федеральное ГОСУДАРСТВЕННОЕ Бюджетное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**"САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ "ЛЭТИ" им.В.И.Ульянова (Ленина)" (СПбГЭТУ)**

Кафедра МОЭВМ

**Отчет по лабораторной работе №2**

«Реализация программы на языке C++»

Выполнил: Тюлин Р.Д.

Факультет КТИ

Группа № 0304

Преподаватель: Смольянинов А.В.

Санкт-Петербург

2013

## Проектирование системы описаний классов как системы файлов на языке C++

main.cpp

Cap.h

cyclist.hpppp

intlist.hpp

Hat.h

Visor.h

point.h

Hat.cpp

Visor.cpp

point.cpp

Segment.h

Line.h

Segment.cpp

line.cpp

## Разработка методики тестирования классов и тестовых наборов данных

Тестирование производится по методу «чёрного ящика».

Для тестирования используются следующие методы:

1. перемещение одного или нескольких изображений на сдвиг ;
2. перемещение всех изображений на сдвиг ;

## Количественные характеристики программы

Сведения о тексте:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя файла | Количество строк-комментариев | Количество строк кода | Количество пустых строк |
| cap.cpp | 4 | 76 | 6 |
| cap.h | 4 | 31 | 4 |
| cyclist.h | 4 | 50 | 6 |
| cyclist.hpp | 4 | 137 | 18 |
| hat.cpp | 4 | 25 | 6 |
| hat.h | 4 | 47 | 6 |
| line.cpp | 4 | 45 | 6 |
| line.h | 4 | 27 | 3 |
| main.cpp | 4 | 198 | 25 |
| point.cpp | 4 | 120 | 15 |
| point.h | 4 | 48 | 6 |
| segment.cpp | 4 | 49 | 6 |
| segment.h | 4 | 29 | 4 |
| slist.h | 4 | 54 | 7 |
| slist.hpp | 4 | 171 | 22 |
| visor.cpp | 4 | 45 | 6 |
| visor.h | 4 | 26 | 3 |

Сведения о классах:

* общее количество классов: 8
* количество новых классов: 5

|  |  |  |
| --- | --- | --- |
| Название класса | Длина текста описания класса | Общее количество функций, описанных вне класса |
| Line | 27 | 6 |
| Segment | 29 | 7 |
| Hat | 25 | 5 |
| Visor | 26 | 5 |
| Cap | 31 | 7 |

Сведения о файлах:

* общее количество заголовочных файлов: 8
* общее количество файлов-кодов: 9

## Вывод

Созданные классы соответствуют заданию и спецификации, и корректно выполняют все поставленные задачи. Тестирующая программа полностью работоспособна и позволяет проверить все возможности разработанных классов.

Достоинство программы состоит в том, что она построена на классах и разбита на модули, с разделением описания и реализации. Разделение классов – составных частей изображения (козырек, шапка), и классов – представлений данных частей (линия, сегмент) делает их подходящими для использования в других программах.

## Список литературы

1. Herb Sutter “Exceptional C++: 47 Engineering Puzzles, Programming Problems, and Solutions”, Addison-Wesley Professional, 1999 г. – 240 с.

## Приложение 1

//file: line.h

//line class header

//autor: Tyulin Roman

//date: 20.04.2013

#ifndef my\_line

#define my\_line

#include "point.h"

class Line

{

private:

Point start;

Point end;

public:

Line(Point start, Point end);

Line(const Line &obj);

~Line();

Point GetStart() const;

Point GetEnd() const;

void MoveBy(Point dxdy);

};

#endif

//file: line.cpp

//line class realisation

//autor: Tyulin Roman

//date: 20.04.2013

#include "line.h"

Line::Line(Point start, Point end)

: start(start), end(end)

{

#ifdef TALKY

cout << "Line created!" << endl;

#endif

}

Line::Line(const Line &obj)

: start(obj.start), end(obj.end)

{

#ifdef TALKY

cout << "Line created!" << endl;

#endif

}

Line::~Line()

{

#ifdef TALKY

cout << "Line removed!" << endl;

#endif

}

Point Line::GetStart() const

{

return start;

}

Point Line::GetEnd() const

{

return end;

}

void Line::MoveBy(Point dxdy)

{

start = start.MoveBy(dxdy);

end = end.MoveBy(dxdy);

