COMP 1406: Arrays and Structs

Arrays and structs in C++

Memory model

Pass by reference

Array Basics

Java/Processing:

```
// Set up variable name
int[] numbers;

// Create and assign array
numbers = new int[3];

// Assign values
numbers[0] = 1;
numbers[1] = 4;
numbers[2] = -7;

// Access values
int n = numbers[1];

// Array length
int len = numbers.length;
```

Python:

```
// Creating a list
[1, 4, -7]

// Naming a list
numbers = [1, 4, -7]

// Accessing values
n = numbers[1]

// Reassigning values
numbers[0] = 10

// Adding a value
numbers.append(8)
```

Java/Processing:

```
// Set up variable name
int[] numbers;

// Create and assign array
numbers = new int[3];

// Assign values
numbers[0] = 1;
numbers[1] = 4;
numbers[2] = -7;

// Access values
int n = numbers[1];

// Array length
int len = numbers.length;
```

Python:

```
// Creating a list
[1, 4, -7]

// Naming a list
numbers = [1, 4, -7]

// Accessing values
n = numbers[1]

// Reassigning values
numbers[0] = 10

// Adding a value
numbers.append(8)
```

Arrays don't work the same way in C++ – you get much less for free!

Arrays in C++

Declaring a single variable:

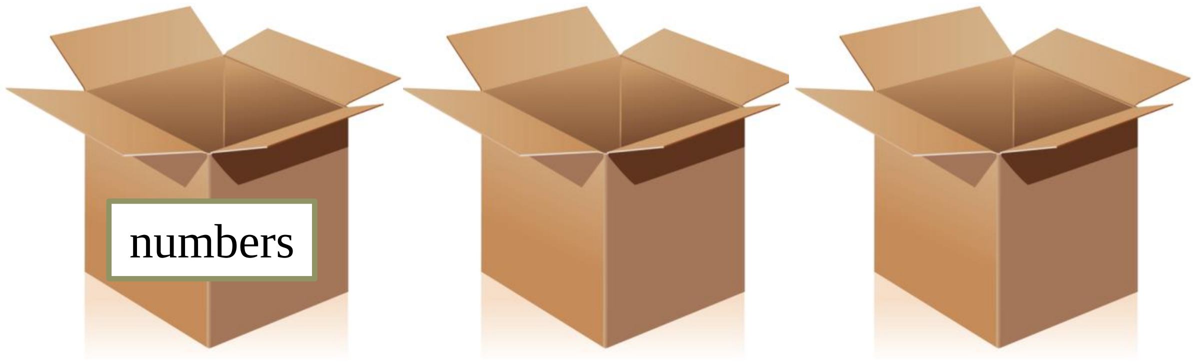
```
int number1;
```

Arrays in C++

Declaring a single variable:

```
int number1;
```

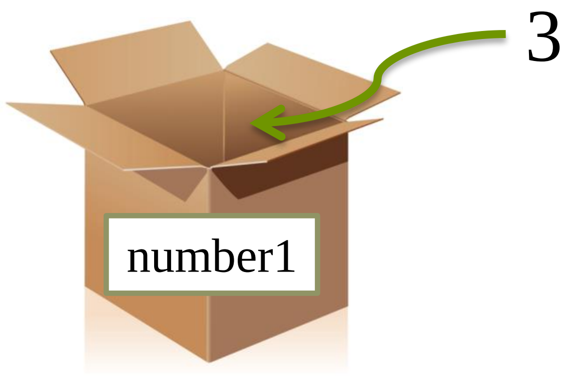
Declaring an array:



```
int numbers[3];
```

Arrays in C++

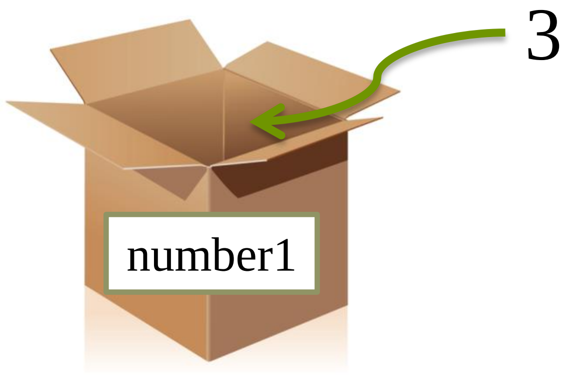
Assigning to a single variable:



```
number1 = 3
```

Arrays in C++

Assigning to a single variable:



```
number1 = 3
```

Assigning to an array:

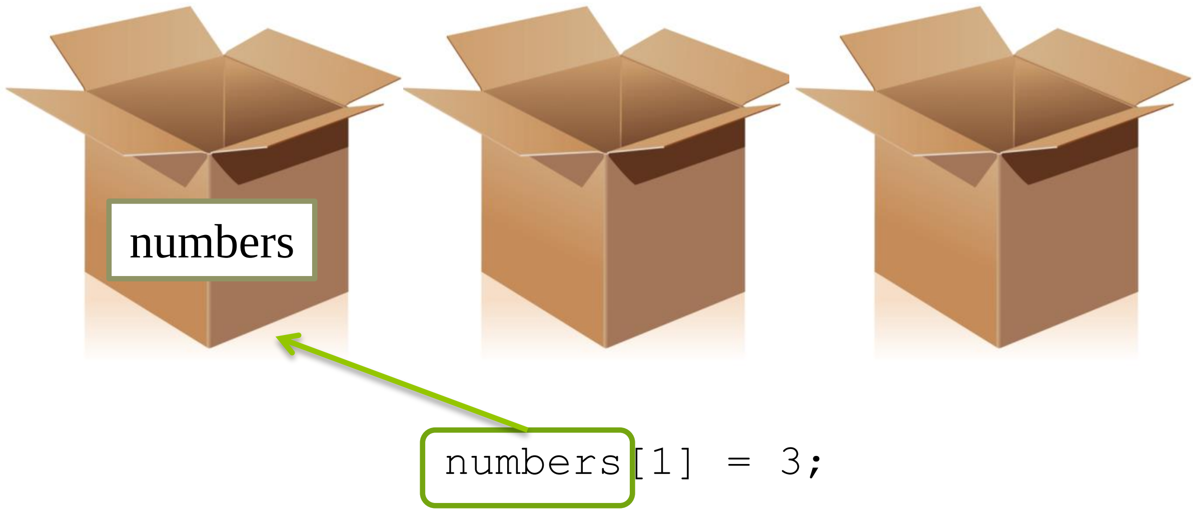
```
numbers[1] = 3;
```

Arrays in C++

Assigning to a single variable:

```
number1 = 3
```

Assigning to an array:



