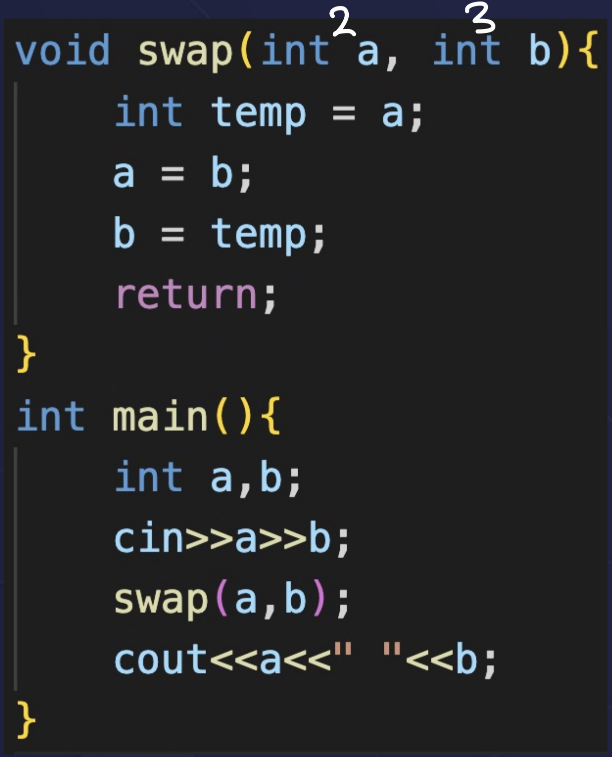
char ch,dh; → ch & dh are both of char

int* p1,p2; → p1 is an int pointer

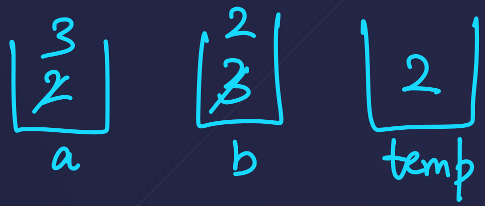
p2 is int

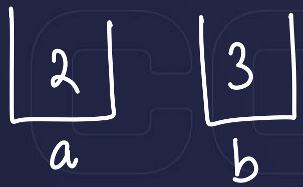
Pass by value and Pass by reference

Writing the correct swap function



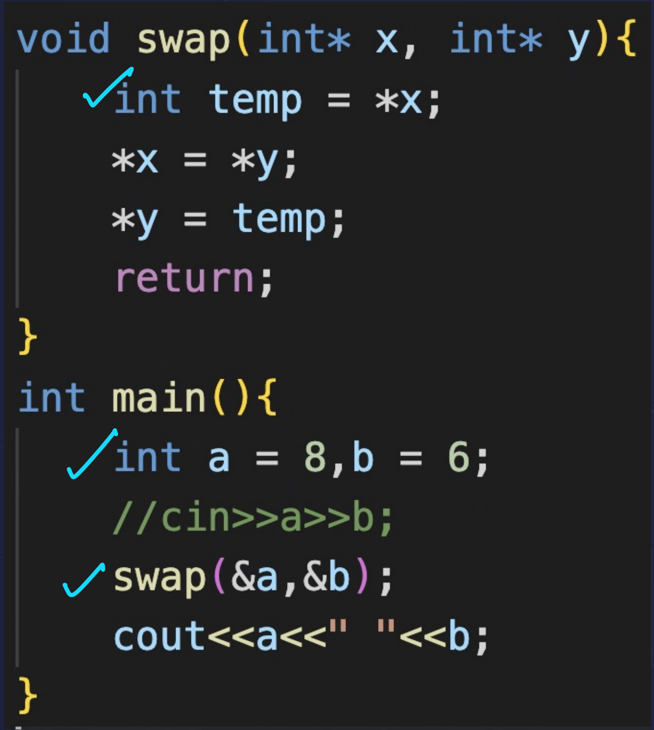
```
void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
    return;
}
int main(){
    int a,b;
    cin>>a>>b;
    swap(a,b);
    cout<<a<<" "<<b;
}
```



