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 №3**

Выполнил: Тюлин Р.Д.

Факультет КТИ

Группа № 0304

Преподаватель: Смольянинов А.В.

Санкт-Петербург

2013

## Объектно-ориентированный анализ задания

Основным объектом задания является изображение «Бейсболка мелкая с лентой», образованное из изображения «Бейсболка мелкая» включением в себя изображения «ленты».

Объект «Лента» представлен «Прямоугольником», заданным положением верхней левой вершины, высотой и шириной, что однозначно определяет прямоугольник на плоскости.

Объект «Вид на витрину» представляется «Витриной» и «Набором изображений»:

* «Витрина» представлена «Равнобочной трапецией», заданным положением левой нижней вершины и габаритами (высотой и размерами оснований);
* «Набор изображений» представлен списком, разработанным в работе № 1.2.

Для построения «Бейсболки мелкой с лентой» используется простое наследование сущности «Бейсболки мелкой».

Для построения «Вида на витрину» используется множественное наследование сущностей «Витрина» и «Набор изображений».

## Построение системы объектов.

### Объект «Бейсболка мелкая с лентой»

|  |  |
| --- | --- |
| Лента. | Sticker sticker; |
| Создаёт новую бейсболку с лентой с указанными в описании модели параметрами. | Goodcap(Point pos, double w1, double w2, double h1, double h2); |
| Возвращает опорные точки изображения. | CycleList<Point> GetPoints() const; |
| Перемещает изображение | void MoveBy(Point dxdy); |

### Объект «Лента»

|  |  |
| --- | --- |
| Прямоугольник. | Rectangle rect; |
| Создаёт новую ленту с необходимыми для ее создания параметрами. | Sticker(Point pos, double width, double height); |
| Возвращает опорные точки изображения. | CycleList<Point> GetPoints() const; |
| Перемещает изображение | void MoveBy(Point dxdy); |

### Объект «Прямоугольник»

|  |  |
| --- | --- |
| Левый верхний угол. | Point lu; |
| Правый верхний угол. | Point ru; |
| Левый нижний угол. | Point ld; |
| Правый нижний угол. | Point rd; |
| Создаёт новый прямоугольник с необходимыми для его создания параметрами. | Rectangle(Point lu, double length, double height); |
| Возвращает левый верхний угол. | Point GetLeftUp() const; |
| Возвращает правый верхний угол. | Point GetRightUp() const; |
| Возвращает левый нижний угол. | Point GetLeftDown() const; |
| Возвращает правый нижний угол. | Point GetRightDown() const; |
| Перемещает объект | void MoveBy(Point dxdy); |

* 1. Объект «Вид на витрину»

|  |  |
| --- | --- |
| Создаёт новый вид на витрину с необходимыми для его создания параметрами. | ShowcaseView(Point pos, double height, double down, double top) |
| Возвращает левый верхний угол. | void Add(const Goodcap &cap) |

* 1. Объект «Витрина»

|  |  |
| --- | --- |
| Трапеция. | Trapezium trap; |
| Создаёт новую витрину с необходимыми для ее создания параметрами. | Showcase(Point pos, double height, double down, double top); |
| Перемещает витрину. | void MoveBy(Point dxdy); |
| Возвращает ключевые точки. | CycleList<Point> GetPoints() const; |
| Проверяет, находится ли изображение на витрине, или же за ее пределами. | int Validate(const Goodcap &cap) const; |

## Объект «Трапеция»

|  |  |
| --- | --- |
| Левый верхний угол. | Point lu; |
| Правый верхний угол. | Point ru; |
| Левый нижний угол. | Point ld; |
| Правый нижний угол. | Point rd; |
| Создаёт новую трапецию с необходимыми для ее создания параметрами. | Trapezium(Point pos, double height, double down, double top); |
| Возвращает левый верхний угол. | Point GetLeftUp() const; |
| Возвращает правый верхний угол. | Point GetRightUp() const; |
| Возвращает левый нижний угол. | Point GetLeftDown() const; |
| Возвращает правый нижний угол. | Point GetRightDown() const; |
| Перемещает объект | void MoveBy(Point dxdy); |

## Математические модели объектов

w1

w2

h

sh

sw

pos

Для построения используется декартовая система координат.

pos – координаты всей кепки, шапки и козырька.

w1 – ширина всей кепки (w1 > w2)  
w2 – ширина козврька   
h – высота шапки

sw = 0.6 \* (w1 – w2) – ширина ленты (параметр вычисляется автоматически)

sh = 0.25 \* h – высота ленты (параметр вычисляется автоматически)

## Проектирование классов объектов на языке C++ на основе контрактов

В графах “Предусловия” и “Постусловия” и “Инвариант” предполагается, что значения типа данных float не содержат значений NaN, ±∞ и соответствуют множеству , а значения Point соответствуют множеству .

