Лабораторная работа № 12

Цель работы

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Использование ассоциативных контейнеров библиотеки STL в ОО программе.

Постановка задачи

Задача 1.

1. Создать ассоциативный контейнер - set
2. Заполнить его элементами стандартного типа (int).
3. Найти максимальный элемент и добавить его в конец контейнера.
4. Найти элемент с заданным ключом и удалить его из контейнера.
5. К каждому элементу добавить среднее арифметическое элементов контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 2.

1. Создать ассоциативный контейнер.
2. Заполнить его элементами пользовательского типа (Money). Для пользовательского типа перегрузить необходимые операции.
3. Найти максимальный элемент и добавить его в конец контейнера.
4. Найти элемент с заданным ключом и удалить его из контейнера.
5. К каждому элементу добавить среднее арифметическое элементов контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 3

1. Создать параметризированный класс Вектор, используя в качестве контейнера ассоциативный контейнер.
2. Заполнить его элементами.
3. Найти максимальный элемент и добавить его в конец контейнера.
4. Найти элемент с заданным ключом и удалить его из контейнера.
5. К каждому элементу добавить среднее арифметическое элементов контейнера.
6. Выполнение всех заданий оформить в виде методов параметризированного класса.

Описание класса

Для выполнения работы были использованы следующие классы и структуры данных:

1. Класс Money (из предыдущих лабораторных работ):

- Хранит денежную сумму в рублях и копейках
- Имеет перегруженные операторы ввода/вывода и сравнения
- Реализует основные арифметические операции

- Поддерживает необходимые операции для работы с ассоциативными контейнерами

2. Шаблонный класс Vector:

- Реализует обертку над ассоциативными контейнерами
- Поддерживает основные операции работы с элементами
- Имеет методы для выполнения заданий лабораторной работы

Определение компонентных функций

Для класса Money были реализованы следующие ключевые методы:

```
// Конструкторы
Money();
Money(long rub, int cheers);
Money(const Money& m);

// Операторы сравнения (необходимы для set)
bool operator<(const Money& m) const;
bool operator==(const Money& m) const;

// Арифметические операции
Money operator+(const Money& m) const;
Money operator/(int n) const; // Для вычисления среднего

// Ввод/вывод
friend istream& operator>>(istream& in, Money& m);
friend ostream& operator<<(ostream& out, const Money& m);
```

Для шаблонного класса Vector:

```
template<typename T>
class Vector {
private:
    set<T> elements;

public:
    // Основные методы
    void insert(const T& value);
    void erase(const T& value);
    void print() const;

    // Методы для заданий
    void addMaxToEnd();
    void removeByKey(const T& key);
    void addAverageToElements();
};
```

Определение глобальных функций

Были реализованы следующие глобальные функции для работы с контейнерами:

```
// Для set<int>
void fillSet(set<int>& s);
void printSet(const set<int>& s);
void processSet(set<int>& s);

// Для set<Money>
void fillMoneySet(set<Money>& s);
void printMoneySet(const set<Money>& s);
void processMoneySet(set<Money>& s);

// Для map<int, string>
void fillMap(map<int, string>& m);
void printMap(const map<int, string>& m);
```

Функция main()

Основная программа демонстрирует выполнение всех задач:

```
int main() {
    // Задача 1 - работа с set<int>
    set<int> intSet;
    fillSet(intSet);
    processSet(intSet);

    // Задача 2 - работа с set<Money>
    set<Money> moneySet;
    fillMoneySet(moneySet);
    processMoneySet(moneySet);

    // Задача 3 - параметризированный Vector
    Vector<int> intVector;
    // ... операции с вектором

    Vector<Money> moneyVector;
    // ... операции с вектором

    return 0;
}
```

Объяснение результатов работы программы

Программа успешно демонстрирует:

1. Работу с ассоциативными контейнерами STL:

- Автоматическую сортировку элементов в set
- Быстрый поиск элементов (O(log n))

- Особенности работы с пользовательскими типами

2. Особенности работы с контейнером set:

- Уникальность элементов
- Неизменяемость ключей
- Эффективные операции поиска и вставки

3. Применение шаблонного класса Vector:

- Инкапсуляцию работы с set
- Реализацию требуемых операций (добавление максимального элемента и др.)

Ответы на контрольные вопросы

1. Что представляет собой ассоциативный контейнер?

Ответ: Ассоциативный контейнер - это контейнер, который хранит элементы в виде пар "ключ-значение" или просто ключей, обеспечивая быстрый поиск по ключу. Элементы упорядочены по ключу согласно заданному критерию сравнения.

2. Перечислить ассоциативные контейнеры библиотеки STL.

Ответ:

- set - множество уникальных ключей
- multiset - множество с возможностью дублирования ключей
- map - ассоциативный массив уникальных ключей
- multimap - ассоциативный массив с возможностью дублирования ключей
- unordered_set - неупорядоченное множество (хеш-таблица)
- unordered_map - неупорядоченный ассоциативный массив

3. Каким образом можно получить доступ к элементам ассоциативного контейнера?

Ответ:

- Через итераторы (begin(), end(), find())
- Для map - оператор [] или метод at()
- Методы find(), lower_bound(), upper_bound() для поиска
- Для set - только через итераторы, так как элементы являются ключами

4. Привести примеры методов, используемых в ассоциативных контейнерах.

Ответ:

- insert() - добавление элемента
- erase() - удаление элемента
- find() - поиск элемента
- count() - подсчет элементов с заданным ключом
- lower_bound(), upper_bound() - поиск границ диапазона
- size(), empty() - информация о размере

5. Каким образом можно создать контейнер map? Привести примеры.

Ответ:

```
// Пустой map с ключами int и значениями string
map<int, string> m1;

// Map с начальной инициализацией
map<string, double> m2 = {
    {"pi", 3.14},
    {"e", 2.71}
};

// Map с пользовательским компаратором
struct CaseInsensitiveCompare {
    bool operator()(const string& a, const string& b) const {
        return lexicographical_compare(
            a.begin(), a.end(),
            b.begin(), b.end(),
            [](char c1, char c2) {
                return tolower(c1) < tolower(c2);
            });
    }
};
map<string, int, CaseInsensitiveCompare> m3;
```

6. Каким образом упорядочены элементы в контейнере map по умолчанию? Как изменить порядок на обратный?

Ответ: По умолчанию элементы map упорядочены по возрастанию ключей с использованием оператора <. Для изменения порядка на обратный можно использовать:

```
map<int, string, greater<int>> reverseMap;
```

Или определить собственный функтор сравнения.

7. Какие операции определены для контейнера map?

Ответ:

- Вставка: insert(), emplace(), operator[]
- Доступ: at(), operator[], find()
- Удаление: erase(), clear()
- Обход: итераторы begin(), end()
- Информация: size(), empty(), count()
- Сравнение: ==, !=, <, > и др.

8. Написать функцию для добавления элементов в контейнер map с помощью функции make_pair().

Ответ:

```
void addToMap(map<int, string>& m, int key, const string& value) {
    m.insert(make_pair(key, value));
}
```

9. Написать функцию для добавления элементов в контейнер map с помощью функции операции прямого доступа [].

Ответ:

```
void addToMap(map<int, string>& m, int key, const string& value) {
    m[key] = value;
}
```

10. Написать функцию для печати контейнера map с помощью итератора.

Ответ:

```
void printMap(const map<int, string>& m) {
    for (auto it = m.begin(); it != m.end(); ++it) {
        cout << "Key: " << it->first << ", Value: " << it->second << endl;
    }
}
```

11. Написать функцию для печати контейнера map с помощью функции операции прямого доступа [].

Ответ: Это неэффективно, так как требует поиска каждого элемента, но возможно:

```
void printMap(const map<int, string>& m) {
    for (auto it = m.begin(); it != m.end(); ++it) {
        cout << "Key: " << it->first << ", Value: " << it->second << endl;
    }
}
```

12. Чем отличаются контейнеры map и multimap?

Ответ:

- map хранит уникальные ключи (один ключ - одно значение)

- multimap позволяет хранить несколько значений с одинаковыми ключами
- В map есть operator[], в multimap его нет
- Метод insert() в map возвращает pair<iterator, bool>, в multimap - просто iterator

13. Что представляет собой контейнер set?

Ответ: set - это ассоциативный контейнер, содержащий уникальные элементы (ключи), упорядоченные по возрастанию. Поддерживает быстрый поиск, вставку и удаление элементов (O(log n)).

14. Чем отличаются контейнеры map и set?

Ответ:

- map хранит пары ключ-значение
- set хранит только ключи
- В map есть доступ к значению по ключу (operator[]), в set такого доступа нет
- Оба контейнера обеспечивают уникальность ключей и быстрый поиск

15. Каким образом можно создать контейнер set? Привести примеры.

Ответ:

```
// Пустой set с элементами int
set<int> s1;

// Set с начальной инициализацией
set<string> s2 = {"apple", "banana", "orange"};

// Set с пользовательским компаратором
set<Money, greater<Money>> s3; // Упорядочивание по убыванию
```

16. Каким образом упорядочены элементы в контейнере set по умолчанию? Как изменить порядок на обратный?

Ответ: По умолчанию элементы set упорядочены по возрастанию с использованием оператора <. Для изменения порядка:

```
set<int, greater<int>> reverseSet;
```

17. Какие операции определены для контейнера set?

Ответ:

- Вставка: insert(), emplace()
- Удаление: erase(), clear()
- Поиск: find(), count(), lower_bound(), upper_bound()
- Обход: итераторы begin(), end()

- Информация: size(), empty()
- Сравнение: ==, !=, <, > и др.

18. Написать функцию для добавления элементов в контейнер set.

Ответ:

```
template<typename T>
void addToSet(set<T>& s, const T& value) {
    s.insert(value);
}
```

19. Написать функцию для печати контейнера set.

Ответ:

```
template<typename T>
void printSet(const set<T>& s) {
    for (const auto& elem : s) {
        cout << elem << " ";
    }
    cout << endl;
}
```

20. Чем отличаются контейнеры set и multiset?

Ответ:

- set хранит только уникальные элементы
- multiset позволяет хранить дубликаты элементов
- Метод count() в set возвращает 0 или 1, в multiset - количество вхождений
- В set insert() возвращает pair<iterator, bool>, в multiset - просто iterator