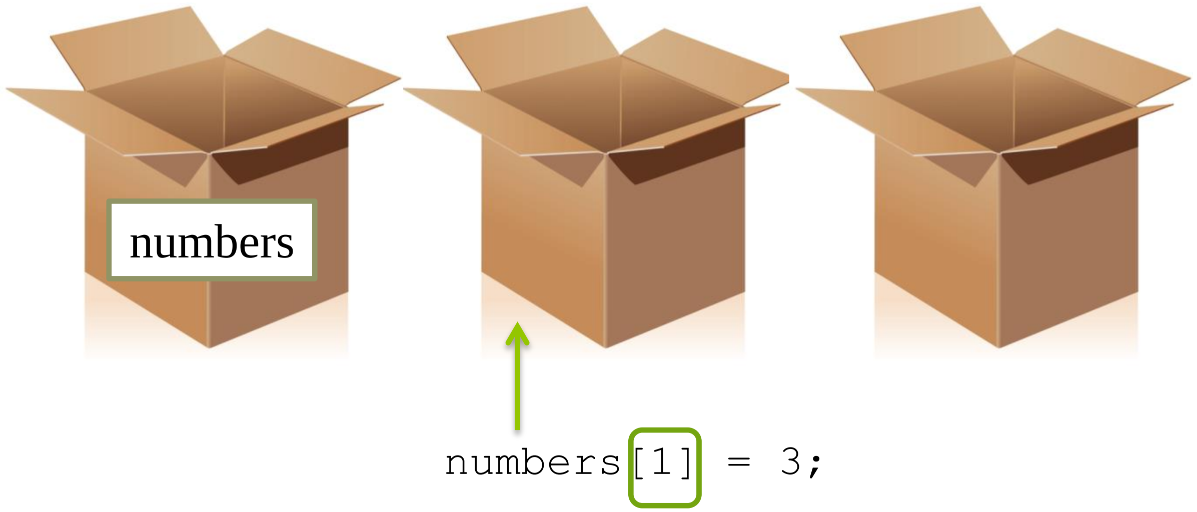
```
numbers[1] = 3;
```

Arrays in C++

Assigning to a single variable:

```
number1 = 3
```

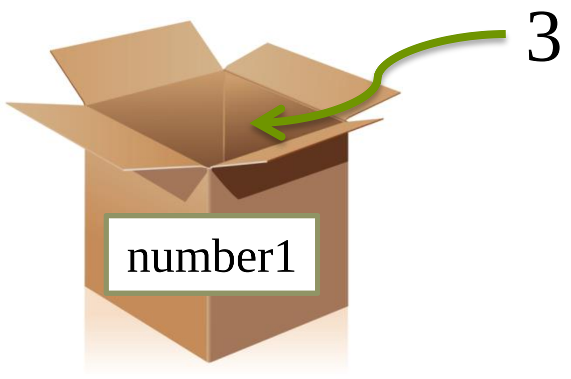
Assigning to an array:



```
numbers[1] = 3;
```

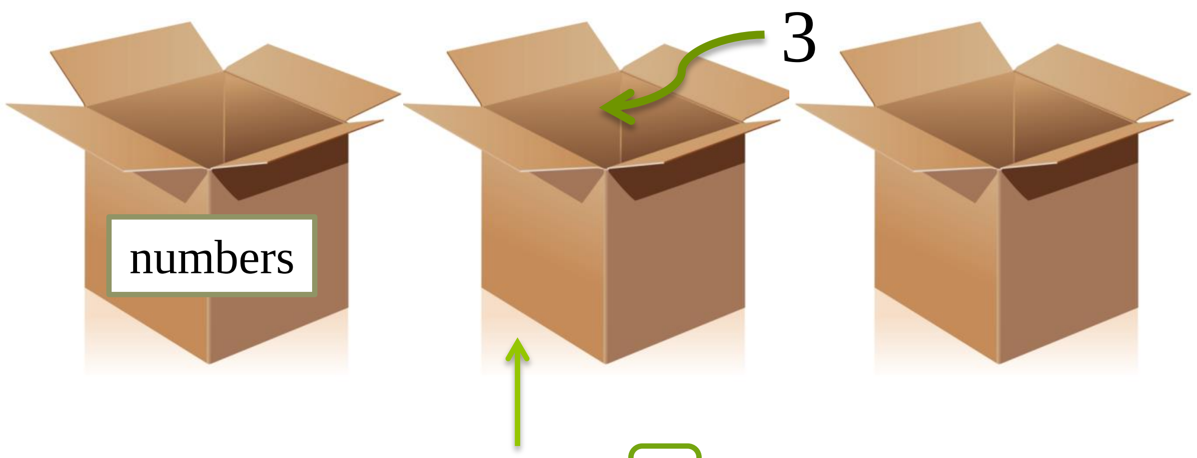
Arrays in C++

Assigning to a single variable:



```
number1 = 3
```

Assigning to an array:



```
numbers[1] = 3;
```

Memory Address	Identifier	Data Stored
500	numbers[0]	
501		
502		
503		
504	numbers[1]	
505		
506		
507		
508	numbers[2]	
509		
510		
511		

```
int numbers[3];
```

Declaring an array saves the **total** number of contiguous bytes it will need in memory, determined by data type and size given when declared.

Memory Address	Identifier	Data Stored
500	numbers[0]	
501		
502		
503		
504	numbers[1]	3
505		
506		
507		
508	numbers[2]	
509		
510		
511		

```
numbers[1] = 3;
```

Assigning a value to an array index saves the value at the appropriate memory offset.

