Правительство Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Кафедра «Компьютерная безопасность»

Отчет
к лабораторной работе №4

по дисциплине

«Языки программирования»

Работу выполнил
студент группы СКБ-201 _____ А.Н. Ушаков
подпись, дата

Работу проверил _____ С.А. Булгаков
подпись, дата

Москва 2021

Содержание

Постановка задачи

Общая часть

Разработать графическое приложение с использованием библиотеки Qt. Приложение состоит из основного окна (наследовать QMainWindow), с панелью инструментов на которой расположены:

1) Залипающие кнопки с выбором типа фигуры *(количество кнопок соответствует количеству фигур в варианте).*

2) Кнопка добавления фигуры. Все фигуры поворачиваются относительно центра описывающего их прямоугольника (по умолчанию против часовой стрелки).
Параметры фигуры выбираются произвольно в допустимом диапазоне. Если ни одна фигура не выбрана, кнопка добавления не активна.

3)Кнопка удаления выделенной фигуры. Если фигура не выделена, кнопка не активна.
Основная часть окна предназначена для размещения фигур (запрещается использовать QGraphicsScene). Фон основной части окна – белый, цвет отрисовки фигур – черный.
При нажатии на фигуре **левой** кнопкой мыши – фигура **выделяется (отрисовывается синим цветом)**. При нажатии на фигуре **правой** кнопкой мыши – открывается всплывающее меню. Меню должно содержать пункты «Удалить» и «Изменить».
При выборе пункта «Изменить» – открывается модальное диалоговое окно, позволяющее изменить параметры фигуры, угол поворота, направление поворота, а также отображающее площадь и периметр фигуры.
При **перетаскивании выделенной** фигуры она меняет свое положение в рамках окна.
При достижении края окна перемещение прекращается. Пересечение с другими фигурами не учитывается.

1. Алгоритм решения задачи

Для решения поставленных задачи были разработаны классы DialogMenu, Figure и MainWindow. Внутри каждого класса были созданы закрытые поля базового типа вместе с закрытыми функциями-членов (методов) класса и открытые функции-члены (методы) класса.

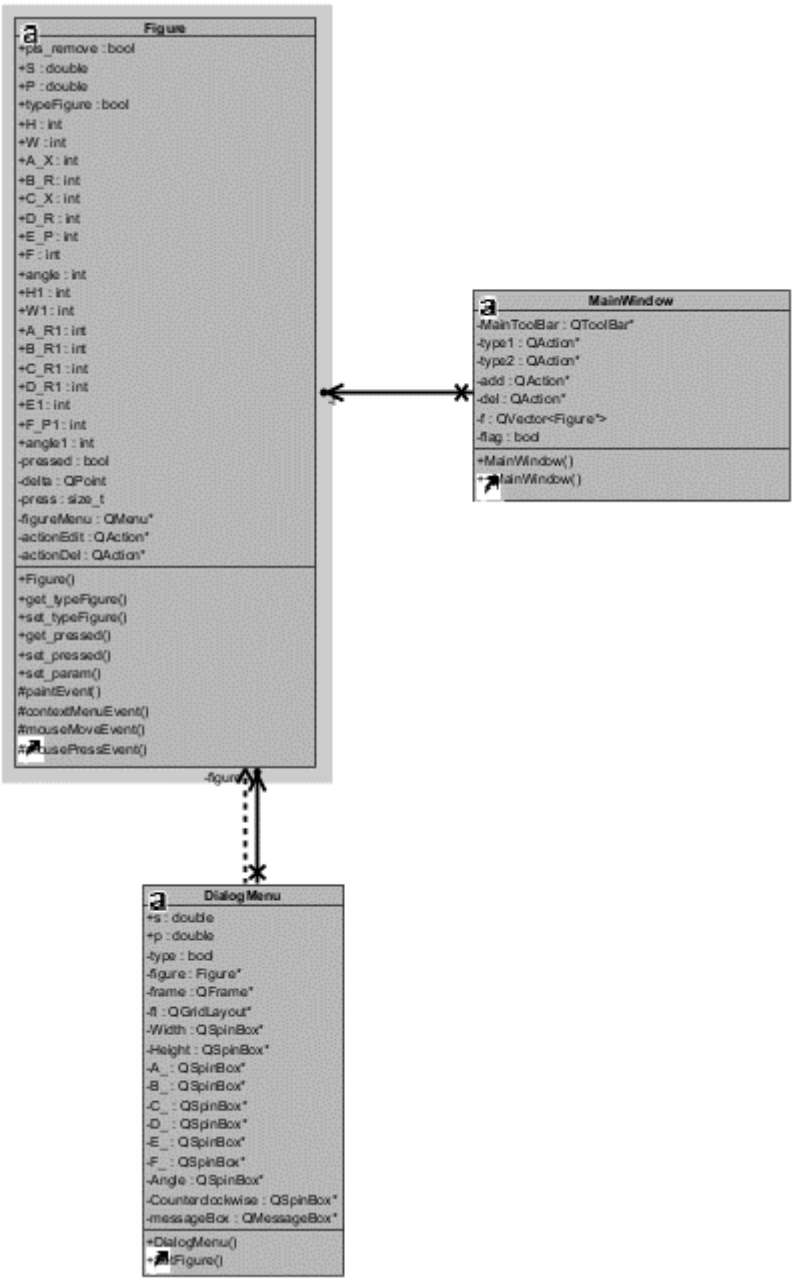


Рис. 1.0: UML-диаграмма классов

1.1 DialogMenu

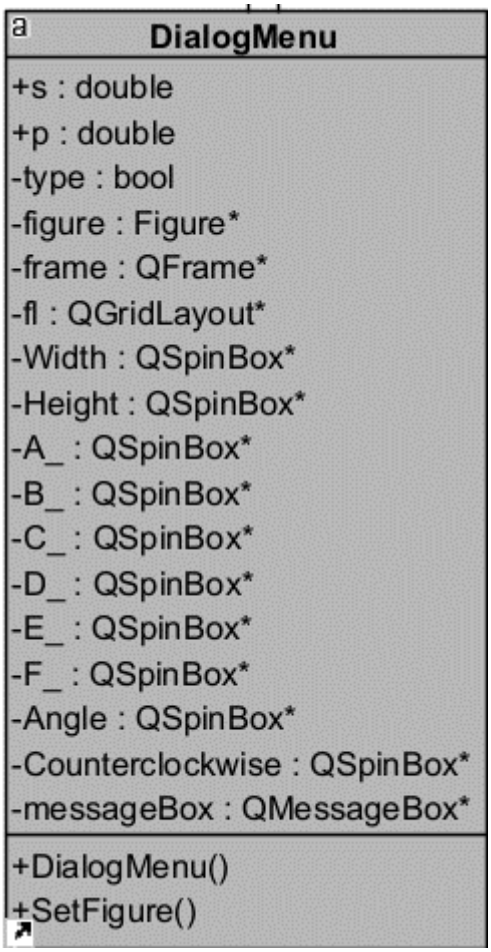


Рис. 1.1: UML-диаграмма класса DialogMenu

1.1.1 Конструкторы

1) DialogMenu()

1.1.2 Методы класса

1) void SetFigure()

1.1.3 Слоты

1) void accept()

