WIDGETS LAYOUT REPORT

2020/2021

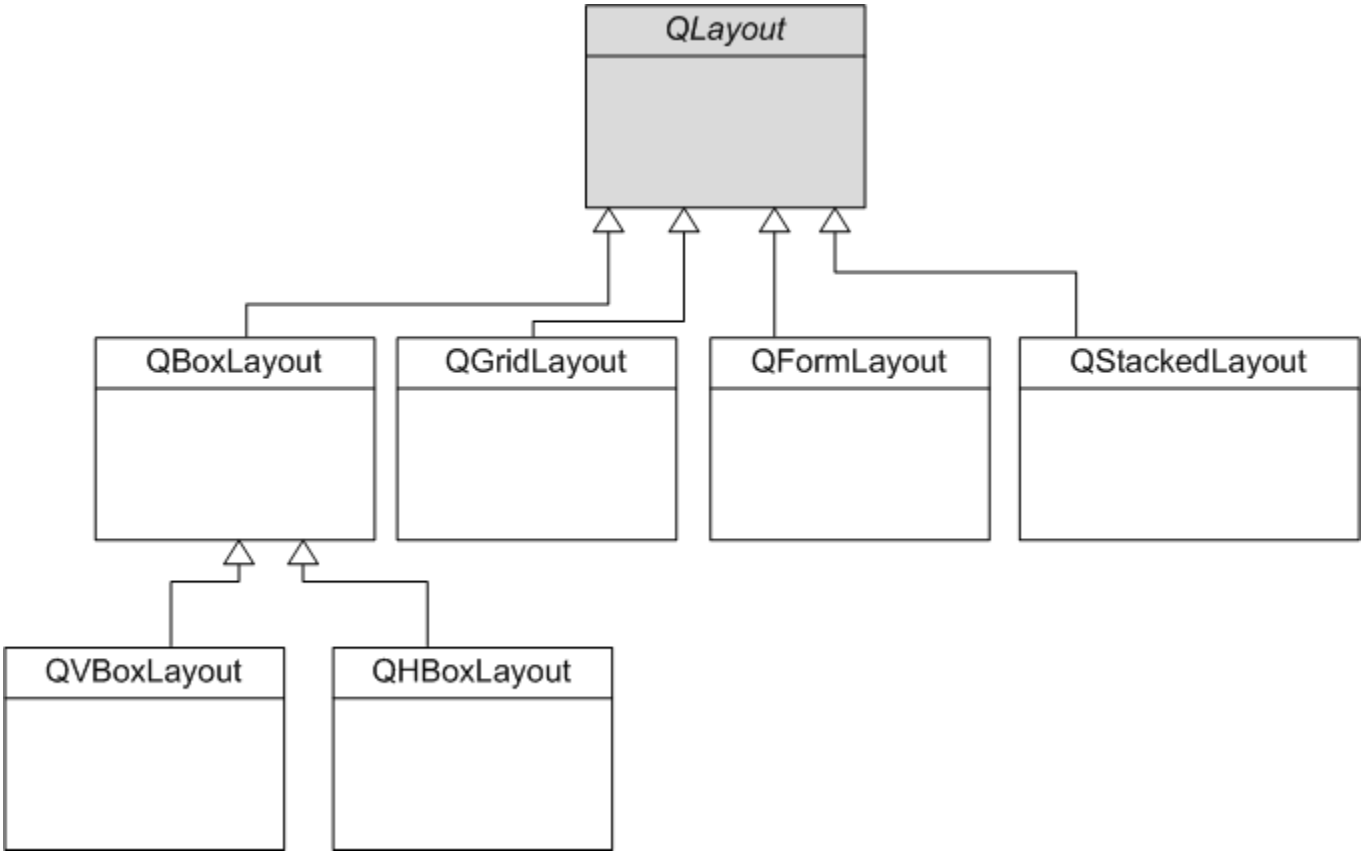
SAYF-ALLAH NAZIHE (CS)

sayf-allah.nazihe@eidia.ueuromed.org

Software development process is getting faster and faster and the technology companies tend to invest fast mechanism. With the creation of the QT Platform and its combination between C++ language, it gives a powerful code to create interfaces that allows users to interact with electronic devices through graphical icons (GUI). Each GUI interface include a lot of widgets and components due to the mechanism that compute the location of each objects and arranging it called Layout Manager. **So, what the purpose of the layout Manager? And how it can help us to create a great GUI interface?**