Arrays in C++

Caution:

C++ will let you use an index that is out of bounds, which means you might be able to read or write data outside of the array! Be very careful!

Arrays in C++

Fixed Arrays:

```
const int data[] = {1, 3, 45, 3, -5, 13};
```

Poll Everywhere Question

What will the following code output?

```
const int AS = 6;
int array[AS] = {67, 43, 98, 87, 50, 78};
int array2[AS] = {3, 4, 0, 1, 5, 2};
double s = 0;

for (int i=0; i < AS && array2[i]; i++)
{
    s += array[array2[i]];
}

cout << s / AS << endl;
```

Text 37607

249455: 18.3333 **249477**: 22.8333 **249490**: 70.5

Structs

Structs in C++

Before making struct variables, you have to define what they will look like:

```
struct character
{
    string name;
    int hitPoints;
    int maxHitPoints;
};
```

Structs in C++

Before making struct variables, you have to define what they will look like:

Keyword to indicate you are going to define a struct

```
struct character
{
        string name;
        int hitPoints;
        int maxHitPoints;
};
```

Structs in C++

Before making struct variables, you have to
defin e:

The name of your struct – this will become a new variable type

```
struct character
{
        string name;
        int hitPoints;
        int maxHitPoints;
};
```

Structs in C++

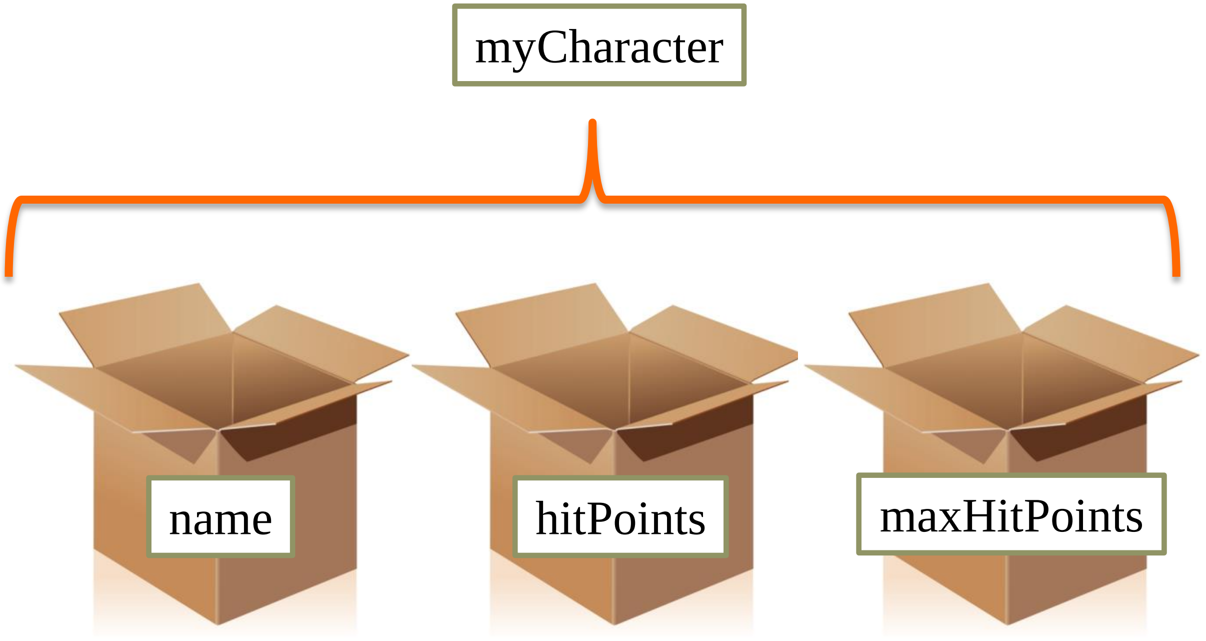
Before making struct variables, you have to define what they will look like:

```
struct character
{
    string name;
    int hitPoints;
    int maxHitPoints;
};
```

These variables will be packaged together into a single "character" type.

Structs in C++

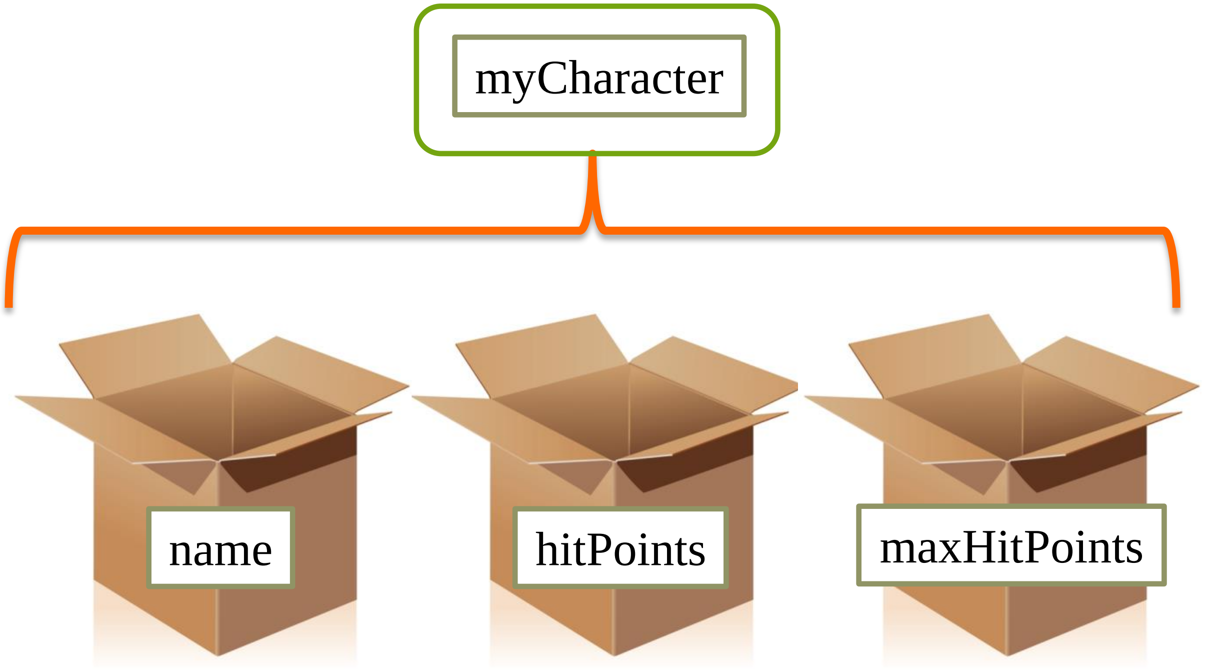
Declaring a struct:



```
character myCharacter;
```