1.2 Figure

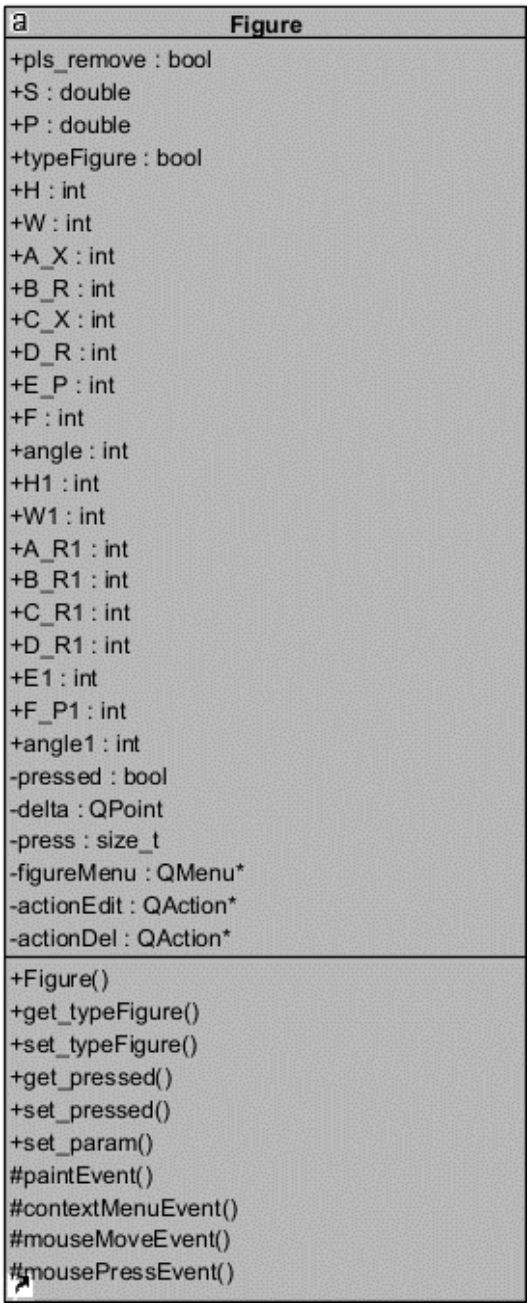


Рис. 1.2: UML-диаграмма класса Figure

1.2.1 Конструкторы

1) Figure()
2) ~Figure()

1.2.2 Методы класса

1) void paintEvent(QPaintEvent *)
2) void contextMenuEvent(QContextMenuEvent *event)
3) void mouseMoveEvent(QMouseEvent *event)
4) void mousePressEvent(QMouseEvent* event)

1.2.3 Методы доступа к компонентам класса

1) bool get_typeFigure() const
2) void set_typeFigure(bool a)
3) bool get_pressed()
4) void set_pressed(bool)
5) void set_param(int, int, int, int, int, int, int, int, int, int)

1.2.4 Слоты

1) void delFigure()
2) void editFigure()

1.3 MainWindow

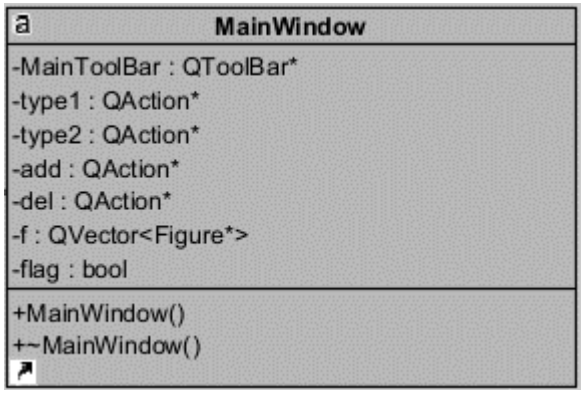


Рис. 1.3: UML-диаграмма класса MainWindow

1.3.1 Конструкторы

1) MainWindow()
2) ~MainWindow()

1.3.2 Слоты

1) void check_widget()
2) void pick1type()
3) void pick2type()
4) void addFigure()
5) void delFigure()

2. Выполнение задач

2.1 Класс DialogMenu

2.1.1 Конструкторы

DialogMenu(QWidget *parent) – конструктор, который создаёт диалоговое окно с полями для ввода данных, а также messageBox, который открывается при неудачном вводе данных в поля в данное диалоговое окно.

~DialogMenu() – деструктор по умолчанию, который реализуется поведением компилятора. В случае не ввода данных, поля будут заполнены автоматически компилятором.

2.1.2 Методы класса

void SetFigure(Figure *f) – **метод класса, который заполняет поля текущего класса данными, а также выполняющий расчёт периметра и площади переданного экземпляра класса Figure с последующим добавлением виджета в рамку.**

2.1.3 Слоты

void accept() – публичный слот класса, который выполняет проверку введенных данных в рамке. В случае неудачного ввода выдает окно с ошибкой. В случае удачного заполнения полей в рамке, данные передаются экземпляру класса Figure и происходит перерисовка данной фигуры с новыми введенными значениями.

2.2 Класс Figure

2.2.1 Конструкторы

Figure() – конструктор класса, добавляющий кнопки в контекстное меню.

~Figure() – – деструктор по умолчанию, который реализуется поведением компилятора. В случае не ввода данных, поля будут заполнены автоматически компилятором.

2.2.2 Методы класса

void paintEvent(QPaintEvent *) – защищенный метод класса, реализующий отрисовку фигуры в зависимости от значения в поле typeFigure.

void contextMenuEvent(QContextMenuEvent *event) – защищенный метод класса, реализующий открытие контекстного меню у данного экземпляра класса.

void mouseMoveEvent(QMouseEvent *event) - защищенный метод класса, реализующий перемещение данного экземпляра класса.

void mousePressEvent(QMouseEvent* event) - - защищенный метод класса, реализующий

выделение экземпляра данного класса.

2.2.3 Методы доступа к компонентам класса

bool get_typeFigure() const - метод класса (геттер), который возвращает значение открытого поля класса, а именно поля - typeFigure.

void set_typeFigure(bool a) - метод класса (сеттер), который устанавливает значение открытого поля класса, а именно поля - typeFigure.

bool get_pressed() - метод класса (геттер), который возвращает значение закрытого поля класса, а именно поля - pressed.

void set_pressed(bool) - метод класса (сеттер), который устанавливает значение закрытого поля класса, а именно поля - pressed.

void set_param(int, int, int, int, int, int, int, int, int, int) - метод класса (сеттер), который устанавливает значение закрытых полей класса, а именно полей - H, W, A_X, B_R, C_X, D_R, E_P, F, angle, если typeFigure равен 0, или H1, W1, A_R1, B_R1, C_R1, D_R1, E1, F_P1, angle1, если typeFigure равен 1.

2.2.4 Слоты

void delFigure() – закрытый слот, устанавливающий значение открытому полю класса - pls_remove - равное 1.

void editFigure() – закрытый слот, создающий экземпляр класса DialogMenu, с дальнейшим вызовом диалоговое окна.

2.3 Класс MainWindow

2.3.1 Конструкторы

MainWindow() – конструктор класса, который создаёт панель инструментов с таймером, для обновления свойств кнопок на данной панели инструментов.

~MainWindow() – деструктор класса, который занимается очищением вектора типа Figure.

2.3.2 Слоты

void check_widget() – закрытый слот класса, который вызывается каждые полсекунды для проверки соответствующих полей экземпляров класса Figure, отвечающих за удаление данных экземпляров, и соответственно для обновления активности кнопок на панели инструментов.

void pick1type() – закрытый слот класса, который выполняет активацию соответствующих кнопок на панели задач и соответствующего заполнения полей данного класса.

void pick2type() – закрытый слот класса, который выполняет активацию соответствующих кнопок на панели задач и соответствующего заполнения полей данного класса.

void addFigure() – закрытый слот класса, который отвечает за создание экземпляров класса Figure и добавления их в вектор.

void delFigure() – закрытый слот класса, отвечающий за закрытие экземпляров класса Figure и удаления их из вектора.

3. Получение исполняемых модулей

С помощью системы сборки qmake производится компоновка (сборка) исполняемого модуля из объектных модулей. Объектные модули – файлы, которые уже могут быть запущены, получаются, соответственно, из файлов исходного кода.

