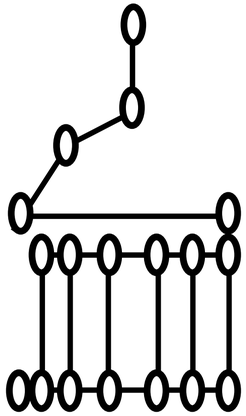
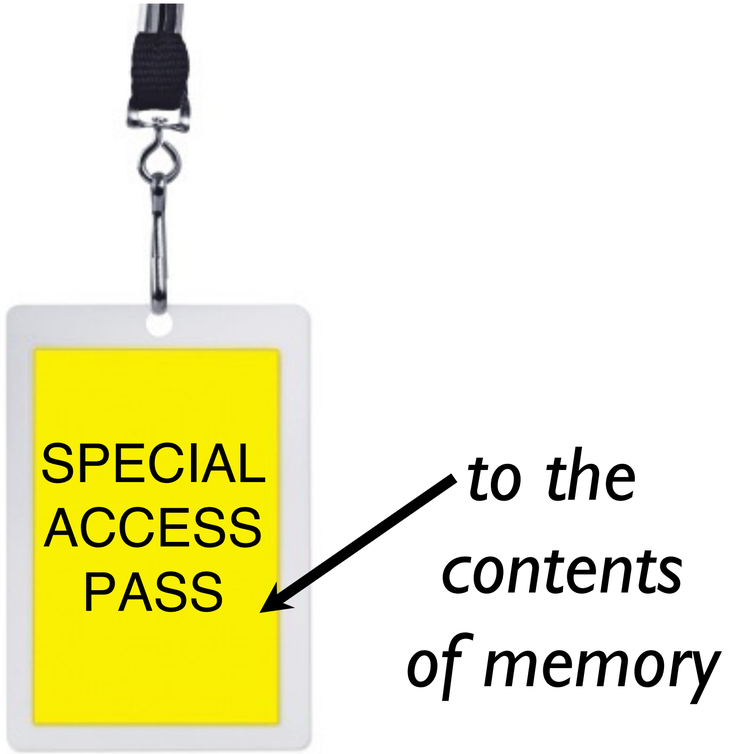
Lecture: Pointer basics



Pointers provide additional power in programming



"With great power comes great responsibility"

Pointers provide additional power in programming

- Simulating pass-by-reference in functions
- Alternate view/use of arrays
- Implementation of more complex data structures (such as linked lists)

to the contents of memory

"With great power comes great responsibility"

Pointer basics

What is a pointer?

A pointer is a *variable* whose value is a *memory address of a particular type*.

Each variable stored in memory has an “address”

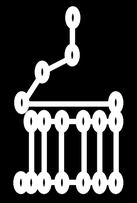
```
#include <iostream>
using namespace std;

int main ()
{
    float myVariable = 5.0;

    cout << "The value of myVariable is “ << myVariable << endl;
    cout << "The address of myVariable is “ << &myVariable << endl;

    return 0;
}
```

Variable	*Memory location (address)*	*Value*
myVariable	0xbffff23c	5.000000

address operator

Example output:

The value of myVariable is 5.000000.
The address of myVariable is 0xbffff23c.



Each variable stored in memory has an “address”

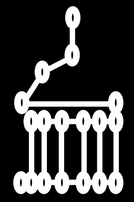
```
#include <iostream>
using namespace std;

int main ()
{
    float myVariable = 5.0;

    cout << "The value of myVariable is “ << myVariable << endl;
    cout << "The address of myVariable is “ << &myVariable << endl;

    return 0;
}
```

Variable	*Memory location (address)*	*Value*
myVariable	0xbffff23c	5.000000

The address reflects the “location” of the variable for a given run of the program.

Example output:

The value of myVariable is 5.000000.
The address of myVariable is 0xbffff23c.

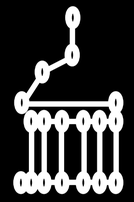


A pointer is a variable whose values are memory addresses

```
#include <iostream>
using namespace std;

void main ()
{
    float myVariable = 5.0;
    float *ptr = nullptr;

  cout << "The value of myVariable is “ << myVariable << endl;
    cout << "The address of myVariable is “ << &myVariable << endl;

    ptr = &myVariable;
    cout << "The value of ptr is “ << ptr << endl;
    cout << "The address of ptr is “ << &ptr << endl;

    return 0;
}
```

(read from right to left)

ptr is a “pointer” to float
Alternatively,
ptr is an “address” to a float

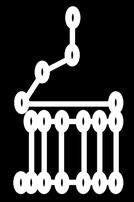


A pointer is a variable whose values are memory addresses

Variable	Memory Location (address)	Value
myVariable	0xbffff23c	5.000000
ptr	0xbffff238	0xbffff23c

```
int main ()
{
    float myVariable = 5.0;
    float *ptr = nullptr;

  cout << "The value of myVariable is “ << myVariable << endl;
    cout << "The address of myVariable is “ << &myVariable << endl;

    ptr = &myVariable;
    cout << "The value of ptr is “ << ptr << endl;
    cout << "The address of ptr is “ << &ptr << endl;

    return 0;
}
```

```
The value of myVariable is 5.000000.
The address of myVariable is 0xbffff23c.
The value of ptr is 0xbffff23c.
The address of ptr is 0xbffff238.
```

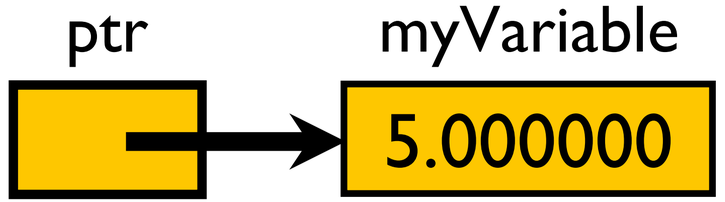


A pointer is a variable whose values are memory addresses

```
int main ()
{
     float myVariable = 5.0;
     float *ptr = nullptr;

    cout << "The value of myVariable is “ << myVariable << endl;
     cout << "The address of myVariable is “ << &myVariable << endl;

     ptr = &myVariable;
     cout << "The value of ptr is “ << ptr << endl;
     cout << "The address of ptr is “ << &ptr << endl;

     return 0;
}
```

Variable	Memory Location (address)	Value
myVariable	0xbffff23c	5.000000
ptr	0xbffff238	0xbffff23c

Often sketched as:



A post office box analogy for memory locations

A post office box analogy for memory locations

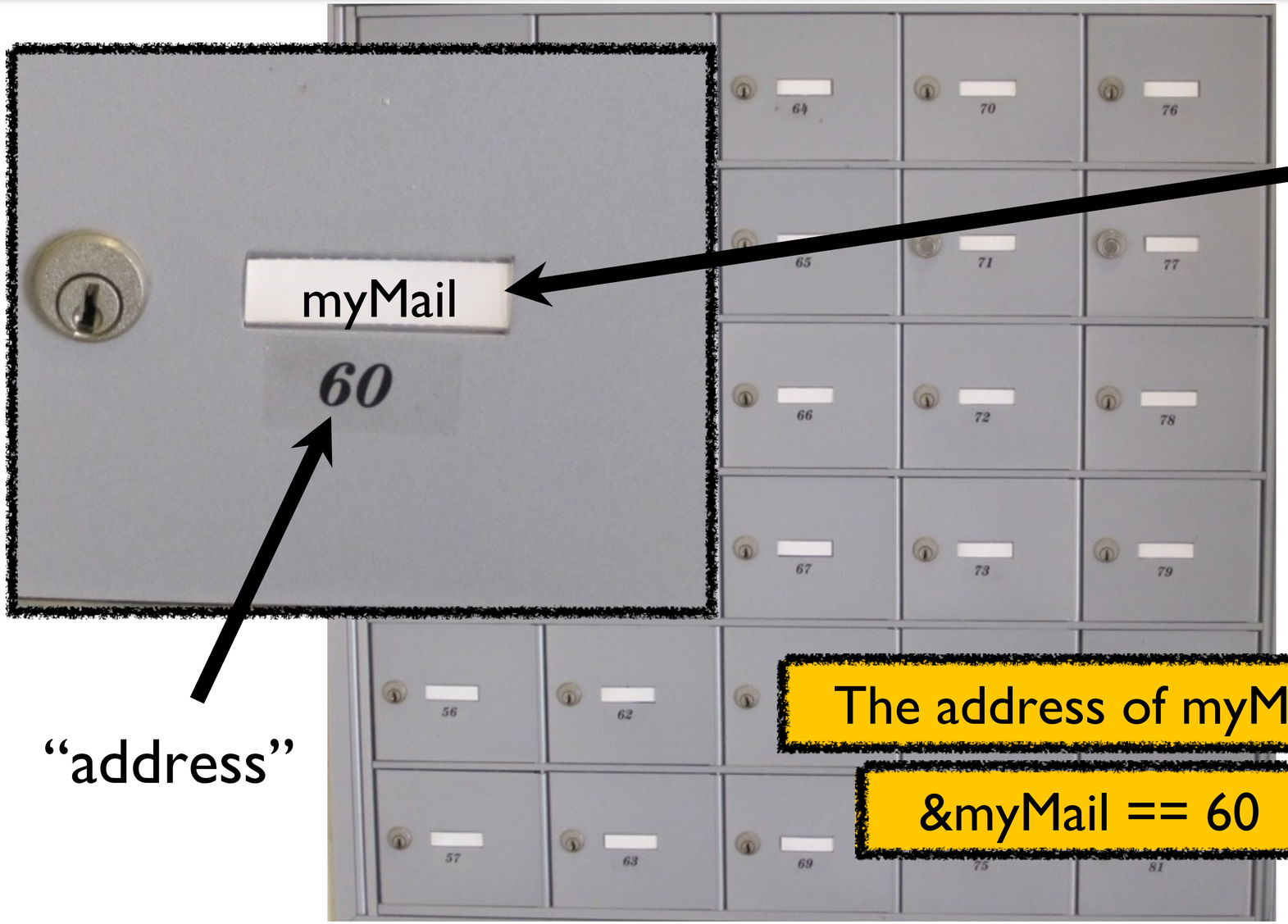
“address”

A post office box analogy for memory locations

A “variable name”
(a convenient label for the mail box)

“address”

A post office box analogy for memory locations



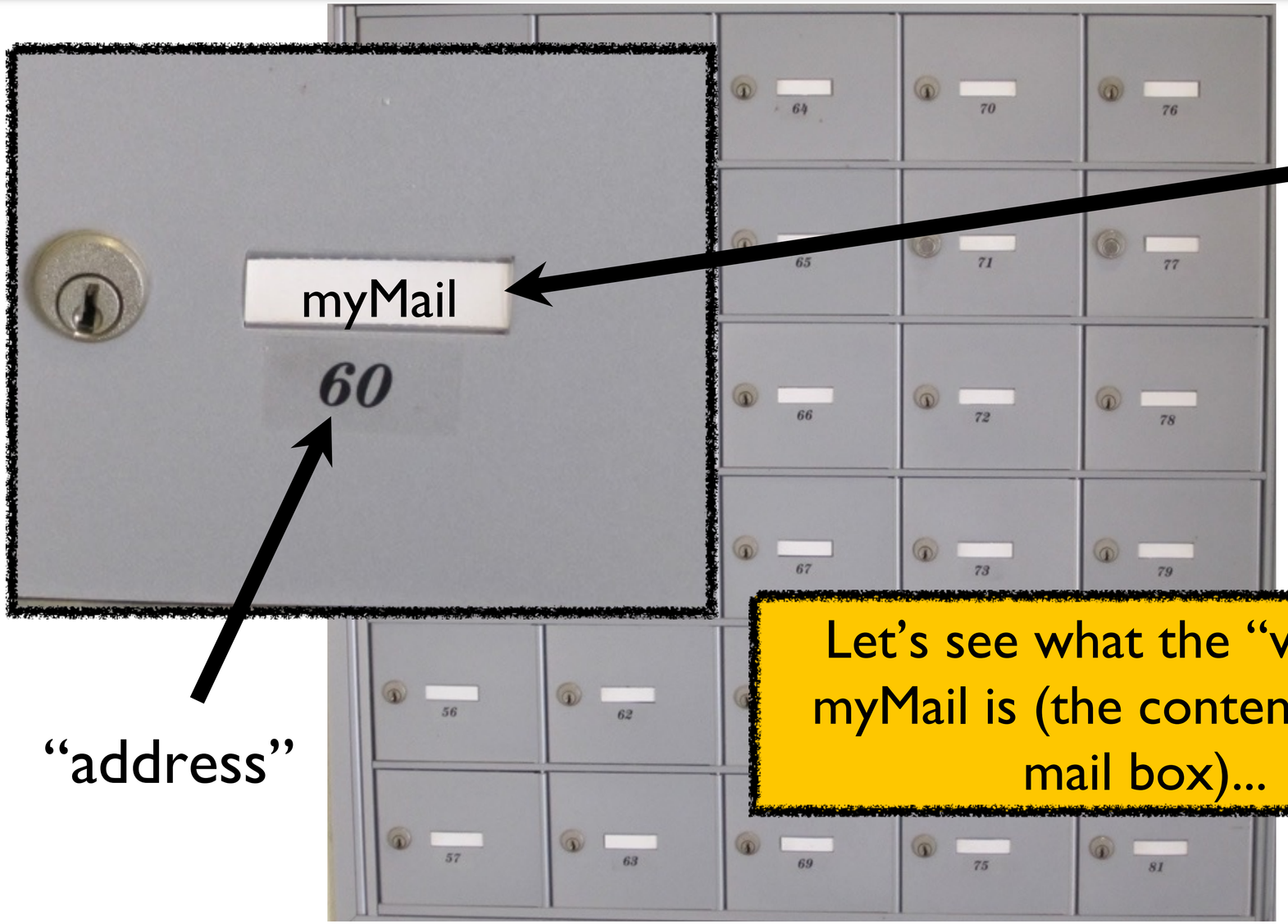
A “variable name” (a convenient label for the mail box)

“address”