Structs in C++

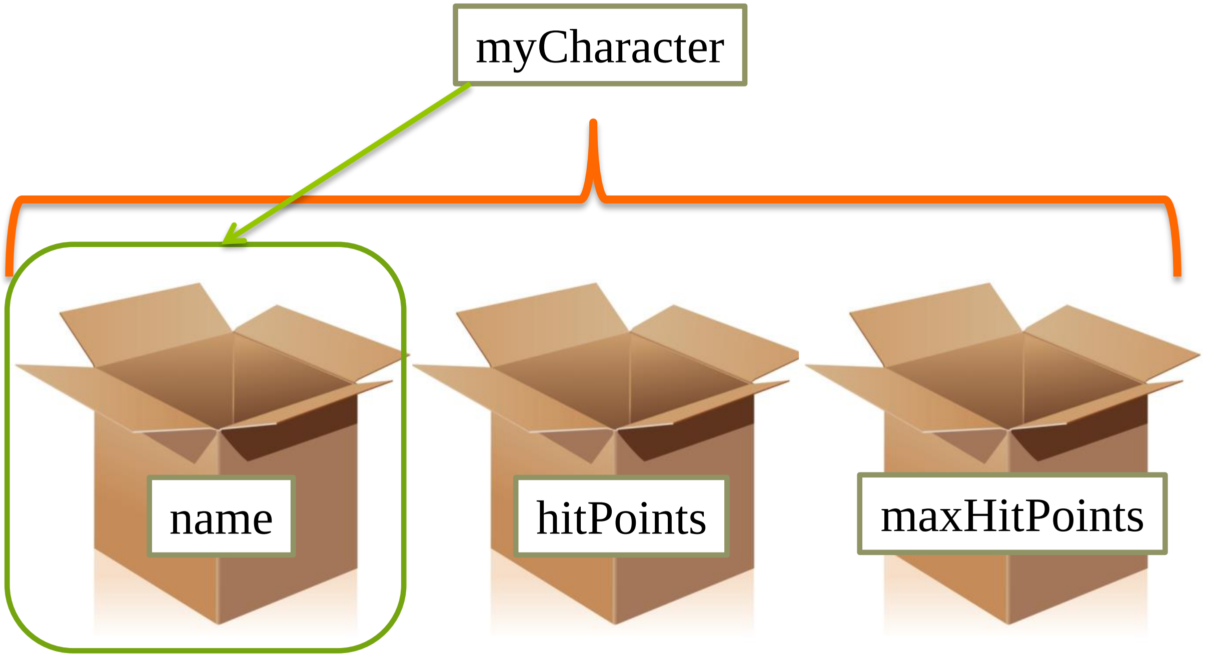
Assigning to a struct:



```
myCharacter.name = "Colonel Mustard";
```

Structs in C++

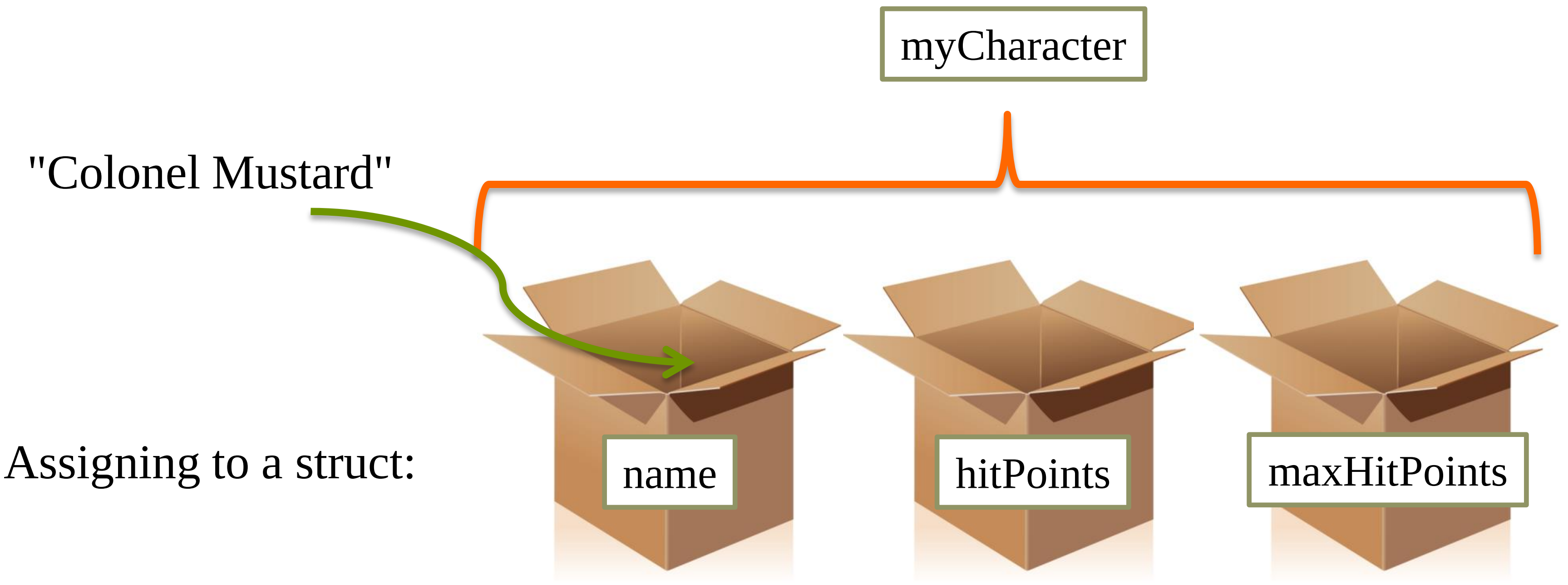
Assigning to a struct:



```
myCharacter.name = "Colonel Mustard";
```

Structs in C++

Assigning to a struct:



```
myCharacter.name = "Colonel Mustard";
```

Memory Address	Identifier		Data Stored
`500`	`myCharacter`	`name`	
…			
`500 + x`		`hitPoints`	
`501 + x`			
`502 + x`			
`503 + x`			
`504 + x`		`maxHitPoints`	
`505 + x`			
`506 + x`			
`507 + x`			

```
character myCharacter;
```

Declaring a struct is like declaring its individual members contiguously in memory, each taking however much space it needs.

