МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**Институт компьютерных технологий и информационной безопасности**

**Кафедра Математического обеспечения и применения ЭВМ**

 

**ОТЧЁТ**

по лабораторной работе №3

по курсу «GoF-паттерны ООП»

на тему «Библиотека STL»

Выполнила:

студентка группы КТмо2-3

Куприянова А.А.

Проверил:

доцент кафедры ТОР

Максимов М.Н.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2017 г.

Таганрог 2017

**Цель работы:** Получение навыка работы с библиотекой STL.

Домашнее задание. Компонентные функции коллекции deque:

Члены

| **Определение типа** | **Описание** |
| --- | --- |
| [deque::const\_iterator](https://msdn.microsoft.com/ru-ru/library/bb385352.aspx) | Тип постоянного итератора для управляемой последовательности. |
| [deque::const\_reference](https://msdn.microsoft.com/ru-ru/library/bb398181.aspx) | Тип постоянной ссылки на элемент. |
| [deque::const\_reverse\_iterator](https://msdn.microsoft.com/ru-ru/library/bb157856.aspx) | Тип константы обратного итератора для контролируемой последовательности. |
| [deque::difference\_type](https://msdn.microsoft.com/ru-ru/library/bb398193.aspx) | Тип расстояния со знаком между двумя элементами. |
| [deque::generic\_container](https://msdn.microsoft.com/ru-ru/library/bb385345.aspx) | Тип универсального интерфейса для контейнера. |
| [deque::generic\_iterator](https://msdn.microsoft.com/ru-ru/library/bb385653.aspx) | Тип итератора для универсального интерфейса для контейнера. |
| [deque::generic\_reverse\_iterator](https://msdn.microsoft.com/ru-ru/library/bb398249.aspx) | Тип обратного итератора для универсального интерфейса для контейнера. |
| [deque::generic\_value](https://msdn.microsoft.com/ru-ru/library/bb386292.aspx) | Тип элемента для универсального интерфейса для контейнера. |
| [deque::iterator](https://msdn.microsoft.com/ru-ru/library/bb385239.aspx) | Тип итератора для управляемой последовательности. |
| [deque::reference](https://msdn.microsoft.com/ru-ru/library/bb385012.aspx) | Тип ссылки на элемент. |
| [deque::reverse\_iterator](https://msdn.microsoft.com/ru-ru/library/bb385639.aspx) | Тип обратного итератора для контролируемой последовательности. |
| [deque::size\_type](https://msdn.microsoft.com/ru-ru/library/bb398045.aspx) | Тип расстояния со знаком между двумя элементами. |
| [deque::value\_type](https://msdn.microsoft.com/ru-ru/library/bb398040.aspx) | Тип элемента. |
| **Функция Member** | **Описание** |
| [deque::assign](https://msdn.microsoft.com/ru-ru/library/bb384998.aspx) | Заменяет все элементы. |
| [deque::at](https://msdn.microsoft.com/ru-ru/library/bb385823.aspx) | Получает элемент в указанной позиции. |
| [deque::back](https://msdn.microsoft.com/ru-ru/library/bb385440.aspx) | Получает последний элемент. |
| [deque::begin](https://msdn.microsoft.com/ru-ru/library/bb386126.aspx) | Задает начало управляемой последовательности. |
| [deque::clear](https://msdn.microsoft.com/ru-ru/library/bb385116.aspx) | Удаляет все элементы. |
| [deque::deque](https://msdn.microsoft.com/ru-ru/library/bb386108.aspx) | Создает объект контейнера. |
| [deque::empty](https://msdn.microsoft.com/ru-ru/library/bb157726.aspx) | Тесты отсутствуют ли какие-либо элементы. |
| [deque::end](https://msdn.microsoft.com/ru-ru/library/bb385270.aspx) | Задает конец управляемой последовательности. |
| [deque::erase](https://msdn.microsoft.com/ru-ru/library/bb385640.aspx) | Удаляет элементы в указанных положениях. |
| [deque::front](https://msdn.microsoft.com/ru-ru/library/bb386160.aspx) | Получение первого элемента. |
| [deque::insert](https://msdn.microsoft.com/ru-ru/library/bb385887.aspx) | Добавляет элементы в указанной позиции. |
| [deque::pop\_back](https://msdn.microsoft.com/ru-ru/library/bb385356.aspx) | Удаляет последний элемент. |
| [deque::pop\_front](https://msdn.microsoft.com/ru-ru/library/bb385347.aspx) | Удаляет первый элемент. |
| [deque::push\_back](https://msdn.microsoft.com/ru-ru/library/bb398182.aspx) | Добавляет новый последний элемент. |
| [deque::push\_front](https://msdn.microsoft.com/ru-ru/library/bb385919.aspx) | Добавляет новый первый элемент. |
| [deque::rbegin](https://msdn.microsoft.com/ru-ru/library/bb385500.aspx) | Задает начало обратной управляемой последовательности. |
| [deque::rend](https://msdn.microsoft.com/ru-ru/library/bb398034.aspx) | Задает конец обратной управляемой последовательности. |
| [deque::resize](https://msdn.microsoft.com/ru-ru/library/bb398073.aspx) | Изменяет количество элементов. |
| [deque::size](https://msdn.microsoft.com/ru-ru/library/bb385635.aspx) | Подсчитывает количество элементов. |
| [deque::swap](https://msdn.microsoft.com/ru-ru/library/bb385350.aspx) | Меняет местами содержимое двух контейнеров. |
| [deque::to\_array](https://msdn.microsoft.com/ru-ru/library/bb386165.aspx) | Копирует контролируемая последовательность в новый массив. |

