CMPT 225 Vector Implementations

Summary

Terminology

Vector Implementation Basics

```
#include "IVector.h"
#include <iostream>

using namespace std;

int main( ) {
const int N = 20;

IVector v ; // Make an int Vector
v.display(); // print its contents

// Store N ints in the Vector
for( int i = 0 ; 1 < N; ++i ) {
  v.push_back(i);
}

// print the contents
v.display();

return 0;
}
```

Methods

`reserve()`

- If the size reaches the capacity, a new array needs to be made

`push_back()`

- Pushes to the back of the vector, calls reserve if necessary

`pop_back()`

- Pops the item off the back of the vector

Templates

- often have algorithms that will work on many data types, with few or no changes.
- In strongly typed languages, we need a way to produce "generic" code
- In C++, templates let us write generic code

- A template function or class definition has a placeholder for one or more data types that is instantiated at compile time'
- The instantiation may be different at different places in the same code.