Memory Address	Identifier		Data Stored
500	myCharacter	name	
…			
500 + x		hitPoints	10
501 + x			
502 + x			
503 + x			
504 + x		maxHitPoints	
505 + x			
506 + x			
507 + x			

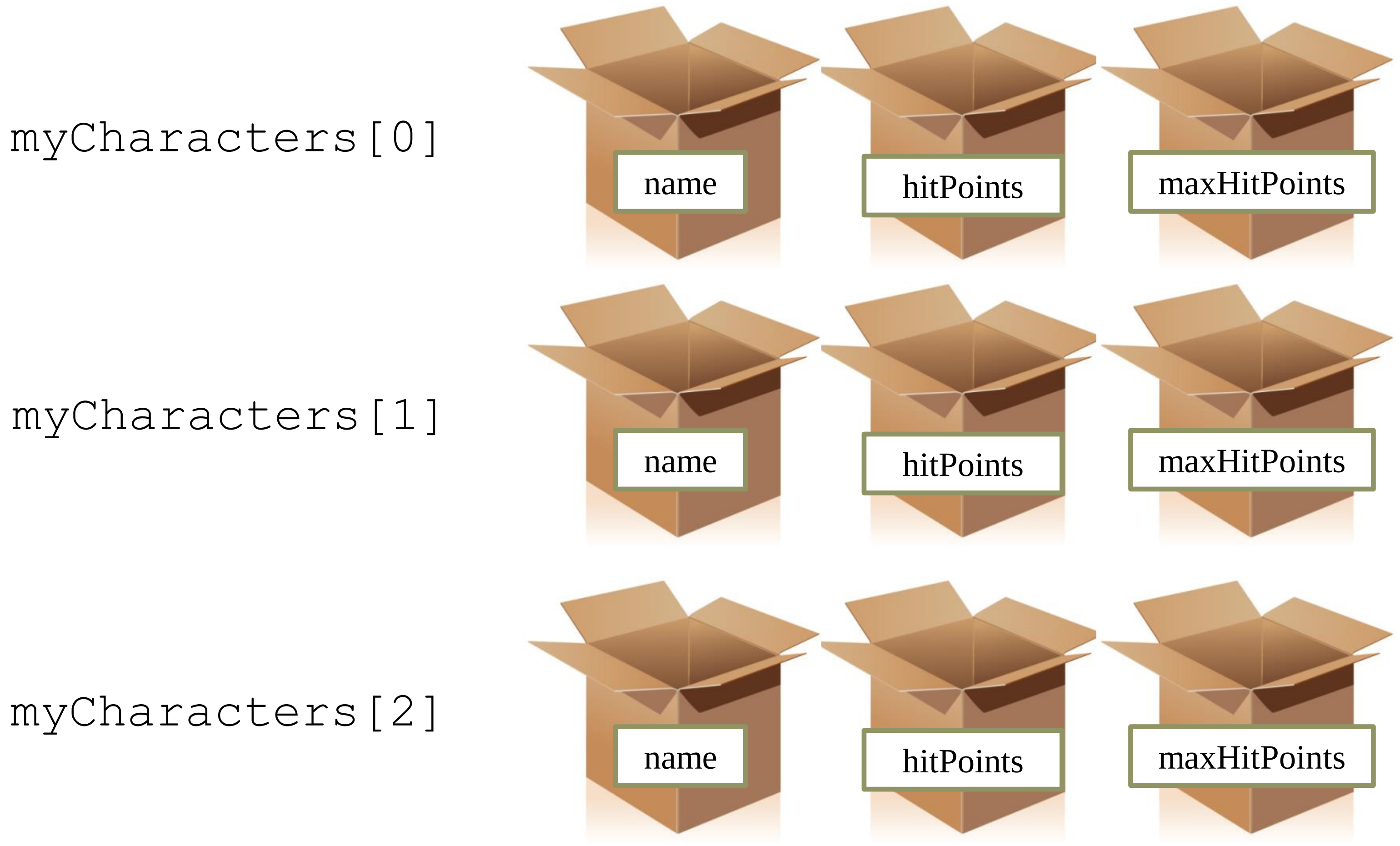
10
name
hitPoints
maxHitPoints

```
myCharacter.hitPoints = 10;
```

To assign to the struct, the name of the member is used to find the right place in memory.

Arrays of Structs

```
character myCharacters[3];
```



Pass by Reference

Which one will work the way we expect?

```
void add1(int num)
{
    num++;
}
```

```
void add1(int &num)
{
    num++;
}
```