| **Свойство** | **Описание** |
| --- | --- |
| [deque::back\_item](https://msdn.microsoft.com/ru-ru/library/bb386019.aspx) | Получает последний элемент. |
| [deque::front\_item](https://msdn.microsoft.com/ru-ru/library/bb157606.aspx) | Получение первого элемента. |
| **Оператор** | **Описание** |
| [deque::operator!=](https://msdn.microsoft.com/ru-ru/library/bb398031.aspx) | Определяет, являются ли два объекта deque неравными. |
| [deque::operator](https://msdn.microsoft.com/ru-ru/library/bb398145.aspx) | Получает элемент в указанной позиции. |
| [operator< (deque)](https://msdn.microsoft.com/ru-ru/library/bb386251.aspx) | Определяет, является ли объект deque меньше другой объект deque. |
| [operator<= (deque)](https://msdn.microsoft.com/ru-ru/library/bb386056.aspx) | Определяет, является ли объект deque меньше или равно другому объекту deque. |
| [operator= (deque)](https://msdn.microsoft.com/ru-ru/library/bb385027.aspx) | Заменяет контролируемая последовательность. |
| [operator== (deque)](https://msdn.microsoft.com/ru-ru/library/bb386074.aspx) | Определяет, является ли объект deque равен другому объекту deque. |
| [operator> (deque)](https://msdn.microsoft.com/ru-ru/library/bb386025.aspx) | Определяет, является ли объект deque больше другого объекта deque. |
| [operator>= (deque)](https://msdn.microsoft.com/ru-ru/library/bb385075.aspx) | Определяет, является ли объект deque больше или равно другому объекту deque. |

Задание 1. Переход от ссылочной семантики к семантике по значению.  
// Три разные коллекции

typedef CountedPtr<comp> IntPtr;

deque<IntPtr> coll1;

list<IntPtr> coll2;

set<IntPtr> coll3;

deque<comp> coll11;

list<comp> coll22;

/\* Вставка общих объектов в коллекции\*/

for (int i = 0; i<5; ++i) {

IntPtr ptr(new comp(i+1, i+1));

coll1.push\_back(ptr);

coll2.push\_front(ptr);

coll3.insert(ptr);

coll11.push\_back(comp(i+1, i+1));

coll22.push\_front(comp(i+1, i+1));