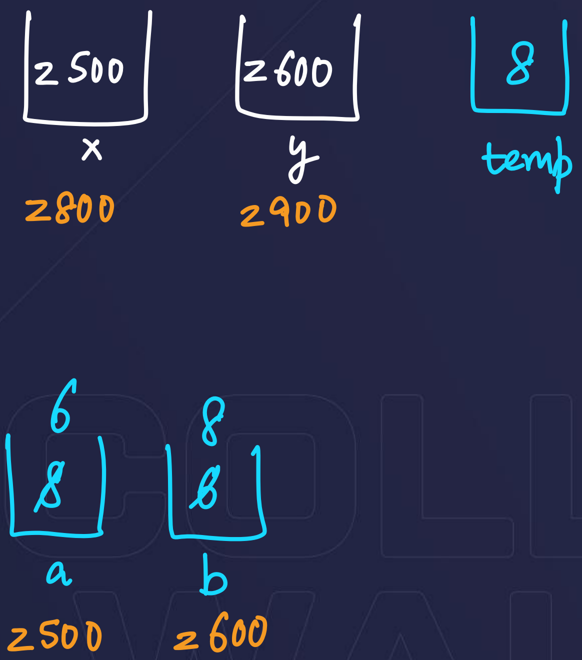


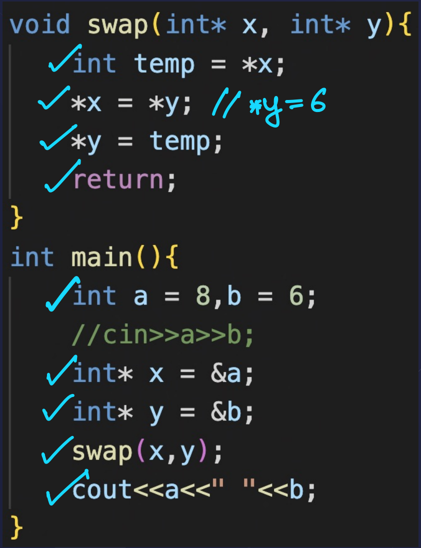
Pass by value and Pass by reference

Writing the correct swap function

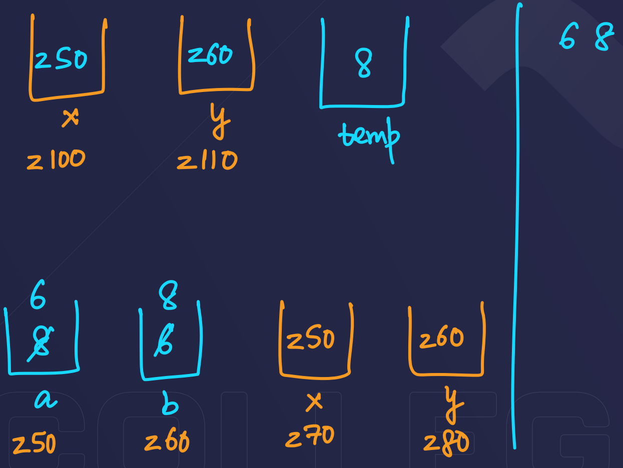


```
void swap(int* x, int* y){
   ✓int temp = *x;
    *x = *y;
    *y = temp;
    return;
}
int main(){
  ✓int a = 8,b = 6;
    //cin>>a>>b;
  ✓swap(&a,&b);
    cout<<a<<" "<<b;
}
```





```
void swap(int* x, int* y){
  ✓int temp = *x;
  ✓*x = *y; // *y=6
  ✓*y = temp;
  ✓return;
}
int main(){
  ✓int a = 8,b = 6;
    //cin>>a>>b;
  ✓int* x = &a;
  ✓int* y = &b;
  ✓swap(x,y);
  ✓cout<<a<<" "<<b;
}
```



Pass by reference (using alias)