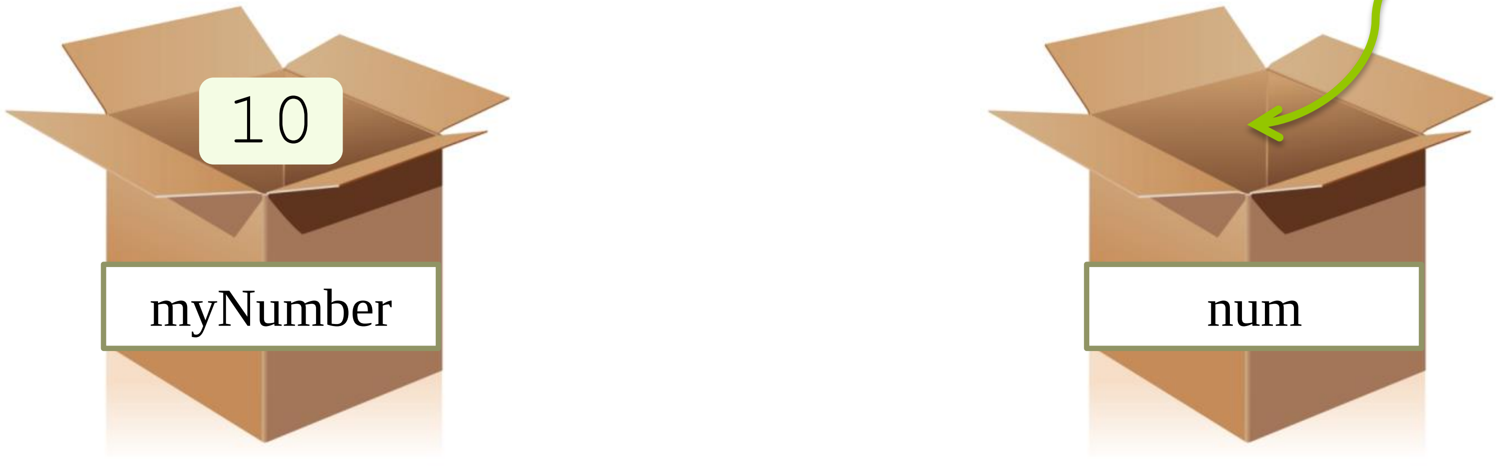
```
int main()
{
    int myNumber = 10;
    add1(myNumber);
    cout << myNumber << endl;
}
```

```
void add1(int num)
{
    num++;
}
```

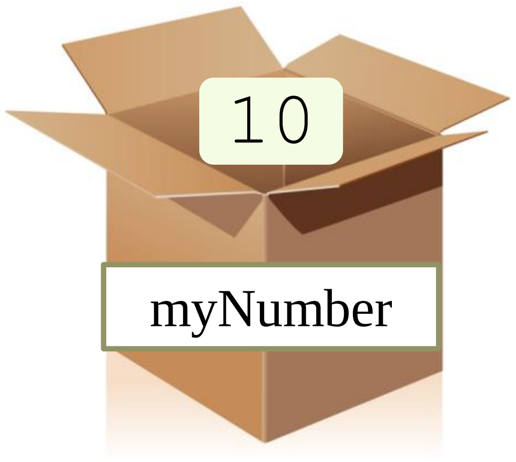
10
myNumber

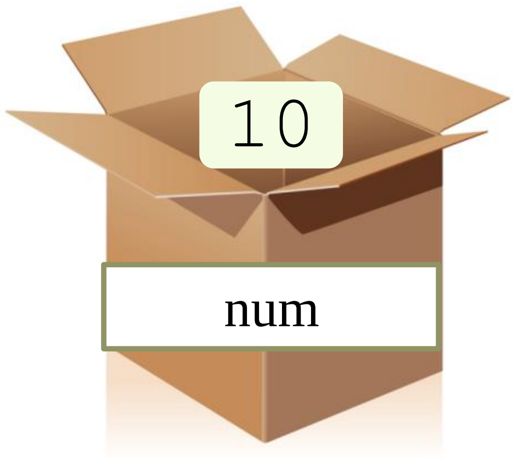
```
void add1(int num)
{
    num++;
}
```

`myNumber`'s value gets copied into `num`

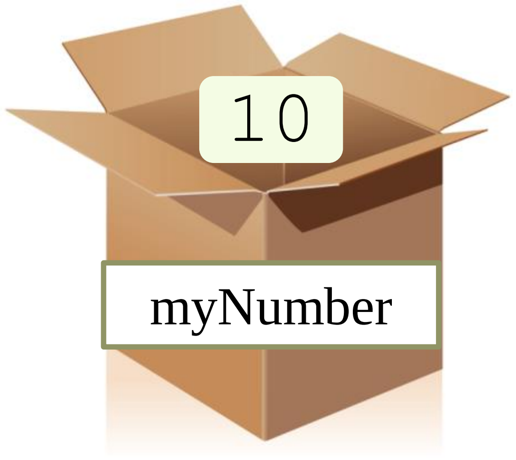


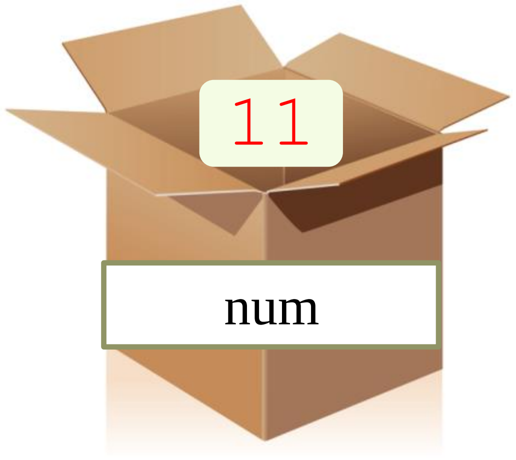
```
void add1(int num)
{
   num++;
}
```





```
void add1(int num)
{
    num++;
}
```



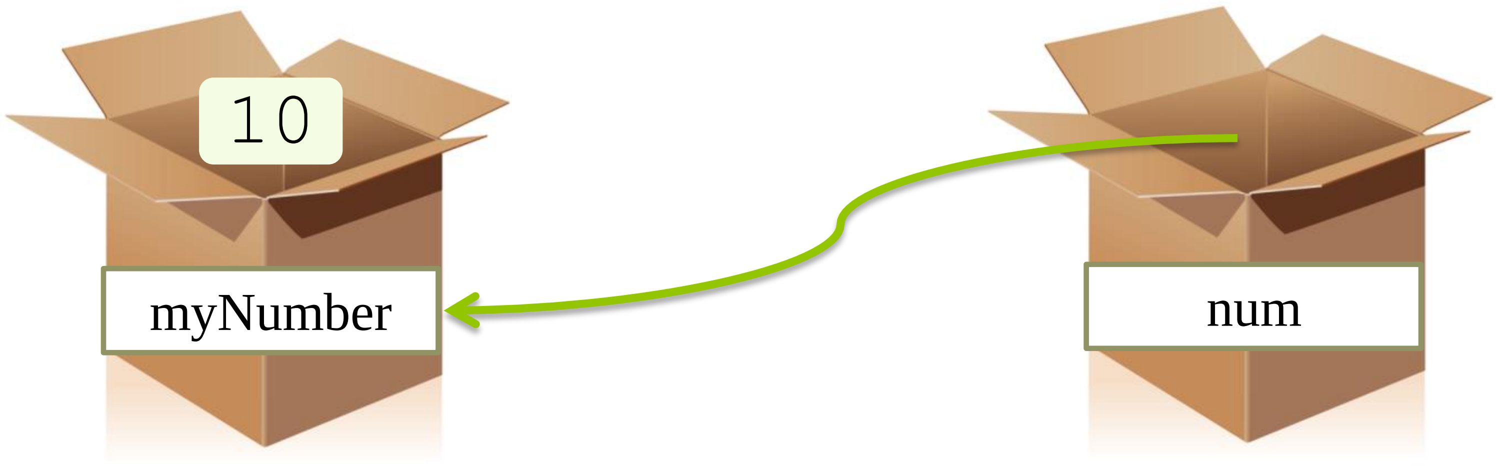


```
void add1(int &num)
{
    num++;
}
```

10
myNumber

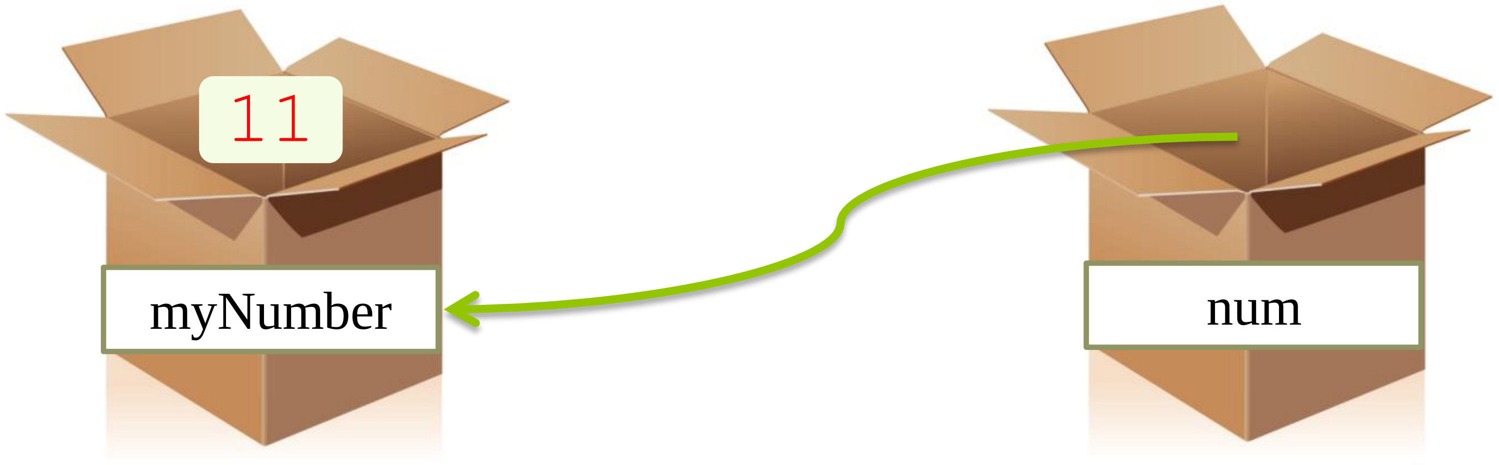
```
void add1(int &num)
{
    num++;
}
```

With `num` being a reference, it's as though an arrow to `myNumber` is put into the `num` box rather than the value 10.



```
void add1(int &num)
{
    num++;
}
```

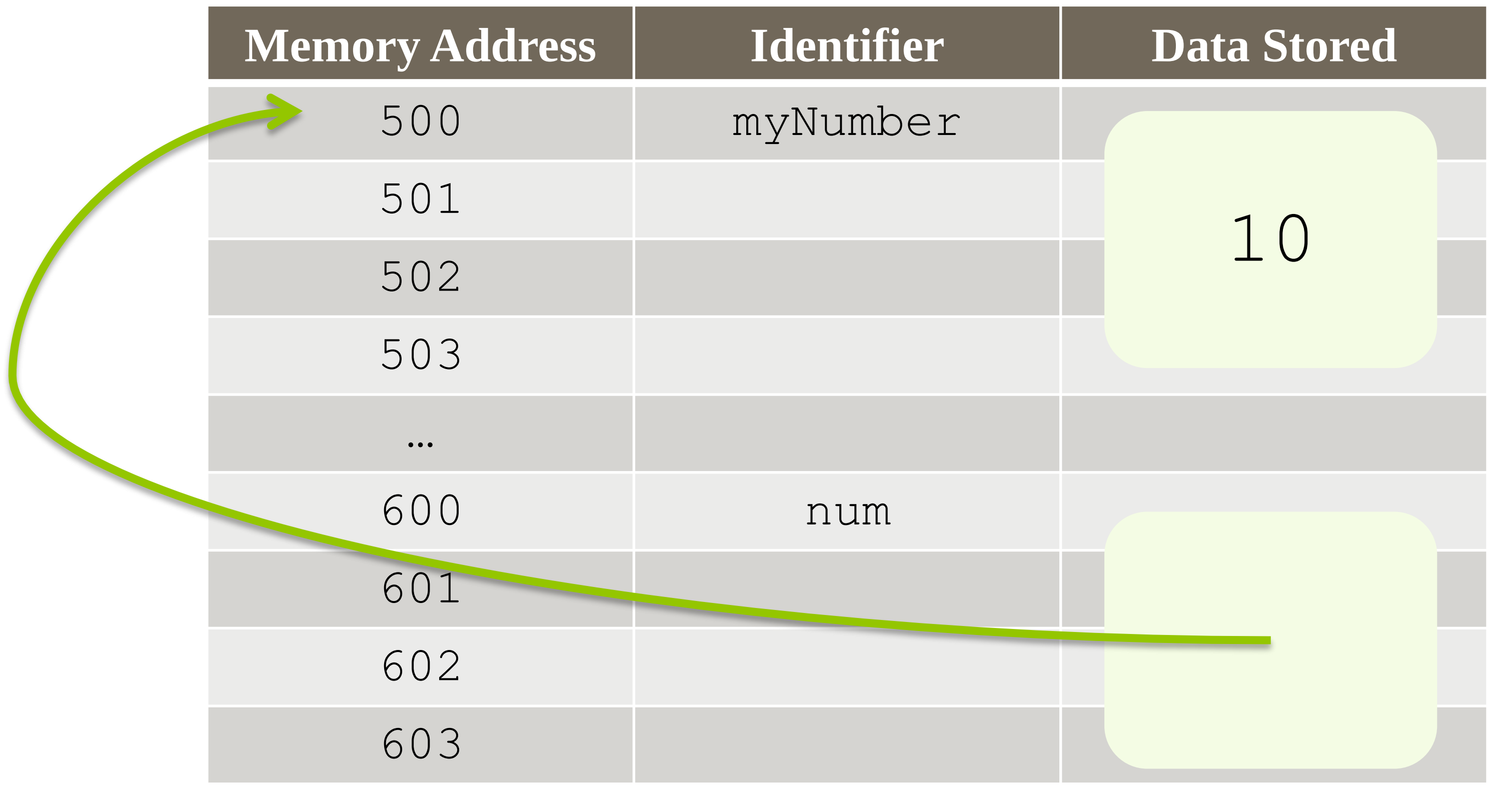
When accessing `num` to add one, C++ automatically follows the arrow and changes the value in the `myNumber` box.



Memory Address	Identifier	Data Stored
500	myNumber	10
501		
502		
503		
…		
600		
601		
602		
603		

```
int myNumber = 10;
```

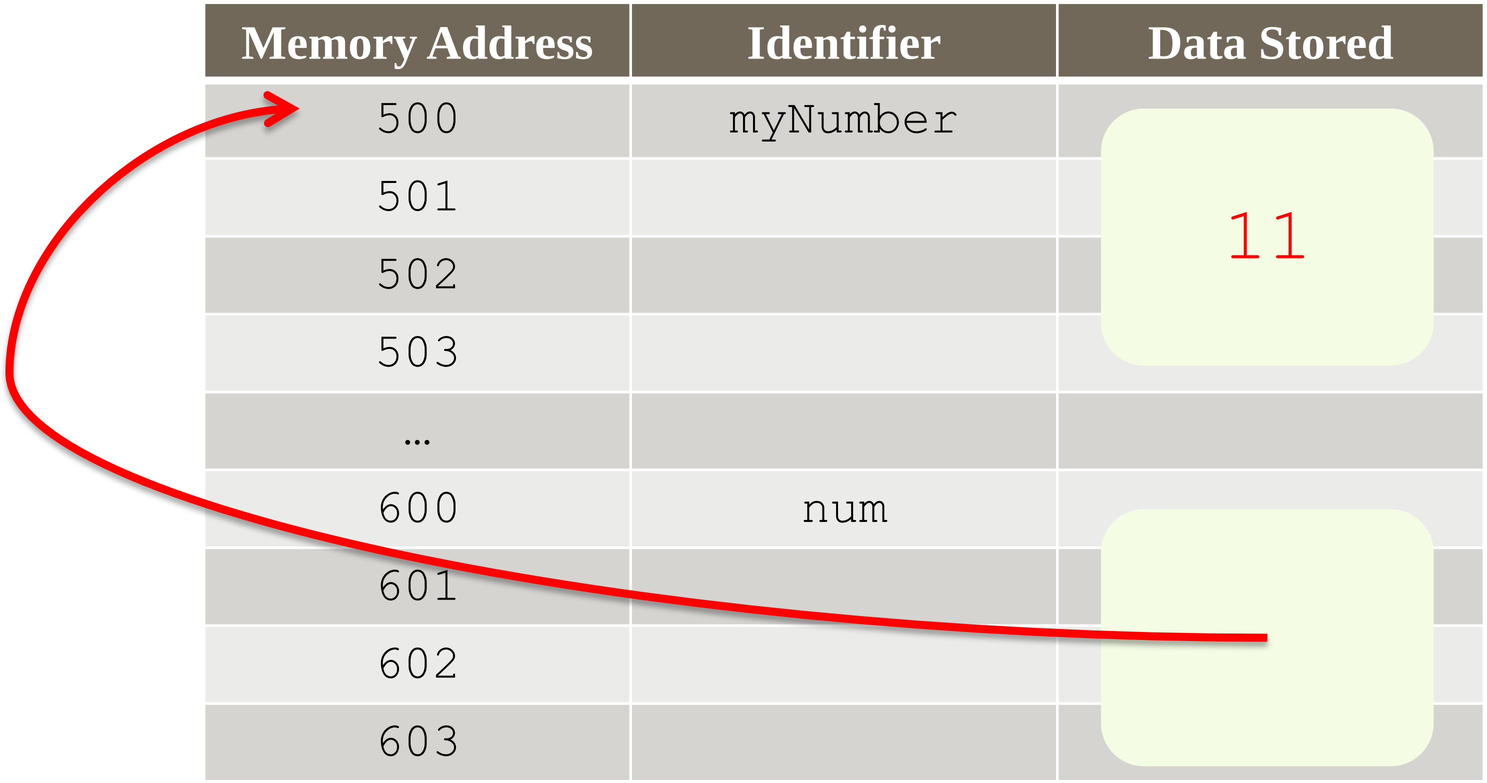
```
void add1(int &num)
{
    num++;
}
```



Memory Address	Identifier	Data Stored
500	myNumber	10
501		
502		
503		
…		
600	num	
601		
602		
603		

```
add1(myNumber);
```

```
void add1(int &num)
{
    num++;
}
```



Memory Address	Identifier	Data Stored
500	myNumber	11
501		
502		
503		
…		
600	num	
601		
602		
603		

```
add1(myNumber);
```

```
void add1(int &num)
{
    num++;
}
```