Practical course report #2

Supervisor:

Dr. Anass Belcaid

[6-11-2021]

الجامعة الأورومتوسطية بفاس
EUROMED UNIVERSITY OF FES
UNIVERSITÉ EUROMED DE FÈS

EIDIA

Authored by: Yahia BENCHEDDI

Executive Summary

This project was a part of the first week assignment of a course on Human-Machine interface in C++ taught by Dr.Belcaid from department of computer science at UEMF. The goal of this project was manipulate multiple types of layout, so the project only contains the interfaces of each given program without and connection between buttons and output.

Program overview

Objectives

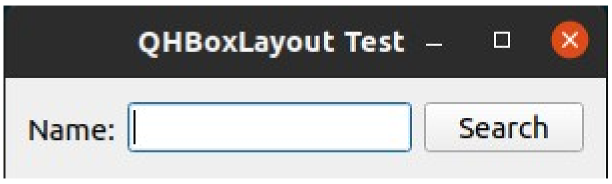
The goal of the program was to write the whole code for each of the four given assignments, strating with manipulating QHBoxLayout, then Nested Layouts, Bug Report Form, and lastly the Grid Layout.

The first two projects were coded in the class with the help of our professor in order to get the hand of it so basically they were a bit easy, the first one was coded whole in the main.cpp:

hbox.cpp:

```cpp
#include "mainwindow.h"
#include <QApplication>
#include <QHBoxLayout>
#include <QLineEdit>
#include <QPushButton>
#include <QLabel>

int main(int argc, char* argv[])
{
  QApplication app(argc, argv);

  QWidget* window = new QWidget();
  window->setWindowTitle("QHBoxLayout Test");
  window->resize(300,50);
  auto layout = new QHBoxLayout;
  window->setLayout(layout);
  auto label = new QLabel("Name:");
  label->show();
  auto text = new QLineEdit;
  auto button = new QPushButton("Search");
  layout->addWidget(label);
  layout->addWidget(text);
  layout->addWidget(button);

  window->show();

  return app.exec();
}
```

Here's the output of the program:



the Nested Layout project was a bit fun specially with all the layouts that needed to be added:

nestelayout.h:

```
#include <QMainWindow>
#include <QLineEdit>
#include <QPushButton>
#include <QHBoxLayout>
#include <QLabel>
#include <QVBoxLayout>
#include <QBoxLayout>
#include <QCheckBox>

class NestedLayout : public QWidget
{
    Q_OBJECT

public:
    //constructor
    NestedLayout(QWidget *parent = nullptr);

protected:
  void createWidgets();
  void placeWidgets();

private:
  QLabel *label;
  QLineEdit *line;
  QPushButton *search;
  QPushButton *close;
  QCheckBox *match;
  QCheckBox *backword;
  QHBoxLayout *layoutP;
  QVBoxLayout *layoutS;
};
```