## Объект «Бейсболка мелкая с лентой»

|  |  |  |  |
| --- | --- | --- | --- |
|  | Имя метода | Предусловия | Постусловия |
| 1 | Goodcap(Point pos, double w1, double w2, double h1, double h2); | w1 > w2, w2 > 0, h1 > 0, h2 > 0 | hat.seg.start = pos  hat.seg.width = w1 – w2  hat.seg.height = h2  visor.line.start = pos  visor.line.end = Point(pos.x – w2, h1)  sticker.rect.pos = pos.MoveBy(Point(0.2 \* (w1 - w2), -0.4 \* h2))  sticker.rect.width = 0.6 \* (w1 - w2)  sticker.rect.height = 0.4 \* h2 |
| 2 | CycleList<Point> GetPoints(); | - | - |
| 3 | void MoveBy(Point dxdy); | - | hat.start = hat.start + dxdy  visor.start = visor.start + dxdy  sticker.rect.pos = sticker.rect.pos + dxdy |

## Объект «Лента»

|  |  |  |  |
| --- | --- | --- | --- |
|  | Имя метода | Предусловия | Постусловия |
| 1 | Sticker(Point pos, double width, double height); | width > 0 height > 0 | rect.pos = pos,  rect.width = width, rect.height = height |
| 4 | CycleList<Point> GetPoints(); | - | - |
| 5 | void MoveBy(Point dxdy); | - | rect.pos = rect.pos + dxdy |

## Объект «Витрина»

|  |  |  |  |
| --- | --- | --- | --- |
|  | Имя метода | Предусловия | Постусловия |
| 1 | Showcase(Point pos, double height, double down, double top); | width > 0 height > 0 | trap.pos = pos,  trap.height = height, trap.top = top, trap.down = down |
| 4 | CycleList<Point> GetPoints(); | - | - |
| 5 | void MoveBy(Point dxdy); | - | trap.pos = trap.pos + dxdy |
| 6 | int Validate(const Goodcap &cap) const; | - | - |

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

Cap.h

cyclist.hpp

Segment.cpp

line.cpp

main.cpp

intlist.hpp

Hat.h

Visor.h

point.h

Hat.cpp

Visor.cpp

point.cpp

Segment.h

Line.h

## 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: 14
* количество новых классов: 6

|  |  |  |
| --- | --- | --- |
| Название класса | Длина текста описания класса | Общее количество функций, описанных вне класса |
| Goodcap | 18 | 3 |
| Rectangle | 22 | 6 |
| Showcase | 24 | 5 |
| ShowcaseView | 24 | 0 |
| Sticker | 18 | 3 |
| Trapezium | 22 | 6 |

Сведения о файлах:

* общее количество заголовочных файлов: 16
* общее количество файлов-кодов: 12

## 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

#ifndef my\_goodcap

#define my\_goodcap

#include "cap.h"

#include "sticker.h"

#include "cyclist.h"

class Goodcap : public Cap

{

private:

Sticker sticker;

public:

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

CycleList<Point> GetPoints() const;

void MoveBy(Point dxdy);

};

#endif

#include "goodcap.h"

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

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

sticker(pos.MoveBy(Point(0.2 \* (w1 - w2), -0.4 \* h2)), 0.6 \* (w1 - w2), 0.4 \* h2)

{

}

CycleList<Point> Goodcap::GetPoints() const

{

CycleList<Point> result = Cap::GetPoints();

CycleList<Point> stickerPoints = sticker.GetPoints();

int i;

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

{

result.Add(stickerPoints.Get(i));

}

return result;

}

void Goodcap::MoveBy(Point dxdy)

{

Cap::MoveBy(dxdy);

sticker.MoveBy(dxdy);

}

#ifndef my\_rectangle

#define my\_rectangle

#include "point.h"

class Rectangle

{

private:

Point lu;

Point ru;

Point ld;

Point rd;

public:

Rectangle(Point lu, double length, double height);

void MoveBy(Point dxdy);

Point GetLeftUp() const;

Point GetRightUp() const;

Point GetLeftDown() const;

Point GetRightDown() const;

};

#endif

#include "rectangl.h"

#include <assert.h>

Rectangle::Rectangle(Point p, double width, double height)

: lu(p), ru(p.MoveBy(Point(width, 0))), ld(p.MoveBy(Point(0, height))), rd(p.MoveBy(Point(width, height)))

{

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

}

void Rectangle::MoveBy(Point dxdy)

