vector and its application

Section 6.7

array vs. struct

- Array is a collection of same type of data residing in consecutive memory spaces.
- Struct is a way to group several related variables (not necessarily the same type) into one group.
- A struct is like a class with all public data members, but without methods on those data members.

Sidenote: struct is like a class with public data members without methods

```
class RowCol {
public:
    int row;
    int col;
};
```

```
struct RowCol {
    int row;
    int col;
};
```

vector is like an intelligent array

- Need to #include <vector> to use vector.
- Vector is a collection of same type elements.
  - To declare a vector of int, use vector<int>.
  - To declare a vector of string, use vector<string>.
  - Suppose we define

```
struct RowCol {
        int row;
        int col;
};
```

  - To represent a vector of cells, using vector<RowCol>.

Declare vector and operations

- Declare a vector
  - vector<int> first; //create an empty vector of int with no element
  - vector<int> second(4); //create a vector starting with 4 elements.
  - vector<int> third(4, 10);
    //create a vector starting with 4 elements, each one is 10.
- Operations for vectors include push_back, size.
- You can access an element of array like it is an array.

An example of vector usage

```
vector<int> vect;
vect.push_back(1);
vect.push_back(2);
vect.push_back(3);

for (int i = 0; i < vect.size(); i++)
    cout << vect[i] << endl;
```

Another example of vector

```
vector<string> vect2;
vect2.push_back("Hello");
vect2.push_back("Hi");
vect2.push_back("Good morning");

for (string str : vect2) //C++ 11
    cout << str << endl;
```