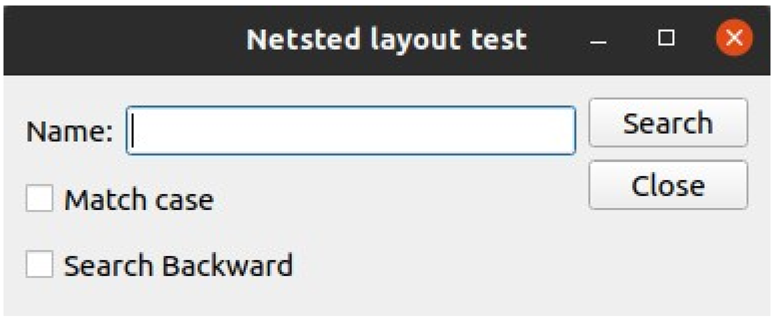
nestedlayout.cpp:

```
#include "nestedlayout.h"
#include "ui_mainwindow.h"
#include <QWidget>

NestedLayout::NestedLayout(QWidget *parent) : QWidget(parent)
{
    createWidgets();
    placeWidgets();
}

void NestedLayout::createWidgets(){
    label = new QLabel("Name:");
    line = new QLineEdit();
    search = new QPushButton("Search");
    close = new QPushButton("Close");
    match = new QCheckBox("Match case");
    backword = new QCheckBox("Search Backward");

   layoutP = new QHBoxLayout();
   layoutS = new QVBoxLayout();
    }
void NestedLayout::placeWidgets(){
    auto mainLayout = new QHBoxLayout;
    auto leftLayout = new QVBoxLayout;
    auto topLeftLayout = new QHBoxLayout;
    auto RightLayout = new QVBoxLayout;

    setLayout(mainLayout);
    leftLayout->addLayout(topLeftLayout);
    leftLayout->addWidget(match);
    leftLayout->addWidget(backword);

    topLeftLayout->addWidget(label);
    topLeftLayout->addWidget(line);

    RightLayout->addWidget(search);
    RightLayout->addWidget(close);
    RightLayout->addStretch(1);

    mainLayout->addLayout(leftLayout);
    mainLayout->addLayout(RightLayout);
}
```

main.cpp:

```
#include "nestedlayout.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    auto  window = new NestedLayout;
       window->setWindowTitle("Netsted layout test");
       window->show();
    return app.exec();
}
```

And here’s the output as requested:



Next we have the Bug Report Form project, this one was a bit of a sticky situation but it was very informing, because I started coding it by declaring every single element I’ll be using, then I discovered in the qt documentation the QFormLayout that made the task really easy, I left the old declared elements in the code but commented:

bugreport.h:

```
#include <QWidget>
#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QLayout>
#include <QSpinBox>
#include <QFormLayout>
#include <QDialogButtonBox>
#include <QVBoxLayout>
#include <QComboBox>
#include <QTextEdit>

class BugReport: public QWidget{
  Q_OBJECT

public:
    BugReport(QWidget * parent = nullptr);

protected:
    void createWidgets();
    void placeWidgets();

protected:
    QVBoxLayout * mainlayout;
//    QLayout * layout1;
//    QLayout * layout2;
//    QLayout * layout3;
//    QLayout * layout4;
//    QLayout * layout5;
//    QLayout * layout6;
//    QLayout * layout7;
//    QLayout * layout8;

    QFormLayout *layout;

    QLineEdit * line1;
    QLineEdit * line2;
    QLineEdit * line3;
    QLineEdit * line4;
    QLineEdit * line5;
    QTextEdit * line6;

    QComboBox * combo;
//    QLabel * label1;
//    QLabel * label2;
//    QLabel * label3;
//    QLabel * label4;
//    QLabel * label5;
//    QLabel * label6;
//    QLabel * label7;

    QVBoxLayout * v1layout;
    QVBoxLayout * vlayout;

//    QPushButton * rst;
//    QPushButton * sbmt;
//    QPushButton * dntSbmt;
//    QSpinBox * spin;

    QDialogButtonBox * buttonbox;

};
```

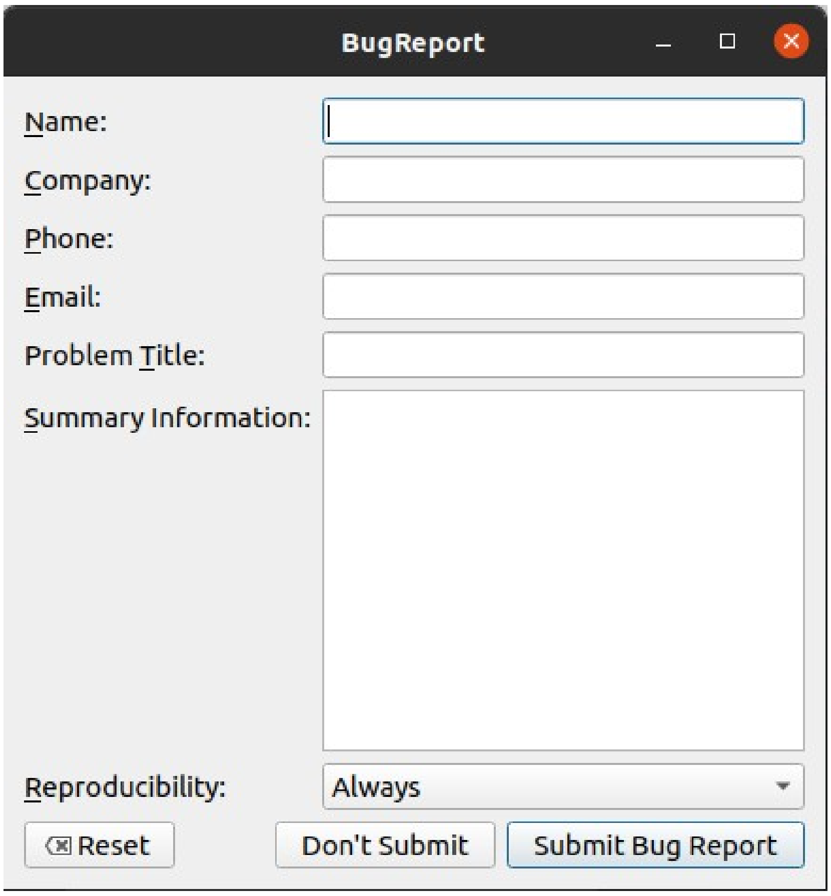
bugreport.cpp:

```
#include "bugreport.h"
#include <QWidget>
#include <QLayout>
#include <QFormLayout>
#include <QApplication>
#include <QVBoxLayout>

BugReport::BugReport(QWidget * parent): QWidget(parent){

    //Creatign the widgets
    createWidgets();

    //place Widgets
    placeWidgets();

}

void BugReport::createWidgets()
{
        mainlayout = new QVBoxLayout();
    //    setLayout(mainlayout);
    //    setWindowTitle("Report Bug");
    //    layout1 = new QHBoxLayout();
    //    layout2 = new QHBoxLayout();
    //    layout3 = new QHBoxLayout();
    //    layout4 = new QHBoxLayout();
    //    layout5 = new QHBoxLayout();
    //    layout6 = new QHBoxLayout();
    //    layout7 = new QHBoxLayout();
    //    layout8 = new QHBoxLayout();
    layout = new QFormLayout;

//    label1 = new QLabel("Name:");
//    label2 = new QLabel("Company:");
//    label3 = new QLabel("Phone:");
//    label4 = new QLabel("Email:");
//    label5 = new QLabel("Problem title:");
//    label6 = new QLabel("Summary information:");
//    label7 = new QLabel("Reproducibility:");

    line1 = new QLineEdit();
    line2 = new QLineEdit();
    line3 = new QLineEdit();
    line4 = new QLineEdit();
    line5 = new QLineEdit();
    line6 = new QTextEdit();

    //    rst = new QPushButton("Reset");
    //    sbmt = new QPushButton("Submit Bug Report");
    //    dntSbmt = new QPushButton("Don't Submit");

    buttonbox = new QDialogButtonBox;

}

void
BugReport::placeWidgets()
{
    layout->addRow(tr("&Name:"), line1);
    layout->addRow(tr("&Company:"), line2);
    layout->addRow(tr("&Phone:"), line3);
    layout->addRow(tr("&Email:"), line4);
    layout->addRow(tr("Problem &Title:"), line5);
    layout->addRow(tr("&Summary Information:"),line6);

    combo = new QComboBox;
    combo->addItem(tr("Always"));
    combo->addItem(tr("Sometimes"));
    combo->addItem(tr("Rarely"));
    layout->addRow(tr("&Reproducibility:"),combo);

    buttonbox->addButton(tr("Submit Bug Report"),
QDialogButtonBox::AcceptRole);
    buttonbox->addButton(tr("Don't Submit"),
QDialogButtonBox::RejectRole);
    buttonbox->addButton(QDialogButtonBox::Reset);

    mainlayout->addLayout(layout);
    mainlayout->addWidget(buttonbox);
    setLayout(mainlayout);

}
```

main.cpp:

```
#include "bugreport.h"
#include <QApplication>

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

And here’s the output of the code:



Reproducibility: Always

And last we have the grid layout or the calculator, this one was so good I had to get some help from my friend:

gridlayout.h:

```
#ifndef GRIDLAYOUT_H
#define GRIDLAYOUT_H

#include <QtWidgets>

#pragma once
namespace Ui {
class calculator;
}
class GridLayout : public QWidget
{

public:
  //constructor
    explicit GridLayout(QWidget *parent = nullptr);
    void creatingWdgets();
    void positionWidgets();
    void makeConnections();
private:
  QPushButton *buttons[10];
  QPushButton *bEnter;
  QLCDNumber *lcd;
  QVBoxLayout *mainLayout;
  QGridLayout *grid;
};
#endif // GRIDLAYOUT_H
```

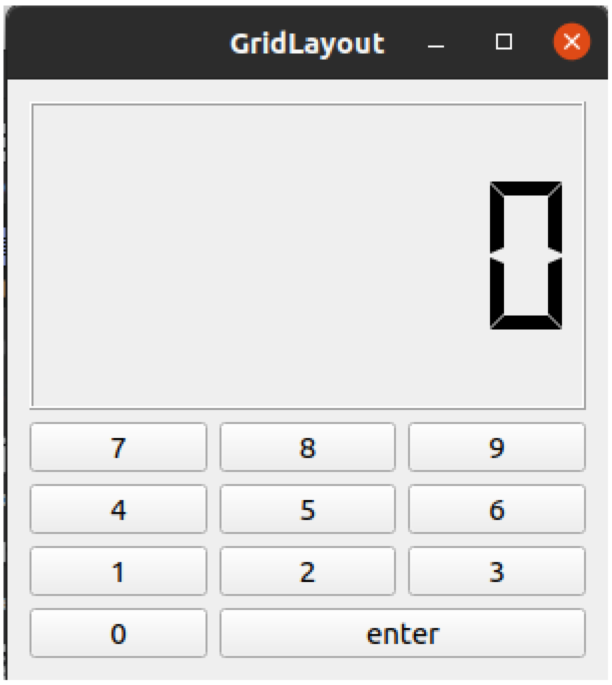
gridlayout.cpp:

```
#include "gridlayout.h"
#include <string>

GridLayout::GridLayout(QWidget* parent):QWidget(parent)
{
    creatingWdgets();
    positionWidgets();

}

void GridLayout ::creatingWdgets(){
    for(int i=0;i<10;i++){
       QString s = QString::number(9-i);
        buttons[i]=new QPushButton(s);
    }
    bEnter =new QPushButton("enter");
    lcd = new QLCDNumber();
    lcd->setSegmentStyle(QLCDNumber::Flat);

}
void GridLayout :: positionWidgets(){
   mainLayout = new QVBoxLayout();
   grid = new QGridLayout();

   int k = 0;
   for(int i=1;i<4;i++){
       for(int j=0;j<3;j++){
        grid->addWidget(buttons[k],i,2-j);
        k++;
        lcd->setMinimumHeight(80);
        lcd->setDigitCount(6);
       }
   }
    grid->addWidget(buttons[9],4,0);
    grid->addWidget(bEnter,4,1,1,2);
    mainLayout->addWidget(lcd);
    mainLayout->addLayout(grid);
    resize(300,300);
    setLayout(mainLayout);

}
```

main.cpp:

```
#include <QApplication>
#include "gridlayout.h"

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

And finally here’s the requested output:



Conclusion

The practical course was very helpful and fun to code, thank you for your time professor and hope you find it intresting.