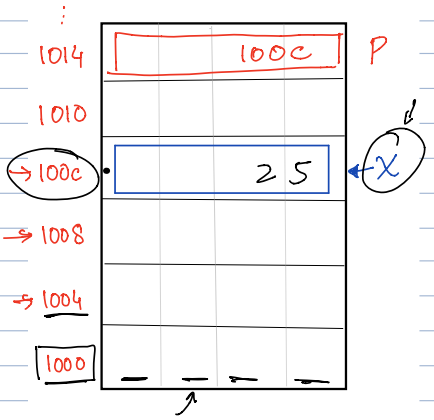
30-01-2019

```
int x;
x = 25;
cout << x ;
```



RAM

"address-of operator" works on an operand.

p = &x ;

&x → 100c ← looks like a number → address

p → data type? → pointer

"pointer". → what is it pointing to?

"integer pointer"

```
pointer p ;
p = &x ;
```

int pointer p ;

int *p ; // p is an integer pointer → hold an address

→ the thing it points to has to be an integer.

"pointer arithmetic"

x ——— x

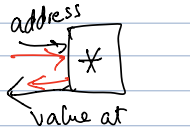
```
int main() {
    int x;      // var allocated at 100c
    x = 25;     // put value
    int *p;     // int pointer
    p = &x;     // 100c   // address of x;
    cout << p ;          // output value of p
    cout << *p ;         // operator "follow the pointer".
}
```

declaration or type

100c

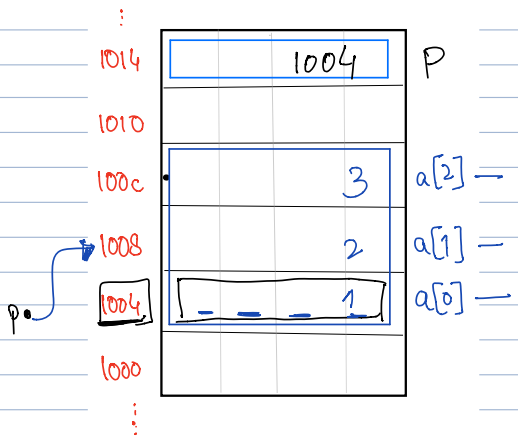
25;

follow



"value at" operator

```
int a[] = {1, 2, 3};

int *p;   // pointer

p = &a[0];   //1004

cout << *p;   // 1
```



"The starting address (address of a[0]) of an array is equal to the address of the array":

$\&a \equiv a$

$a \equiv \&a[0]$ (equal)

a → 1004 ; $\&a[0]$ → 1004

```
cout << a;   // 1004 - by definition

p = a;   // a → &a[0]

cout << *p;   // 1

p = p + 1;   // p: 1004, 1: 1005   X ✓

p = p + 4;   X   // machine dependant

✓ p = p + 1;   // 1 "int" → because p is an int pointer
   // old value: 1004
   // new value: 1008
```

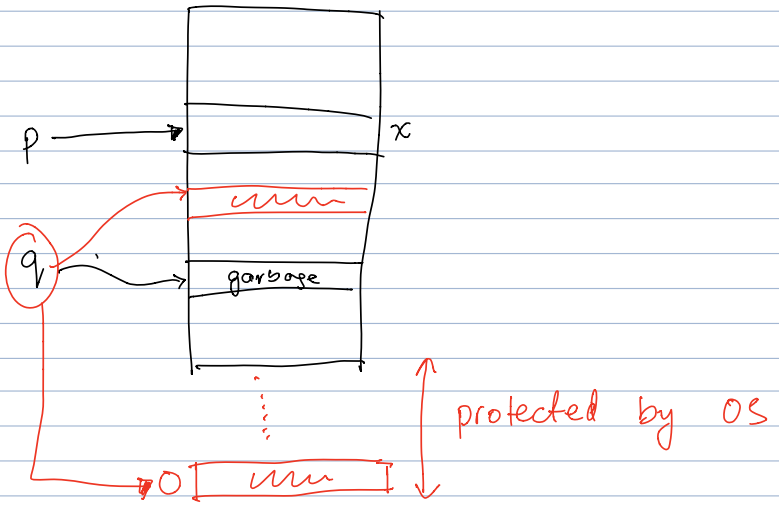
```
p = &a[1];
cout << *p;   //2
```

```
int a[] = {1, 2, 3};
int *p;

p = a;

for (int i=0; i<3; i++) {
    cout << *p;
    p = p+1;   //p++;
}
```

3- times

// *1004 → 1
// *1008 → 2
// *100C → 3

p++ ≡ p = p+1



→ follow the pointer

*p

NULL pointer de-referencing → error / crash