The address of myMail is 60

&myMail == 60

A post office box analogy for memory locations



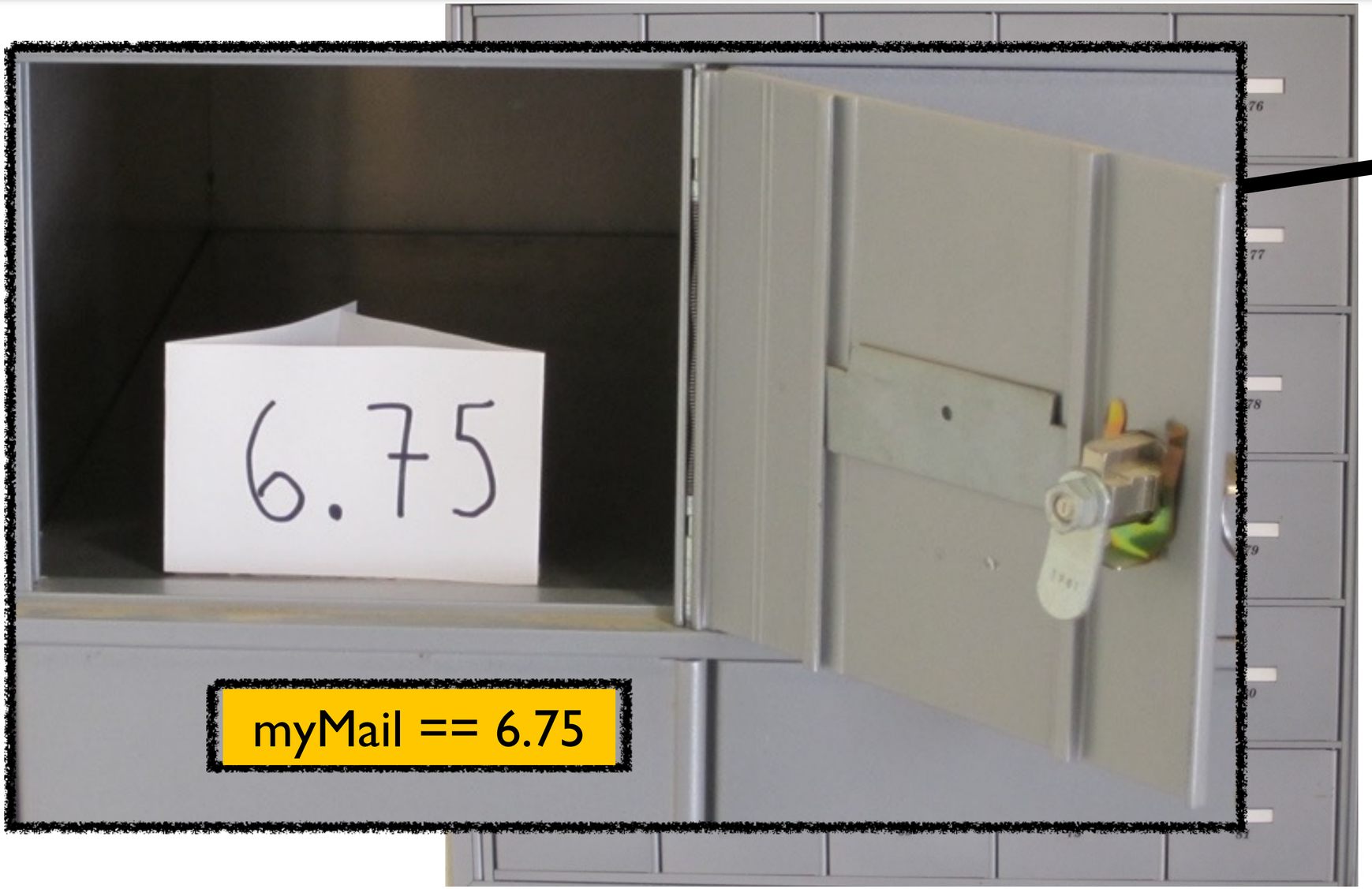
A "variable name"

(a convenient label for the mail box)

"address"

Let's see what the "value" of myMail is (the contents of the mail box)...

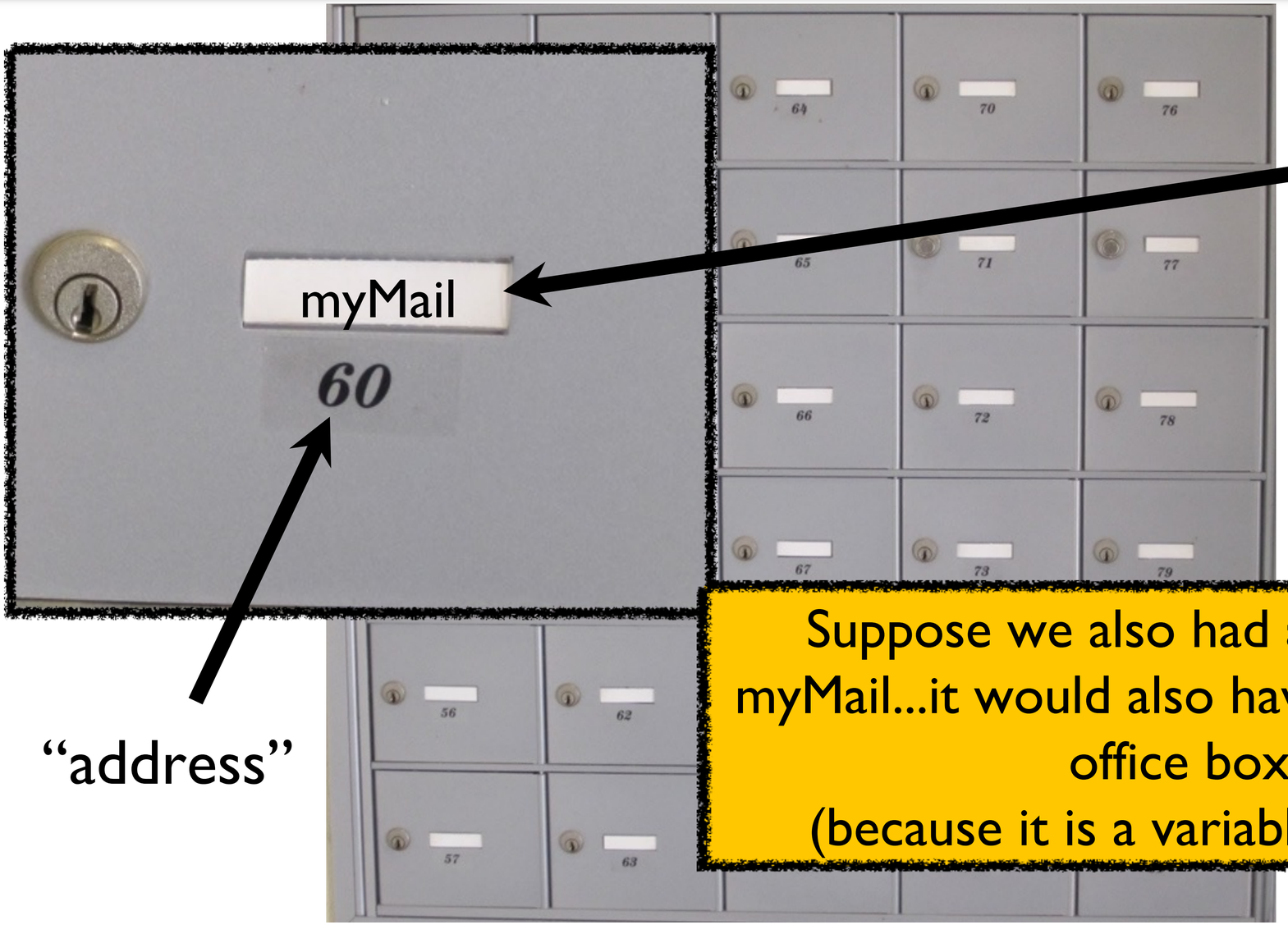
A post office box analogy for memory locations