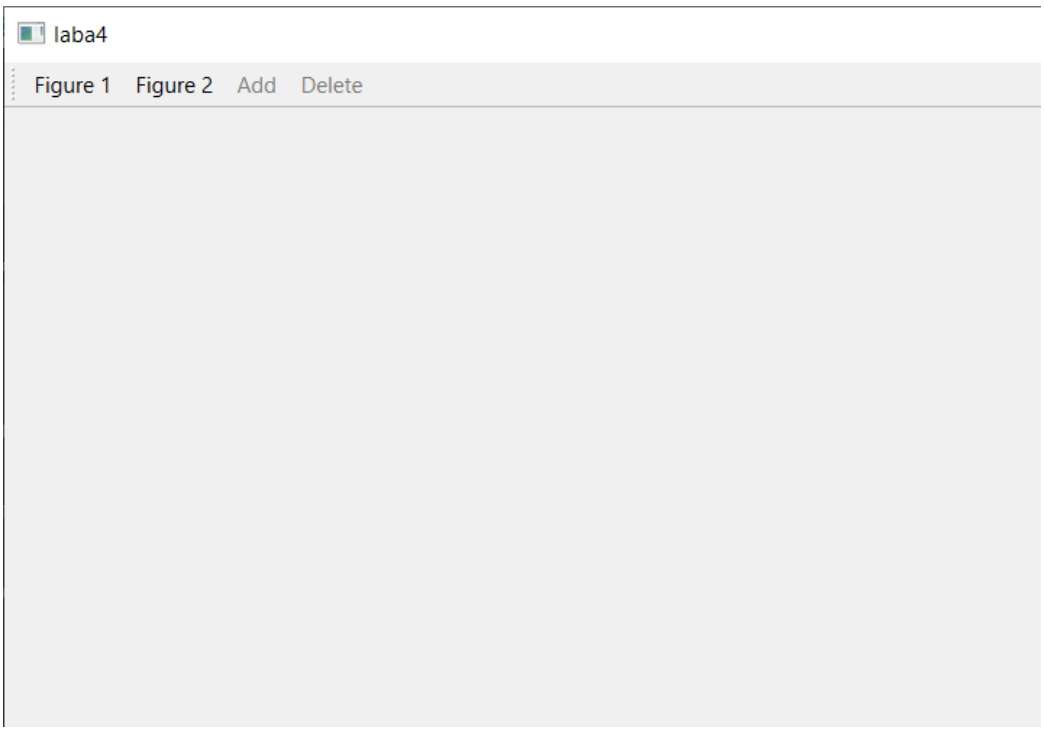
Листинг-1 laba4.pro

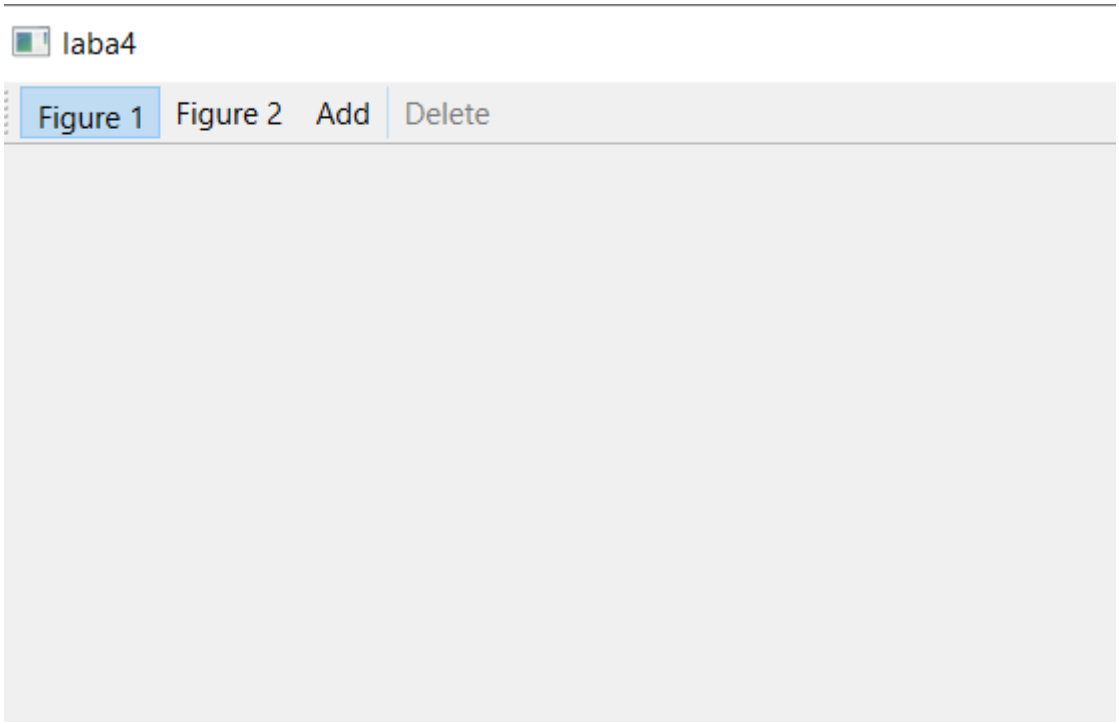
```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs
deprecated before Qt 6.0.0

SOURCES += \
    DialogMenu.cpp \
    Figure.cpp \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    DialogMenu.h \
    Figure.h \
    mainwindow.h

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

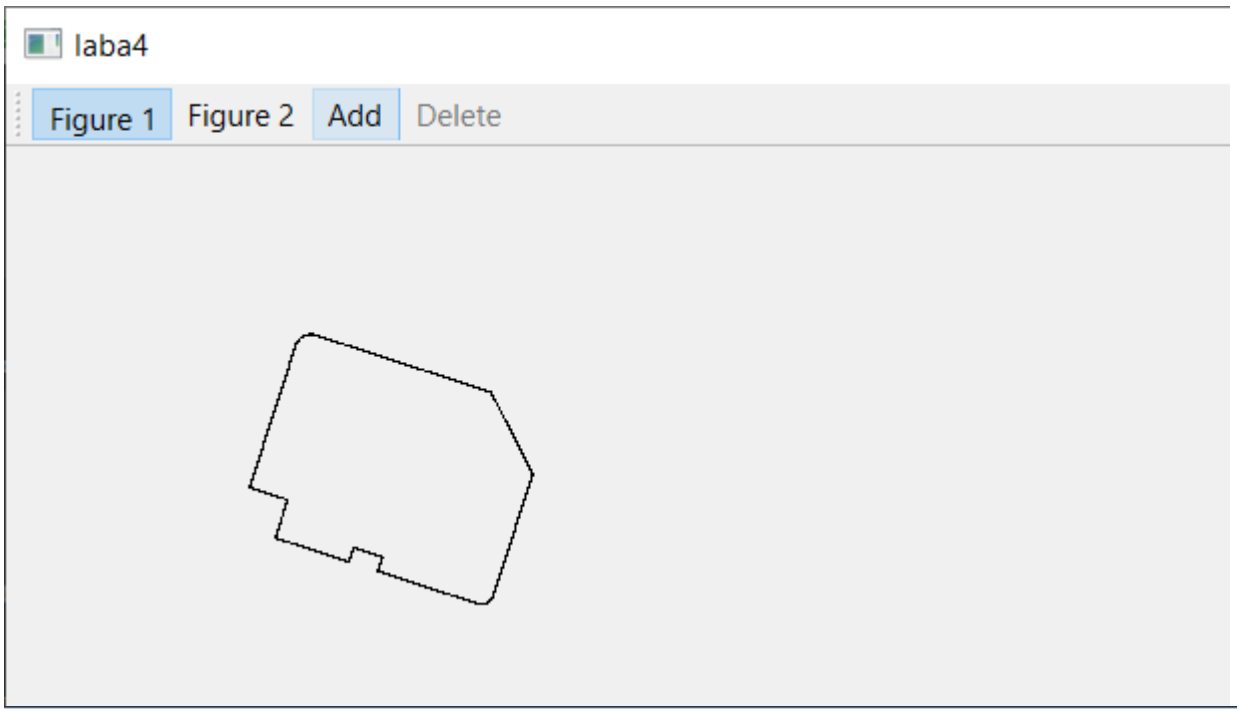
4. Тестирование

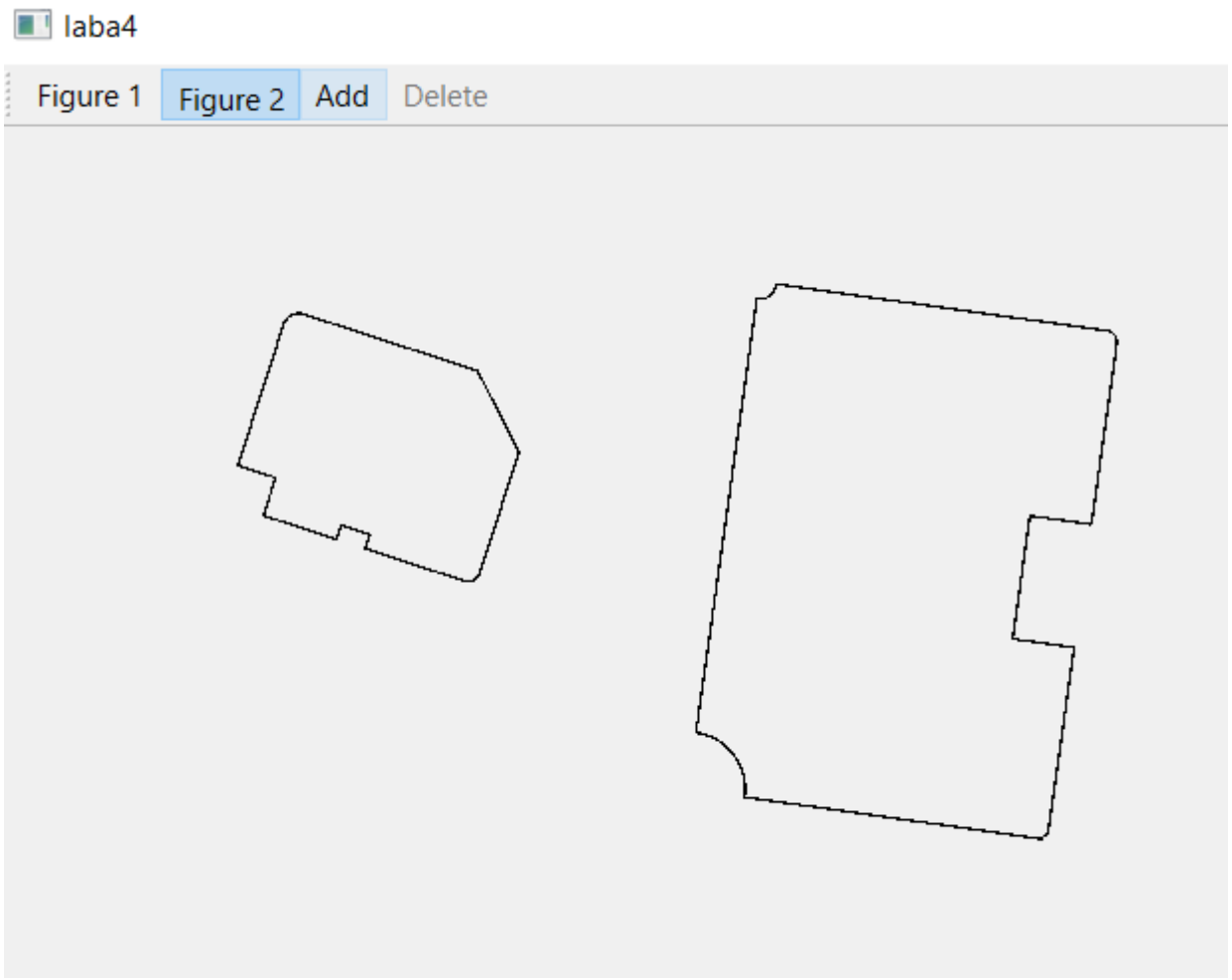
1) **Test №1**

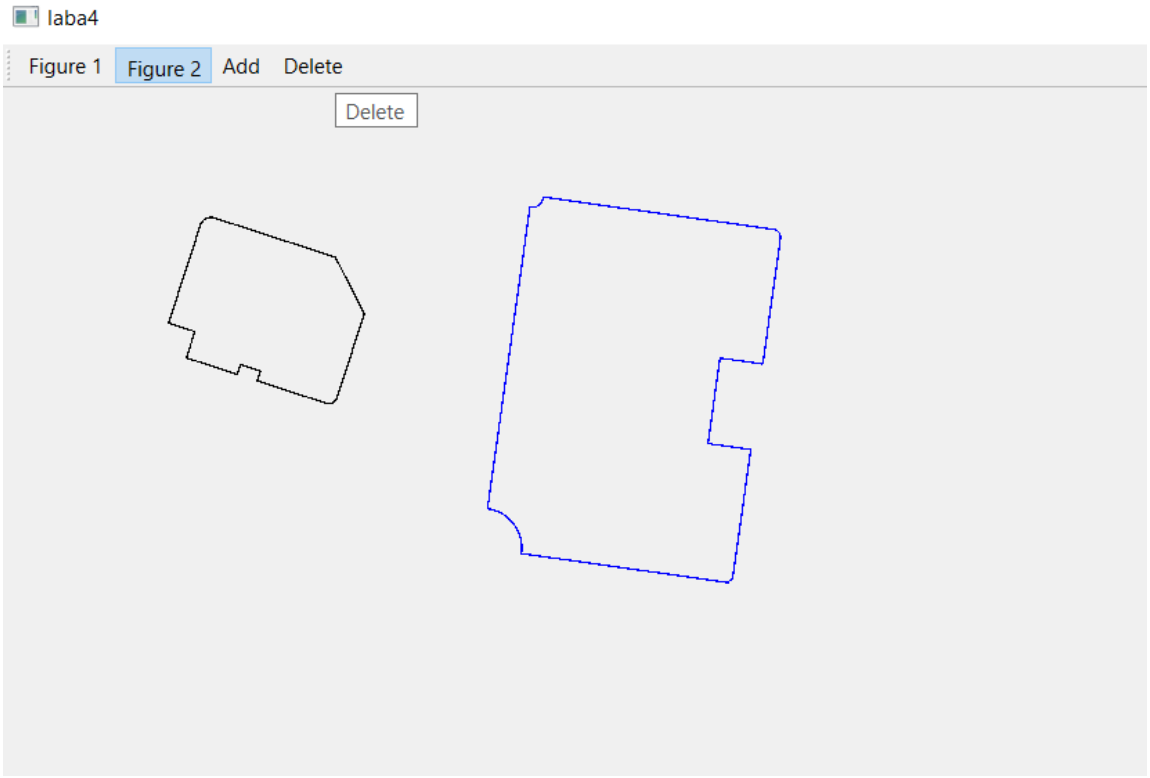
Демонстрируется результат работы программы.

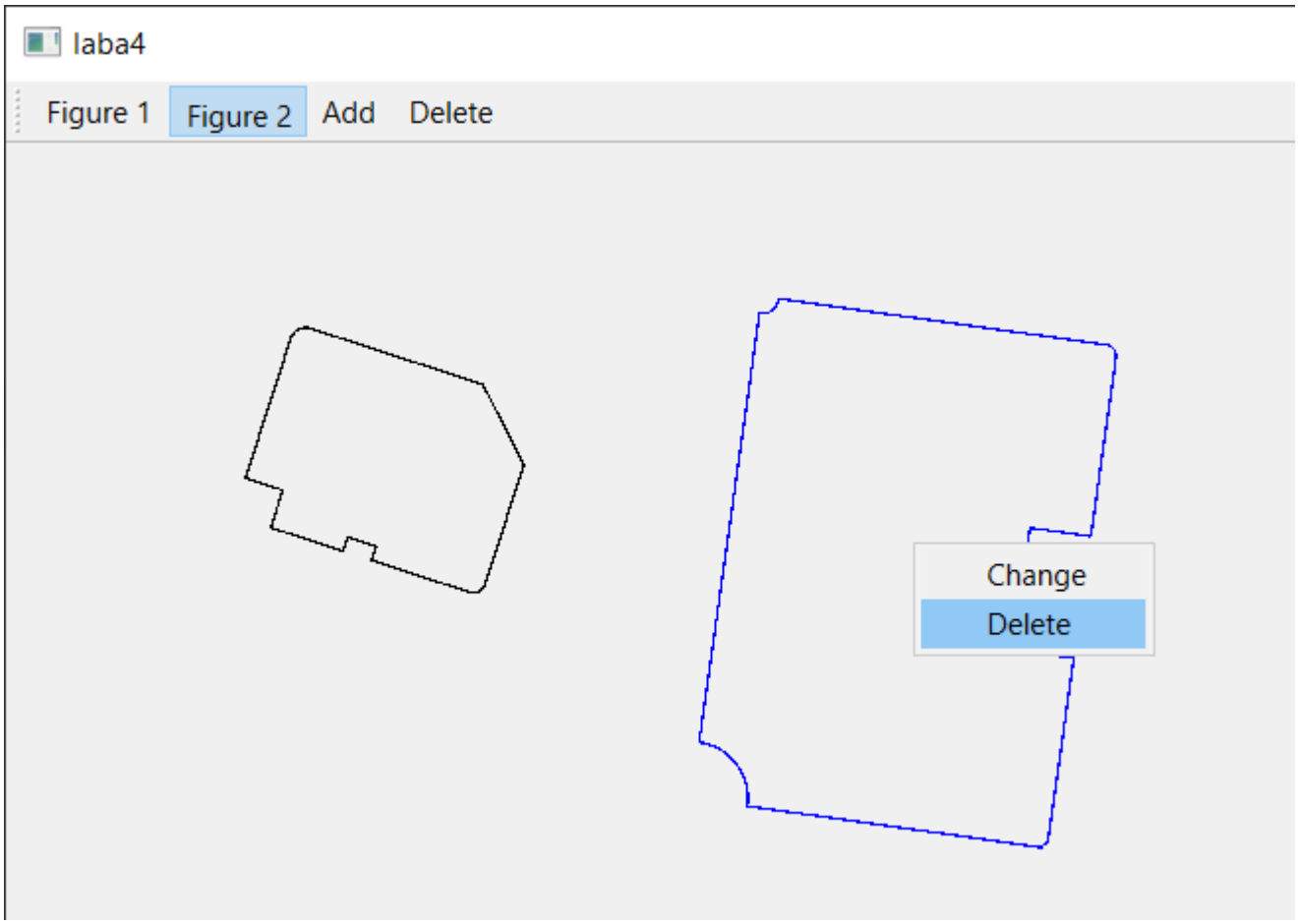

laba4
Figure 1
Figure 2
Add
Delete

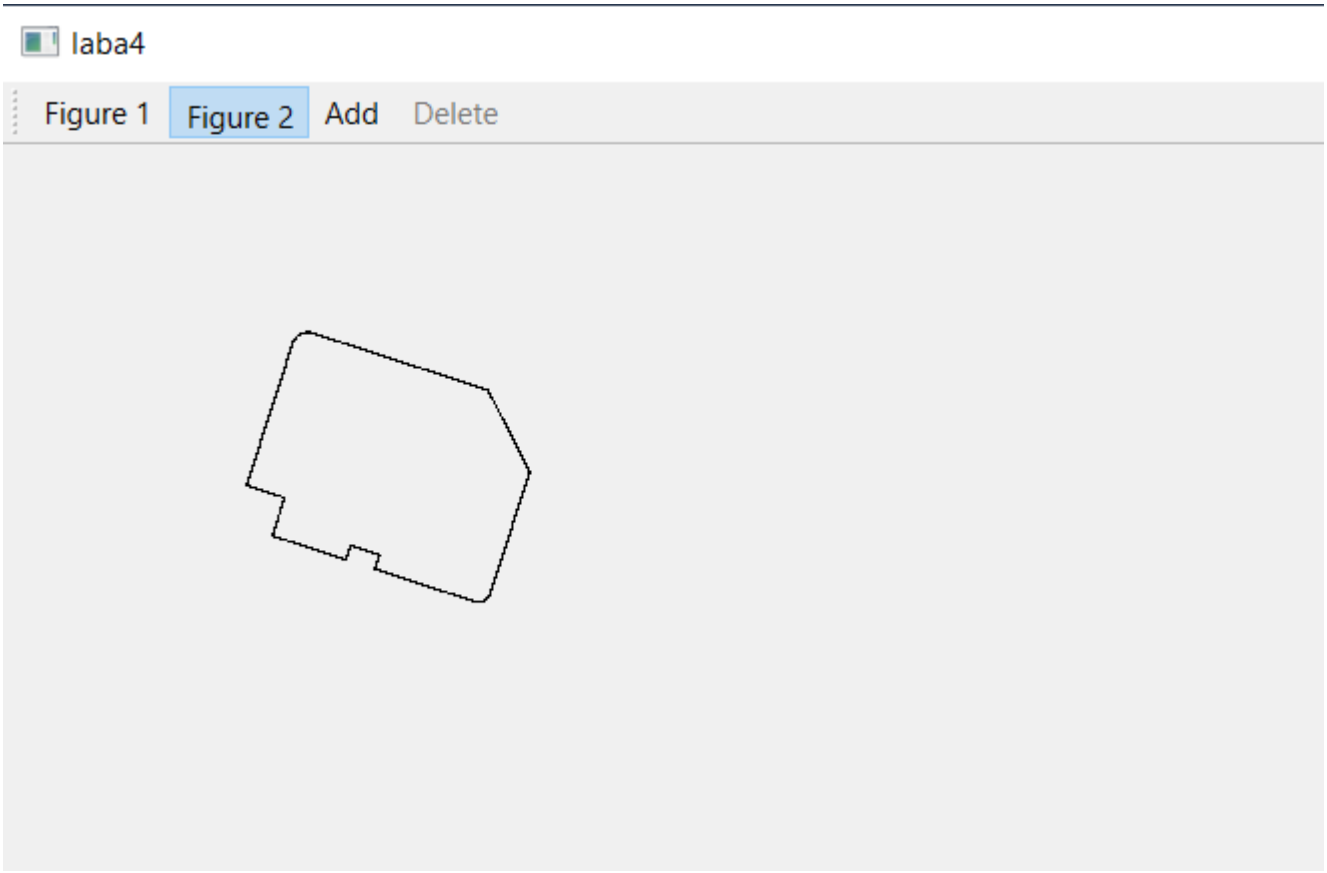

laba4
Figure 1
Figure 2
Add
Delete

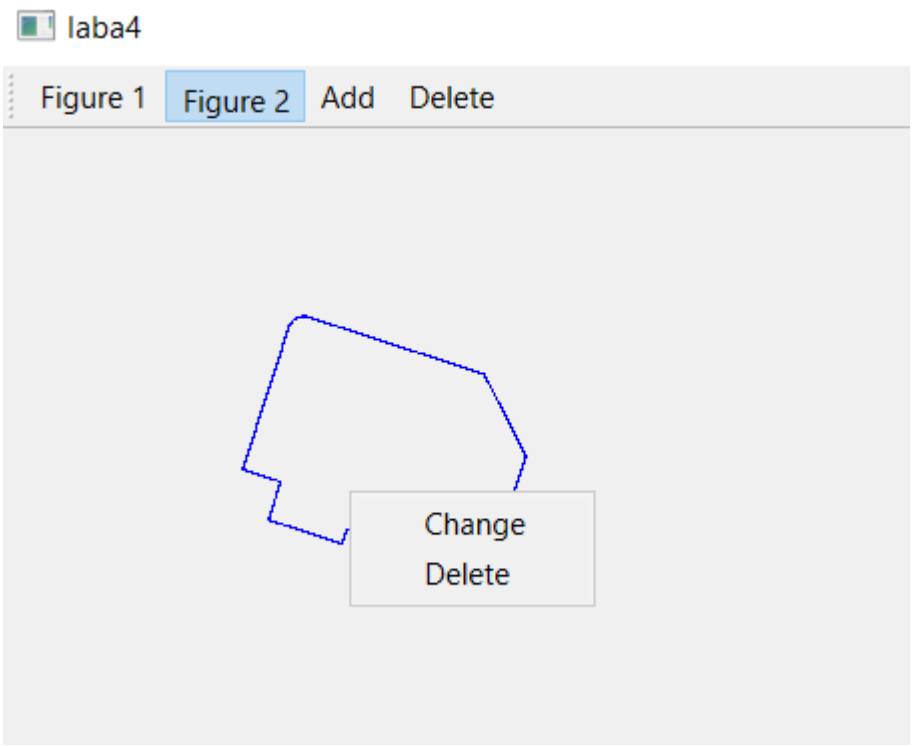

laba4
Figure 1
Figure 2
Add
Delete

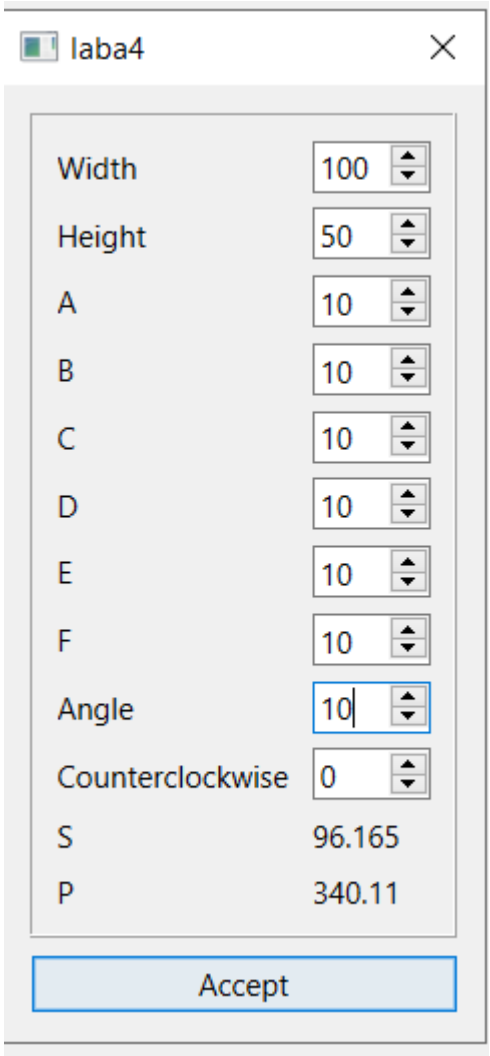

