



Methods

Passing Parameters by Reference

Method Parameters: pass by reference

Pass by reference = ALIAS

- When you pass parameters by reference to a method, **the method creates a new name for the variable** instead of copying it to a new variable.
- Changes made to the variable inside the method affect the variable used when calling the method.

Pass by Reference (ALIAS)

```
int n1 = 5, n2 = 2;
```

```
int result = t1000.sum(n1, n2);
```

New name for

New name for

```
int Calculator::sum(int& number1, int& number2)
```

```
{
```

```
    return number1 + number2;
```

```
}
```



Methods

const modifier

const modifier on methods

- The **const** modifier (which stands for constant) tells the compiler that the method is NOT ALLOWED to change member variables.
- If code is added to the method that changes a value of the class, the compiler will generate a build error.
- The const modifier added to a method is a way to communicate that the method will not have side effects on the class.

const modifier on methods

```
class Sample
{
    private:
        int mSomeData = 5;
        int WhatIsTheData() const
        {
            mSomeData += 10; //not allowed to change fields!
            return mSomeData;
        }
};
```



Parameters

const modifier

const modifier

- Pass by reference allows the method to change the variable in the calling code. If that is not intended, you can add the **const** modifier to the pass by reference parameter.
- The compiler will then prevent the variable from being modified.

const modifier

```
float average(const std::vector<int>& scores)
{
    scores.push_back(5); //not allowed because scores is marked as const
```



`std::vector<type>`

[Vector in C++ STL - GeeksforGeeks](#)



`std::vector<type>` Size vs capacity

- The `size()` of a vector will tell you how many items are in the vector.
- The `capacity()` of a vector will tell you how big the internal array is.
- NOTE: these numbers do not always match.


`std::vector<type>.reserve(n)`

`reserve(n)` will pre-size the internal array

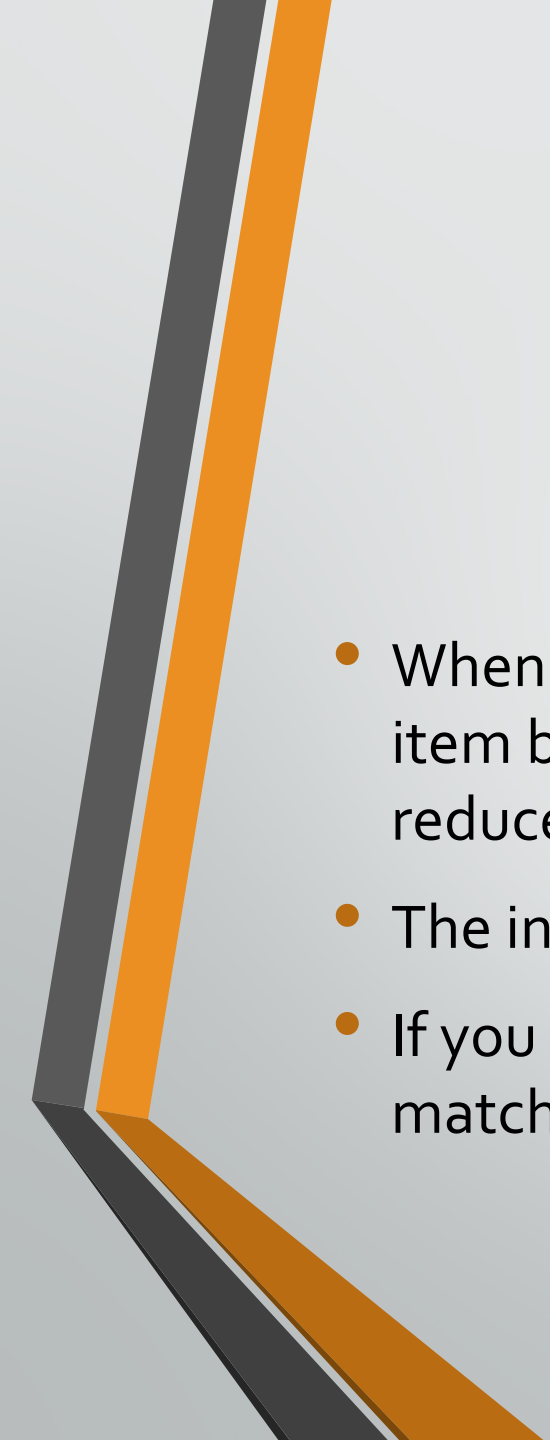
```
std::vector<int> scores;
```

```
scores.reserve(10); //makes the internal array to hold 10 items.
```

- The vector will then not need to be resized until it goes beyond 10 items.
- Will not remove any items that currently exist in the vector.
- If `n` is smaller than the current size, it will do nothing.



`std::vector<type>`
Removing in a loop



`std::vector<type>` Removing in a loop

- When you remove an item from the vector, every item to the right of the item being removed is shifted down (or left) by 1 spot and the size is reduced by 1.
- The internal array however stays the same size.
- If you remove the items in a loop, you need to ensure that all the items that match the criteria are removed.

`std::vector<type>` Removing in a loop

Size is 6



Erasing 7 from the vector...

Size is 5

