Министерство Науки и Образования Российской Федерации

Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ»

Кафедра МО ЭВМ

Отчёт по лабораторной работе №1

**«Программирование контейнерных классов»**

**Выполнил: Эмман П.А.**

**гр. 3351**

**Проверил: Романенко С.А.**

**Санкт-Петербург**

**2007**

Оглавление

[1. Постановка задачи 3](#_Toc184387411)

[2. Ход работы 4](#_Toc184387412)

[2.1. Разработка программ в среде MS Visual C++ 4](#_Toc184387413)

[2.1.1. Настройка среды. Выполнение индивидуального задания 4](#_Toc184387414)

[2.1.2. Работа в режиме отладки 4](#_Toc184387415)

[2.1.3. Исследование программы при помощи Profiler 4](#_Toc184387416)

[2.2. Применение стандартной библиотеки STL 4](#_Toc184387417)

[2.2.1. Составить консольные приложения, демонстрирующие основные операции с контейнерами и итераторами STL 4](#_Toc184387418)

[2.2.2. Реализовать новый шаблон контейнера и шаблон итератора для него по индивидуальному заданию 4](#_Toc184387419)

[3. Выводы 5](#_Toc184387420)

# Постановка задачи

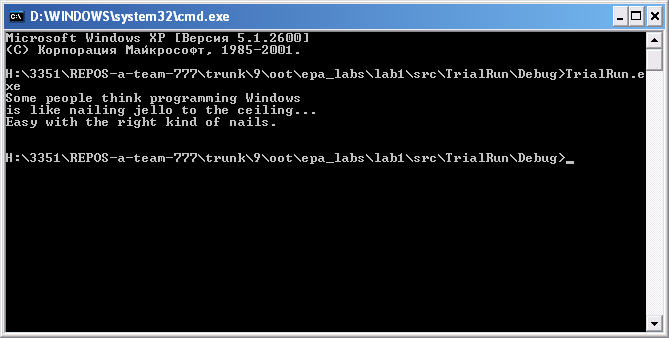
* 1. Разработка программ в среде MS Visual C++
     1. Настройка среды. Выполнение индивидуального задания
     2. Работа в режиме отладки
     3. Исследование программы при помощи Profiler
  2. Применение стандартной библиотеки STL
     1. Составить консольные приложения, демонстрирующие основные операции с контейнерами и итераторами STL
     2. Реализовать новый шаблон контейнера и шаблон итератора для него по индивидуальному заданию

# Ход работы

## Разработка программ в среде MS Visual C++

## Настройка среды. Выполнение индивидуального задания

1. Найти проекты MFC Tutorial. Откомпилировать и выполнить пример TrialRun.



1. Индивидуальное задание:

*Написать классы для создания графических объектов. Классы должны иметь общий абстрактный базовый класс Shape с чистыми виртуальными функциями.*

*Необходимо использовать множественное наследование. В классах должны быть предусмотрены виртуальные функции для вывода информации об объектах в поток, а Shape должен иметь дружественный перегруженный оператор <<.*

В соответствии с вариантом задания необходимо реализовать элементы:

* квадрат
* ромб
* текст
* текст в квадрате

1. Диаграмма разработанных и реализованных классов:



1. Заголовочные файлы для этих классов:

### CShape.h

#ifndef \_CSHAPE\_H

#define \_CSHAPE\_H

#include "iostream.h"

//фигура

//##ModelId=47532D07006D

class CShape{

private:

//##ModelId=47532D07006E

static int KOL;

//##ModelId=47532D07006F

const int id;

protected:

//##ModelId=47532D070070

double x;

//##ModelId=47532D07007D

double y;

//constructors & desctuctors

//##ModelId=47532D07007E

CShape(double x, double y);

//re-defined function

//##ModelId=47532D070081

virtual ostream& printInfo(ostream&os) const;

public:

//constructors & desctuctors

//##ModelId=47532D07008D

virtual ~CShape();

//getters

//##ModelId=47532D07008F

const double getX() const;

//##ModelId=47532D070091

const double getY() const;

//##ModelId=47532D070093

const int getID() const;