laba4
Figure 1
Figure 2
Add
Delete

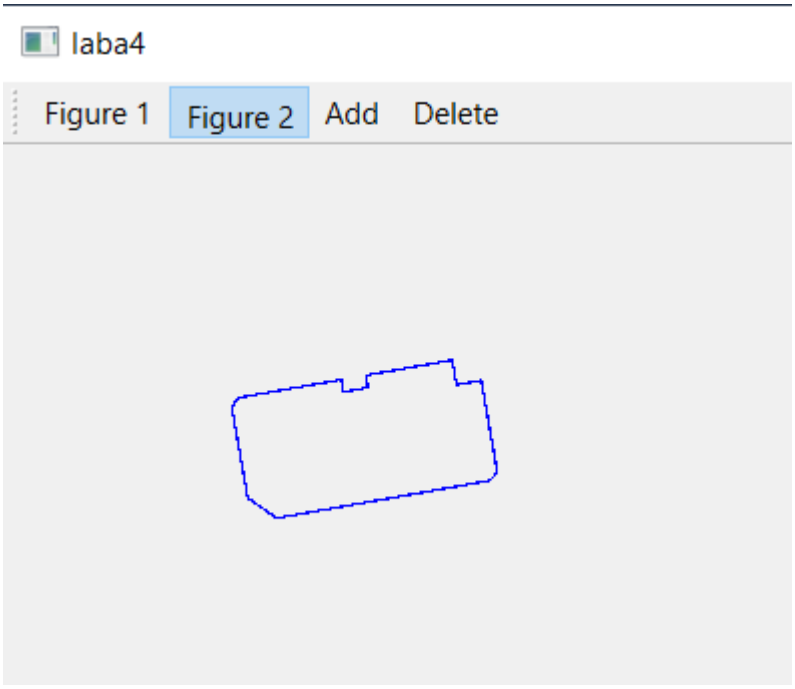

laba4
Figure 1
Figure 2
Add
Delete
Delete


laba4
Figure 1
Figure 2
Add
Delete
Change
Delete


laba4
Figure 1
Figure 2
Add
Delete


laba4
Figure 1
Figure 2
Add
Delete
Change
Delete


laba4
Width
100
Height
50
A
10
B
10
C
10
D
10
E
10
F
10
Angle
10
Counterclockwise
0
S
96.165
P
340.11
Accept



Приложение А

Листинг-2 А1 DialogMenu.h