}

//file: main.cpp

//entery point

//autor: Tyulin Roman

//date: 20.04.2013

#include <iostream.h>

#include <strstream.h>

#include <assert.h>

#include "slist.h"

#include "cap.h"

#include "point.h"

const int HELP = 0;

const int EXIT = 1;

const int ADD = 2;

const int REMOVE = 3;

const int MOVE = 4;

const int SHOW = 5;

int ProcessCommand(int command, istream &args);

void Add(istream &args);

void Remove(istream &args);

void Move(istream &args);

void Show(istream &args);

void Show(const Cap &cap);

void Help();

template<class T>

T Read(istream &is, T \*fake);

SingleList<Cap> caps;

int main()

{

Help();

while (1)

{

char line[255];

cin.getline(line, 255);

istrstream str(line);

int command;

str >> command;

if (!ProcessCommand(command, str))

break;

}

return 0;

}

int ProcessCommand(int command, istream &args)

{

while (!args.eof())

{

switch(command)

{

case HELP:

Help();

return 1;

case EXIT:

return 0;

case ADD:

Add(args);

break;

case REMOVE:

Remove(args);

break;

case MOVE:

Move(args);

break;

case SHOW:

Show(args);

break;

default:

cout << "Wrong command, try 0." << endl;

return 1;

}

}

cout << "Command done." << endl;

return 1;

}

void Add(istream &args)

{

Point p;

args >> p;

if (args.eof())

return;

double w1 = Read(args, (double \*)0);

if (args.eof())

return;

double w2 = Read(args, (double \*)0);

if (args.eof())

return;

double h1 = Read(args, (double \*)0);

if (args.eof())

return;

double h2 = Read(args, (double \*)0);

if (args.eof())

return;

Cap cap(p, w1, w2, h1, h2);

caps.Add(cap);

}

void Remove(istream &args)

{

int index = Read(args, (int \*)0);

if (args.eof())

return;

if (index == -1)

{

caps.Clear();

return;

}

assert(index >= 0 && index < caps.Count());

caps.RemoveAt(index);

}

void Move(istream &args)//{10, 10} 10 5 2 5 {10, 10} 10 5 2 5 {11, 12} 10 6 2 5

{

int index = Read(args, (int \*)0);

if (args.eof())

return;

Point dxdy;

args >> dxdy;

if (args.eof())

return;

if (index == -1)

{

for (int i = 0; i < caps.Count(); ++i)

{

caps.Get(i).MoveBy(dxdy);

}

return;

}

assert(index >=0 && index < caps.Count());

caps.Get(index).MoveBy(dxdy);

}

void Show(istream &args)

{

int index = Read(args, (int \*)0);

if (args.eof())

return;

if (index == -1)

{

for (int i = 0; i < caps.Count(); ++i)

{

cout << "[" << i << "]";

Show(caps.Get(i));

cout << endl;

}

return;

}

assert(index >= 0 && index < caps.Count());

cout << "[" << index << "]";

Show(caps.Get(index));

cout << endl;

}

void Show(const Cap &cap)

{

CycleList<Point> capPoints = cap.GetPoints();

cout << "( ";

for (int i = 0; i < capPoints.Count() - 1; ++i)

{

cout << capPoints.Get(i) << ", ";

}

cout << capPoints.Get(capPoints.Count() - 1) << " )";

}

void Help()

{

cout << "0 - Show this message." << endl

<< "1 - Exit program." << endl

<< "2 - Add new caps in list (pos, w1, w2, h1, h2; w1 > w2)." << endl

<< "3 - Remove caps by indices (-1 to clear)." << endl

<< "4 - Move caps by indices (-1 to move all)." << endl

<< "5 - Show caps by indices (-1 to show all)." << endl;

}

template<class T>

T Read(istream &is, T \*fake)

{

fake = fake;

//assert(!is.eof());

T var;

is >> var;

return var;

}

//file: point.h

//point class header

//autor: Tyulin Roman

//date: 20.04.2013

#ifndef my\_point

#define my\_point

#ifdef BORLAND

#include <iostream.h>

#else

#include <iostream>

#endif

#ifndef BORLAND

using namespace std;

#endif

class Point

{

private:

double x;

double y;

public:

Point();

Point(double x, double y);

Point(double xy);

Point &operator =(const Point &obj);

int operator ==(const Point &obj);

int operator !=(const Point &obj);

int GetX() const;

int GetY() const;

