Пермский Национальный Исследовательский  
Политехнический Университет

**Лабораторная работа №13**

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

Стандартные обобщенные алгоритмы библиотеки STL

1. **Цель задания**.

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

1.2. Использование стандартных обобщенных алгоритмов из библиотеки STL в ОО программе

1. **Задание.**

Задача №1

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

Задача №2

* 1. Адаптер контейнера – стек

Задача №3

3.1 Ассоциативный контейнер - множество

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 lab13\_1\_main.cpp**

#include <iostream>

#include <algorithm>

#include <iterator>

#include <vector>

#include <string>

#include <numeric>

#include <functional>

#include "time\_class.h"

typedef std::vector<Time> MyContainer;

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

void fill(MyContainer&);

void replace\_max(MyContainer&);

void erase\_min(MyContainer&);

void sort\_ascending(MyContainer&);

void sort\_descending(MyContainer&);

void find(MyContainer&);

void add\_average(MyContainer&);

int main()

{

MyContainer vec;

fill(vec);

print(vec, "Initial container: ");

replace\_max(vec);

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

erase\_min(vec);

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

sort\_ascending(vec);

print(vec, "After sort by ascending: ");

sort\_descending(vec);

print(vec, "After sort by descending: ");

find(vec);

add\_average(vec);

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

std::cin.ignore();

std::cin.get();

return 0;

}

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

{

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

std::copy(c.begin(), c.end(), std::ostream\_iterator<Time>(std::cout, " "));

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.push\_back(t);

}

}

void replace\_max(MyContainer& c)

{

std::cout << "Enter replacing element:\n";

Time t;

std::cin >> t;

MyContainer::iterator max = std::max\_element(c.begin(), c.end());

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

std::replace(c.begin(), c.end(), \*max, t);

}

void erase\_min(MyContainer& c)

{

MyContainer::iterator min = std::min\_element(c.begin(), c.end());

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

c.erase(std::remove(c.begin(), c.end(), \*min), c.end());

}

void sort\_ascending(MyContainer& c)

{

std::sort(c.begin(), c.end());

}

void sort\_descending(MyContainer& c)

{

std::sort(c.begin(), c.end(), std::greater<Time>());

}

void find(MyContainer& c)

{

Time t;

std::cout << "Enter value to find:\n";

std::cin >> t;

if (std::find(c.begin(), c.end(), t) != c.end())

std::cout << "Element found!\n\n";

else

std::cout << "Element NOT found!\n\n";

}

void add\_average(MyContainer& c)

{

Time avg = std::accumulate(c.begin(), c.end(), Time()) / c.size();

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

std::transform(c.begin(), c.end(), c.begin(),

[avg](const Time& t) { return t + 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&);

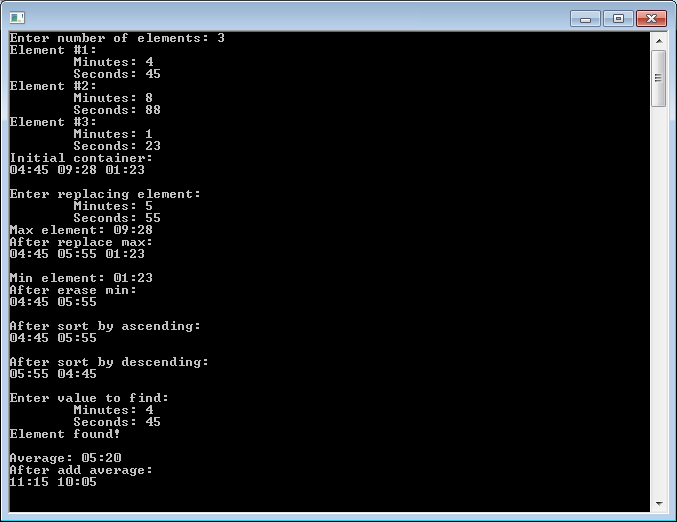
private:

int minutes, seconds;

};

#endif

**Результат работы программы**



**Задание №2**

**Файл lab13\_2\_main.cpp**

#include <iostream>

#include <algorithm>

#include <iterator>

#include <stack>

#include <vector>

#include <string>

#include <numeric>

#include <functional>

#include "time\_class.h"

typedef std::stack<Time> MyContainer;

typedef std::vector<Time> TempContainer;

void stack\_to\_vector(MyContainer&, TempContainer&);

void vector\_to\_stack(TempContainer&, MyContainer&);

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

void fill(MyContainer&);

void replace\_max(MyContainer&);

void erase\_min(MyContainer&);

void sort\_ascending(MyContainer&);

void sort\_descending(MyContainer&);

void find(MyContainer&);

void add\_average(MyContainer&);

int main()

{

MyContainer vec;

fill(vec);

print(vec, "Initial container: ");

replace\_max(vec);

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

erase\_min(vec);

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

sort\_ascending(vec);

print(vec, "After sort by ascending: ");

sort\_descending(vec);

print(vec, "After sort by descending: ");

find(vec);

add\_average(vec);

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

std::cin.ignore();

std::cin.get();

return 0;

}

void stack\_to\_vector(MyContainer& c, TempContainer& v)

{

while (!c.empty()) {

v.push\_back(c.top());

c.pop();

}

}

void vector\_to\_stack(TempContainer& v, MyContainer& c)

{

for (TempContainer::reverse\_iterator it = v.rbegin(); it != v.rend(); ++it)

c.push(\*it);

}

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

{

TempContainer temp;

stack\_to\_vector(c, temp);

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

std::copy(temp.begin(), temp.end(), std::ostream\_iterator<Time>(std::cout, " "));

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

vector\_to\_stack(temp, c);

}

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.push(t);

}

}

void replace\_max(MyContainer& c)

{

std::cout << "Enter replacing element:\n";

Time t;

std::cin >> t;

TempContainer temp;

stack\_to\_vector(c, temp);

TempContainer::iterator max = std::max\_element(temp.begin(), temp.end());

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

std::replace(temp.begin(), temp.end(), \*max, t);

vector\_to\_stack(temp, c);

}

void erase\_min(MyContainer& c)

{

TempContainer temp;

stack\_to\_vector(c, temp);

TempContainer::iterator min = std::min\_element(temp.begin(), temp.end());

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

temp.erase(std::remove(temp.begin(), temp.end(), \*min), temp.end());

vector\_to\_stack(temp, c);

}

void sort\_ascending(MyContainer& c)

{

TempContainer temp;

stack\_to\_vector(c, temp);

std::sort(temp.begin(), temp.end());

vector\_to\_stack(temp, c);

}

void sort\_descending(MyContainer& c)

{

TempContainer temp;

stack\_to\_vector(c, temp);

std::sort(temp.begin(), temp.end(), std::greater<Time>());

vector\_to\_stack(temp, c);

}

void find(MyContainer& c)

{

TempContainer temp;

stack\_to\_vector(c, temp);

Time t;

std::cout << "Enter value to find:\n";

std::cin >> t;

if (std::find(temp.begin(), temp.end(), t) != temp.end())

std::cout << "Element found!\n\n";

else

std::cout << "Element NOT found!\n\n";

vector\_to\_stack(temp, c);

}

void add\_average(MyContainer& c)

{

TempContainer temp;

stack\_to\_vector(c, temp);

Time avg = std::accumulate(temp.begin(), temp.end(), Time()) / temp.size();

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

std::transform(temp.begin(), temp.end(), temp.begin(),

[avg](const Time& t) { return t + avg; });

vector\_to\_stack(temp, c);

}

**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&);

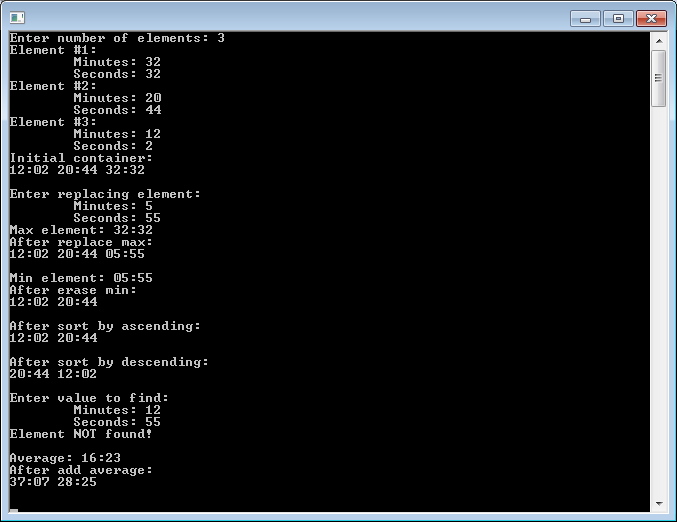
private:

int minutes, seconds;

};

#endif

**Результат выполнения программы**



**Задание №3**

**Файл lab13\_3\_main.cpp**

#include <iostream>

#include <algorithm>

#include <iterator>

#include <set>

#include <string>

#include <numeric>

#include <functional>

#include "time\_class.h"

typedef std::set<Time> MyContainer;

typedef std::set<Time, std::greater<Time>> TempContainer;

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

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

void fill(MyContainer&);

void replace\_max(MyContainer&);

void erase\_min(MyContainer&);

TempContainer sort\_descending(MyContainer&);

void find(MyContainer&);

TempContainer add\_average(MyContainer&);

int main()

{

MyContainer vec;

fill(vec);

print(vec, "Initial container: ");

replace\_max(vec);

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

erase\_min(vec);

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

print(sort\_descending(vec), "After sort by descending: ");

find(vec);

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

std::cin.ignore();

std::cin.get();

return 0;

}

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

{

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

std::copy(c.begin(), c.end(), std::ostream\_iterator<Time>(std::cout, " "));

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

}

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

{

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

std::copy(c.begin(), c.end(), std::ostream\_iterator<Time>(std::cout, " "));

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.insert(t);

}

}

void replace\_max(MyContainer& c)

{

std::cout << "Enter replacing element:\n";

Time t;

std::cin >> t;

MyContainer::iterator max = std::max\_element(c.begin(), c.end());

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

c.erase(max);

c.insert(t);

}

void erase\_min(MyContainer& c)

{

MyContainer::iterator min = std::min\_element(c.begin(), c.end());

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

c.erase(min);

}

TempContainer sort\_descending(MyContainer& c)

{

TempContainer temp;

std::copy(c.begin(), c.end(), std::inserter(temp, temp.begin()));

return temp;

}

void find(MyContainer& c)

{

Time t;

std::cout << "Enter value to find:\n";

std::cin >> t;

if (std::find(c.begin(), c.end(), t) != c.end())

std::cout << "Element found!\n\n";

else

std::cout << "Element NOT found!\n\n";

}

TempContainer add\_average(MyContainer& c)

{

Time avg = std::accumulate(c.begin(), c.end(), Time()) / c.size();

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

TempContainer temp;

std::transform(c.begin(), c.end(), std::inserter(temp, temp.begin()),

[avg](const Time& t) { return t + avg; });

return temp;

}

**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**