```
#ifndef DIALOGMENU_H
#define DIALOGMENU_H
#include "Figure.h"
#include <QWidget>
#include <QDialog>
#include <QSpinBox>
#include<QFrame>
#include <QMessageBox>
#include <QGridLayout>

class DialogMenu : public QDialog {
    Q_OBJECT

public:
    DialogMenu(QWidget *p=0);
    void SetFigure(Figure *f);
     double s=0,p=0;
public slots:
    void accept();

private:
    bool type;
    Figure *figure;
    QFrame *frame;
     QGridLayout *fl;
    QSpinBox *Width;
    QSpinBox *Height;
    QSpinBox *A_;
    QSpinBox *B_;
```

```
    QSpinBox *C_;
    QSpinBox *D_;
    QSpinBox *E_;
    QSpinBox *F_;
    QSpinBox *Angle;
    QSpinBox *Counterclockwise;
    QMessageBox* messageBox;
};
#endif // DIALOGMENU_H
```

Листинг-3 A2 Figure.h

```
#ifndef FIGURE_H
#define FIGURE_H
#include <QWidget>
#include <QPainter>
#include <QPainterPath>
#include <QMenu>

class Figure : public QWidget {

    Q_OBJECT

public:
  Figure(QWidget *parent = 0);
  bool pls_remove=false;
  bool get_typeFigure() const;
  double S=10,P=10;
  void set_typeFigure(bool a);
  bool get_pressed();
  void set_pressed(bool);
  void set_param(int, int, int, int, int, int, int, int, int, int);
   bool typeFigure;
   int H, W, A_X, B_R, C_X, D_R, E_P, F, angle = 0;
   int H1, W1, A_R1, B_R1, C_R1, D_R1, E1, F_P1, angle1 = 0;
protected:
  void paintEvent(QPaintEvent *);
  void contextMenuEvent(QContextMenuEvent *event);
  void mouseMoveEvent(QMouseEvent *event);
  void mousePressEvent(QMouseEvent* event);
private:


  bool pressed = false;
  QPoint delta;
  size_t press=0;
  QMenu *figureMenu;
  QAction *actionEdit;
  QAction *actionDel;
private slots:
  void delFigure();
  void editFigure();
};

#endif // FIGURE_H
```

Листинг-4 A3 mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
```

```
#include <QMenuBar>
#include <QVector>
#include <QCoreApplication>
#include <QContextMenuEvent>
#include <QMainWindow>
#include <QPainter>
#include <Figure.h>

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private:
    QToolBar *MainToolBar;
    QAction *type1,*type2,*add,*del;
    QVector <Figure*> f;
    bool flag;

private slots:
    void check_widget();
    void pick1type();
    void pick2type();
    void addFigure();
    void delFigure();

};
#endif // MAINWINDOW_H
```