Point MoveBy(Point dxdy);

int operator >(double value) const;

int operator <(double value) const;

friend ostream &operator <<(ostream &os, Point &p);

friend istream &operator >>(istream &is, Point &p);

};

Point \*ReadPoint(istream &is);

void SkipSpaces(istream &is);

#endif

//file: point.cpp

//point class realisation

//autor: Tyulin Roman

//date: 20.04.2013

#include "point.h"

#include <ctype.h>

Point::Point()

: x(0), y(0)

{

#ifdef TALKY

cout << "Point created!" << endl;

#endif

}

Point::Point(double x, double y)

: x(x), y(y)

{

#ifdef TALKY

cout << "Point created!" << endl;

#endif

}

Point::Point(double xy)

: x(xy), y(xy)

{

#ifdef TALKY

cout << "Point created!" << endl;

#endif

}

Point &Point::operator =(const Point &obj)

{

x = obj.x;

y = obj.y;

return \*this;

}

int Point::operator ==(const Point &obj)

{

return x == obj.x && y == obj.y;

}

int Point::operator !=(const Point &obj)

{

return x != obj.x || y != obj.y;

}

int Point::GetX() const

{

return x;

}

int Point::GetY() const

{

return y;

}

int Point::operator >(double value) const

{

return x \* y > value;

}

int Point::operator <(double value) const

{

return x \* y < value;

}

Point Point::MoveBy(Point dxdy)

{

return Point(x + dxdy.x, y + dxdy.y);

}

ostream &operator <<(ostream &os, Point &p)

{

return os << "{ " << p.GetX() << ", " << p.GetY() << " }";

}

istream &operator >>(istream &is, Point &p)

{

Point \*readed = ReadPoint(is);

is.clear(istream::goodbit);

if (!readed)

{

is.clear(istream::eofbit);

return is;

}

p = \*readed;

return is;

}

Point \*ReadPoint(istream &is)

{

SkipSpaces(is);

if(is.peek() != '{')

return 0;

is.get();

SkipSpaces(is);

double x = 0;

is >> x;

SkipSpaces(is);

if(is.peek() != ',')

return 0;

is.get();

SkipSpaces(is);

double y = 0;

is >> y;

SkipSpaces(is);

if(is.peek() != '}')

return 0;

is.get();

return new Point(x, y);

}

void SkipSpaces(istream &is)

{

while(isspace(is.peek()))

is.get();

}

//file: segmrnt.h

//segment class header

//autor: Tyulin Roman

//date: 20.04.2013

#ifndef my\_segment

#define my\_segment

#include "point.h"

class Segment

{

private:

Point start;

double width;

double height;

public:

Segment(Point start, double width, double height);

Segment(const Segment &obj);

~Segment();

Point GetStart() const;

double GetWidth() const;

double GetHeight() const;

void MoveBy(Point dxdy);

};

#endif

//file: segment.cpp

//segment class realisation

//autor: Tyulin Roman

//date: 20.04.2013

#include "segment.h"

Segment::Segment(Point start, double width, double height)

: start(start), width(width), height(height)

{

#ifdef TALKY

cout << "Segment created!" << endl;

#endif

}

Segment::Segment(const Segment &obj)

: start(obj.start), width(obj.width), height(obj.height)

{

#ifdef TALKY

cout << "Segment created!" << endl;

#endif

}

Segment::~Segment()

{

#ifdef TALKY

cout << "Segment removed!" << endl;

#endif

}

Point Segment::GetStart() const

{

return start;

}

double Segment::GetWidth() const

{

return width;

}

double Segment::GetHeight() const

{

return height;

}

void Segment::MoveBy(Point dxdy)

{

start = start.MoveBy(dxdy);

}

//file: visor.h

//visor class header

//autor: Tyulin Roman

//date: 20.04.2013

#ifndef my\_visor

#define my\_visor

#include "cyclist.h"

#include "point.h"

#include "line.h"

class Visor

{

private:

Line line;

public:

Visor(Point position, double dx, double dy);

Visor(const Visor &obj);

~Visor();

CycleList<Point> GetPoints() const;

void MoveBy(Point dxdy);

};

#endif

//file: visor.cpp

//visor class realisation

//autor: Tyulin Roman

//date: 20.04.2013

#include "visor.h"

#include <assert.h>

Visor::Visor(Point position, double dx, double dy)

: line(position, position.MoveBy(Point(dx, dy)))

{

#ifdef TALKY

cout << "Visor created!" << endl;

