Министерство образования и науки Российской Федерации
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»
Факультет компьютерных технологий и информатики
Кафедра вычислительной техники

Отчёт по лабораторной работе №6
на тему
«Стандартная библиотека шаблонов»
по дисциплине
«Алгоритмы и структуры данных»

Вариант 18

Выполнили студенты
факультета КТИ группы №3305
Лоуцкер Алексей и Григорьева Анна

Проверил старший преподаватель
Колинько Павел Георгиевич

Санкт-Петербург
2015

Оглавление

1. Задание.....3
2. Описание структуры данных.....4
2. Тестовый пример.....5
3. Оценки временной сложности операций.....6
4. Выводы.....7
5. Список использованной литературы.....7
6. Приложение: исходные тексты.....7

1. Задание

Переделать программу, составленную по теме «Последовательности», под использование контейнеров из стандартной библиотеки шаблонов. Использовать возможности библиотеки алгоритмов. Программа должна реализовывать цепочку операций над множествами в соответствии с заданием по теме 2 и операций с последовательностями — о теме 3.

Операции над последовательностями

1. Слияние (merge)
2. Включение (subst)
3. Замена (change)

2. Использованные контейнеры и функции STL

std::multiset<int> data - контейнер, хранящий элементы в виде дерева двоичного поиска; может содержать элементы с одинаковым ключом.

std::vector<std::multiset<int>::iterator> sequence — вектор итераторов на элементы data. Позволяет поддерживать произвольную последовательность.

std::initializer_list<int> args — специальный тип для инициализации контейнеров

std::default_random_engine generator – линейно-конгруэнтный генератор случайных чисел

xx::iterator – объект-указатель на элемент контейнера

xx.begin() - возвращает итератор на первый элемент контейнера

xx.end() - итератор на *past-the-end* element

sequence.push_back() – вставка в конец вектора

data.insert() – вставка в multiset

xx.size() - возвращает количество элементов в контейнере

sequence.resize() - изменить размер

data.erase() – удаление элемента из контейнера

data.find() - поиск элемента

std::set_union(), std::set_intersection(), std::set_difference() – соответственно объединение, пересечение, вычитание элементов контейнера.

xx.clear() - удаление всех элементов из контейнера

std::merge() – слияние отсортированных последовательностей.

3. Тестовый пример

```
{ 1 2 3 4 5 }
{ 7 8 9 }
{ 0 0 0 0 0 }

{ 1 2 3 4 5 7 8 9 }
{ 7 8 9 1 2 3 4 5 7 8 9 }
{ 7 8 9 0 0 0 0 0 7 8 9 }

ВЫПОЛНЕНИЕ FINISHED; значение выхода 0,; в реальном времени: 10ms;
```

4. Оценки временной сложности операций

Вектор итераторов sequence обеспечивает доступ к элементу за константное время

Однако, необходимость поддерживать эту последовательность увеличивает время вставки/удаления с логарифмического до линейного.

Все двухместные операции в худшем случае сводятся к последовательной вставке элементов в бинарное дерево, поэтому их сложность пропорциональна n*log(n)

Операция	Сложность
доступ по позиции	O(1)
вставка, удаление	O(n)
пересечение, объединение, вычитание, слияние, подстановка, замена	O(n*log(n))

5. Выводы

Стандартная библиотека шаблонов позволяет легко хранить и выполнять различные действия над наборами элементов любого типа; имеет хорошо спроектированный интерфейс и алгоритмы с наилучшей оценкой сложности.

6. Список использованной литературы

1) Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Часть 2 / сост. П. Г. Колинько. - СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2015

2) http://cplusplus.com

6. Приложение: исходные тексты

Исходный код доступен в репозитарии по адресу:

github.com/alout1/avl-tree-mk2