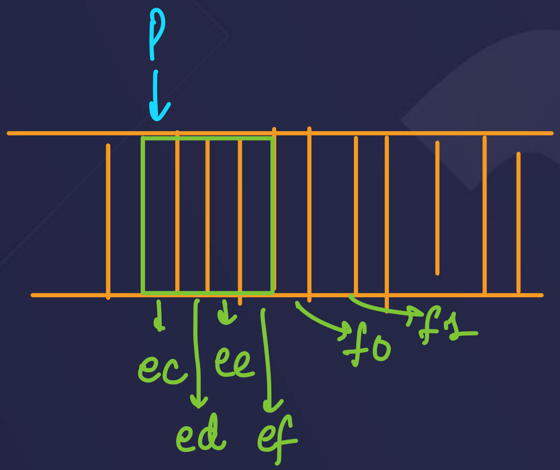
Alias names using & operator

Pointer Arithmetic

Increment and Decrement

```
int x = 5;
int* p = &x;
p = p+1;
```

ec
f0

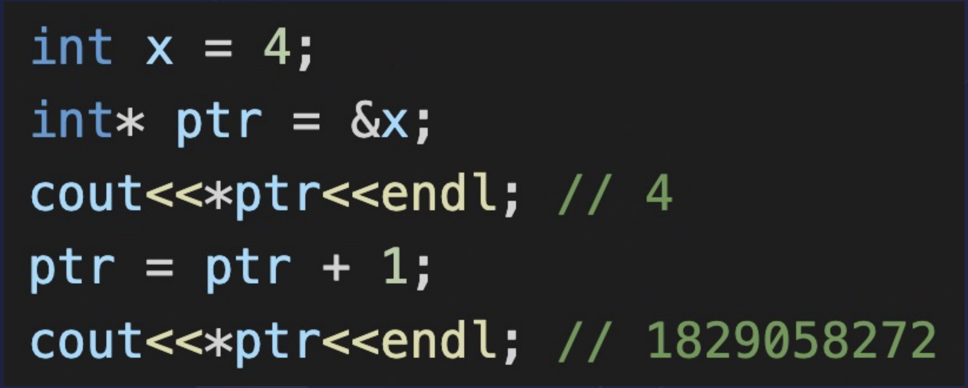


0 1 2 3 4 5 6 7 8 9 a b c d e f

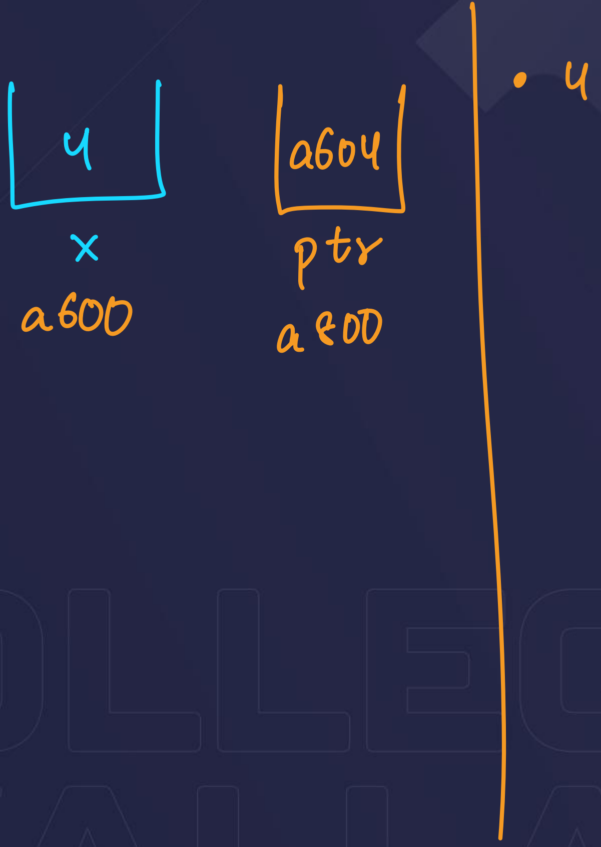
10 11 12 13 14 15

Pointer Arithmetic

Increment and Decrement



```
int x = 4;
int* ptr = &x;
cout<<*ptr<<endl; // 4
ptr = ptr + 1;
cout<<*ptr<<endl; // 1829058272
```



Pointer Arithmetic

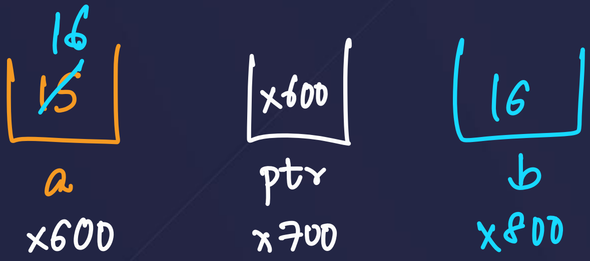
The dependence of addition and subtraction to pointers on the data type

int → 4 bytes se aage

bool/char → 1 byte se aage

Predict the output

```
int a = 15;
int *ptr = &a;
int b = ++*ptr;
cout << a << ' ' << b;
```



16 16

Assume the address of a is 1000.

Practice

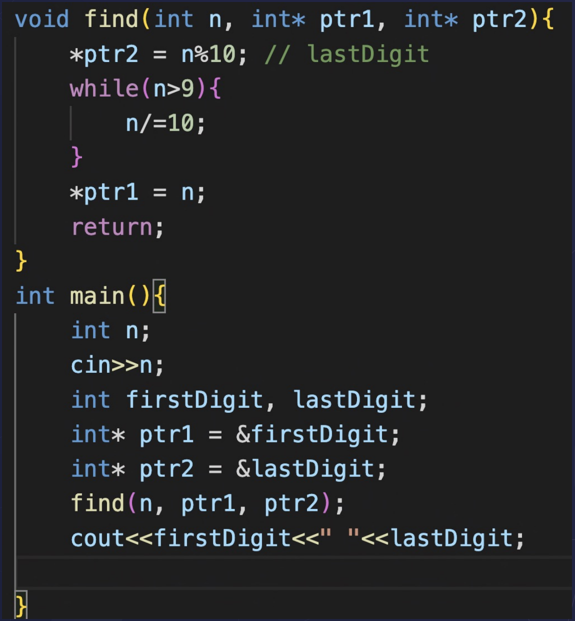
Write a function to find out the first and last digit of a number without returning anything.

```
int  n;
cin>>n;
```