}

// Вывод содержимого коллекций

cout << "\n deque, coll1\n";

for\_each(coll1.begin(), coll1.end(), printCountedPtr);

cout << "\n list, coll2\n";

for\_each(coll2.begin(), coll2.end(), printCountedPtr);

cout << "\n set, coll3\n";

for\_each(coll3.begin(), coll3.end(), printCountedPtr);

cout << endl << endl;

cout << "\n deque, coll11:\n";

for\_each(coll11.begin(), coll11.end(), printCountedPtr2);

cout << "\n list, coll22\n";

for\_each(coll22.begin(), coll22.end(), printCountedPtr2);

cout << endl << endl;

\*coll2.front() = -(\*coll2.front());

\*coll1[1] = \*coll1[1] \* \*coll1[1];

coll11[1] = coll11[1] \* coll11[1];

coll22.front() = -(coll22.front());

coll22.back() = -(coll22.back());

// Повторный вывод содержимого коллекций

cout << "\n deque, coll1\n";

for\_each(coll1.begin(), coll1.end(), printCountedPtr);

cout << "\n list, coll2\n";

for\_each(coll2.begin(), coll2.end(), printCountedPtr);

cout << "\n set, coll3\n";

for\_each(coll3.begin(), coll3.end(), printCountedPtr);

cout << endl << endl;

cout << "\n deque, coll11:\n";

for\_each(coll11.begin(), coll11.end(), printCountedPtr2);

cout << "\n list, coll22\n";

for\_each(coll22.begin(), coll22.end(), printCountedPtr2);

cout << endl << endl;

Задание 2. Определение своего класса и формирование коллекции из объектов этого класса.

class ComplexNumber {

double Re;

double Im;

public:

~ComplexNumber(){}; //Деструктор

ComplexNumber(double r); // Конструктор2

ComplexNumber(double r, double i); // Конструктор1

ComplexNumber(){ //Конструктор по умолчанию

Re = 0;

Im = 0;

//cout << "\ndefault constructor, this=" << this;

}

friend ComplexNumber& operator+(ComplexNumber& A, ComplexNumber& B);

friend ComplexNumber& operator-(ComplexNumber& A, ComplexNumber& B);

friend ComplexNumber& operator\*(ComplexNumber& A, ComplexNumber& B);

friend ComplexNumber& operator\*(ComplexNumber& A, double K);

friend ComplexNumber& operator\*(double K, ComplexNumber& A);

friend void operator+=(ComplexNumber& A, ComplexNumber& B);

friend bool operator==(const ComplexNumber& A, const ComplexNumber& B);

friend bool operator<(const ComplexNumber& A, const ComplexNumber& B);

friend bool operator!=(ComplexNumber& A, ComplexNumber& B);

friend ostream& operator<<(ostream& stream, const ComplexNumber& A);

friend double abs(const ComplexNumber& A);

friend void display2(ComplexNumber& A);

};

void printComplexNumberDeque(ComplexNumber elem)

{

cout << "\n" << elem;

}

ComplexNumber::ComplexNumber(double r, double i) {

Re = r;

Im = i;

//cout << "\nconstructor1 for ComplexNumber, this="<<this;

}

ComplexNumber::ComplexNumber(double r){

Re = r;

Im = 0;

//cout << "\nconstructor2 for ComplexNumber, r=" << r<<" , this="<<this;

}

ostream& operator<<(ostream& stream, const ComplexNumber& A) {

if (A.Im>0) stream << A.Re << "+" << A.Im << "i";

if (A.Im<0) stream << A.Re << "-" << (A.Im\*(-1.0)) << "i";

if (A.Im == 0) stream << A.Re;

return stream;

}

double abs(const ComplexNumber& A){

//cout << "\n double abs" << A;

//double result;

//cout << "\nA.Re=" << A.Re << ", A.Im=" << A.Im;

//result = sqrt(A.Re\*A.Re + A.Im\*A.Im);

//cout << "\nresult abs=" << result;

return sqrt(A.Re\*A.Re + A.Im\*A.Im);

}

bool operator==(const ComplexNumber& A, const ComplexNumber& B) {

if ((A.Re == B.Re) && (A.Im == B.Im)) return true;

else return false;