A “variable name”

(a convenient label for the mail box)

A post office box analogy for memory locations



A "variable name"

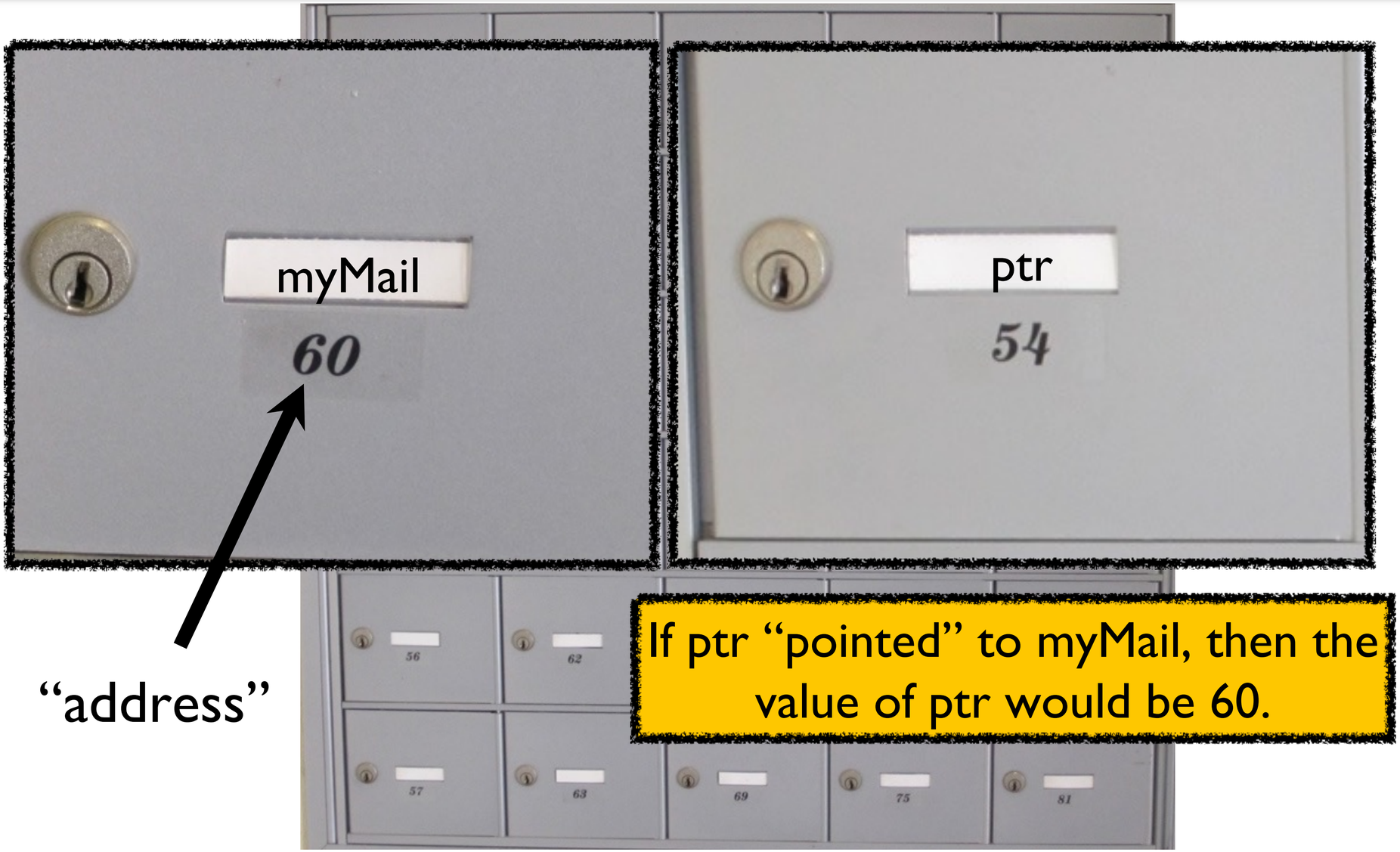
(a convenient label for the mail box)

"address"

Suppose we also had a pointer to myMail...it would also have its own post office box

(because it is a variable as well)!!!

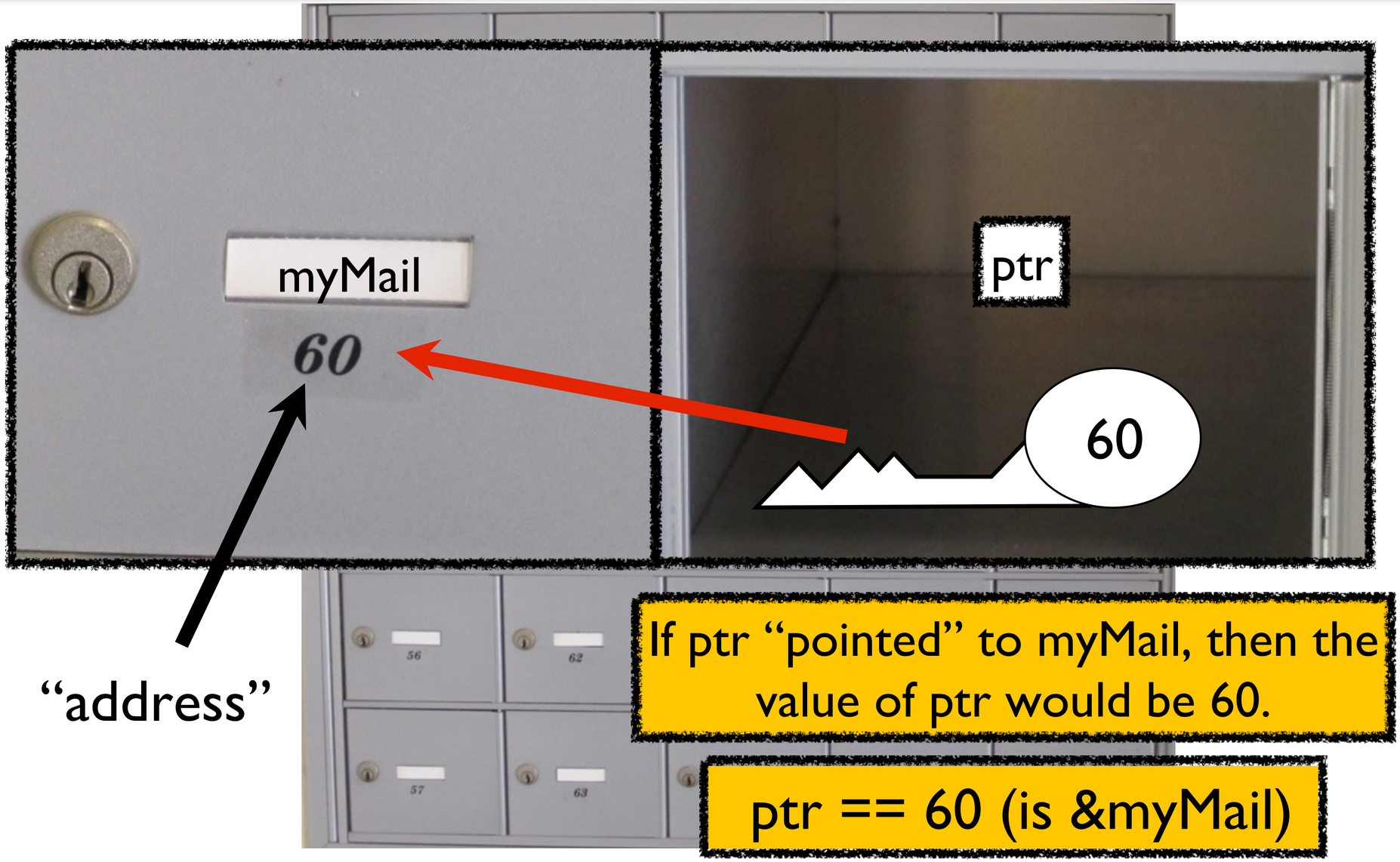
A post office box analogy for memory locations



"address"

If ptr "pointed" to myMail, then the value of ptr would be 60.