{

lu.MoveBy(dxdy);

ru.MoveBy(dxdy);

ld.MoveBy(dxdy);

rd.MoveBy(dxdy);

}

Point Rectangle::GetLeftUp() const

{

return lu;

}

Point Rectangle::GetRightUp() const

{

return ru;

}

Point Rectangle::GetLeftDown() const

{

return ld;

}

Point Rectangle::GetRightDown() const

{

return rd;

}

#ifndef my\_showcase

#define my\_showcase

#include "trap.h"

#include "cyclist.h"

#include "goodcap.h"

class Showcase

{

private:

Trapezium trap;

double k1;

double k2;

double b1;

double b2;

int Inside(const Point &p) const;

public:

Showcase(Point pos, double height, double down, double top);

void MoveBy(Point dxdy);

CycleList<Point> GetPoints() const;

int Validate(const Goodcap &cap) const;

};

#endif

#include "showcase.h"

Showcase::Showcase(Point pos, double height, double down, double top)

: trap(pos, height, down, top)

{

k1 = 2 \* height / (down - top);

b1 = pos.GetY() - pos.GetX() \* k1;

k2 = -k1;

b2 = b1 + k1 \* down;

}

void Showcase::MoveBy(Point dxdy)

{

trap.MoveBy(dxdy);

}

int Showcase::Inside(const Point &p) const

{

if (p.GetY() < trap.GetLeftDown().GetY())

return 0;

if (p.GetY() > trap.GetLeftUp().GetY())

return 0;

if (p.GetY() > k1 \* p.GetX() + b1)

return 0;

if (p.GetY() > k2 \* p.GetX() + b2)

return 0;

return 1;

}

CycleList<Point> Showcase::GetPoints() const

{

CycleList<Point> result;

result.Add(trap.GetLeftUp());

result.Add(trap.GetRightUp());

result.Add(trap.GetLeftDown());

result.Add(trap.GetRightDown());

return result;

}

int Showcase::Validate(const Goodcap &cap) const

{

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

int i;

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

{

if (!Inside(capPoints.Get(i)))

return 0;

}

return 1;

}

#ifndef my\_showview

#define my\_showview

#include "cyclist.h"

#include "showcase.h"

class ShowcaseView : public Showcase, public CycleList<Goodcap>

{

public:

ShowcaseView(Point pos, double height, double down, double top)

: Showcase(pos, height, down, top)

{

}

void Add(const Goodcap &cap)

{

if (!Validate(cap))

return;

CycleList<Goodcap>::Add(cap);

}

};

#endif

#ifndef my\_sticker

#define my\_sticker

#include "rectangl.h"

#include "point.h"

#include "cyclist.h"

class Sticker

{

private:

Rectangle rect;

public:

Sticker(Point pos, double width, double height);

CycleList<Point> GetPoints() const;

void MoveBy(Point dxdy);

};

#endif

#include "sticker.h"

Sticker::Sticker(Point pos, double width, double height)

: rect(pos, width, height)

{

}

CycleList<Point> Sticker::GetPoints() const

{

CycleList<Point> result;

result.Add(rect.GetLeftUp());

result.Add(rect.GetRightUp());

result.Add(rect.GetLeftDown());

result.Add(rect.GetRightDown());

return result;

}

void Sticker::MoveBy(Point dxdy)

{

rect.MoveBy(dxdy);

}

#ifndef my\_trap

#define my\_trap

#include "point.h"

class Trapezium

{

private:

Point lu;

Point ru;

Point ld;

Point rd;

public:

Trapezium(Point pos, double height, double down, double top);

void MoveBy(Point dxdy);

Point GetLeftUp() const;

Point GetRightUp() const;

Point GetLeftDown() const;

Point GetRightDown() const;

};

#endif

#include "trap.h"

Trapezium::Trapezium(Point pos, double height, double down, double top)

: lu(pos.MoveBy(Point(down / 2 - top / 2, height))), ru(pos.MoveBy(Point(down / 2 + top / 2, height))),

ld(pos), rd(pos.MoveBy(Point(down, 0)))

{

}

void Trapezium::MoveBy(Point dxdy)

{

lu.MoveBy(dxdy);

ru.MoveBy(dxdy);

ld.MoveBy(dxdy);

rd.MoveBy(dxdy);

}

Point Trapezium::GetLeftUp() const

{

return lu;

}

Point Trapezium::GetRightUp() const

{

return ru;

}

Point Trapezium::GetLeftDown() const

{

return ld;

}

Point Trapezium::GetRightDown() const

{

return rd;

}

# Тестирование по методу черного ящика: