C++ STL – Lesson 4

Data Structure

1 Tic-Tac-Toe

[STL/04_data_structure/task01_tic_tac_toe.cpp]

Разработать игру крестики-нолики. Соперником является компьютер.

Пользователь вводит через пробел номер строки и столбца. В данной ячейке ставится крестик. Компьютер ставит нолик в произвольную свободную клетку.

2 Set

[STL/04_data_structure/task02_set.cpp]

Разработать обобщенный (шаблонный) класс множество. Множество содержит набор уникальных элементов, т.е. в множестве не может встретится два и более элементов с одинаковым значением. Например, набор {1, 2, 3, 4, 5} является множеством, а набор {1, 2, 3, 2, 4} не является. Примерный интерфейс класса:

`add(value)` – добавляет элемент со значением `value` в множество. Если элемент уже существует, ничего не происходит

`exist(value)` – проверяет наличие элемента `value` в множестве, возвращает `true`, если элемент присутствует

`remove(value)` – удаляет значение `value` из множества

3 List

[STL/04_data_structure/task03_list.cpp]

Разработать обобщенный (шаблонный) класс список `List<T>`. Примерный интерфейс класса:

`T value_` – поле типа `T`, хранящее текущий элемент списка

`List<T>* next_item_` - поле, хранящее указатель на следующий элемент списка, если такой еще не добавлен, то хранит значение `std::nullptr`

`void append(T value)` – добавляет новый элемент в конец списка и присваивает ему значение `value`

`T get()` – возвращает значение элемента списка

`set(T value)` – устанавливает значение для элемента списка

`List<T>* next()` – возвращает следующий элемент списка

`remove(List<T>* item)` – удаляет элемент `item` из списка

Создайте список фамилий. Добавьте пять фамилий. Измените вторую фамилию и удалите третью фамилию.

4 Map

[STL/04_data_structure/task04_map.cpp]

Разработать класс «англо-русский словарь», реализующий следующий интерфейс:

`add(word, translate)` – добавляет слово `word` и его перевод `translate` в словарь. Если слово word уже имеется в словаре, генерируется исключение

`update(word, translate)` – изменяет перевод `translate` для слова `word`. Если слово word не существует, генерируется исключение

`exist(word)` – проверяет наличие слова word в словаре, возвращает true, если слово существует

`at(word)` – возвращает перевод translate, если слово word в словаре отсутствует генерируется исключение

`remove(word)` – удаляет слово и его перевод из словаря

Создать объект типа словарь, добавить несколько слов. Вывести в цикле все слова из словаря и соответствующий им перевод. Здесь вы столкнетесь с неразрешимой проблемой. Решите её. Постарайтесь, чтобы в цикле не фигурировали целочисленные индексы. Реализуйте методы типа `begin(), next(), curr(), end()` для итерирования по словарю.

Попробуйте сделать тип обобщенным. Тогда станет возможным использовать словарь не по назначению. Например, для сохранения сведений об успеваемости. В качестве ключа будет использоваться имя типа `string`, а в качестве значения средний балл `double`.

Пример:

```
    <template class TKey, class TValue> Dict { };
    Dict<string, double> academicPerformance;
    academicPerformance.add(“Petrov”, 4.5);
```

Подсказка: создайте структуру пар ключ-значение `Pair<TKey,TValue>{}`, храните в словаре массив `Pair[]`