Листинг-5 A4 DialogMenu.cpp

```
#include "DialogMenu.h"
#include <QLabel>
#include <QFrame>
#include <QGridLayout>
#include <QPushButton>
#include <QString>
#include <QLineEdit>


DialogMenu::DialogMenu(QWidget *parent) : QDialog(parent), figure(nullptr) {
    frame=new QFrame(this);
    frame->setFrameStyle(QFrame::Box | QFrame::Raised);


    Width=new QSpinBox(frame);
    Height=new QSpinBox(frame);
    A_=new QSpinBox(frame);
    B_=new QSpinBox(frame);
    C_=new QSpinBox(frame);
    D_=new QSpinBox(frame);
    E_=new QSpinBox(frame);
    F_=new QSpinBox(frame);
    Angle=new QSpinBox(frame);
    Counterclockwise=new QSpinBox(frame);

     Width->setRange(0, 2000);
     Height->setRange(0, 2000);
     A_->setRange(0, 2000);
```

```
 B_->setRange(0, 2000);
 C_->setRange(0, 2000);
 D_->setRange(0, 2000);
 E_->setRange(0, 2000);
 F_->setRange(0, 2000);
 Angle->setRange(-180, 180);
 Counterclockwise->setRange(0,1);

messageBox=new QMessageBox();
messageBox->setWindowTitle("Warning!");
messageBox->setText("Wrong input!");
messageBox->addButton(QMessageBox::Ok);
QLabel *LWidth=new QLabel("Width ",frame);
QLabel *LHeight=new QLabel("Height ", frame);
QLabel *LA=new QLabel("A ", frame);
QLabel *LB=new QLabel("B ", frame);
QLabel *LC=new QLabel("C ", frame);
QLabel *LD=new QLabel("D ", frame);
QLabel *LE=new QLabel("E ", frame);
QLabel *LF=new QLabel("F ", frame);
QLabel *LAngle=new QLabel("Angle ", frame);
QLabel *LProtive=new QLabel("Counterclockwise ", frame);
QLabel *LabelS=new QLabel("S ",frame);
QLabel *LabelP=new QLabel("P ",frame);

fl=new QGridLayout(frame);
fl->addWidget(LWidth,0,0);
fl->addWidget(Width,0,1);

fl->addWidget(LHeight,1,0);
fl->addWidget(Height,1,1);

fl->addWidget(LA,2,0);
fl->addWidget(A_,2,1);

fl->addWidget(LB,3,0);
fl->addWidget(B_,3,1);
fl->addWidget(LC,4,0);
fl->addWidget(C_,4,1);

fl->addWidget(LD,5,0);
fl->addWidget(D_,5,1);
fl->addWidget(LE,6,0);
fl->addWidget(E_,6,1);
fl->addWidget(LF,7,0);
fl->addWidget(F_,7,1);
fl->addWidget(LAngle,8,0);
fl->addWidget(Angle,8,1);
fl->addWidget(LProtive,9,0);
fl->addWidget(Counterclockwise,9,1);
fl->addWidget(LabelS,10,0);
fl->addWidget(LabelP,11,0);
frame->setLayout(fl);

QPushButton *accbutt=new QPushButton("Accept", this);
connect(accbutt, SIGNAL(clicked()), this, SLOT(accept()));

QGridLayout *gl=new QGridLayout(this);
gl->addWidget(frame,0,0);
gl->addWidget(accbutt,1,0);
setLayout(gl);
setModal(true);
```

```
}
void DialogMenu::SetFigure(Figure *f) {

figure=f;
type=figure->typeFigure;
if (!type){
    p=f->H-f->B_R+2*3.14*f->B_R/4+f->W-f->B_R-f->C_X+sqrt(f->A_X)+f->H-f->A_X-f-
>D_R+2*3.14*f->D_R/4+f->W-f->D_R+f->E_P;
    s=f->H*f->S-f->A_X*f->A_X-f->B_R*f->B_R+3.14*f->B_R*f->B_R/4-f->D_R*f-
>D_R+3.14*f->D_R*f->D_R/4-f->C_X*f->C_X/2-f->E_P/2*f->E_P;

}
else {
p=f->H1-f->A_R1+2*3.14*f->A_R1/4-f->B_R1+2*3.14*f->B_R1+f->W1-f->B_R1+f->F_P1-f-
>C_R1+2*3.14*f->C_R1+f->H1-f->C_R1-f->D_R1+2*3.14*f->D_R1+f->W1-f->D_R1-f->A_R1;
s=(p*f->W1-2*f->W1*f->W1)/2;
}
QLabel *ppp=new QLabel(QString::number(p),frame);
QLabel *sss=new QLabel(QString::number(s),frame);
fl->addWidget(ppp,11,1);
fl->addWidget(sss,10,1);

    }

void DialogMenu::accept() {
    bool flag1=0;

    if (type==0){
        if (Width->value()<700&&Height->value()<500&&Width->value()>Height-
>value()&&Height->value()>0&&A_->value()>0&&A_->value()<Height->value()/3&&B_-
>value()>0&&B_->value()<Height->value()/3&&C_->value()>0&&C_->value()<Height-
>value()/3&&D_->value()>0&&D_->value()<Height->value()/3&&E_->value()>0&&E_-
>value()<Width->value()/4){
            flag1=0;
        }
        else {
            flag1=1;
            messageBox->open();
        }
    }
    else {
        if (Width->value()<700&&Height->value()<500&&Width->value()>Height-
>value()&&Height->value()>0&&A_->value()>0&&A_->value()<Height->value()/3&&B_-
>value()>0&&B_->value()<Height->value()/3&&C_->value()>0&&C_->value()<Height-
>value()/3&&D_->value()>0&&D_->value()<Height->value()/3&&F_->value()>0&&F_-
>value()<Width->value()/4){
            flag1=0;
        }
        else {
            flag1=1;
            messageBox->open();
        }
    }
    if (flag1==0){

    figure->set_param(Width->value(),Height->value(),A_->value(),B_->value(),C_-
>value(),D_->value(),E_->value(),F_->value(),Angle->value(),Counterclockwise-
>value());
}
    figure->repaint();
    close();
}
```