The layout Management

It’s a system that provides a simple and powerful way of automatically arranging child widgets within a widget to ensure that they make a good use for spacing, and for sure that the layout include a lot of types that each one provides use with a different forms and concepts

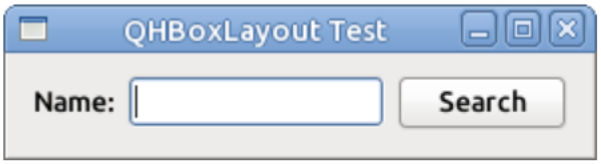


Objective: Programming Widgets Layout

To dive a little deeper into layouts and their principles we will show you a little main example to understand how its work:

Experimenting with QHBOXLayout

Our goal is to create a little interface that display the following form using the QHBOXlayout :



A QHBoxLayout example.

But first before coding we need to know what is QHBOXlayout?. As we say before that is a type of layouts that class lines up widgets horizontally. To see the effect of this type of layout we will use class named exo1 that include:

- **<QApplication>** manage the GUI application control flow and main settings
- **<Qwidgets>** that represent the base class of all user interface objects
- layout **<QHBoxlayout>**
- label **<QLabel>** that provide a text display
- button **<Qpushbutton>** that provide the button
- line **<QLineEdit>** that provide a one line-text editor

File.h: stands for the creating of the widgets and the methods
File.cpp: stands for the implementation of widgets, place of widgets
Main class: stands for displaying the result code

Exo1.h

```
class exo1 : public QWidget //inheritance from Qwidget to use the
components that is include in
{
    Q_OBJECT
public:
    explicit exo1(QWidget *parent = nullptr); //is a default arguments for
C++ to not specify argument
protected:// so that the members are accessible just in the class
void createwidgets();
void placewidgets();          // methods that we will use to create,place
                                  and connect the widgets
void connexionwidgets();
QHBoxLayout *layout; //create a layout
QLabel *label;//create a label
QPushButton *search;// create a button
QLineEdit *line;// create the line edit
};
```

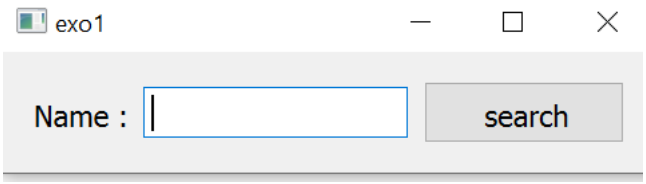
Exo1.cpp

```
exo1::exo1(QWidget *parent) : QWidget(parent)
{
createwidgets();
placewidgets();
connexionwidgets();
}
void exo1::createwidgets(){//implement the method with the widgets that we
                                                 will use
    line = new QLineEdit;
    search = new QPushButton("search");
    layout = new QHBoxLayout;
    label = new QLabel("Name :");
    setLayout(layout);//to see the layout components that we have been
                                            created
}
void exo1::placewidgets(){//place the widgets into layout
     layout->addWidget(label);
      layout->addWidget(line);      // adding the widgets to the layout
    layout->addWidget(search);
}
void exo1::connexionwidgets(){
                 }
```

Main class

```
#include <QApplication>
#include<exo1.h>//to call the exo1 class
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
auto q= new exo1;//create a "q" to call the class
q->show();//show the result


    return app.exec();//show the result and exist
}
```