//setters

//##ModelId=47532D070095

void setX(double newX);

//##ModelId=47532D07009C

void setY(double newY);

//re-defined

//##ModelId=47532D0700B0

friend ostream& operator<<(ostream& o, const CShape& rhs);

//##ModelId=47532D07009E

virtual bool operator==(const CShape& rhs) const;

//some user-functions

//##ModelId=47532D0700A1

void moveX(double deltaX);

//##ModelId=47532D0700A3

void moveY(double deltaY);

//pure virtual function

//##ModelId=47532D0700AC

virtual double S()const = 0;

//##ModelId=47532D0700AE

virtual double P()const = 0;

};

#endif

### CDiamond.h

#ifndef \_CCDIAMOND\_H

#define \_CCDIAMOND\_H

#include "CShape.h"

//фигура

//##ModelId=47532D0700BB

class CDiamond: public CShape{

private:

//##ModelId=47532D0700CC

static int KOLdiamond;

protected:

//##ModelId=47532D0700CD

const int idDiamond;

//длина стороны

//##ModelId=47532D0700CE

double l;

//##ModelId=47532D0700CF

double u; //угол

//re-defined function

//##ModelId=47532D0700DA

virtual ostream& printInfo(ostream&os) const;

public:

//constructors & desctuctors

//##ModelId=47532D0700DD

CDiamond(double x, double y, double l, double u);

//constructors & desctuctors

//##ModelId=47532D0700E2

~CDiamond();

//getters

//##ModelId=47532D0700EA

const double getL() const;

//##ModelId=47532D0700EC

const double getU() const;

//setters

//##ModelId=47532D0700EE

void setL(double newL);

//##ModelId=47532D0700F0

void setU(double newU);

//pure virtual function

//##ModelId=47532D0700F2

double S()const;

//##ModelId=47532D0700FB

double P()const;

};

#endif

### CSquare.h

#ifndef \_CSQUARE\_H

#define \_CSQUARE\_H

#include "CShape.h"

//фигура

//##ModelId=47532D070023

class CSquare: public virtual CShape{

private:

//##ModelId=47532D07002F

static int KOLsquare;

protected:

//##ModelId=47532D070030

const int idSquare;

//длина стороны

//##ModelId=47532D070031

double l;

//re-defined function

//##ModelId=47532D070032

virtual ostream& printInfo(ostream&os) const;

public:

//constructors & desctuctors

//##ModelId=47532D070040

CSquare(double x, double y, double l);

//constructors & desctuctors

//##ModelId=47532D07004F

~CSquare();

//getters

//##ModelId=47532D070050

const double getL() const;

//setters

//##ModelId=47532D070052

void setL(double newL);

//pure virtual function

//##ModelId=47532D070054

double S()const;

//##ModelId=47532D070056

double P()const;

};

#endif

### CText.h

#ifndef \_CTEXT\_H

#define \_CTEXT\_H

#include "CShape.h"

#include <string>

//текст

//##ModelId=47532D0603AF

class CText : public virtual CShape

{

private:

static KOLtext;

protected:

//##ModelId=47532D0603BA

int IDtext;

//##ModelId=47532D0603BB

CText(std::string content, float x, float y);

//содержимое текста

//##ModelId=47532D0603D9

std::string \_content;

//вывести состояние прямоугольника в поток

//##ModelId=47532D07000F

virtual ostream& printInfo(ostream& os) const;

public:

//##ModelId=47532D070012

const std::string& get\_\_content() const;

//##ModelId=47532D070014

void set\_\_content(std::string& value);

//##ModelId=47532D070016

virtual ~CText();

//вычисляет площадь фигуры

//##ModelId=47532D07001F

virtual double S() const;

//##ModelId=47532D070021

virtual double P() const;

};

#endif

### CTextInSquare.h

#ifndef \_CTEXTINSQUARE\_H

#define \_CTEXTINSQUARE\_H

//////////////////////////////////////////////////////////////////////////

#include "CSquare.h"

#include "CText.h"

//////////////////////////////////////////////////////////////////////////

class ostream;

