Пермский Национальный Исследовательский  
Политехнический Университет

**Лабораторная работа №11**

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

Последовательные контейнеры библиотеки STL

1. **Цель задания**.

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

1.2. Использование последовательных контейнеров STL в ОО программе

1. **Задание.**

Задача №1

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

Задача №2

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

Задача №3

3.1 Параметризированный класс – Вектор

Задача №4

4.1 Адаптер контейнера – стек.

Задача №5

5.1 Параметризированный класс – Вектор

5.2 Адаптер контейнера – стек.

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 Lab11\_1\_main.cpp**

#include <vector>

#include <iostream>

#include <iomanip>

#include <random>

#include <string>

typedef std::vector<double> my\_vector;

void print\_vector(my\_vector & vec, const std::string & message);

void fill\_vector(my\_vector & vec);

void add\_max\_to\_begin(my\_vector & vec);

void erase\_min(my\_vector & vec);

void inc\_average(my\_vector & vec);

int main()

{

my\_vector vec;

fill\_vector(vec);

print\_vector(vec, "\nInitial vector: ");

add\_max\_to\_begin(vec);

print\_vector(vec, "\nAfter add max to begin: ");

erase\_min(vec);

print\_vector(vec, "\nAfter erase min: ");

inc\_average(vec);

print\_vector(vec, "\nAfter inc all element on average: ");

std::cin.ignore();

std::cin.get();

return 0;

}

void print\_vector(my\_vector & vec, const std::string & message)

{

std::cout << message << "\n" << std::setprecision(3) << std::fixed;

for (size\_t i = 0; i < vec.size(); ++i)

std::cout << vec[i] << " ";

std::cout << "\n";

}

void fill\_vector(my\_vector & vec)

{

std::cout << "Enter count elements: ";

int n;

std::cin >> n;

std::random\_device rd;

std::uniform\_real\_distribution<double> distr(0, 1);

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

vec.push\_back(distr(rd));

}

void add\_max\_to\_begin(my\_vector & vec)

{

double max = vec[0];

for (size\_t i = 0; i < vec.size(); ++i)

{

if (vec[i] > max)

max = vec[i];

}

vec.insert(vec.begin(), max);

}

void erase\_min(my\_vector & vec)

{

my\_vector::iterator min = vec.begin();

for (my\_vector::iterator it = vec.begin(); it != vec.end(); ++it)

{

if (\*it < \*min)

min = it;

}

vec.erase(min);

}

void inc\_average(my\_vector & vec)

{

double avg = 0;

for (size\_t i = 0; i < vec.size(); ++i)

avg += vec[i];

avg /= vec.size();

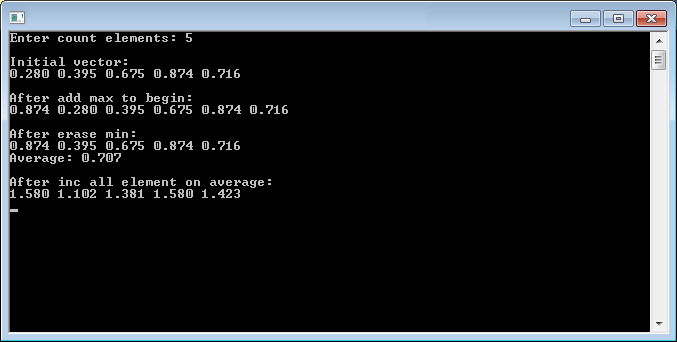
std::cout << "Average: " << avg << "\n";

for (size\_t i = 0; i < vec.size(); ++i)

vec[i] += avg;

}

**Результат выполнения программы**



**Задание №2**

**Файл Lab11\_2\_main.cpp**

#include <vector>

#include <iostream>

#include <iomanip>

#include <random>

#include <string>

#include "time\_class.h"

typedef std::vector<Time> my\_vector;

void print\_vector(my\_vector & vec, const std::string & message);

void fill\_vector(my\_vector & vec);

void add\_max\_to\_begin(my\_vector & vec);

void erase\_min(my\_vector & vec);

void inc\_average(my\_vector & vec);

int main()

{

my\_vector vec;

fill\_vector(vec);

print\_vector(vec, "Initial vector: ");

add\_max\_to\_begin(vec);

print\_vector(vec, "After add max to begin: ");

erase\_min(vec);

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

inc\_average(vec);

print\_vector(vec, "After inc all element on average: ");

std::cin.ignore();

std::cin.get();

return 0;

}

void print\_vector(my\_vector & vec, const std::string & message)

{

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

for (size\_t i = 0; i < vec.size(); ++i)

std::cout << vec[i] << " ";

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

}

void fill\_vector(my\_vector & vec)

{

std::cout << "Enter count elements: ";

int n;

std::cin >> n;

std::random\_device rd;

std::uniform\_int\_distribution<int> distr(0, 59);

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

vec.push\_back(Time(distr(rd), distr(rd)));

}

void add\_max\_to\_begin(my\_vector & vec)

