**Templates**   
  
Workshop 3

In this workshop, you design and code a class template and test it on two different classes.

**LEARNING OUTCOMES**

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

* design and code a class template
* store key-value information in a pair of parallel arrays
* reflect on the material learned in this workshop

**SPECIFICATIONS**

The source files for this workshop include:

* **KVList.h** - defines a class template for a list of key-value pairs
* **w4.cpp** - the application that uses your class template

**KVList Template**

Design and code a class template named **KVList** for managing lists of key-value pairs.  The classes generated by your template contain two parallel arrays of dimension **N** - a key array of type **K** and a value array of type **V**.  **K**, **V** and **N**are template parameters, which the porgrammer who uses your template can specify.  Save your template in a header file named **KVList.h**.

Your design includes the following member functions:

* **KVList()** - default constructor - adopts a safe empty state
* **size\_t size() const** - returns the number of entires in the key-value list
* **const K& key(int i) const** - returns an unmodifiable reference to the key of element **i** in the list
* **const V& value(int i) const** - returns an unmodifiable reference to the value of element **i** in the list
* **KVList& add(const K&, const V&)** - adds a new element to the list if room exists and returns a reference to the current object, does nothing if no room exists
* **int find(const K& k) const** - returns the index of the first element in the list with a key equal to **k** - defaults to 0
* **KVList& replace(int i, const K& k, const V& v)** - replaces element **i** in the list with the key and value received and returns a reference to the current object

**Main Program**

The main program that uses your template definition manages

1. an inventory of product-price pairs
2. a glossary of acronym-definition pairs

For each list, this main program sets the maximum number of entries to 5.

|  |
| --- |
| **// Workshop 4 - Templates**  **// w4.cpp**  **#include <iostream>**  **#include <iomanip>**  **#include <string>**  **#include "KVList.h"**  **template <typename K, typename V, int N>**  **void display(const std::string& msg, const KVList<K, V, N>& list, int w) {**  **std::cout << msg;**  **for (int i = 0; i < list.size(); i++)**  **std::cout << std::setw(w) << list.key(i)**  **<< " : " << list.value(i) << std::endl;**  **}**  **int main(int argc, char\*\* argv) {**  **if (argc != 1) {**  **std::cerr << argv[0] << ": too many arguments\n";**  **return 1;**  **}**  **int width;**  **bool keepreading;**  **std::cout << std::fixed << std::setprecision(2);**  **std::cout << "\nInventory\n=========\n";**  **KVList <std::string, double, 5> inventory;**  **std::string str;**  **double price;**  **keepreading = true;**  **do {**  **std::cout << "Product : ";**  **getline(std::cin, str);**  **if (str.compare("quit") == 0) {**  **keepreading = false;**  **} else {**  **std::cout << "Price : ";**  **std::cin >> price;**  **std::cin.ignore();**  **inventory.add(str, price);**  **}**  **} while(keepreading);**  **display("\nPrice List\n----------\n", inventory, 13);**  **std::cout << "\nCorrections\n-----------\n";**  **keepreading = true;**  **do {**  **std::cout << "Product : ";**  **getline(std::cin, str);**  **if (str.compare("quit") == 0) {**  **keepreading = false;**  **} else {**  **int i = inventory.find(str);**  **if (i != -1) {**  **std::cout << "Price : ";**  **std::cin >> price;**  **std::cin.ignore();**  **inventory.replace(i, str, price);**  **}**  **}**  **} while(keepreading);**  **display("\nPrice List\n----------\n", inventory, 13);**  **std::cout << "\nGlossary\n========\n";**  **KVList <std::string, std::string, 5> glossary;**  **std::string key, definition;**  **keepreading = true;**  **do {**  **std::cout << "Key : ";**  **getline(std::cin, key);**  **if (key.compare("quit") == 0) {**  **keepreading = false;**  **} else {**  **std::cout << "Definition : ";**  **getline(std::cin, definition);**  **glossary.add(key, definition);**  **}**  **} while(keepreading);**  **display("\nEntries\n-------\n", glossary, 5);**  **}** |

For the input listed below the main program with your template produces the output shown:

|  |  |
| --- | --- |
| **Inventory**  **=========**  **Product : Pizza**  **Price : 4.49**  **Product : Pierogi**  **Price : 2.56**  **Product : Potato Chips**  **Price : 2.29**  **Product : Black Tea**  **Price : 4.49**  **Product : Green Tea**  **Price : 3.46**  **Product : Fruit Tea**  **Price : 2.29**  **Product : quit**  **Price List**  **----------**  **Pizza : 4.49**  **Pierogi : 2.56**  **Potato Chips : 2.29**  **Black Tea : 4.49**  **Green Tea : 3.46**  **Corrections**  **-----------**  **Product : Black Tea**  **Price : 5.29**  **Product : quit**  **Price List**  **----------**  **Pizza : 4.49**  **Pierogi : 2.56**  **Potato Chips : 2.29**  **Black Tea : 5.29**  **Green Tea : 3.46** | **Glossary**  **========**  **Key : CPU**  **Definition : central processing unit**  **Key : ALU**  **Definition : arithmetic logic unit**  **Key : quit**  **Entries**  **-------**  **CPU : central processing unit**  **ALU : arithmetic logic unit** |

Note that the input data is only stored for the first N items, which is the size specified in the definitions of **inventory**and **glossary**.

**SUBMISSION**

Follow your professor's submission instructions