//////////////////////////////////////////////////////////////////////////

//##ModelId=47532D06038E

class CTextInSquare : public CSquare, public CText

{

private:

static KOLtextInSquare;

protected:

//##ModelId=47532D060391

int idTextInSquare;

//вывести состояние текста в овале в поток

//##ModelId=47532D060399

virtual ostream& printInfo(ostream& os) const;

public:

//##ModelId=47532D06039C

CTextInSquare(std::string content, double x, double y, double l);

//##ModelId=47532D0603A9

virtual ~CTextInSquare();

//вычисляет площадь фигуры

//##ModelId=47532D0603AB

virtual double S() const;

//##ModelId=47532D0603AD

virtual double P() const;

};

#endif

1. Требуемые в задании функции реализованы следующим образом:

* перегруженный оператор вывода в поток

ostream& operator<<( ostream& o, const CShape& rhs )

{

return rhs.printInfo(o);

}

* чистые виртуальные функции (переопределяют CShape::S() CShape::P() ):

double CDiamond::S()const{

return this->l \* this->l;

};

double CDiamond::P()const{

return 4 \* this->l;

};

double CSquare::S()const{

return this->l \* this->l;

};

double CSquare::P()const{

return 4 \* this->l;

};

double CText::S() const

{

return -1;

}

double CText::P() const

{

return -1;

}

double CTextInSquare::S() const

{

return CSquare::S();

}

double CTextInSquare::P() const

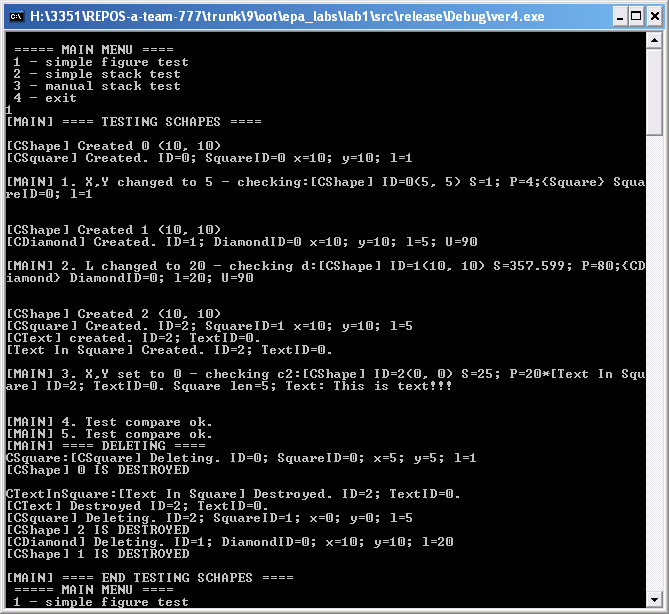
{

return CSquare::P();

}

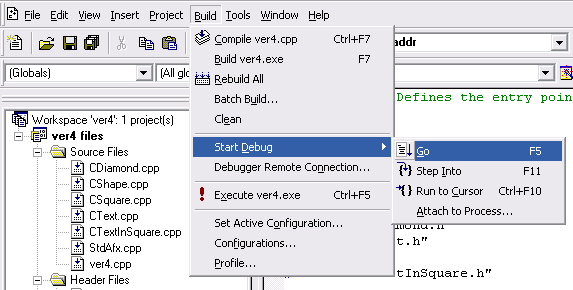
1. В простейшем случае тестирующая программа выглядит так:

|  |  |
| --- | --- |
| void runTestFig()  {  CShape\* c = new CSquare(10,10,1);  c->setX(5);  c->setY(5);  cout << endl << "[MAIN] 1. X,Y changed to 5 - checking:" << \*c << endl << endl;  CDiamond d = CDiamond(10,10,5,90);  d.setL(20);  cout << endl << "[MAIN] 2. L changed to 20 - checking d:" << d<< endl<< endl;  CShape\* c2 = new CTextInSquare("This is text!!!", 10,10,5);  c2->setX(0);  c2->setY(0);  cout << endl << "[MAIN] 3. X,Y set to 0 - checking c2:" << \*c2<< endl<< endl;  CShape \* c3 = c;  if(c3 == c) cout << "[MAIN] 4. Test compare ok."<< endl; else cout << "[MAIN] 4. Test compare FAILED."<< endl;  if(c2 == c) cout << "[MAIN] 5. Test compare FAILED."<< endl; else cout << "[MAIN] 5. Test compare ok."<< endl;  cout << "[MAIN] ==== DELETING ==== "<< endl << "CSquare:";  delete c;  cout <<endl << "CTextInSquare:";  delete c2;  } | [MAIN] ==== TESTING SCHAPES ====  [CShape] Created 0 (10, 10)  [CSquare] Created. ID=0; SquareID=0 x=10; y=10; l=1  [MAIN] 1. X,Y changed to 5 - checking:[CShape] ID=0(5, 5) S=1; P=4;{Square} Squa  reID=0; l=1  [CShape] Created 1 (10, 10)  [CDiamond] Created. ID=1; DiamondID=0 x=10; y=10; l=5; U=90  [MAIN] 2. L changed to 20 - checking d:[CShape] ID=1(10, 10) S=357.599; P=80;{CD  iamond} DiamondID=0; l=20; U=90  [CShape] Created 2 (10, 10)  [CSquare] Created. ID=2; SquareID=1 x=10; y=10; l=5  [CText] created. ID=2; TextID=0.  [Text In Square] Created. ID=2; TextID=0.  [MAIN] 3. X,Y set to 0 - checking c2:[CShape] ID=2(0, 0) S=25; P=20\*[Text In Squ  are] ID=2; TextID=0. Square len=5; Text: This is text!!!  [MAIN] 4. Test compare ok.  [MAIN] 5. Test compare ok.  [MAIN] ==== DELETING ====  CSquare:[CSquare] Deleting. ID=0; SquareID=0; x=5; y=5; l=1  [CShape] 0 IS DESTROYED  CTextInSquare:[Text In Square] Destroyed. ID=2; TextID=0.  [CText] Destroyed ID=2; TextID=0.  [CSquare] Deleting. ID=2; SquareID=1; x=0; y=0; l=5  [CShape] 2 IS DESTROYED  [CDiamond] Deleting. ID=1; DiamondID=0; x=10; y=10; l=20  [CShape] 1 IS DESTROYED  [MAIN] ==== END TESTING SCHAPES ==== |



## Работа в режиме отладки

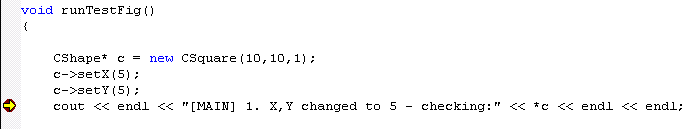
* + - 1. Запустить программу и просмотреть ее работу по шагам (Build -> Start Debug -> Go)



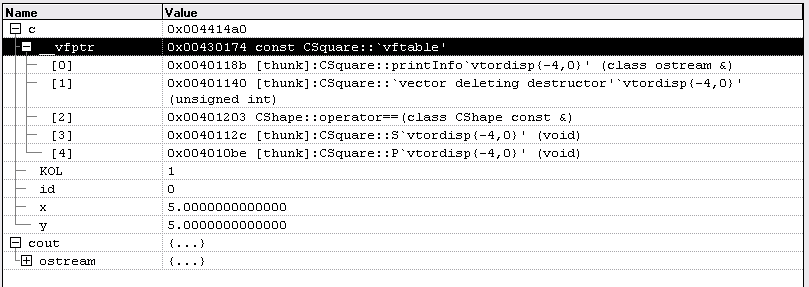
* + - 1. Просмотреть иерархию классов и найти примеры множественного наследования.

Пример множественного наследования — класс «Текст в квадрате» (CTextInSquare). Он наследует одновременно классы «Текст» (CText) и «Квадрат» (CSquare).

* + - 1. Расставить точки превывания программы (Break Points – F9) и протестировать её работу.



* + - 1. Для выяснения текущих значений переменных, использовать механизм "Watch variable".