A post office box analogy for memory locations



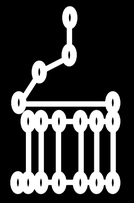
Code for the mail box analogy

```
int main ()
{
    float myMail = 6.75;
    float *ptr = nullptr;

    ptr = &myMail;

    cout << "The value of myMail is " << myMail << ".\n" ;
    cout << "The address of myMail is " << &myMail << ".\n" ;
    cout << "The value of ptr is " << ptr << ".\n";

    return 0;
}
```

variable	*address*
myMail	0xbffff238
ptr	0xbffff234

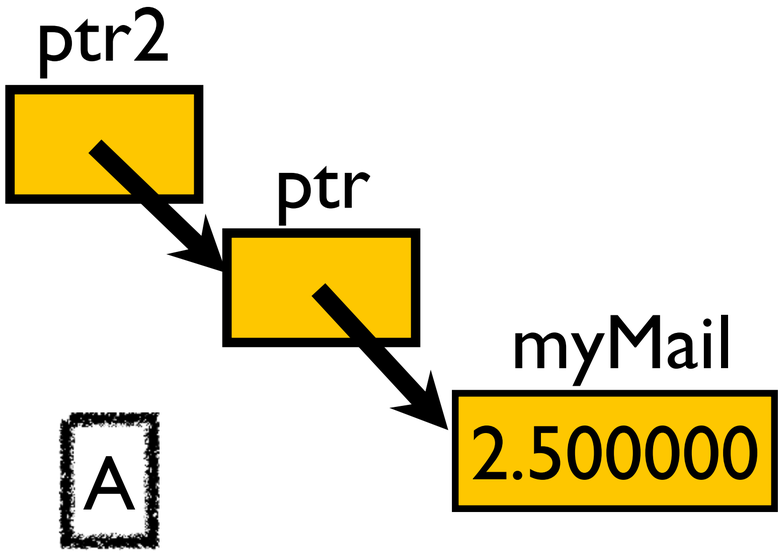
value
6.75
0xbffff238

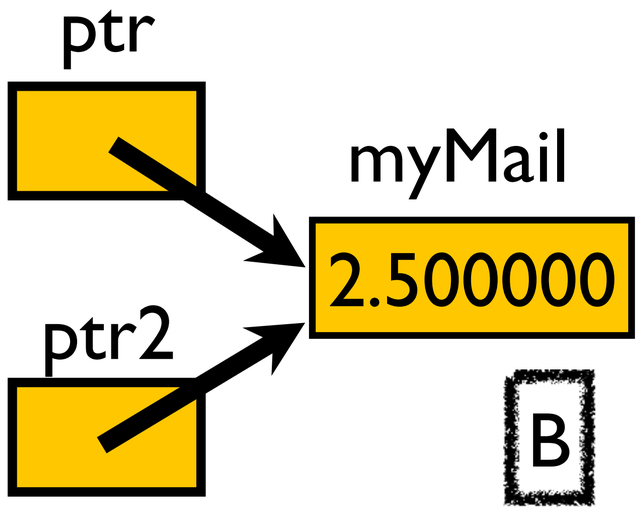
The value of myMail is 6.750000.
The address of myMail is 0xbffff238.
The value of ptr is 0xbffff238.

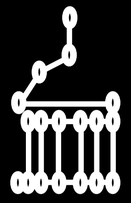


CQ: Which of the following sketches best reflects the given code segment?

```
float myMail = 2.5;
float *ptr;
float **ptr2;

ptr = &myMail;
ptr2 = &ptr;
```





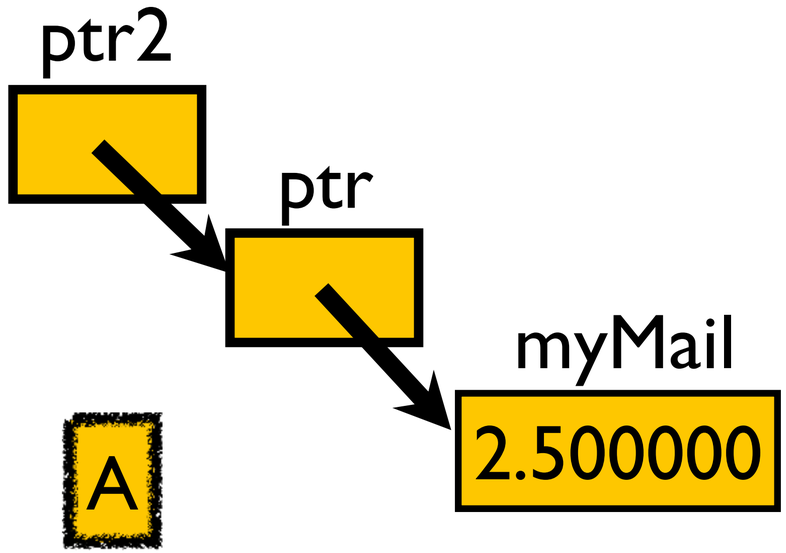


CQ: Which of the following sketches best reflects the given code segment?

```
float myMail = 2.5;
float *ptr;
float **ptr2;
```

"pointer to pointer to float"

```
ptr = &myMail;
ptr2 = &ptr;
```



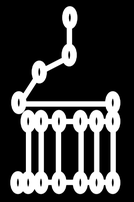
Variable	*Address*	*Value*
myMail	1004	2.500
ptr	1008	1004
ptr2	1016	1008

With pointers, you can modify the contents of memory locations (without knowing the corresponding variable name)

"Dereferencing a pointer"

(use the contents of the memory location for which the pointer "points")

```
int main ()
{
    float myMail = 6.75;
    float *ptr = nullptr;

    ptr = &myMail;

    cout << "The value of myMail is " << myMail << ".\n" ;
    cout << "The address of myMail is " << &myMail << ".\n" ;
    cout << "The value of ptr is " << ptr << ".\n" << endl;

    *ptr = *ptr + 1;
    cout << "The value of myMail is "<< myMail << ".\n" ;
    cout << "The value of *ptr is "<< *ptr << ".\n" ;

    return 0;
}
```