Листинг-6 A6 Figure.cpp

```
#include "Figure.h"
#include "DialogMenu.h"
#include <cmath>
#include <QMouseEvent>
//#include "m_window1.h"
//#include "m_window2.h"
#include <QMessageBox>
#include <QCoreApplication>

Figure::Figure(QWidget *parent) : QWidget(parent) {
    figureMenu=new QMenu;
    actionEdit=figureMenu->addAction("&Change");
    actionDel=figureMenu->addAction("&Delete", this->parentWidget(),
SLOT(delFigure()));
    connect(actionEdit, SIGNAL(triggered()), this, SLOT(editFigure()));

}
void Figure::editFigure(){
    DialogMenu edit;
    edit.SetFigure(this);
    edit.exec();
    repaint();
}
void Figure::delFigure(){
    pls_remove=true;
}

void Figure::set_typeFigure(bool a){
    typeFigure =a;
    if (!a){
        W = rand()%375+65;
        H = rand()%(W-65)+65;
        A_X = rand()%(H/3);
        B_R = rand()%(H/3);
        C_X = rand()%(H/3);
        D_R=rand()%(H/3);
        E_P=rand()%(W/4);
        F=0;
        angle = rand()% 360;
         setGeometry(rand() % 750 + 40, rand() % 500 + 40, W*sqrt(2)+1,H*sqrt(2)+1);
P=2;
    }
    else {
        W1 = rand() % 375 + 65;
        H1 = rand() % (W1 - 65 ) + 65;
        A_R1 = rand()% (H1 / 3),
        B_R1 = rand()% (H1 / 3),
        C_R1 = rand()% (H1 / 3),
        D_R1 = rand()% (H1 / 3),
        E1 = rand()% (W1 / 4),
        F_P1 = rand()% (W1 / 4);
        angle1 = rand()% 360;
         setGeometry(rand() % 750 + 40, rand() % 500 + 40,
W1*sqrt(2)+1,H1*sqrt(2)+1);
         P=2;

    }
```

```
}
void Figure::set_param(int W_,int  H_, int A_, int B_, int C_, int D_, int E_, int
F_, int angle_,int a){

    if (!typeFigure){
    W = W_;
    H = H_;
    A_X = A_;
    B_R = B_;
    C_X = C_;
    D_R=D_;
    E_P=E_;
    F=F_;
    angle = angle_ % 360;
    if (a==0){
        angle=360-angle;
    }
    }
else{
        W1 = W_;
        H1 = H_;
        A_R1 = A_;
        B_R1 = B_;
        C_R1 = C_;
        D_R1 = D_;
        E1 = E_;
        F_P1 = F_;
        angle1 = angle_% 360;
        if (a==0)
            angle1=180-angle1;
    }
}

void Figure::paintEvent(QPaintEvent *) {

    QPainterPath path;

    QPainter painter(this);
    if(!typeFigure){


    this->resize(sqrt(W*W + H*H) + 1, sqrt(W*W + H*H) + 1);


    if (!pressed){
        QPen pen(Qt::black);
        pen.setWidth(1);
        painter.setPen(pen);
    }
    else{
        QPen pen(Qt::blue);
        pen.setWidth(1);
        painter.setPen(pen);

    }

    painter.translate((sqrt(W*W + H*H) + 1)/2, (sqrt(W*W + H*H) + 1)/2);
    painter.rotate(angle);


    path.moveTo(0+E_P/2,-H/2);
    path.lineTo(W/2 - A_X, -H/2);
```

```
    path.lineTo(W/2-A_X, -H/2 + A_X);
    path.lineTo(W/2, -H/2 + A_X);
    path.lineTo(W/2, H/2 - B_R/2);

    path.moveTo(W/2 - B_R,H/2);
    QRectF rectangle( W/2 - B_R,H/2 - B_R ,B_R, B_R);
    path.arcTo(rectangle, 270,90);

    path.moveTo(W/2 - B_R,H/2);
   path.lineTo(0-W/2+C_X,H/2);
    path.lineTo(0-W/2,H/2-C_X);

    path.lineTo(-W/2, -H/2 + D_R/2 );
    path.moveTo(- W/2+ D_R/2 , -H/2);
    QRectF rectangle4(- W/2 , -H/2 , D_R, D_R);
    path.arcTo(rectangle4, 90 ,90);

    path.moveTo(- W/2 + D_R/2, - H/2 );
    path.lineTo(0 - E_P/2,-H/2);
    path.lineTo(-E_P/2, -H/2);
    path.lineTo(-E_P/2, -H/2 + E_P/2);
    path.lineTo(E_P/2, -H/2 + E_P/2);
    path.lineTo(E_P/2, -H/2);
    painter.drawPath(path);
    }
else{
        this->resize(sqrt(W1*W1 + H1*H1) + 1, sqrt(W1*W1 + H1*H1) + 1);


        if (!pressed){
            QPen pen(Qt::black);
            pen.setWidth(1);
            painter.setPen(pen);

        }
        else{
            QPen pen(Qt::blue);
            pen.setWidth(1);
            painter.setPen(pen);
        }
        painter.translate(sqrt(W1*W1 + H1*H1)/2+1, sqrt(W1*W1 + H1*H1)/2+1);
        painter.rotate(angle1);
        path.moveTo(0 + F_P1/2, -H1/2);
        path.lineTo(W1/2 - A_R1/2,-H1/2);
        QRectF rectangle(W1/2 - A_R1/2, -H1/2 - A_R1/2, A_R1, A_R1);
        path.arcTo(rectangle, 180, 90);
        path.moveTo(W1/2, -H1/2 + A_R1/2);
        path.lineTo(W1/2, -H1/2 + A_R1/2);
        path.lineTo(W1/2,H1/2 - B_R1/2);
        path.moveTo(W1/2 - B_R1,H1/2);
        QRectF rectangle9( W1/2 - B_R1,H1/2 - B_R1 ,B_R1, B_R1);
        path.arcTo(rectangle9, 270,90);
        path.moveTo(W1/2 - B_R1,H1/2);
        path.lineTo(0 + F_P1/2,H1/2);
        path.lineTo(0+ F_P1/2,H1/2-F_P1/2);
        path.lineTo(0- F_P1/2,H1/2-F_P1/2);
         path.lineTo(0- F_P1/2,H1/2);
        path.moveTo(0 - F_P1/2, H1/2);
        path.lineTo(-W1/2 + C_R1/2,H1/2);
        QRectF rectangle3( -W1/2 - C_R1/2, H1/2 - C_R1/2 ,C_R1, C_R1);
        path.arcTo(rectangle3, 0,90);
        path.moveTo(-W1/2, H1/2 - C_R1/2);
```

```
        path.lineTo(-W1/2, -H1/2 + D_R1/2 );
        QRectF rectangle4(- W1/2-D_R1/2 , -H1/2-D_R1/2, D_R1, D_R1);
        path.arcTo(rectangle4, 270 ,90);
        path.moveTo(- W1/2 + D_R1/2, - H1/2 );
        path.lineTo(-W1/2+ D_R1/2, -H1/2 );
        path.lineTo(W1/2 - A_R1/2,-H1/2);
         painter.drawPath(path);

    }
}

void Figure::set_pressed(bool status){
    pressed = status;
}

bool Figure::get_pressed(){
    return pressed;
}

void Figure::mouseMoveEvent(QMouseEvent *event){

    pressed = true;
    update();
    if (event->buttons() & Qt::LeftButton){
        if (((event->globalPos() - delta).x() + width() > 1500) ||
            ((event->globalPos() - delta).y() + height() > 1000) ||
            ((event->globalPos() - delta).x() <= 0 ) ||
            ((event->globalPos() - delta).y() <= 30))
                delta = event->globalPos() - pos();
        else
            move(event->globalPos() - delta);
    }

}

void Figure::mousePressEvent(QMouseEvent* event){

    if (event->button() & Qt::LeftButton){
        press++;
        if (press % 2 == 0)
            pressed = false;
        else
            pressed = true;
        update();
        delta = event->globalPos() - pos();
    }}

void Figure::contextMenuEvent(QContextMenuEvent *event) {
    if(pressed==true) {
        figureMenu->exec(event->globalPos());
    }
}
```

Листинг-7 A7 mainwindow.cpp

```
#include "mainwindow.h"
#include <QToolBar>
#include <QTimer>
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
{
```

```
    MainToolBar=new QToolBar(this);

    type1=MainToolBar->addAction(tr("Figure 1"));
    type2=MainToolBar->addAction(tr("Figure 2"));
    add=MainToolBar->addAction(tr("Add"));
    del=MainToolBar->addAction(tr("Delete"));

    addToolBar(MainToolBar);

    type1->setCheckable(true);
    type2->setCheckable(true);
    add->setDisabled(true);
    del->setDisabled(true);

    connect(type1,SIGNAL(triggered()),this,SLOT(pick1type()));
    connect(type2,SIGNAL(triggered()),this,SLOT(pick2type()));
    connect(add,SIGNAL(triggered()),this,SLOT(addFigure()));
    connect(del,SIGNAL(triggered()),this,SLOT(delFigure()));

    QTimer *timer = new QTimer(this);
    connect(timer, SIGNAL(timeout()), this, SLOT(check_widget()));
    timer->start(500);


}
void MainWindow::check_widget(){
    bool a=0;
    for (size_t i=0;i<f.size();i++){
        if (f[i]->get_pressed()){
            a=1;
        }
    }
    if (a){
        del->setDisabled(false);
    }
    else {
        del->setDisabled(true);
    }
}
MainWindow::~MainWindow()
{
    f.clear();
}

void MainWindow::pick1type(){
    type2->setChecked(false);
    type1->setChecked(true);
    add->setDisabled(false);
    flag=0;
   update();
}
void MainWindow::pick2type(){
    type1->setChecked(false);
    type2->setChecked(true);
    add->setDisabled(false);
    flag=1;
    update();
}
void MainWindow::addFigure(){

    Figure *p = new Figure(this);
    p->set_typeFigure(flag);
```

```
    f.push_back(p);
    p->show();
    update();
}
void MainWindow::delFigure(){
for(size_t i=0;i<f.size();i++){
    if (f[i]->get_pressed()==true||f[i]->pls_remove){
        f[i]->close();
        f.remove(i);
    }

}
update();
}
```

Листинг-8 A8 main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    MainWindow a;
    a.resize(1500,1000);
    a.show();
    return app.exec();
}
```