## Исследование программы при помощи Profiler

* Изучить возможности оптимизации программы в интегрированной среде, в отчете перечислить и объяснить параметры (опции), влияющие на оптимизацию.

Компилятор MS Visual Studio предоставляет следующие виды оптимизации:

* максимизировать скорость выполнения программы (Maximize Speed)
* минимизировать размер исполяемой программы (Minimize Size)
* не оптимизировать при отладке (Disable: Debug)
* по умолчанию (Default)
* Построить несколько вариантов, отличающихся способом оптимизации, проанализировать время работы и объем памяти полученных вариантов. С помощью Profiler определить наиболее долго выполнявшиеся функции. С помощью Profiler определить не исполнявшиеся участки программы.
* Изменить текст main так, чтобы выполнялись все участки программы.

## Применение стандартной библиотеки STL

## Составить консольные приложения, демонстрирующие основные операции с контейнерами и итераторами STL

* Заполняя 3 контейнера строками из <cstring> или другими элементами, продемонстрировать отличия  
    - последовательностей (vector, list, dequeue);  
    - адаптеров последовательностей (stack, queue, priority\_queue);  
    - ассоциативных контейнеров на базе map.
* На примере заполнения одного контейнера-последовательности из предыдущего задания целыми числами, протестировать интерфейсы контейнера и итератора.

Аналогично протестировать ассоциативный контейнер, заполняя его указателями на разные графические объекты из разд. 1.1. Протестировать алгоритны-методы и алгоритмы-классы на множестве графических элементов.

## Реализовать новый шаблон контейнера и шаблон итератора для него по индивидуальному заданию

* Предусмотреть обработку исключительных ситуаций.
* Протестировать контейнер, заполнив его графическими объектами.
* В отчете формально описать реализуемую структуру данных и абстракцию итерации, перечислить все отношения между классами, описать интерфейсы классов и особенности реализации.

По индивидуальному заданию, необходимо реализовать стек на базе массива. В связи с этим имеем две сущности: элемент стека, стек. Реализцем внутренний итератор, который непосредственно хранит информацию об объекте стек и управляет перемещением по стеку. Отметим, что во время «обхода» стека, никакие изменяющие состояния стека действия не должны исполняться.

Для проектирования исключений, было наложено ограничение на количество элементов в стеке. Таким образом, имеется возможность реализовать два вида исключений:

* стек пуст и из него нельзя ничего достать
* стек переполнен и в него нельзя ничего добавить



#### Реализация стека на основе массива:

Сгенерированная диаграмма ПО Rational Rose



****TIterator и InternalIterator реализуют интерфейс IIterator. Внешний итератор TIterator знает о TStack и использует InternalIterator для обхода коллекции. InternalIterator при обходе использует CItemStack для временного хранения информации об объекте. TStack использует CItemStack для хранения всех элементов стека.

#### Основная тестирующая программа

Пользователь может посмотреть на выполнения автоматических тестов фигур и контейнера. А так же работать с контейнером. Доступны операции добавления в стек, извлечения и стека, очистка и просмотр стека. Обход коллекции осуществляется при выводе информации о стеке:

===== MAIN MENU ====

1 - simple figure test

2 - simple stack test

3 - manual stack test

4 - exit

|  |  |
| --- | --- |
| TStack<CShape>\* myS = new TStack<CShape>();  CShape\* c1 = NULL;  try  {  c1 = myS->pop();  }  catch (StackEmptyException\* e)  {  cout<<"[MAIN] Got StackEmptyException!" <<endl;  e->printException();  }  CShape\*q1 = new CSquare(10,10,1);  myS->push(q1);  CShape\*q2 = new CSquare(20,20,2);  myS->push(q2);  CShape\*q3 = new CSquare(30,30,3);  myS->push(q3);  myS->push(new CSquare(40,40,4));  myS->push(new CSquare(50,50,5));  CShape\* c = NULL;  try  {  c = new CSquare(60,60,1);  myS->push(c);  }  catch (StackFullException\* e)  {  cout<<"[MAIN] Got StackFullException!" <<endl;  e->printException();  }  if(c != NULL) delete c;  c1 = myS->pop();  cout << "[MAIN] Pop shape is " << \*c1 << endl;  delete c1;  c1 = myS->pop();  cout << "[MAIN] Pop shape is " << \*c1 << endl;  delete c1;  myS->clear();  delete q1;  delete q2;  delete q3;  delete myS; | [MAIN] ==== TESTING STACK ====  [TStack] Created.  [MAIN] Got StackEmptyException!  [StackEmptyException]  Stack has no elements!  [CShape] Created 0 (10, 10)  [CSquare] Created. ID=0; SquareID=0 x=10; y=10; l=1  [TStack] Push element:[CShape] ID=0(10, 10) S=1; P=4;{Square} SquareID=0; l=1  [TStack] Element pushed.  [CShape] Created 1 (20, 20)  [CSquare] Created. ID=1; SquareID=1 x=20; y=20; l=2  [TStack] Push element:[CShape] ID=1(20, 20) S=4; P=8;{Square} SquareID=1; l=2  [TStack] Element pushed.  [CShape] Created 2 (30, 30)  [CSquare] Created. ID=2; SquareID=2 x=30; y=30; l=3  [TStack] Push element:[CShape] ID=2(30, 30) S=9; P=12;{Square} SquareID=2; l=3  [TStack] Element pushed.  [CShape] Created 3 (40, 40)  [CSquare] Created. ID=3; SquareID=3 x=40; y=40; l=4  [TStack] Push element:[CShape] ID=3(40, 40) S=16; P=16;{Square} SquareID=3; l=4  [TStack] Element pushed.  [CShape] Created 4 (50, 50)  [CSquare] Created. ID=4; SquareID=4 x=50; y=50; l=5  [TStack] Push element:[CShape] ID=4(50, 50) S=25; P=20;{Square} SquareID=4; l=5  [TStack] Element pushed.  [CShape] Created 5 (60, 60)  [CSquare] Created. ID=5; SquareID=5 x=60; y=60; l=1  [MAIN] Got StackFullException!  [StackFullException]  Stack is full!  [CSquare] Deleting. ID=5; SquareID=5; x=60; y=60; l=1  [CShape] 5 IS DESTROYED  [TStack] Pop element: [CShape] ID=4(50, 50) S=25; P=20;{Square} SquareID=4; l=5  [MAIN] Pop shape is [CShape] ID=4(50, 50) S=25; P=20;{Square} SquareID=4; l=5  [CSquare] Deleting. ID=4; SquareID=4; x=50; y=50; l=5  [CShape] 4 IS DESTROYED  [TStack] Pop element: [CShape] ID=3(40, 40) S=16; P=16;{Square} SquareID=3; l=4  [MAIN] Pop shape is [CShape] ID=3(40, 40) S=16; P=16;{Square} SquareID=3; l=4  [CSquare] Deleting. ID=3; SquareID=3; x=40; y=40; l=4  [CShape] 3 IS DESTROYED  [TStack] Clearing...  [TStack] Cleared.  [CSquare] Deleting. ID=0; SquareID=0; x=10; y=10; l=1  [CShape] 0 IS DESTROYED  [CSquare] Deleting. ID=1; SquareID=1; x=20; y=20; l=2  [CShape] 1 IS DESTROYED  [CSquare] Deleting. ID=2; SquareID=2; x=30; y=30; l=3  [CShape] 2 IS DESTROYED  [TStack] Clearing...  [TStack] Cleared.  [TStack] Destroyed.  [MAIN] ==== END TESTING STACK ==== |

Меню для ручного тестировании контейнера:

===== MANUAL MENU ====

1 - PRINT

2 - POP

3 - PUSH

4 - CLEAR

5 - back to main menu

Доступные фигуры:

Creating CShape:

1 - CSquare

2 - CDiamond

3 - CText

4 - CTextInSquare

# Выводы

В ходе лабораторной работы были созданы системы классов, представляющих графические объекты, шаблон контейнера «стек», информацию о возникших исключительных ситуациях. Освоены навыки работы с ПО Rational Rose и MS Visual Studio 6.0.