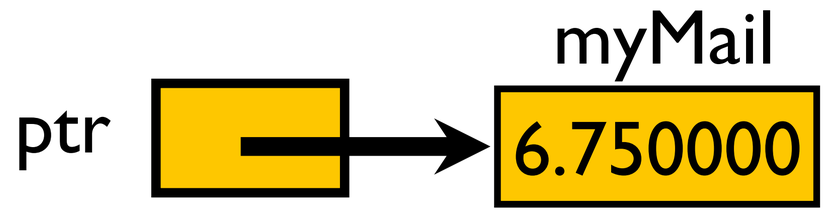


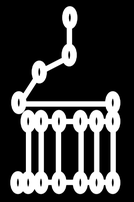
With pointers, you can modify the contents of memory locations (without knowing the corresponding variable name)

```
int main ()
{
    float myMail = 6.75;
    float *ptr = nullptr;

    ptr = &myMail;

    cout << "The value of myMail is " << myMail << ".\n" ;
    cout << "The address of myMail is " << &myMail << ".\n" ;
    cout << "The value of ptr is " << ptr << ".\n" << endl;

    *ptr = *ptr + 1;
    cout << "The value of myMail is "<< myMail << ".\n" ;
    cout << "The value of *ptr is "<< *ptr << ".\n" ;

return 0;
}
```

"Dereferencing a pointer"

(use the contents of the memory location for which the pointer "points")



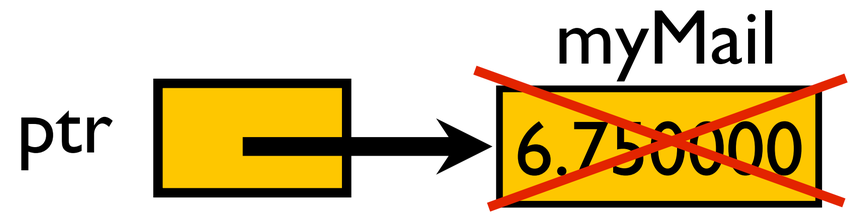


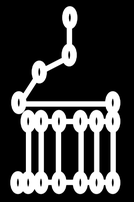
With pointers, you can modify the contents of memory locations (without knowing the corresponding variable name)

```
int main ()
{
    float myMail = 6.75;
    float *ptr = nullptr;

    ptr = &myMail;

    cout << "The value of myMail is " << myMail << ".\n" ;
    cout << "The address of myMail is " << &myMail << ".\n" ;
    cout << "The value of ptr is " << ptr << ".\n" << endl;

    *ptr = *ptr + 1;
    cout << "The value of myMail is "<< myMail << ".\n" ;
    cout << "The value of *ptr is "<< *ptr << ".\n" ;

return 0;
}
```

“Dereferencing a pointer”

(use the contents of the memory location for which the pointer “points”)



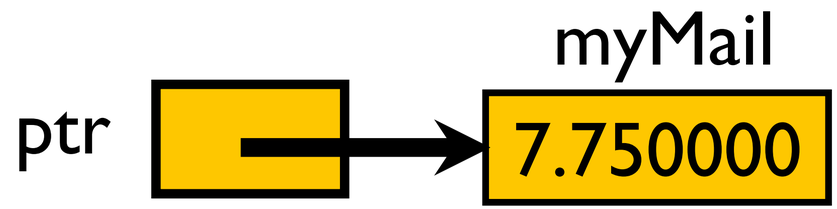


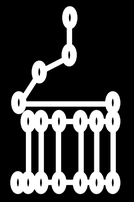
With pointers, you can modify the contents of memory locations (without knowing the corresponding variable name)

```
int main ()
{
    float myMail = 6.75;
    float *ptr = nullptr;

    ptr = &myMail;

    cout << "The value of myMail is " << myMail << ".\n" ;
    cout << "The address of myMail is " << &myMail << ".\n" ;
    cout << "The value of ptr is " << ptr << ".\n" << endl;

    *ptr = *ptr + 1;
    cout << "The value of myMail is "<< myMail << ".\n" ;
    cout << "The value of *ptr is "<< *ptr << ".\n" ;

return 0;
}
```

"Dereferencing a pointer"

(use the contents of the memory location for which the pointer "points")





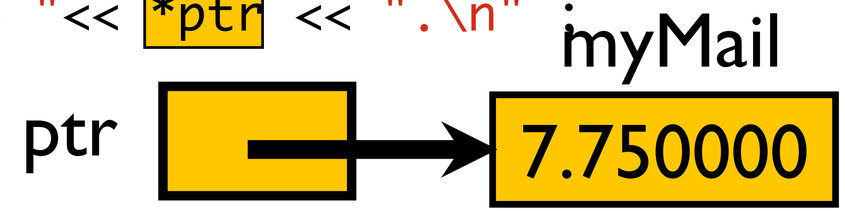
With pointers, you can modify the contents of memory locations (without knowing the corresponding variable name)

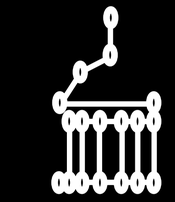
"Dereferencing a pointer"

(use the contents of the
emory location for which
the pointer "points")

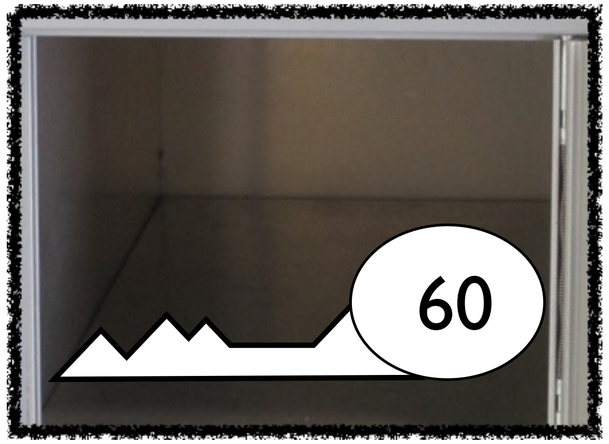
```
The value of myMail is 6.750000.
The address of myMail is 0xbffff238.
The value of ptr is 0xbffff238.
The value of myMail is 7.750000.
The value of *ptr is 7.750000.
```

```
                                    " << myMail << ".\n" ;
    cout << "The address of myMail is " << &myMail << ".\n" ;
    cout << "The value of ptr is " << ptr << ".\n" << endl;

    *ptr = *ptr + 1;
    cout << "The value of myMail is "<< myMail << ".\n" ;
    cout << "The value of *ptr is "<< *ptr << ".\n" ;

    return 0;
}
```

