```
int studentId00 = 9000; int studentId20 = 9020; int studentId40 = 9040; int studentId60 = 9060;
int studentId01 = 9001; int studentId21 = 9021; int studentId41 = 9041; int studentId61 = 9061;
int studentId02 = 9002; int studentId22 = 9022; int studentId42 = 9042; int studentId62 = 9062;
int studentId03 = 9003; int studentId23 = 9023; int studentId43 = 9043; int studentId63 = 9063;
int studentId04 = 9004; int studentId24 = 9024; int studentId44 = 9044; int studentId64 = 9064;
int studentId05 = 9005; int studentId25 = 9025; int studentId45 = 9045; int studentId65 = 9065;
int studentId06 = 9006; int studentId26 = 9026; int studentId46 = 9046; int studentId66 = 9066;
int studentId07 = 9007; int studentId27 = 9027; int studentId47 = 9047; int studentId67 = 9067;
int studentId08 = 9008; int studentId28 = 9028; int studentId48 = 9048; int studentId68 = 9068;
int studentId09 = 9009; int studentId29 = 9029; int studentId49 = 9049; int studentId68 = 9069;
int studentId10 = 9010; int studentId30 = 9030; int studentId50 = 9050; int studentId70 = 9070;
int studentId11 = 9011; int studentId31 = 9031; int studentId51 = 9051; int studentId71 = 9071;
int studentId12 = 9012; int studentId32 = 9032; int studentId52 = 9052; int studentId72 = 9072;
int studentId13 = 9013; int studentId33 = 9033; int studentId53 = 9053; int studentId73 = 9073;
int studentId14 = 9014; int studentId34 = 9034; int studentId54 = 9054; int studentId74 = 9074;
int studentId15 = 9015; int studentId35 = 9035; int studentId55 = 9055; int studentId75 = 9075;
int studentId16 = 9016; int studentId36 = 9036; int studentId56 = 9056; int studentId76 = 9076;
int studentId17 = 9017; int studentId37 = 9037; int studentId57 = 9057; int studentId77 = 9077;
int studentId18 = 9018; int studentId38 = 9038; int studentId58 = 9058; int studentId78 = 9078;
int studentId19 = 9019; int studentId39 = 9039; int studentId59 = 9059; int studentId79 = 9079;
```

I figured out how to store 80 student numbers!!!!

Vectors

Slides #9 -- Section 8.11

Topics

1) How can we store many values at once?
2) How can we pass vectors to functions?

Vectors

Vector

- Vector Object:
  -
  - Can dynamically grow and shrink, and report its size.

price =

0	1	2	3
1	5	12	20

Vector example

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

int main() {
    // Create a vector of unsigned ints
    vector<unsigned int> myFavNums;

    // Insert my favourite numbers
    myFavNums.push_back(42);
    myFavNums.push_back(0);
    myFavNums.push_back(3141590000);

    // Print out the three numbers
    cout << "Num 1: " << myFavNums[0] << endl;
    cout << "Num 2: " << myFavNums[1] << endl;
    cout << "Num 3: " << myFavNums[2] << endl;

    return 0;
}
```

Must include <vector> and name-space std.

When created, must specify type of values it will hold:

Add an element to the vector with:

Use [] notation to access elements. Ex: unsigned int k = data[i];

```
Num 1: 42
Num 2: 0
Num 3: 3141590000
```


Vectors

- Vector is in the Standard Template Library (STL):
  - STL is programmer-created data types and algorithms (not part of 'core' C++).
  - It is a template class:
    It can be used to hold...
- State type of data to hold when creating vector:

```
vector<int>             vectOfInts;
vector<double>          vectOfDoubles;
vector<string>          vectOfStrings;
vector<unsigned short>  vectOfUShorts;
```


Vector Element Access

- Directly access to any element:
  - For N elements...

```
daysPerMonth[0] = 31;          // January
```

Pronounced...

- Ex:

```
daysPerMonth[11] = 31;              // December
int a = daysPerMonth[1];            // February
int guess = daysPerMonth[i + 1];    // Depends on i.
cout << daysPerMonth[1];            // Outputs 28
cin >> daysPerMonth[9];             // Read in oct.
```

Vector object daysPerMonth

	Idx	Val
Jan	0	31
Feb	1	28
Mar	2	31
Apr	3	30
May	4	31
Jun	5	30
Jul	6	31
Aug	7	31
Sep	8	30
Oct	9	31
Nov	10	30
Dec	11	31

Vector Elements vs Values

- An element's value and its index are different:

```
vector<int> price;
// price.push_back(...)
```

price =

0	1	2	3
1	5	12	20

  - Add 2 elements:

```
int a = price[1] + price[2];      //
```

  - Add 2 indices:

```
int b =price [1 + 2];             //
```

- Two (independent) ++ Example:

```
int i = 2;
cout << price[i]++;
```

```
int i = 2;
cout << price[i++];
```

Vector methods

Function	Description
myVect[i]	Access the i'th element of myVect (where i is an integer) Ex: int val = myVect[i]; Ex: myVect[i] = 77;
myVect.clear()	Removes all elements from the vector.
myVect.empty()	Returns true if the vector is empty, false otherwise.
myVect.pop_back()	Removes the last element from the vector.
myVect.push_back(42)	Adds the number 42 to the end of the vector. The value must match vector type.
myVect.size()	

See text or online documentation for more vector methods and constructors.

Vector example

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

int main () {
      vector<double> data;

      cout << "Enter values (0 to exit):\n";
      int inVal = 0;
      do {
            cin >> inVal;
            data.push_back(inVal);
      } while (inVal != 0);

      cout << "\nData:\n";
      for (int i = 0; i < data.size(); i++) {
            cout << i << ": " << data[i] << endl;
      }

      return 0;
}
```

size() method returns the number of elements.

Review

- Write some code which creates a vector to hold characters and insert the first 2 letters of your name.

- Write a loop to output the contents of the above vector. Do not hardcode the size!

Vectors as function arguments

Passing elements

- Single elements of a vector can be passed to function...

```
void showOneElement(char ch) {
      cout << "Element: " << ch << endl;
}
void changeOneElement(char &ch) {
      char newVal = 'x';
      cout << "Changing " << ch << " to "
            << newVal << "." << endl;
      ch = newVal;
}
```

```
int main () {
      vector<char> greeting;
      greeting.push_back( 'H' );
      greeting.push_back( 'i' );
      greeting.push_back( '!' );

      // Pass an element by value.
      showOneElement( greeting[0] );

      // Pass an element by reference.
      changeOneElement( greeting[0] );
      showOneElement( greeting[0] );
      ...
}
```


Passing a whole vector

- You can pass a to a function using pass by value, or pass by reference.

```
void changeA( vector<int> changeMe ) {
      changeMe.push_back(42);
}
void changeB( vector<int> &changeMe ) {
      changeMe.push_back(1337);
}
```

```
int main () {
      // Create the vectors
      vector<int> v1, v2;

      // Pass by value example:
      changeA(v1);

      // Pass by reference example:
      changeB(v2);
      ...
}
```


Summary

- C++ vectors store many items of the same type.
  - Can grow & shrink.
- Passing to functions:
  - Pass elements or whole vector.
  - Pass by value or pass by reference.