Result



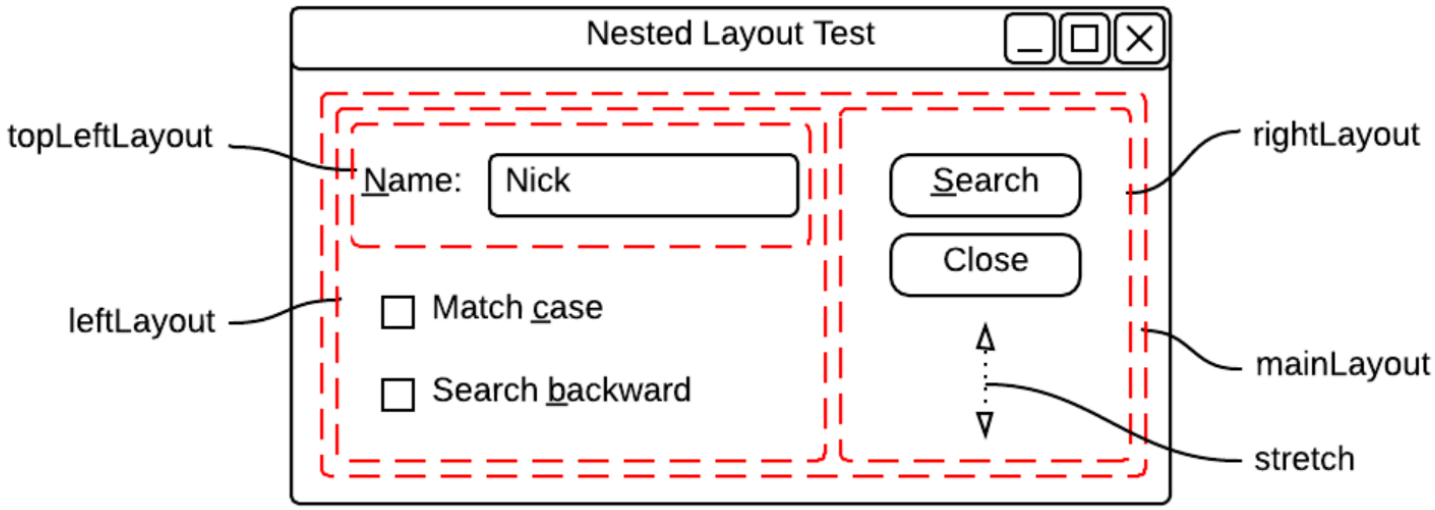
What we learned?

We learned how can we create a simple interface window and also the main function of the layout (arranging widgets with each other's Horizontally by QHBOxLayout)

If you are able to do this, so we jump over the next test. It's a simple test that includes also layout in an easier way to understand it using Nested layout (one layout inside of others)

Nested Layout TEST

Our goal is to do this form of dialogue using the nested layout:



Nested Layout.

To see the result, we use a class named exo2 including the following

objects:

- **<QApplication>** manage the GUI application control flow and main settings
- **<Qwidgets>** that represent the base class of all user interface objects
- layout **<QHBoxlayout>** or **<QVBoxlayout>**
- label **<QLabel>** that provide a text display
- button **<Qpushbutton>** that provide the button
- line **<QLineEdit>** that provide a one line-text editor
- boxes **<QcheckBox>** that provide a checkbox with a text label

Exo2.h

```
class exo2 : public QWidget//inheritance from Qwidget to use the
components that is include in
{

    Q_OBJECT
public:
    explicit exo2(QWidget *parent = nullptr); is a default arguments for
C++ to not specify argument
    virtual ~exo2();//create a destructor
protected:// so that the members are accessible just in the class

    void createwidgets();
    void connexionwidgets();      // methods that we will use
                                          tocreate,place
                                       and connect the widgets
    void placewidgets();

QPushButton *search; //create a button
QPushButton *close; //create a button
QLineEdit *line; //create a line edit
QLabel *n; //create a label
QHBoxLayout *layout; //create a main layout
QVBoxLayout *leftlayout;
QHBoxLayout *topleftlayout;   //create the nested layouts
QVBoxLayout *rightlayout;
QCheckBox *box1; //create a checkbox
QCheckBox *box2;}; //create a checkbox
```

