ITP 365

Lecture 2

**Std::strings:**

* “Nathan” + “Greenfield” are string literals and not technically std::string classes
  + Need to cast one -> std::string(“Nathan”) + “Greenfield” works
* **4 basic data types**
  + **char**
  + **Boolean**
  + **Int**
  + **Double**
* Can access string characters by doing varName[3] sort of like arrays
* since std::string is a class, can access member functions
  + name.length();
* String member functions
  + Clear – clears entire string so its empty
  + Empty – tells you whether or not string is empty
    - Returns bool true/false
  + Front – gets first character in string
    - Returns char
  + Back – gets last character in string (before null terminator)
    - Returns char
  + Find – searches for the first appearance of another string
    - Parameters (string searching for, index you want to start at)
    - Returns index at which the thing appears
      * But since indexes are never negative, you can’t have the thing return like -1 if it doesn’t exist bc its an unsigned int so you have to use std::string::npos
      * If you don’t find the string then you get the value that std::string::npos is equal to (some large ass #)
      * Combat this by doing this:
      * If (title.find(“hello”) == std::string::npos){
        + Cout << “Didn’t find string << endl;}
  + **Substring – returns a new string that’s a substring**
    - **Parameters (index to start at, the number of characters to include)**
    - **Returns a new string**
  + **Overloading a function: when function named same but can have different number of parameters, like substring bc it has optional second parameter**

**Function parameters:**

* Pass by value (makes copy, changes don’t persist)
* Pass by reference (int& x, int& y)
* **\*\*\*\*Practice this stuff\*\*\*\***
* Another reason to pass by reference is that it makes things much faster
* Basic types, usually pass by value unless you DO want parameter modifications to persist
* Non-basic types, usually pass by reference to avoid the copy (string, ifstream, ofstream)

**Stanford C++ Library:**

* Convert to/from strings
* Graphical drawing
* Collections (data structures)
* GWindow class
  + GWindow gw(500, 500)
    - Creates instance of GWindow that’s 500x500
  + Origin is top left corner (0,0)
* drawRect function to draw rectangle
  + gw.drawRect(x, y, width, height)
    - x/y are top left coordinates, width/height of rect.
* fillRect function to fill rectangle
  + gw.fillRect(x, y, width, height)
* setColor
  + gw.setColor(color)
  + Ex. gw.setColor(“blue”);
  + gw.fillRect(50, 50, 200, 100) 🡪 will be in blue
  + gw.setColor(“red”);
  + gw.fillRect(50, 50, 200, 100) 🡪 will be in red
* draw a text string (label)
  + gw.drawLabel(text, x, y)
    - text is the std::string you want to draw
    - x/y are coordinates of the center of the text

**Collections – type of data structure that can store elements:**

* Arrays, lists, dictionaries,
* **Strings ARE collections (of char’s)**
* **Enums ARE NOT collections (can only hold one type of enum at a time)**
* Array
  + **Problem 1: C++ arrays are created with a fixed size**
  + **Problem 2: Arrays don’t know their own size. Can go too far in a for loop trying to access array elements**
  + **Problem 3: If we don’t initialize the array to values, it will have random garbage data**
* Vector is a good fix to an array
  + Automatically grows (and shrinks)
  + Allows to insert and remove elements whenever you want to
  + Can check to make sure you aren’t accessing invalid index
  + Stanford Vector:
    - #include “vector.h”
    - Vector<int> myVector;
  + Size function
    - myVector.size()
  + Add an element to the end of the vector
    - myVector.add(element) -> the element type must match type of Vector
  + Remove an element from vector at specified index
    - myVector.remove(index)
    - If you were to remove beginning element, vector has to copy/shift all indices to the left by 1
  + Insert an element **before** the specified index
    - myVector.insert(index, element)
  + Check if a vector is empty
    - myVector.isEmpty()
    - returns a Boolean true or false
  + Removes everything from the vector
    - myVector.clear()