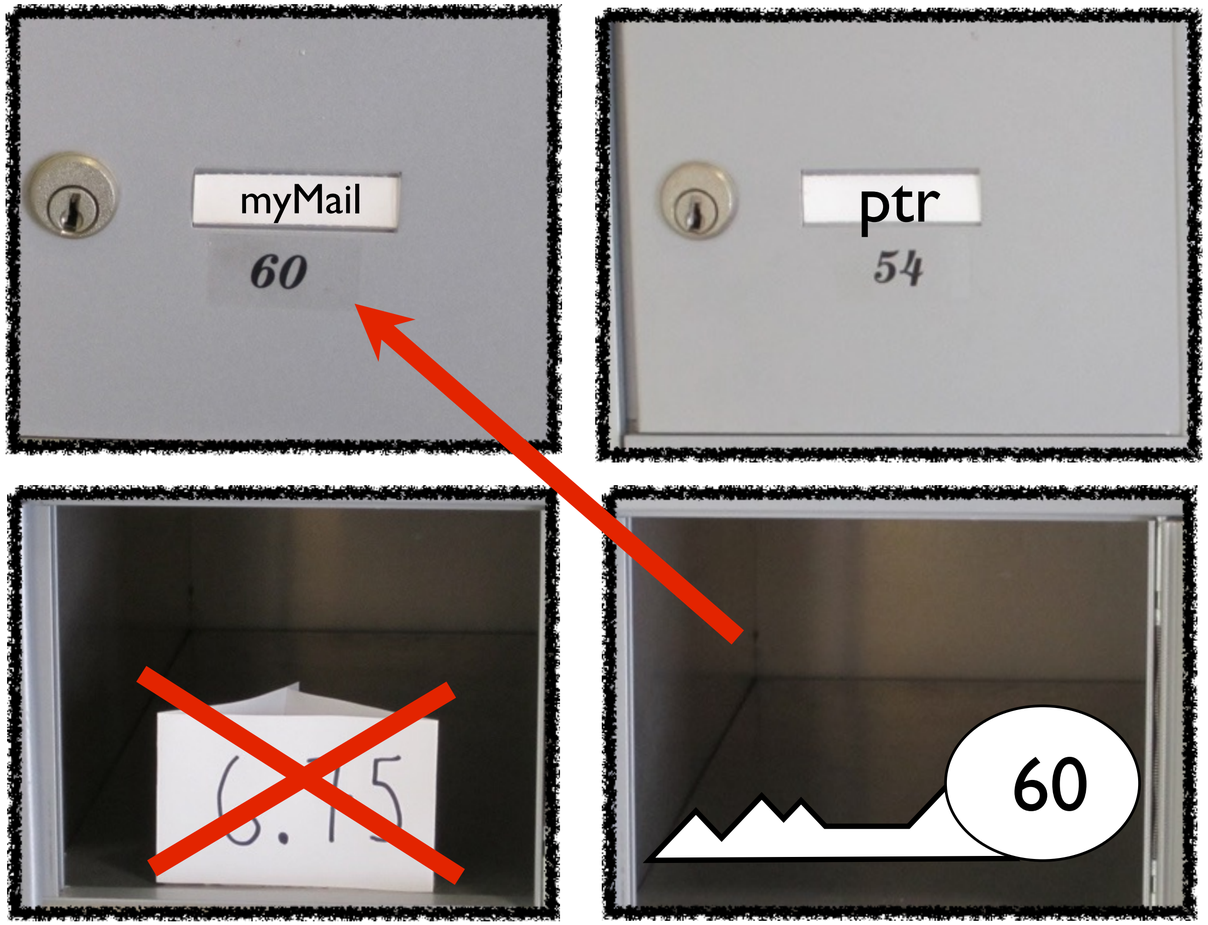




Back to our post office box analogy...



Back to our post office box analogy...

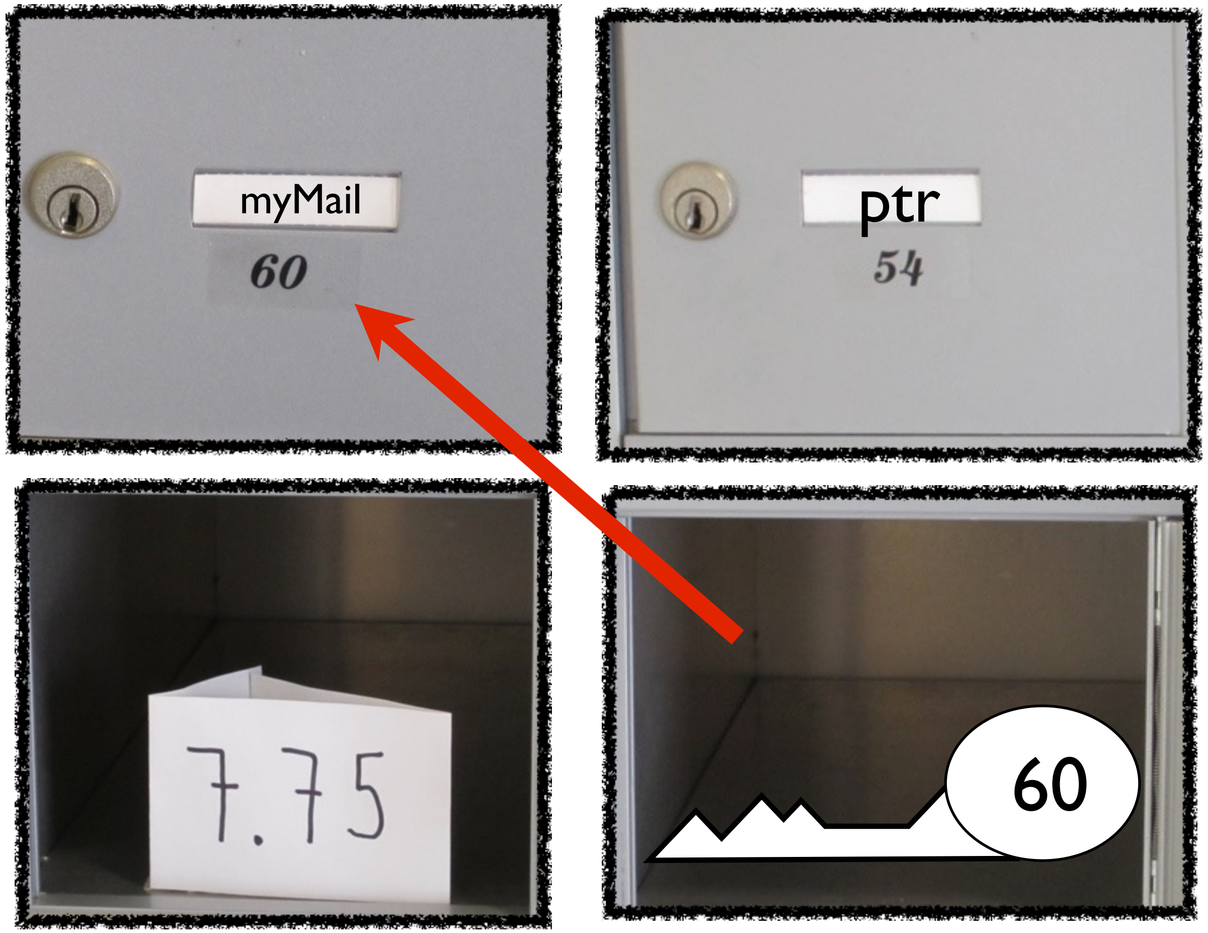


7.75

*ptr = 7.75

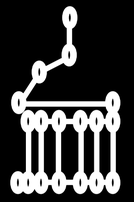
(using the key to open the post office box)

Back to our post office box analogy...