}

bool operator<(const ComplexNumber& A, const ComplexNumber& B) {

//if ((A.Re < B.Re) && (A.Im < B.Im)) return true;

//cout << "\n operator<";

if (abs(A) < abs(B)) return true;

else return false;

}

bool operator!=(ComplexNumber& A, ComplexNumber& B) {

return (!(A == B));

}

ComplexNumber& operator+(ComplexNumber& A, ComplexNumber& B) {

ComplexNumber result = ComplexNumber(A.Re + B.Re, A.Im + B.Im);

return result;

}

void operator+=(ComplexNumber& A, ComplexNumber& B) {

ComplexNumber result = ComplexNumber(A.Re + B.Re, A.Im + B.Im);

A = result;

return;

}

ComplexNumber& operator-(ComplexNumber& A, ComplexNumber& B) {

ComplexNumber result = ComplexNumber(A.Re - B.Re, A.Im - B.Im);

return result;

}

ComplexNumber& operator\*(ComplexNumber& A, ComplexNumber& B) {

ComplexNumber result = ComplexNumber(A.Re\*B.Re - A.Im\*B.Im, A.Re\*B.Im + A.Im\*B.Re);

//cout << "\noperator\*(ComplexNumber& A, ComplexNumber& B)\nA.Re=" << A.Re << ", A.Im=" << A.Im << ", B.Re=" << B.Re << ", B.Im=" << B.Im;

//cout << "\nresult=" << result;

return result;

}

ComplexNumber& operator\*(ComplexNumber& A, double K) {

ComplexNumber result = ComplexNumber(A.Re\*K, A.Im\*K);

return result;

}

ComplexNumber& operator\*(double K, ComplexNumber& A) {

ComplexNumber result = ComplexNumber(K\*A.Re, K\*A.Im);

return result;

}

void display2(ComplexNumber& A){

if ((A.Im)<0) {

printf("%5.2f%5.2fi\n", A.Re, A.Im);

return;

}

if ((A.Im)==0) {

printf("%5.2fi\n", A.Re, A.Im);

return;

}

printf("%5.2f+%5.2fi", A.Re, A.Im);

}

deque<ComplexNumber> coll4;

list<ComplexNumber> coll5;

set<ComplexNumber> coll6;

for (int i = 0; i<5; ++i) {

ComplexNumber C = ComplexNumber(i, i+1);

coll4.push\_back(C); //coll1.push\_back(comp(i,i));

coll5.push\_front(C);

coll6.insert(C);

}

// Вывод содержимого коллекций

cout << "\n deque, coll4";

for\_each(coll4.begin(), coll4.end(), printComplexNumberDeque);

cout << "\n list, coll5";

for\_each(coll5.begin(), coll5.end(), printComplexNumberDeque);

cout << "\n set, coll6";

for\_each(coll6.begin(), coll6.end(), printComplexNumberDeque);

cout << endl << endl;

/\* Модификация значений в разных коллекциях

coll4[0] = \*(new ComplexNumber(coll4[0] \* coll4[0]));

coll4[1] = \*(new ComplexNumber(ComplexNumber(coll4[0]) \* ComplexNumber(0, 1)));

coll4[2] = \*(new ComplexNumber(ComplexNumber(0, 100) + ComplexNumber(coll4[2])));

coll4[3] = \*(new ComplexNumber(ComplexNumber(abs(coll4[3])) + ComplexNumber(1, 0)));

ComplexNumber c;

c = ComplexNumber(10, 10) \* coll4[4];

coll4[4] = c;

c = c \* 2;

coll5.front() = \*(new ComplexNumber(coll5.front()\*(-1.0)));

// Вывод содержимого коллекций

cout << "\n deque, coll4:";

for\_each(coll4.begin(), coll4.end(), printComplexNumberDeque);

cout << "\n list, coll5";

for\_each(coll5.begin(), coll5.end(), printComplexNumberDeque);

cout << "\n set, coll6";

for\_each(coll6.begin(), coll6.end(), printComplexNumberDeque);

cout << endl << endl;

Результаты работы программы:

 