Пермский Национальный Исследовательский  
Политехнический Университет

**Лабораторная работа №12**

Основы алгоритмизации и программирования

за 2 семестр

Вариант № 1

Выполнил:

Студент группы РИС 20-1-бз

Курганов Н.В.

20-ЭТФ-631

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

2022

Ассоциативные контейнеры библиотеки STL

1. **Цель задания**.

1.1 Создание консольного приложения, состоящего из нескольких файлов в системе программирования VisualStudio.

1.2. Использование ассоциативных контейнеров библиотеки STL в ОО программе

1. **Задание.**

Задача №1

* 1. Контейнер – multiamp
  2. Тип элементов – double

Задача №2

* 1. Тип элементов Time (см. лабораторную работу №3).

Задача №3

3.1 Параметризированный класс – Вектор

Задача №4

Задание №3

Найти максимальный элемент т добавить его в начало контейнера.

Задание№4

Найти минимальный элемент и удалить его из контейнера.

Задание №5

К каждому элементу добавить среднее арифметическое контейнера

1. **Создаем:**

**Задание №1**

**Файл lab12\_1\_main.cpp**

#include <iostream>

#include <map>

#include <string>

typedef std::multimap<int, double> MyContainer;

void print(MyContainer&, const std::string&);

void fill(MyContainer&);

void insert\_max(MyContainer&);

void erase\_min(MyContainer&);

void add\_average(MyContainer&);

int main()

{

MyContainer vec;

fill(vec);

print(vec, "Initial container: ");

insert\_max(vec);

print(vec, "After replace max: ");

erase\_min(vec);

print(vec, "After erase min: ");

add\_average(vec);

print(vec, "After add average: ");

std::cin.ignore();

std::cin.get();

return 0;

}

void print(MyContainer& c, const std::string& message)

{

std::cout << message << "\n";

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

std::cout << "\t" << it->first << ": " << it->second << "\n";

std::cout << "\n\n";

}

void fill(MyContainer& c)

{

std::cout << "Enter number of elements: ";

int n;

std::cin >> n;

for (int i = 0; i < n; ++i)

{

std::cout << "Element #" << i + 1 << ": ";

double t;

std::cin >> t;

c.emplace(i, t);

}

}

void insert\_max(MyContainer& c)

{

MyContainer::iterator max = c.begin();

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

{

if (max->second < it->second)

max = it;

}

std::cout << "Max element: " << max->second << "\n";

c.insert(\*max);

}

void erase\_min(MyContainer& c)

{

MyContainer::iterator min = c.begin();

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

{

if (it->second < min->second)

min = it;

}

std::cout << "Min element: " << min->second << "\n";

c.erase(min);

}

void add\_average(MyContainer& c)

{

double avg = 0.0;

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

avg += it->second;

avg /= c.size();

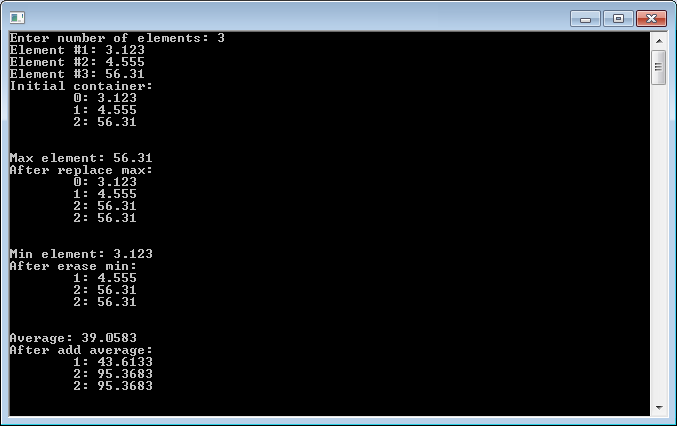
std::cout << "Average: " << avg << "\n";

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

it->second += avg;

}

**Результат работы программы**



**Задание №2**

**Файл lab12\_2\_main.cpp**

#include <iostream>

#include <map>

#include <string>

#include "time\_class.h"

typedef std::multimap<int, Time> MyContainer;

void print(MyContainer&, const std::string&);

void fill(MyContainer&);

void insert\_max(MyContainer&);

void erase\_min(MyContainer&);

void add\_average(MyContainer&);

int main()

{

MyContainer vec;

fill(vec);

print(vec, "Initial container: ");

insert\_max(vec);

print(vec, "After replace max: ");

erase\_min(vec);

print(vec, "After erase min: ");

add\_average(vec);

print(vec, "After add average: ");

std::cin.ignore();

std::cin.get();

return 0;

}

void print(MyContainer& c, const std::string& message)

{

std::cout << message << "\n";

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

std::cout << "\t" << it->first << ": " << it->second << "\n";

std::cout << "\n\n";

}

void fill(MyContainer& c)

{

std::cout << "Enter number of elements: ";

int n;

std::cin >> n;

for (int i = 0; i < n; ++i)

{

std::cout << "Element #" << i + 1 << ":\n";

Time t;

std::cin >> t;

c.emplace(i, t);

}

}

void insert\_max(MyContainer& c)

{

MyContainer::iterator max = c.begin();

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

{

if (max->second < it->second)

max = it;

}

std::cout << "Max element: " << max->second << "\n";

c.insert(\*max);

}

void erase\_min(MyContainer& c)

{

MyContainer::iterator min = c.begin();

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

{

if (it->second < min->second)

min = it;

}

std::cout << "Min element: " << min->second << "\n";

c.erase(min);

}

void add\_average(MyContainer& c)

{

Time avg;

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

avg += it->second;

avg = avg / c.size();

std::cout << "Average: " << avg << "\n";

for (MyContainer::iterator it = c.begin(); it != c.end(); ++it)

it->second += avg;

}

**Файл time\_class.cpp**

#include <iostream>

#include <iomanip>

#include <fstream>

#include "time\_class.h"

Time::Time() : minutes(0), seconds(0) {}

Time::Time(int m, int s) : minutes(m + s / 60), seconds(s % 60) {}

Time::Time(const Time& t) : minutes(t.minutes), seconds(t.seconds) {}

Time Time::operator=(const Time& t)

{

minutes = t.minutes;

seconds = t.seconds;

return \*this;

}

Time& Time::operator+=(const Time& t)

{

\*this = Time(minutes + t.minutes, seconds + t.seconds);

return \*this;

}

std::ostream& operator<<(std::ostream& out, const Time& t)

{

return out << std::setfill('0') << std::setw(2) << t.minutes << ":"

<< std::setw(2) << t.seconds << std::setfill(' ');

}

std::istream& operator>>(std::istream& in, Time & t)

{

int m, s;

std::cout << "\tMinutes: "; in >> m;

std::cout << "\tSeconds: "; in >> s;

t = Time(m, s);

return in;

}

std::ofstream& operator<<(std::ofstream& out, const Time& t)

{

out << t.minutes << " " << t.seconds << "\n";

return out;

}

std::ifstream& operator>>(std::ifstream& in, Time& t)

{

in >> t.minutes >> t.seconds;

return in;

}

bool operator==(const Time& t1, const Time& t2)

{

return (t1.minutes == t2.minutes) && (t1.seconds == t2.seconds);

}

bool operator!=(const Time& t1, const Time& t2)

{

return !(t1 == t2);

}

bool operator<(const Time& t1, const Time& t2)

{

return (t1.minutes < t2.minutes) ||

((t1.minutes == t2.minutes) && (t1.seconds < t2.seconds));

}

bool operator>(const Time& t1, const Time& t2)

{

return !(t1 < t2) && (t1 != t2);

}

Time operator+(const Time& t1, const Time& t2)

{

return Time(t1.minutes + t2.minutes, t1.seconds + t2.seconds);

}

Time operator/(const Time& t, int x)

{

int secs = (t.minutes \* 60 + t.seconds) / x;

return Time(0, secs);

}

**Файл time\_class.h**

#ifndef TIME\_CLASS\_H

#define TIME\_CLASS\_H

#include <iosfwd>

class Time

{

friend std::ostream& operator<<(std::ostream&, const Time&);

friend std::istream& operator>>(std::istream&, Time&);

friend std::ofstream& operator<<(std::ofstream&, const Time&);

friend std::ifstream& operator>>(std::ifstream&, Time&);

friend bool operator==(const Time&, const Time&);

friend bool operator!=(const Time&, const Time&);

friend bool operator<(const Time&, const Time&);

friend bool operator>(const Time&, const Time&);

friend Time operator+(const Time&, const Time&);

friend Time operator/(const Time&, int);

public:

Time();

Time(int, int);

Time(const Time&);

Time operator=(const Time&);

Time& operator+=(const Time&);

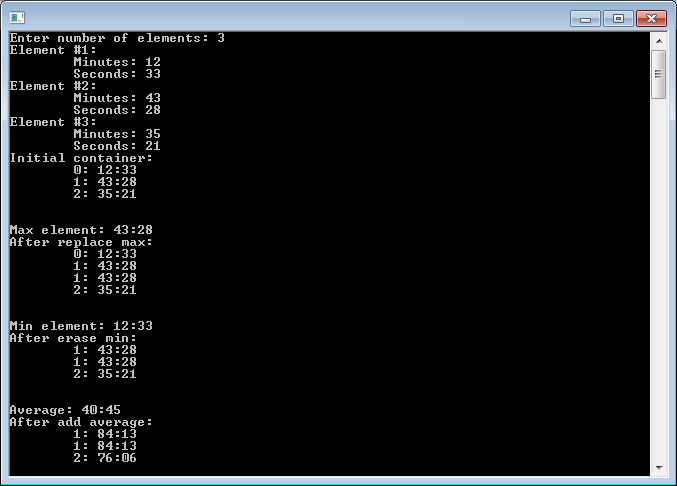
private:

int minutes, seconds;

};

#endif

**Результат выполнения программы**



**Задание №3**

**Файл lab12\_3\_main.cpp**

#include <iostream>

#include <string>

#include "time\_class.h"

#include "my\_vector\_12\_3.h"

typedef MyVector<Time> MyContainer;

int main()

{

std::cout << "Enter number of elements: ";

int n;

std::cin >> n;

MyContainer vec(n);

vec.print("Initial container: ");

vec.insert\_max();

vec.print("After replace max: ");

vec.erase\_min();

vec.print("After erase min: ");

vec.inc\_average();

vec.print("After add average: ");

std::cin.ignore();

std::cin.get();

return 0;

}

**Файл my\_vector\_12\_3.h**

#ifndef MY\_VECTOR\_H

#define MY\_VECTOR\_H

#include <map>

#include <string>

#include <iostream>

template <typename T>

class MyVector

{

public:

MyVector() {}

MyVector(int n);

void print(const std::string & message) const;

void insert\_max();

void erase\_min();

void inc\_average();

private:

std::multimap<int, T> vec;

};

template <typename T>

MyVector<T>::MyVector(int n)

{

for (int i = 0; i < n; ++i) {

std::cout << "Element #" << i + 1 << ":\n";

T el;

std::cin >> el;

vec.emplace(i, el);

}

}

template <typename T>

void MyVector<T>::print(const std::string & message) const

{

std::cout << message << "\n";

for (std::multimap<int, T>::const\_iterator it = vec.begin(); it != vec.end(); ++it)

std::cout << "\t" << it->first << ": " << it->second << "\n";

std::cout << "\n\n";

}

template <typename T>

void MyVector<T>::insert\_max()

{

std::multimap<int, T>::iterator max = vec.begin();

for (std::multimap<int, T>::iterator it = vec.begin(); it != vec.end(); ++it)

{

if (max->second < it->second)

max = it;

}

std::cout << "Max element: " << max->second << "\n";

vec.insert(\*max);

}

template <typename T>

void MyVector<T>::erase\_min()

{

std::multimap<int, T>::iterator min = vec.begin();

for (std::multimap<int, T>::iterator it = vec.begin(); it != vec.end(); ++it)

{

if (it->second < min->second)

min = it;

}

std::cout << "Min element: " << min->second << "\n";

vec.erase(min);

}

template <typename T>

void MyVector<T>::inc\_average()

{

T avg;

for (std::multimap<int, T>::iterator it = vec.begin(); it != vec.end(); ++it)

avg += it->second;

avg = avg / vec.size();

std::cout << "Average: " << avg << "\n";

for (std::multimap<int, T>::iterator it = vec.begin(); it != vec.end(); ++it)

it->second += avg;

}

#endif

**Файл time\_class.cpp**

#include <iostream>

#include <iomanip>

#include <fstream>

#include "time\_class.h"

Time::Time() : minutes(0), seconds(0) {}

Time::Time(int m, int s) : minutes(m + s / 60), seconds(s % 60) {}

Time::Time(const Time& t) : minutes(t.minutes), seconds(t.seconds) {}

Time Time::operator=(const Time& t)

{

minutes = t.minutes;

seconds = t.seconds;

return \*this;

}

Time& Time::operator+=(const Time& t)

{

\*this = Time(minutes + t.minutes, seconds + t.seconds);

return \*this;

}

std::ostream& operator<<(std::ostream& out, const Time& t)

{

return out << std::setfill('0') << std::setw(2) << t.minutes << ":"

<< std::setw(2) << t.seconds << std::setfill(' ');

}

std::istream& operator>>(std::istream& in, Time & t)

{

int m, s;

std::cout << "\tMinutes: "; in >> m;

std::cout << "\tSeconds: "; in >> s;

t = Time(m, s);

return in;

}

std::ofstream& operator<<(std::ofstream& out, const Time& t)

{

out << t.minutes << " " << t.seconds << "\n";

return out;

}

std::ifstream& operator>>(std::ifstream& in, Time& t)

{

in >> t.minutes >> t.seconds;

return in;

}

bool operator==(const Time& t1, const Time& t2)

{

return (t1.minutes == t2.minutes) && (t1.seconds == t2.seconds);

}

bool operator!=(const Time& t1, const Time& t2)

{

return !(t1 == t2);

}

bool operator<(const Time& t1, const Time& t2)

{

return (t1.minutes < t2.minutes) ||

((t1.minutes == t2.minutes) && (t1.seconds < t2.seconds));

}

bool operator>(const Time& t1, const Time& t2)

{

return !(t1 < t2) && (t1 != t2);

}

Time operator+(const Time& t1, const Time& t2)

{

return Time(t1.minutes + t2.minutes, t1.seconds + t2.seconds);

}

Time operator/(const Time& t, int x)

{

int secs = (t.minutes \* 60 + t.seconds) / x;

return Time(0, secs);

}

**Файл time\_class.h**

#ifndef TIME\_CLASS\_H

#define TIME\_CLASS\_H

#include <iosfwd>

class Time

{

friend std::ostream& operator<<(std::ostream&, const Time&);

friend std::istream& operator>>(std::istream&, Time&);

friend std::ofstream& operator<<(std::ofstream&, const Time&);

friend std::ifstream& operator>>(std::ifstream&, Time&);

friend bool operator==(const Time&, const Time&);

friend bool operator!=(const Time&, const Time&);

friend bool operator<(const Time&, const Time&);

friend bool operator>(const Time&, const Time&);

friend Time operator+(const Time&, const Time&);

friend Time operator/(const Time&, int);

public:

Time();

Time(int, int);

Time(const Time&);

Time operator=(const Time&);

Time& operator+=(const Time&);

private:

int minutes, seconds;

};

#endif

**Результат работы программы**