Exo2.cpp

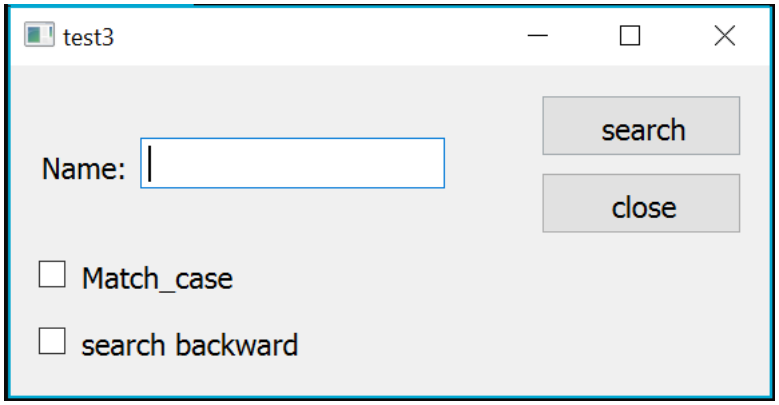
```
exo2::exo2(QWidget *parent) : QWidget(parent)
{
 createwidgets();
 placewidgets();
 connexionwidgets();
}
exo2::~exo2(){//implement the widgets that will be deleted after been
executed
    delete search;
    delete close;
    delete n;
    delete line;
    delete layout;
    delete box1;
    delete box2;
    delete leftlayout;
    delete rightlayout;
```

```
    delete topleftlayout;
}
void exo2::createwidgets(){//implement the method with the widgets that we
                                                will use
    line=new QLineEdit;
    search= new QPushButton("search");
    close= new QPushButton("close");
    box1= new QCheckBox("Match_case");
    box2= new QCheckBox("search backward");
    n= new QLabel("Name:");
    leftlayout = new QVBoxLayout;
    topleftlayout = new QHBoxLayout;
    rightlayout= new QVBoxLayout;
    layout = new QHBoxLayout;
    setLayout(layout); //to see the layout components that we have been
                                          created
}

}
void exo2::placewidgets(){
    topleftlayout->addWidget(n);
    topleftlayout->addWidget(line);
    rightlayout->addWidget(search);
    rightlayout->addWidget(close);
     rightlayout->addSpacerItem(new QSpacerItem(30,50));      //adding
                                     space(stretch) to the layouts
     topleftlayout->addSpacerItem(new QSpacerItem(30,50));
         leftlayout->addLayout(topleftlayout);
         leftlayout->addWidget(box1);
         leftlayout->addWidget(box2);
    layout->addLayout(topleftlayout);
    layout->addLayout(leftlayout);          //add the nested layouts to the
                                                  main layout
    layout->addLayout(rightlayout);

}
void exo2::connexionwidgets(){
connect(close, &QPushButton::clicked, qApp,&QApplication::exit);//make a
connection of the button close to an exit button
}
```

Main class

```
#include"exo2.h" //to call the exo2 class
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    auto name= new exo2; //create "name" to call the class
name->show(); //show of results
return app.exec(); //exit after running
```

Result

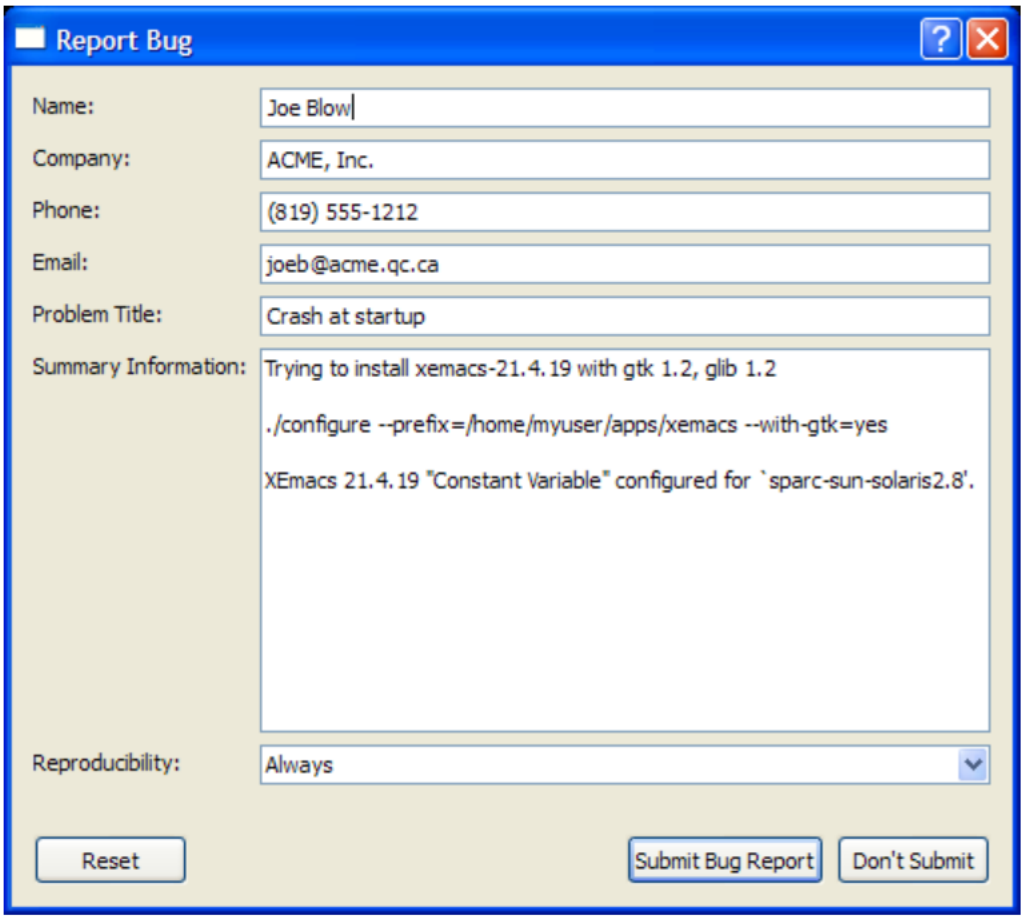


What we learned?

We learned that a lot of layouts can be in a layout (nested layout) by the addlayout() method and also how to add spacing between the layouts by the method addspacerItem()

The next task is to create a Bug report form ass the following:

Bug Report Form Test



Dialog to report a form.

To create our interface we use a important class of layout called QFormlayout that manages forms of input widgets and their associated labels,QcomboBox that is a combined button and popup list and the following widgets :

- **<QApplication>** manage the GUI application control flow and main settings
- **<Qwidgets>** that represent the base classes of all user interface objects
- layout **<QHBoxlayout>** or **<QVBoxlayout>**
- layout2 **<QFormLayout>**
- label **<QLabel>** that provide a text display
- line edit **<QLineedit>** that provide a one line-text editor
- button **<Qpushbutton>** that provide the button
- combo**<QComboBox>**
- text **<QTextEdit>** that is used to edit and display both plain and rich text

Exo3.h

```
class exo3 : public QWidget
{
    Q_OBJECT
public:
    explicit exo3(QWidget *parent = nullptr);
   // ~exo3();

protected:

   void createwidgets();
   void placewidgets();
   void connexionwidgets();
   QFormLayout *layout;
   QLineEdit *name;
   QLineEdit *age;
   QLineEdit *email;
   QLineEdit *problem;
   QLineEdit *Phone;
   QLabel *summ;
   QLabel *prod;
   QHBoxLayout *layout2;
   QHBoxLayout *layout3;
   QHBoxLayout *layout4;
   QVBoxLayout *layout5;
   QPushButton *but1;
   QPushButton *but2;
   QPushButton *but3;
QComboBox *combo;
QTextEdit *text;
};
```

Exo3.cpp

```
exo3::exo3(QWidget *parent) : QWidget(parent)
{
createwidgets();
placewidgets();
connexionwidgets();
}
void exo3::createwidgets(){
     but1 = new QPushButton("Reset");
     but2= new QPushButton("Submit Bug report");
     but3 = new QPushButton("Don't SUBMIT");
     summ=new QLabel("Summary :") ;
      prod=new QLabel("Reproducibility :");
     layout2 =new QHBoxLayout ;
      layout3= new QHBoxLayout;
```

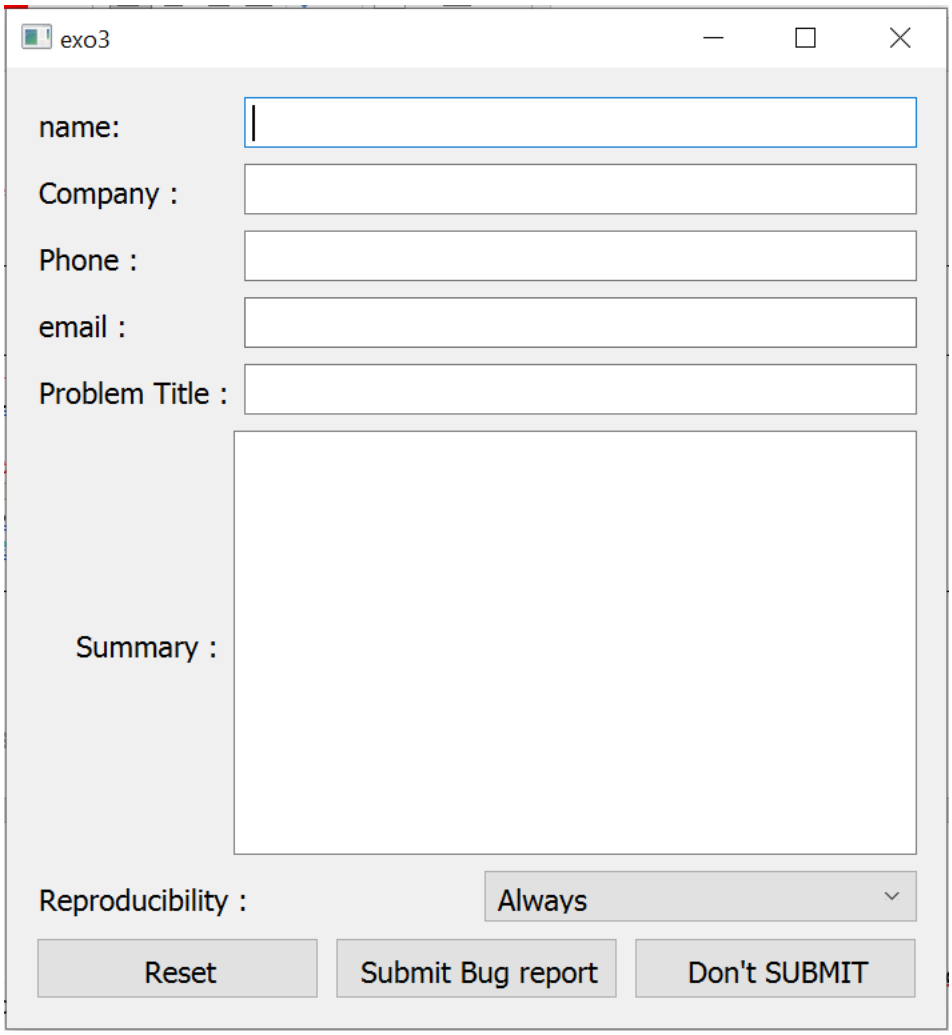
```
      layout4= new QHBoxLayout;
      layout5=new QVBoxLayout;
     /*line1 = new QLineEdit;
     line2 = new QLineEdit;
    line3= new  QLineEdit;
    line4 =new QLineEdit ;
     line5= new QLineEdit;*/
  combo = new QComboBox();
  text = new QTextEdit;
  layout= new QFormLayout;
  name =new QLineEdit;
  age =new QLineEdit;
  email=new QLineEdit;
  problem=new QLineEdit;
  Phone=new QLineEdit;
 setLayout(layout5);
                          }

void exo3::placewidgets(){
    combo->addItem(tr("Always"));
    combo->addItem(tr("Sometimes"));
    combo->addItem(tr("Rarely"));
    layout->setAlignment(Qt::AlignRight | Qt::AlignHCenter);//The
positioning of the layout that include the widgets
    layout->addRow("name: ", name);
    layout->addRow("Company :", age);
    layout->addRow("Phone :", Phone);
    layout->addRow("email :",email );
    layout->addRow("Problem Title :", problem);
    layout2->addSpacerItem(new QSpacerItem(14,20));//add a space to the
layout2
    layout2->addWidget(summ);
    layout2->addWidget(text);
    layout5->addLayout(layout);
    layout5->addLayout(layout2);
    layout5->addLayout(layout3);
    layout5->addLayout(layout4);
    layout3->addWidget(prod);
    layout3->addWidget(combo);
    layout4->addWidget(but1);
     layout4->addWidget(but2);
      layout4->addWidget(but3);



}
void exo3::connexionwidgets(){

}
```

// add the rows of the form layout

Main Class

```
#include"exo3.h"
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
auto q= new exo3;
q->show();
    return app.exec();
}
```

Result



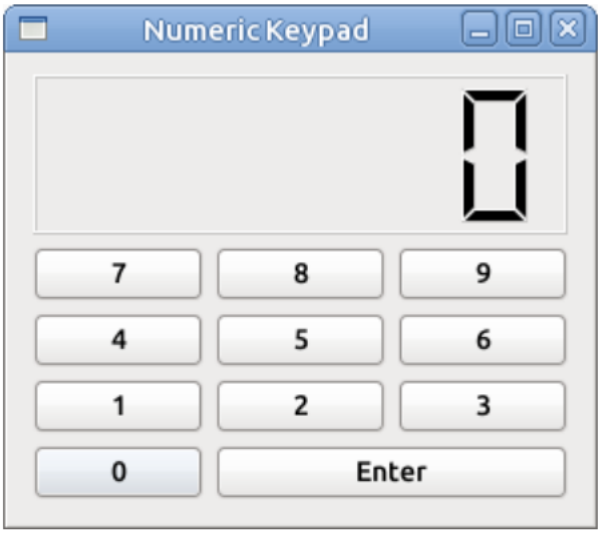
What we learned?

We learned that the QFormlayout is an easy way to associated widgets and their labels by a simple structure code

Our final test is build on a calculator form that include a important type of layout called QGridLayout , let find out what is it and how it can help us :

Grid Layout Test

QGridLayout is a class that lays out widgets in a grid, our challenge is to create the following form using the GridLayout and a QLCDNumber widgets that displays a number with LCD-like digits:



Calculator using the Grid Layout.

To create our interface, we will use a class named exo5 that include the following objects:

- **<QApplication>** manage the GUI application control flow and main settings
- **<Qwidgets>** that represent the base class of all user interface objects
- layout **<QHBoxlayout>** or **<QVBoxlayout>**
- button **<Qpushbutton>** that provide the button
- grid**<QGridLayout>**
- LCDnumber **<QLCDNumber>**

Exo5.h

```
class exo5 : public QWidget
{
    Q_OBJECT
public:
    explicit exo5(QWidget *parent = nullptr);

protected:
    void createwidgets();
    void placewidgets();
    void Connexionwidgets();

QPushButton *b1;
QPushButton *b2;
QPushButton *b3;
QPushButton *b4;
QPushButton *b5;
QPushButton *b6;
QPushButton *b7;
QPushButton *b9;        //create the buttons of the calculator
QPushButton *b8;
QPushButton *b10;
QPushButton *b11;
QGridLayout *grid;
QLCDNumber *number;//create the LCD number
QVBoxLayout *layout;//create a layout

};
```

