федеральное ГОСУДАРСТВЕННОЕ Бюджетное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

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

Кафедра МОЭВМ

**Отчет по курсовой работе.**

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».

## Изменения классов для реализации принципа подстановки.

### Объект «Бейсболка мелкая с лентой»

|  |  |  |
| --- | --- | --- |
| Описание члена класса | Объявление в ЛР 3 | Измененное объявление |
| Список, содержащий проблемные объекты | CycleList<Goodcap> | CycleList<Cap \*> |
| Метод добавления объекта на вид витрины | void Add(const Goodcap &cap) | void Add(Cap \* const &cap) |

## 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

//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);

virtual ~Cap();

int operator ==(const Cap &obj);

int operator !=(const Cap &obj);

virtual CycleList<Point> GetPoints() const;

virtual 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);

}

#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);

virtual ~Goodcap();

virtual CycleList<Point> GetPoints() const;

virtual 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)

{

}

Goodcap::~Goodcap()

{

}

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\_showview

#define my\_showview

#include "cyclist.h"

#include "showcase.h"

class ShowcaseView : public Showcase, public CycleList<Cap \*>

{

public:

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

: Showcase(pos, height, down, top)

{

}

void Add(Cap \* const &cap)

{

if (!Validate(\*cap))

return;

CycleList<Cap \*>::Add(cap);

}

};

#endif

//file: main.cpp

//entry point

//author: Tyulin Roman

//date: 20.04.2013

#include <iostream.h>

#include <strstream.h>

#include <assert.h>

#include <conio.h>

#include "slist.h"

#include "cap.h"

#include "point.h"

#include "showview.h"

const int HELP = 0;

const int EXIT = 1;

const int ADD = 2;

const int ADD\_ = 3;

const int REMOVE = 4;

const int MOVE = 5;

const int SHOW = 6;

int ProcessCommand(int command, istream &args);

void Add(istream &args);

void AddSticker(istream &args);

void Remove(istream &args);

void Move(istream &args);

void Move(Cap \*cap, Point dxdy);

void Show(istream &args);

void Show(const Cap \*cap);

void Help();

void GenerateShowcaseView();

template<class T>

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

ShowcaseView \*view;

int main()

{

clrscr();

cout << "Caps manipulator." << endl

<< "created by Tyulin Roman(c)" << endl;

GenerateShowcaseView();

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 = new Cap(p, w1, w2, h1, h2);

if (!view->Validate(\*cap))

{

cout << "Wrong arguments. Cap is out of range." << endl;

return;

}

view->Add(cap);

}

void AddSticker(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;

Goodcap \*cap = new Goodcap(p, w1, w2, h1, h2);

if (!view->Validate(\*cap))

{

cout << "Wrong arguments. Cap is out of range." << endl;

return;

}

view->Add(cap);

}

void Remove(istream &args)

{

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

if (args.eof())

return;

if (index == -1)

{

view->Clear();

return;

}

assert(index >= 0 && index < view->Count());

Cap \*toDelete = view->Get(index);

delete toDelete;

view->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 < view->Count(); ++i)

{

Move(view->Get(i), dxdy);

}

return;

}

assert(index >=0 && index < view->Count());

Move(view->Get(index), dxdy);

}

void Move(Cap \*cap, Point dxdy)

{

cap->MoveBy(dxdy);

if (!view->Validate(\*cap))

{

cout << "Wrong parameters. Cap is out of range." << endl;

cap->MoveBy(Point(-dxdy.GetX(), -dxdy.GetY()));

return;

}

}

void Show(istream &args)

{

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

if (args.eof())

return;

if (index == -1)

{

for (int i = 0; i < view->Count(); ++i)

{

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

Show(view->Get(i));

cout << endl;

}

return;

}

assert(index >= 0 && index < view->Count());

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

Show(view->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 << HELP << " - Show this message." << endl

<< EXIT << " - Exit program." << endl

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

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

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

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

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

}

void GenerateShowcaseView()

{

cout << "input showcase view parameters: (position, height, down side, top side)" << endl;

Point pos;

cin >> pos;

double h;

cin >> h;

double d;

cin >> d;

double t;

cin >> t;

view = new ShowcaseView(pos, h, d, t);

}

template<class T>

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

{

fake = fake; //не стирать, руки оторву!

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

T var;

is >> var;

return var;

}

# Тестирование по методу черного ящика: