МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ

по практической работе №7

по дисциплине «Объектно-ориентированное программирование»

Студенты гр. 8363 _____ Нерсисян А.С.

_____ Панфилович А.И.

Преподаватель _____ Егоров С.С.

Санкт-Петербург

2021

Задание на практическую работу

В примере к работе №7 решается задача, аналогичная задаче работы №6, дополненная тем, что вершины многоугольника нумеруются и среди них выделяется активная.

Это реализуется созданием класса-наследника многоугольника (отношение обобщения). В нем вводится атрибут активной вершины и метод ее изменения. Его формальный параметр имеет тип bool, что говорит о по крайней мере двух стратегиях ее обхода вершин. В примере - это по часовой (true) и против часовой стрелки.

Класс холста дополнен обработчиком событий нажатия кнопки мыши. В нем анализируется, какая кнопка нажата, правая или левая. В зависимости от этого изменяется и отрисовывается активная вершина многоугольника.

В приложении, которое требуется реализовать, иерархия TSample <- TDerivedSample соответствует иерархии "граф" <- "граф событий", а нажатие мыши - генерации 2-х событий. События, соответствующие вашему графу событий, должны у вас формироваться на у холста, а в интерфейсе, на котором размещается холст с визуальным представлением текущего состояния графа событий.

Требуется реализовать приложение, которое рисует правильный многоугольник по заданным параметрам. Необходимые параметры следует задавать в интерфейсе. а сам холст должен быть встроен в этот интерфейс.

Реализовать и отладить программу, удовлетворяющие сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программы. Оформить отчет, сделать выводы по работе.

Спецификации классов

Класс DrawPoly

Атрибуты:

Тип	Наименование	Область видимости
int	count	protected

Класс DrawPoly реализовывает отрисовку правильного многоугольника.

Конструктор DrawPoly**(int)** имеет 1 формальный параметр типа int, область видимости public, реализовывает создание объекта и задает число углов многоугольника.

Метод **draw(QPainter*, QRect, QColor)** не возвращает никакие параметры (void), имеет 3 формальных параметров, область видимости public, реализовывает отрисовку многоугольника по 3-ем параметрам: указатель на объект **QPainter**, по параметрам окна и цвету для отрисовки, (число углов задается при создании объекта в конструкторе).

Класс DeriveSample

Атрибуты:

Тип	Наименование	Область видимости
int	active	private

Класс DeriveSample является наследником класса DrawPoly, реализовывает отрисовку активной вершины правильного многоугольника.

Класс Widget

Атрибуты:

Тип	Наименование	Область видимости
DeriveSample*	s	private

Класс Widget реализовывает графический интерфейс программы.

Скриншоты программы на контрольных примерах

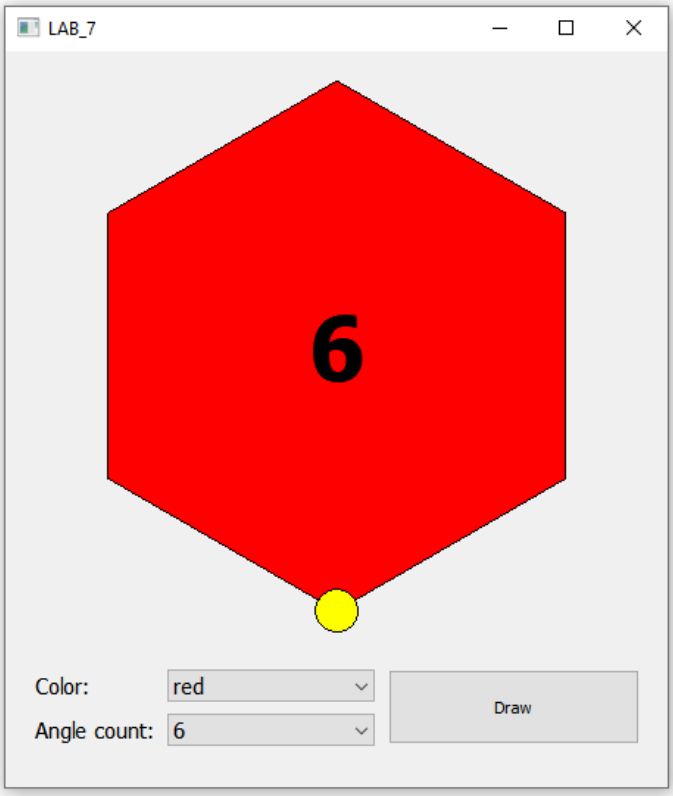

Рисунок 2 – Интерфейс программы

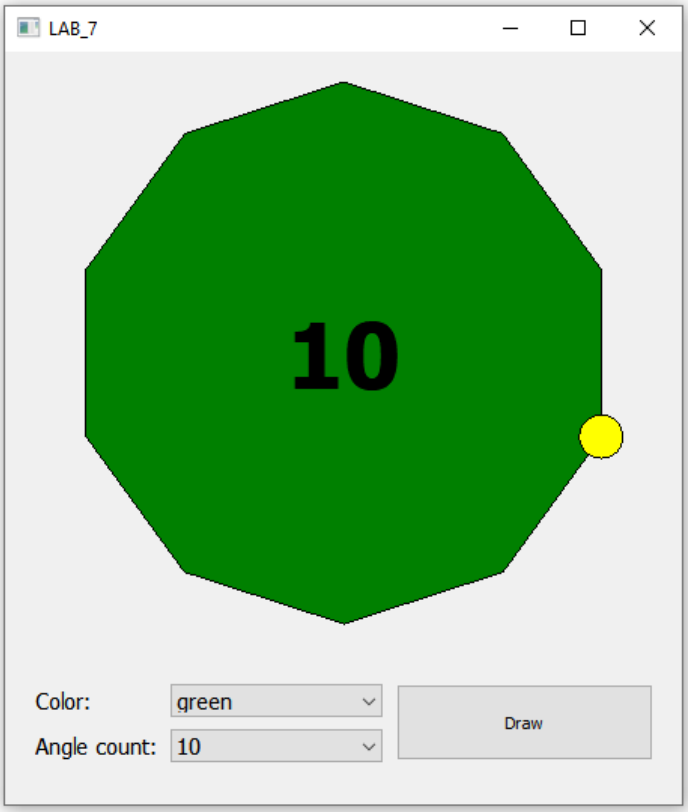

Рисунок 3 – Интерфейс программы

Выводы по выполнению работы

В рамках данной практической работы была реализована и отлажена программа, удовлетворяющая сформулированным требованиям и заявленным целям. Разработаны контрольные примеры, и программа оттестирована на них.

ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ КОД

FILE LAB_7.pro

```
QT       += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
TARGET = painter
TEMPLATE = app
CONFIG += c++11
SOURCES += \
    derivesample.cpp \
    drawpoly.cpp \
    main.cpp \
    widget.cpp
HEADERS += \
    derivesample.h \
    drawpoly.h \
    widget.h
FORMS += \
    widget.ui
```

FILE drawpoly.h

```
#ifndef DRAWPOLY_H
#define DRAWPOLY_H

#include <QPainter>

class DrawPoly
{
protected:
    int count;
public:
    DrawPoly(int);
    ~DrawPoly() = default;
    void draw(QPainter*,QRect,QColor);
};


#endif // DRAWPOLY_H
```

FILE drawpoly.cpp

```
#include "drawpoly.h"
#include <math.h>

DrawPoly::DrawPoly(int n)
{
    count=n;
}

void DrawPoly::draw(QPainter* p, QRect r, QColor c)
{
    qreal cw = 0.5*r.width();
    qreal ch = 0.4*r.height();
    qreal cr = 0.9*(cw>ch?ch:cw);
    qreal a = 2.0*acos(-1.0)/count;
    QPointF *t = new QPointF[count];
    for (int i=0; i<count; i++)
    {
        t[i] = QPointF(cw+cr*sin(i*a),ch-cr*cos(i*a));
    }
    p->setPen(QPen(Qt::black));
    p->setBrush(QBrush(c));
    p->drawPolygon(t,count);
    QFont font;
    qreal cf = 0.25*cr;
    font.setPointSize(cf);
    font.setBold(true);
    p->setFont(font);
    p->drawText(QRectF(cw-cf,ch-cf,2.0*cf,2.0*cf),
                QString().setNum(count),
                QTextOption(Qt::AlignCenter));
    delete [] t;
}
```

FILE derivesample.h

```
#ifndef DERIVESAMPLE_H
#define DERIVESAMPLE_H
#include "drawpoly.h"

class DeriveSample : public DrawPoly
{
    int active;
public:
    DeriveSample(int);
    void draw(QPainter*,QRect,QColor);
    void newEvent(bool);
};
#endif // DERIVESAMPLE_H
```

FILE derivesample.cpp

```
#include "derivesample.h"
#include <math.h>

DeriveSample::DeriveSample(int n) : DrawPoly(n)
{
    active = 0;
}

void DeriveSample::draw(QPainter* p, QRect r, QColor c)
{
    DrawPoly::draw(p,r,c);
    qreal cw = 0.5*r.width();
    qreal ch = 0.4*r.height();
    qreal cr = 0.9*(cw>ch?ch:cw);
    qreal a = 2.0*acos(-1.0)/count;
    QPointF t(cw+cr*sin(active*a),ch-cr*cos(active*a));
    p->setPen(QPen(Qt::black));
    p->setBrush(QBrush(Qt::yellow));
    p->drawEllipse(t,0.08*cr,0.08*cr);
}

void DeriveSample::newEvent(bool direction)
{
    if (direction) active++;
    else active = --active + count;
    active = active % count;
}
```

FILE widget.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include <QPainter>
#include <QMouseEvent>
#include <drawpoly.h>
#include "derivesample.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Widget; }
QT_END_NAMESPACE

class Widget : public QWidget
{
    Q_OBJECT
```

```
    DeriveSample* s;

public:
    explicit Widget(QWidget *parent = nullptr);
    ~Widget();

protected:
    void paintEvent(QPaintEvent *event);
    void mousePressEvent(QMouseEvent*);

private slots:
    void on_DrawButton_clicked();

private:
    Ui::Widget *ui;
};
#endif // WIDGET_H
```

FILE widget.cpp

```
#include "widget.h"
#include "ui_widget.h"

Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);
    s = new DeriveSample(ui->AngleCountCB->currentText().toInt());

    setWindowTitle("LAB_7");
}

Widget::~Widget()
{
    delete ui;
}

void Widget::paintEvent(QPaintEvent *event)
{
    Q_UNUSED(event);
    QPainter p;
    p.begin(this);
    s->draw(&p,rect(),ui->ColorCB->currentText());
    p.end();
}

void Widget::mousePressEvent(QMouseEvent *event)
```

```
{
    if (event->button() == Qt::LeftButton)  s->newEvent(false);
    if (event->button() == Qt::RightButton) s->newEvent(true);
    update();
}

void Widget::on_DrawButton_clicked()
{
    s = new DeriveSample(ui->AngleCountCB->currentText().toInt());
    repaint();
}
```

FILE main.cpp

```
#include "widget.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Widget w;
    w.show();
    return a.exec();
}
```

FILE widget.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Widget</class>
 <widget class="QWidget" name="Widget">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>450</width>
    <height>500</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Widget</string>
  </property>
  <widget class="QPushButton" name="DrawButton">
   <property name="geometry">
```

```
  <rect>
   <x>260</x>
   <y>420</y>
   <width>171</width>
   <height>51</height>
  </rect>
 </property>
 <property name="text">
  <string>Draw</string>
 </property>
</widget>
<widget class="QComboBox" name="ColorCB">
 <property name="geometry">
  <rect>
   <x>110</x>
   <y>420</y>
   <width>141</width>
   <height>22</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <pointsize>11</pointsize>
  </font>
 </property>
 <item>
  <property name="text">
   <string>red</string>
  </property>
 </item>
 <item>
  <property name="text">
   <string>green</string>
  </property>
 </item>
 <item>
  <property name="text">
   <string>blue</string>
  </property>
 </item>
 <item>
  <property name="text">
   <string>brown</string>
  </property>
 </item>
 <item>
  <property name="text">
   <string>yellow</string>
  </property>
 </item>
```

```
  <item>
   <property name="text">
    <string>black</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>white</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>purple</string>
   </property>
  </item>
 </widget>
 <widget class="QLabel" name="ColorLabel">
  <property name="geometry">
   <rect>
    <x>20</x>
    <y>420</y>
    <width>47</width>
    <height>21</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <pointsize>11</pointsize>
   </font>
  </property>
  <property name="text">
   <string>Color:</string>
  </property>
 </widget>
 <widget class="QComboBox" name="AngleCountCB">
  <property name="geometry">
   <rect>
    <x>110</x>
    <y>450</y>
    <width>141</width>
    <height>22</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <pointsize>11</pointsize>
   </font>
  </property>
  <item>
   <property name="text">
```

```
  <string>3</string>
 </property>
</item>
<item>
 <property name="text">
  <string>4</string>
 </property>
</item>
<item>
 <property name="text">
  <string>5</string>
 </property>
</item>
<item>
 <property name="text">
  <string>6</string>
 </property>
</item>
<item>
 <property name="text">
  <string>7</string>
 </property>
</item>
<item>
 <property name="text">
  <string>8</string>
 </property>
</item>
<item>
 <property name="text">
  <string>9</string>
 </property>
</item>
<item>
 <property name="text">
  <string>10</string>
 </property>
</item>
<item>
 <property name="text">
  <string>11</string>
 </property>
</item>
<item>
 <property name="text">
  <string>12</string>
 </property>
</item>
<item>
 <property name="text">
```

```
      <string>13</string>
     </property>
    </item>
    <item>
     <property name="text">
      <string>14</string>
     </property>
    </item>
    <item>
     <property name="text">
      <string>15</string>
     </property>
    </item>
    <item>
     <property name="text">
      <string>16</string>
     </property>
    </item>
   </widget>
   <widget class="QLabel" name="AngleCountLabel">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>450</y>
      <width>91</width>
      <height>21</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <pointsize>11</pointsize>
     </font>
    </property>
    <property name="text">
     <string>Angle count:</string>
    </property>
   </widget>
  </widget>
  <resources/>
  <connections/>
 </ui>
```