{

Time max = vec[0];

for (size\_t i = 0; i < vec.size(); ++i)

{

if (vec[i] > max)

max = vec[i];

}

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

vec.insert(vec.begin(), max);

}

void erase\_min(my\_vector & vec)

{

my\_vector::iterator min = vec.begin();

for (my\_vector::iterator it = vec.begin(); it != vec.end(); ++it)

{

if (\*it < \*min)

min = it;

}

std::cout << "Min: " << \*min << "\n";

vec.erase(min);

}

void inc\_average(my\_vector & vec)

{

Time avg;

for (size\_t i = 0; i < vec.size(); ++i)

avg += vec[i];

avg = avg / vec.size();

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

for (size\_t i = 0; i < vec.size(); ++i)

vec[i] += 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&);

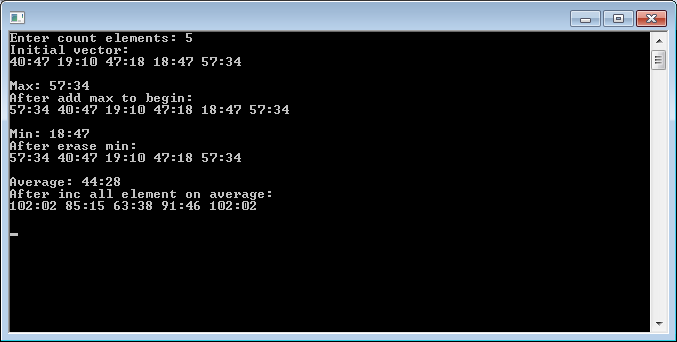
private:

int minutes, seconds;

};

#endif

**Результат выполнения программы**



**Задание №3**

**Файл Lab11\_3\_main.cpp**

#include <iostream>

#include "my\_vector\_11\_3.h"

#include "time\_class.h"

int main()

{

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

int n;

std::cin >> n;

MyVector<Time> vec(n);

vec.print("Initial vector: ");

vec.add\_max\_to\_begin();

vec.print("After add max to begin: ");

vec.erase\_min();

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

vec.inc\_average();

vec.print("After inc all element on average: ");

std::cin.ignore();

std::cin.get();

return 0;

}

**Файл my\_vector\_11\_3.h**

#ifndef MY\_VECTOR\_H

#define MY\_VECTOR\_H

#include <vector>

#include <string>

#include <iostream>

#include <algorithm>

#include <iterator>

#include <numeric>

template <typename T>

class MyVector

{

public:

MyVector() {}

MyVector(int n);

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

void add\_max\_to\_begin();

void erase\_min();

void inc\_average();

private:

std::vector<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.push\_back(el);

}

}

template <typename T>

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

{

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

for (size\_t i = 0; i < vec.size(); ++i)

std::cout << vec[i] << " ";

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

}

template <typename T>

void MyVector<T>::add\_max\_to\_begin()

{

T max = \*std::max\_element(vec.begin(), vec.end());

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

vec.insert(vec.begin(), max);

}

template <typename T>

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

{

std::vector<T>::const\_iterator min = std::min\_element(vec.begin(), vec.end());

std::cout << "Min: " << \*min << "\n";

vec.erase(min);

}

template <typename T>

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

{

T avg = T();

for (size\_t i = 0; i < vec.size(); ++i)

avg += vec[i];

avg = avg / vec.size();

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

for (size\_t i = 0; i < vec.size(); ++i)

vec[i] += 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&);

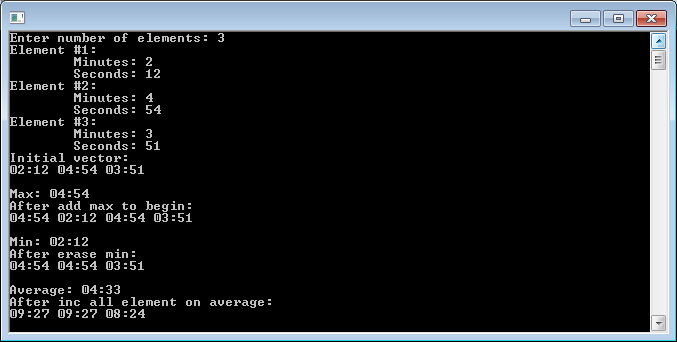
private:

int minutes, seconds;

};

#endif

**Результат выполнения программы**



**Задание №4**

**Файл Lab11\_4\_main.cpp**

#include <iostream>

#include <stack>

#include <algorithm>

#include <iterator>

#include <string>

#include "time\_class.h"

typedef std::stack<Time> MyStack;

void print\_stack(MyStack & vec, const std::string & message);

void fill\_stack(MyStack & vec);

void add\_max\_to\_begin(MyStack & vec);

void erase\_min(MyStack & vec);

void inc\_average(MyStack & vec);

int main()

{

MyStack vec;

fill\_stack(vec);

print\_stack(vec, "Initial vector: ");

add\_max\_to\_begin(vec);

print\_stack(vec, "After add max to begin: ");

erase\_min(vec);