#endif

}

Visor::Visor(const Visor &obj)

: line(obj.line)

{

#ifdef TALKY

cout << "Visor created!" << endl;

#endif

}

Visor::~Visor()

{

#ifdef TALKY

cout << "Visor removed!" << endl;

#endif

}

CycleList<Point> Visor::GetPoints() const

{

CycleList<Point> list;

list.Add(line.GetStart());

list.Add(line.GetEnd());

return list;

}

void Visor::MoveBy(Point dxdy)

{

line.MoveBy(dxdy);

}

//file: cap.h

//cap class header

//autor: Tyulin Roman

//date: 20.04.2013

#ifndef my\_cap

#define my\_cap

#include "hat.h"

#include "visor.h"

#include "point.h"

#include "cyclist.h"

class Cap

{

private:

Hat hat;

Visor visor;

public:

Cap(Point pos, double w1, double w2, double h1, double h2);

Cap(const Cap &obj);

~Cap();

int operator ==(const Cap &obj);

int operator !=(const Cap &obj);

CycleList<Point> GetPoints() const;

void MoveBy(Point dxdy);

};

#endif

//file: cap.cpp

//Cap class realisation

//autor: Tyulin Roman

//date: 20.04.2013

#include "cap.h"

#include <assert.h>

Cap::Cap(Point pos, double w1, double w2, double h1, double h2)

: hat(pos, w1 - w2, h2), visor(pos, -w2, h1)

{

assert(w1 > w2 && h2 > 0 && w1 > 0 && w2 > 0);

#ifdef TALKY

cout << "Cap created!" << endl;

#endif

}

Cap::Cap(const Cap &obj)

: hat(obj.hat), visor(obj.visor)

{

#ifdef TALKY

cout << "Cap created!" << endl;

#endif

}

Cap::~Cap()

{

#ifdef TALKY

cout << "Cap deleted!" << endl;

#endif

}

int Cap::operator ==(const Cap &obj)

{

CycleList<Point> my = GetPoints();

CycleList<Point> his = obj.GetPoints();

if (my.Count() != his.Count())

return 0;

for (int i = 0; i < my.Count(); ++i)

{

if (my.Get(i) != his.Get(i))

return 0;

}

return 1;

}

int Cap::operator !=(const Cap &obj)

{

return !(\*this == obj);

}

CycleList<Point> Cap::GetPoints() const

{

CycleList<Point> list;

CycleList<Point> hatPoints = hat.GetPoints();

int i = 0;

for (i = 0; i < hatPoints.Count(); ++i)

{

list.Add(hatPoints.Get(i));

}

CycleList<Point> visorPoints = visor.GetPoints();

for (i = 0; i < visorPoints.Count(); ++i)

{

list.Add(visorPoints.Get(i));

}

return list;

}

void Cap::MoveBy(Point dxdy)

{

hat.MoveBy(dxdy);

visor.MoveBy(dxdy);

}

//file: hat.h

//hat class header

//autor: Tyulin Roman

//date: 20.04.2013

#ifndef my\_hat

#define my\_hat

#include "segment.h"

#include "cyclist.h"

class Hat

{

private:

Segment seg;

public:

Hat(Point position, float len, float height);

Hat(const Hat &obj);

~Hat();

CycleList<Point> GetPoints() const;

void MoveBy(Point dxdy);

};

#endif

//file: hat.cpp

//hat class realisation

//autor: Tyulin Roman

//date: 20.04.2013

#include "hat.h"

#include <assert.h>

Hat::Hat(Point position, float len, float height)

: seg(position, len, height)

{

assert(len > 0 && height > 0);

#ifdef TALKY

cout << "Hat created!" << endl;

#endif

}

Hat::Hat(const Hat &obj)

: seg(obj.seg)

{

#ifdef TALKY

cout << "Hat created!" << endl;

#endif

}

Hat::~Hat()

{

#ifdef TALKY

cout << "Hat deleted!" << endl;

#endif

}

CycleList<Point> Hat::GetPoints() const

{

CycleList<Point> list;

list.Add(seg.GetStart());

list.Add(seg.GetStart().MoveBy(Point(seg.GetWidth() / 2, seg.GetHeight())));

list.Add(seg.GetStart().MoveBy(Point(seg.GetWidth(), 0)));

return list;

}

void Hat::MoveBy(Point dxdy)

{

seg.MoveBy(dxdy);

}