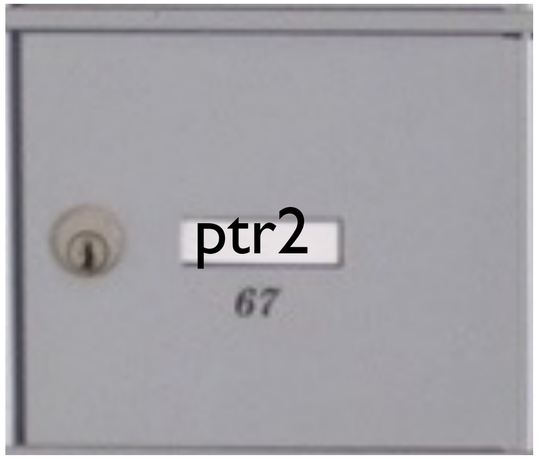


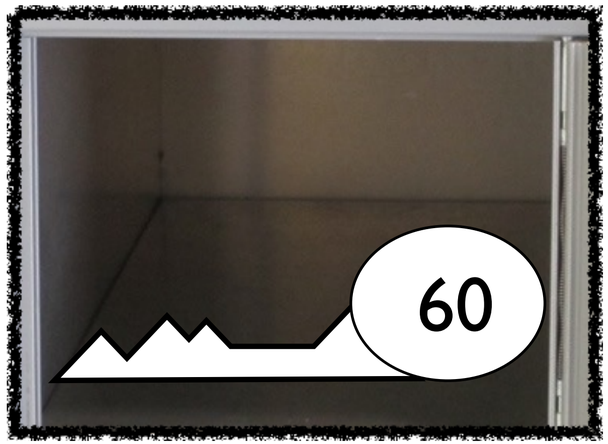
*ptr = 7.75

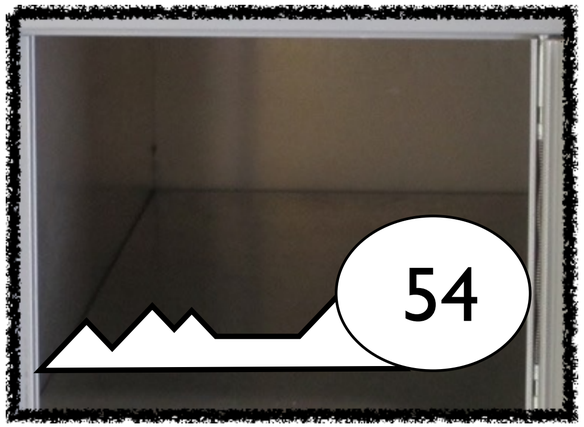
(using the key to open the post office box)



CQ: In our post office box analogy, suppose we have another post office box at location 67 named ptr2 and set its contents to 54. What is the value of **ptr2?







A: 60

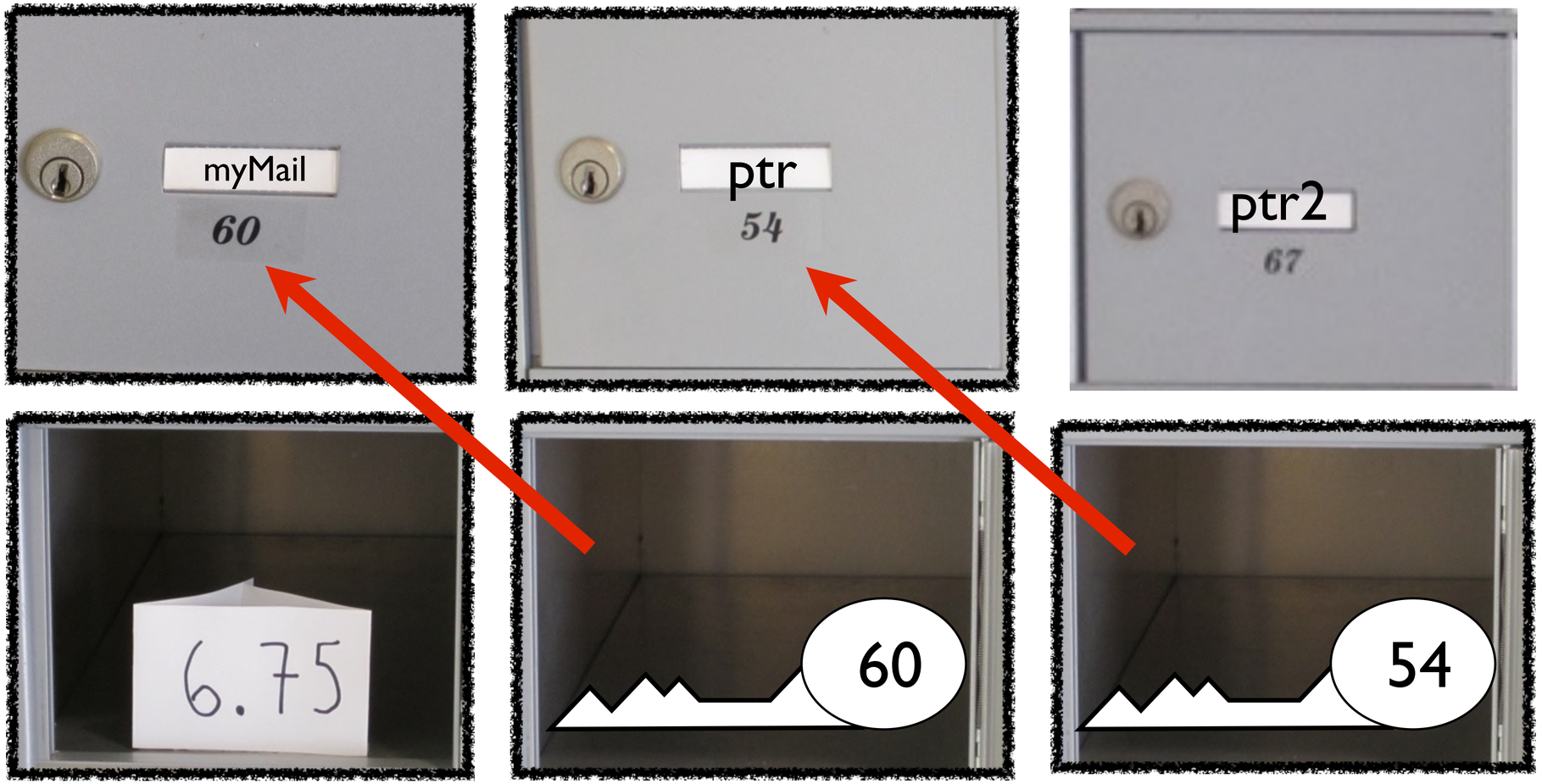
B: 6.75

C: 54

D: 67

E: unknown

CQ: In our post office box analogy, suppose we have another post office box at location 67 named ptr2 and set its contents to 54. What is the value of **ptr2?



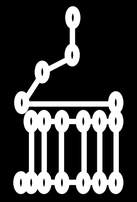
A: 60

B: 6.75

C: 54

D: 67

E: unknown



CQ: What is the final value of x in the following example?

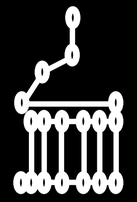
```
int main ()
{
    int x = 5;
    int *ptr;

    x = x + 2;
    ptr = &x;
    x = x + 1;
    *ptr = *ptr + 2;

    cout << "x = " << x << endl;

    return 0;
}
```

A. 5

B. 7

C. 8

D. 10



Take Home: What is the output of the following program?

```
void mysteryFunction(int *val);

int main ()
{
    int x = 5;
    int *ptr = &x;

    mysteryFunction(&x);
    mysteryFunction(ptr);

    cout << x << endl;

    return 0;
}

void mysteryFunction(int *val)
{
    *val = *val + *val;
}
```

A. 5

B. 10

C. 15

D. 20