**Compound Types**   
  
Workshop 2

In this workshop, you review simple fstream input and output and use the string class of the Standard Library to manage character string data.

**LEARNING OUTCOMES**

Upon successful completion of this workshop, you will have demonstrated the abilities to

* work with objects of the string class
* declare and use enumeration constants
* use the correct constant type in initializations
* retrieve data from and backup data to a text file
* move data between unsigned and signed integers

**SPECIFICATIONS**

**Station**

Design and code a class named **Station** that holds information about the monthly passes for sale at a monitored Subway station.  There are two types of passes:

* Student passes
* Adult passes

The information stored in each **Station** object includes:

* the name of the station - stored as a **string** object
* the number of student passes - stored as an **unsigned int**
* the number of adult passes - stored as an **unsigned int**

You can find information about the **string** class [here](https://scs.senecac.on.ca/~oop345/pages/content/string.html) and more information [here](https://www.cplusplus.com/reference/string/string/getline).

Your class design includes the following member functions:

* **Station()** - a default constructor that initializes the instance variables through an initialization list
* **void set(const std::string&, unsigned, unsigned)** - sets the instance variables to the values received in its parameters
* **void update(PassType, int)** - updates the number of passes - sold if negative, added if positive
* **unsigned inStock(PassType) const** - returns the number of passes of the requested type
* **const std::string& getName() const** - returns a reference to a string object holding the name of the station

**PassType** is an enumeration type that identifies pass types.

The deduction of passes sold in **update()** from the instance variable requires mixing signed and unsigned integers and may cause some strange results.  Discuss this issue with other students in your class.  See also the note on the ["The Unsigned Trap"](https://scs.senecac.on.ca/~oop345/pages/content/expre.html#uns) in the chapter on [Expressions](https://scs.senecac.on.ca/~oop345/pages/content/expre.html).

**Stations**

Code a class named **Stations** that processes information about the monthly passes stored at a set of monitored Subway stations.  The **Stations** constructor receives the name of a file in the form of a C-style, null-terminated string.  The file contains all of the data for the monitored stations (see below).  The first record holds the number of stations and the delimiter character for name input.  Each subsequent record holds the name of a station followed by the delimiter, the number of students passes and the number of adult passes.  Upon instantiation, a **Stations** object opens the file, allocates memory for the monitored stations and copies the data from the file to each **Station** object.  Upon destruction, a **Stations** object copies the updated data from memory to the same file, overwriting the previous records.

You can find notes on file objects [here](https://scs.senecac.on.ca/~oop244/pages/content/files.html).

Your class design includes three member functions:

* **void update() const** - accepts the number of passes sold and updates the data for each station in turn
* **void restock() const** - accepts the number of passes added and updates the data for each station in turn
* **void report() const** - reports the number of passes currently available at each station

**Main Program**

The main program that uses your two classes is listed below:

|  |
| --- |
| **// Workshop 2 - unsigned int, Enumerations, String Class and File I/O**  **// w2.cpp**  **#include <iostream>**  **#include "Stations.h"**  **int main(int argc, char\* argv[]) {**  **std::cout << "\nCommand Line : ";**  **for (int i = 0; i < argc; i++) {**  **std::cout << argv[i] << ' ';**  **}**  **std::cout << std::endl;**  **if (argc != 2) {**  **std::cerr << "\nIncorrect number of arguments\n";**  **return 1;**  **}**  **w2::Stations stations(argv[1]);**  **stations.update();**  **stations.restock();**  **stations.report();**  **return 0;**  **}** |

The test file (**Stations.dat**) contains the following data

|  |
| --- |
| **4;**  **Spadina;76 156**  **Bathurst;121 291**  **Keele;70 61**  **Bay;158 158** |

The output produced by this main program along with your classes is shown below:

|  |
| --- |
| **Command Line : w2 Stations.dat**  **Passes Sold :**  **-------------**  **Spadina**  **Student Passes sold : 5**  **Adult Passes sold : 8**  **Bathurst**  **Student Passes sold : 34**  **Adult Passes sold : 52**  **Keele**  **Student Passes sold : 21**  **Adult Passes sold : 13**  **Bay**  **Student Passes sold : 67**  **Adult Passes sold : 65**  **Passes Added :**  **--------------**  **Spadina**  **Student Passes added : 29**  **Adult Passes added : 52**  **Bathurst**  **Student Passes added : 113**  **Adult Passes added : 61**  **Keele**  **Student Passes added : 51**  **Adult Passes added : 52**  **Bay**  **Student Passes added : 109**  **Adult Passes added : 107**  **Passes in Stock : Student Adult**  **-------------------------------**  **Spadina 100 200**  **Bathurst 200 300**  **Keele 100 100**  **Bay 200 200** |

**SUBMISSION**

**Typescript**

On matrix, create a typescript of your complete solution using the following commands:

**+ At the prompt, type: script w2.txt**

**+ At the prompt, type: whoami**

**+ At the prompt, type: cat Station.h Station.cpp Stations.h Stations.cpp**

**+ At the prompt, type: g++ -o w2 w2.cpp Station.cpp Stations.cpp**

**+ At the prompt, type: w2 Stations.dat**

**+ At the prompt, enter input shown**

**+ At the prompt type: exit**

These commands will produce a file named **w2.txt**.

Download your typescript file to your local computer.

**Moodle**

* Login to [OOP244](https://scs.senecac.on.ca/~oop244/index.html)
* Select OOP345 if necessary
* Select W2 under Workshops
* Upload your typescript file to Moodle
* Press "Edit"
* Summarize to your instructor the concepts that you have learned in doing this particular workshop.  Add any other comments you wish to make.
* Press "Save Changes"
* When ready to submit, press "Send for Marking"

**MySeneca**

* Login to [IPC144](https://scs.senecac.on.ca/~ipc144/index.html)
* Select OOP345 if necessary
* Select Assignments or Workshops
* Select W2
* Press "Browse My Computer" to upload your typescript
* Press "Edit"
* Summarize to your instructor the concepts that you have learned in doing this particular workshop.  Add any other comments you wish to make in the comment box provided.
* Press "Submit" IMPORTANT: If you "Save As Draft" your instructor does not receive your submission unitl you press "Submit"