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

inc\_average(vec);

print\_stack(vec, "After inc all element on average: ");

std::cin.ignore();

std::cin.get();

return 0;

}

void print\_stack(MyStack & vec, const std::string & message)

{

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

MyStack temp;

while (!vec.empty()) {

temp.push(vec.top());

vec.pop();

}

while (!temp.empty()) {

std::cout << temp.top() << " ";

vec.push(temp.top());

temp.pop();

}

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

}

void fill\_stack(MyStack & vec)

{

std::cout << "Enter count elements: ";

int n;

std::cin >> n;

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

Time t;

std::cin >> t;

vec.push(t);

}

}

void add\_max\_to\_begin(MyStack & vec)

{

Time max = vec.top();

MyStack temp;

while (!vec.empty()) {

if (vec.top() > max)

max = vec.top();

temp.push(vec.top());

vec.pop();

}

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

vec.push(max);

while (!temp.empty()) {

vec.push(temp.top());

temp.pop();

}

}

void erase\_min(MyStack & vec)

{

Time min = vec.top();

MyStack temp;

while (!vec.empty()) {

if (vec.top() < min)

min = vec.top();

temp.push(vec.top());

vec.pop();

}

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

while (!temp.empty()) {

if (temp.top() != min)

vec.push(temp.top());

temp.pop();

}

}

void inc\_average(MyStack & vec)

{

Time avg;

MyStack temp;

while (!vec.empty()) {

avg = avg + vec.top();

temp.push(vec.top());

vec.pop();

}

avg = avg / temp.size();

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

while (!temp.empty()) {

vec.push(temp.top() + avg);

temp.pop();

}

}

**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&);

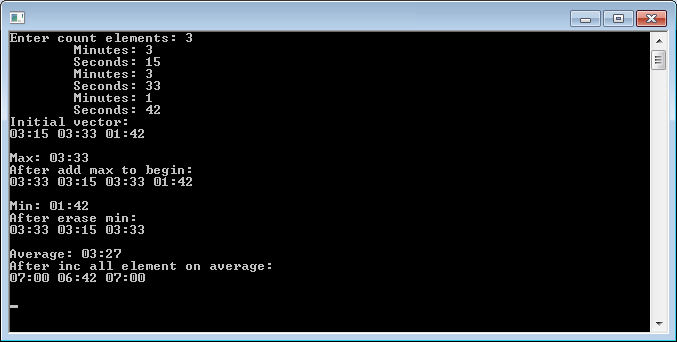
private:

int minutes, seconds;

};

#endif

**Результат выполнения программы**



**Задание №5**

**Файл Lab11\_5\_main.cpp**

#include <iostream>

#include "my\_vector\_11\_5.h"

#include "time\_class.h"

int main()

{

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

int n;

std::cin >> n;

MyVector<Time> vec(n);

vec.print("Initial vector: ");

vec.add\_max\_to\_begin();

vec.print("After add max to begin: ");

vec.erase\_min();

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

vec.inc\_average();

vec.print("After inc all element on average: ");

std::cin.ignore();

std::cin.get();

return 0;

}

**Файл my\_vector\_11\_5.h**

#ifndef MY\_VECTOR\_H

#define MY\_VECTOR\_H

#include <stack>

#include <string>

#include <iostream>

template <typename T>

class MyVector

{

public:

MyVector() {}

MyVector(int n);

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

void add\_max\_to\_begin();

void erase\_min();

void inc\_average();

private:

std::stack<T> vec;

};

template <typename T>

MyVector<T>::MyVector(int n)

{

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

T t;

std::cin >> t;

vec.push(t);

}

}

template <typename T>

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

{

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

std::stack<T> temp;

while (!vec.empty()) {

temp.push(vec.top());

vec.pop();

}

while (!temp.empty()) {

std::cout << temp.top() << " ";

vec.push(temp.top());

temp.pop();

}

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

}

template <typename T>

void MyVector<T>::add\_max\_to\_begin()

{

T max = vec.top();

std::stack<T> temp;

while (!vec.empty()) {

if (vec.top() > max)

max = vec.top();

temp.push(vec.top());

vec.pop();

}

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

vec.push(max);

while (!temp.empty()) {

vec.push(temp.top());

temp.pop();

}

}

template <typename T>

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

{

T min = vec.top();

std::stack<T> temp;

while (!vec.empty()) {

if (vec.top() < min)

min = vec.top();

temp.push(vec.top());

vec.pop();

}

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

while (!temp.empty()) {

if (temp.top() != min)

vec.push(temp.top());

temp.pop();

}

}

template <typename T>

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

{

T avg;

std::stack<T> temp;

while (!vec.empty()) {

avg = avg + vec.top();

temp.push(vec.top());

vec.pop();

}

avg = avg / temp.size();

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

while (!temp.empty()) {

vec.push(temp.top() + avg);

temp.pop();

}

}

#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 выполнения программы**