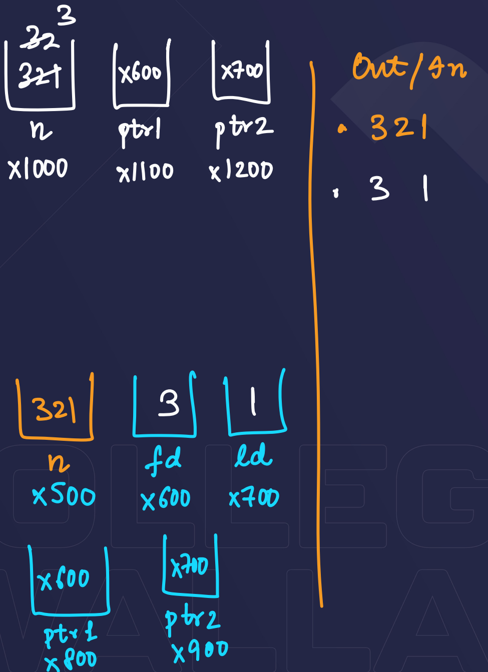
n = ~~12345~~ ~~1234~~ ~~123~~ ~~12~~ 1

ld = n%10 → 5

```
while(n>9){
        n = n/10;
}
```



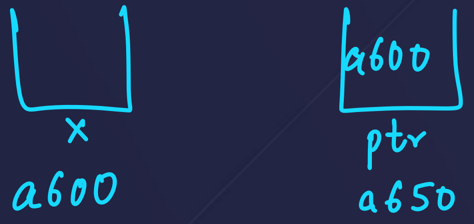
```
void find(int n, int* ptr1, int* ptr2){
    *ptr2 = n%10; // lastDigit
    while(n>9){
        n/=10;
    }
    *ptr1 = n;
    return;
}
int main(){
    int n;
    cin>>n;
    int firstDigit, lastDigit;
    int* ptr1 = &firstDigit;
    int* ptr2 = &lastDigit;
    find(n, ptr1, ptr2);
    cout<<firstDigit<<" "<<lastDigit;

}
```



Null Pointer → will be of great importance

Good practices of using pointers

```
int x ;
int* ptr = &x
```



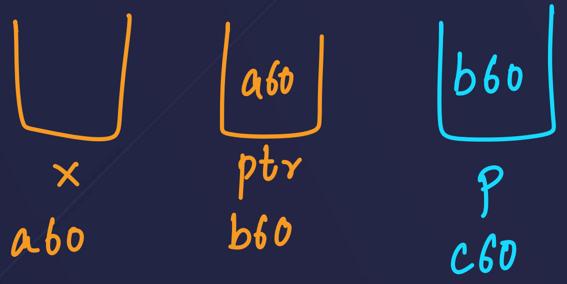
```
int* ptr = NULL;
```

Double pointers

→ used to store address of a single pointer

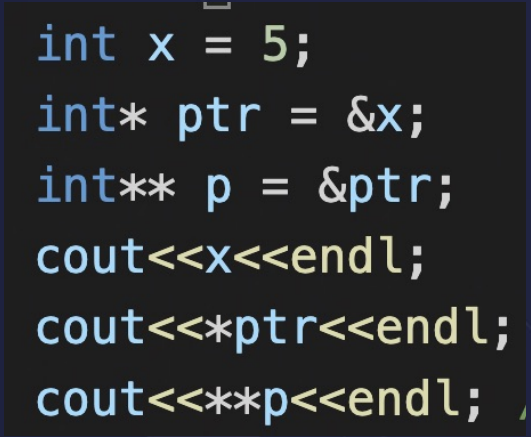
Playing with double pointers

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

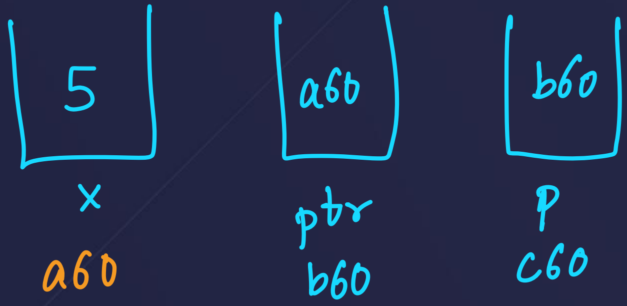


Double pointers

Playing with double pointers



```
int x = 5;
int* ptr = &x;
int** p = &ptr;
cout<<x<<endl;
cout<<*ptr<<endl;
cout<<**p<<endl;
```



·5

·5

·5

Next Lecture

↓

DSA → Arrays