Exo5.cpp

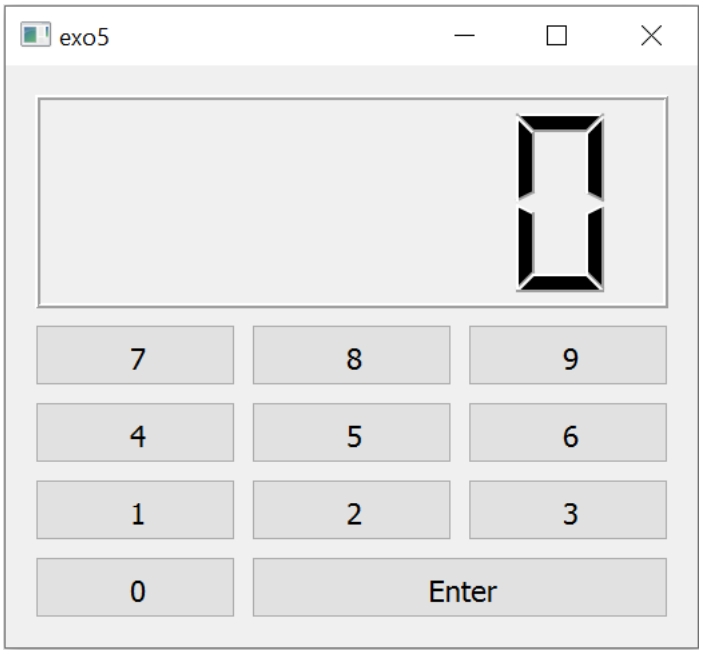
```
exo5::exo5(QWidget *parent) : QWidget(parent)
{
createwidgets();
placewidgets();
Connexionwidgets();
}
void exo5::createwidgets(){//implementation of the widgets
    b1= new QPushButton("7");
    b2= new QPushButton("8");
    b3= new QPushButton("9");
    b4= new QPushButton("4");
    b5= new QPushButton("5");
    b6= new QPushButton("6");
    b7= new QPushButton("1");
     b8= new QPushButton("2");
      b9= new QPushButton("3");
       b10= new QPushButton("0");
       b11= new QPushButton("Enter");
    grid = new QGridLayout;
    //window= new QWidget;
    number =new QLCDNumber;
   layout=new QVBoxLayout;
    setLayout(layout); //make the layout visible
}
void exo5::placewidgets(){
number->setMinimumHeight(80);//set the height of the LCD nUmber
    grid->addWidget(b1,0,0,1,1);
    grid->addWidget(b2,0,1,1,1);
        grid->addWidget(b3,0,2,1,1);
layout->addWidget(number);
        // 1st row
        grid->addWidget(b4,1,0,1,1);
        grid->addWidget(b5,1,1,1,1);
        grid->addWidget(b6,1,2,1,1);

        grid->addWidget(b7,2,0,1,1);


        grid->addWidget(b8,2,1,1,1);
        grid->addWidget(b9,2,2,1,1);
        grid->addWidget(b10,3,0,1,1);
        grid->addWidget(b11,3,1,1,2);

layout->addLayout(grid);//add the grid to the layout

}
void exo5::Connexionwidgets(){
}
```

//implement the coordinates of the buttons and add them to the grid

Main Class

```
#include"exo5.h" //to call the exo5 class
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    auto name= new exo5; //create "name" to call the class
name->show(); //show of results
return app.exec(); //exit after running
}
```

Result



What we learned?

We learned the purpose of the gridlayout , how it work and also the new type of displaying numbers is the LCDNUmber as a new technique of displaying difits-numbers

Conclusion

Besides that, the layouts automatically arranging child widgets within a widget

it's takes charge of the following tasks:

- ✓ POSITIONING OF CHILD WIDGETS.
- ✓ SENSIBLE DEFAULT SIZES FOR WINDOWS.
- ✓ SENSIBLE MINIMUM SIZES FOR WINDOWS.
- ✓ RESIZE HANDLING.
- ✓ AUTOMATIC UPDATES WHEN CONTENTS CHANGE:
  - Font size, text or other contents of child widgets.
  - Hiding or showing a child widget.
  - Removal of Child widgets.

I hope you enjoy my report and see you in other ones😊