Sidenote : aveti numele fisierului in care se afla fiecare tutorial langa nume :)

T1

Qt este cross platform, poate fi utilizat pe unix, windows :)

T2 hello-t2

#include <QCoreApplication>

#include <QDebug> //folosim qdebug pt meniul de debug

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv); //creeaza o instanta a aplicatiei

qDebug() << "Hello World"; //afiseaza hello world

return a.exec();

}

T3 T3

din QT

MainWindow - numele clasei

.ui – extensia din qt pt form-uri

Structural, toate fisierele sunt xml

Q Object - baselevel pt orice din qt (ca object in java)

Pentru a edita Gui-ul intram in mainwindow.ui

si putem da drag and drop componentelor dorite

apoi mergem in mainwindow.cpp si putem modifica textul

ui->pushButton->setText

in design -> signals and slots -> drag de la componenta -> se deschide Configure Connection

apasam pe clicked() apoi bifam pe show signals si dam close din main window pt a

->> buton va inchide fereastra la apasare

T4 - T3

Signals and slots

-Fiecare signal si slot trb sa aibe acelasi numar de argumente

n cazul nostru avem valueChanged(int) din horizontal slider si setValue(int) , ambele avand 1 arg

-Pentru a sterge o connectiune, o selectam si apasam delete

putem avea mai multe signal-uri conectate la un singur slot;

T5 -T3

\*in design\*

Typehere :: File

Typehere :: New File

^ creeaza o actiune (action) careia ii putem da drag and drop in ui

daca dam click dreapta pe action-> go to slot -> triggered -> genereaza cod

pt un form nou -> source files -> add new design form class

T6 - T6

Layouts - butoane in ribbon in .ui - se folosesc cu selectarea componentelor dorite

- se folosesc deoarece qt e cross platform

-Pt butonul de ok am folosit accept() din slots -> acesta accepta optiunea si inchide fereastra de dialog

Buddies - alege ce 2 sau mai multe componente se pot bolda

Tab order - alege ordinea in care se boldeaza un obiect atunci cand se apasa tab

T7 - minimal

HTML Aware Widgets - toate label-urile sunt HTML aware, deci putem folosi sintaxe din html

ca si <b> </b> pt bold si <i> </i> pt italic

minimal.pro

QT += widgets

TARGET = minimal

SOURCES += \

main.cpp

main.cpp

#include <QApplication>

#include <QLabel>

int main(int argc, char \*argv[]){

QApplication app(argc,argv);

//QLabel \*label= new QLabel("<b>Hello</b> <i>Worlddd</i>dddd:)");

QLabel \*label= new QLabel("<h2>Hello</h2> <font color = red><i>Worlddd</i></font>dddd:)");

label->show(); //se incapsuleaza automat intr o fereastra

return app.exec();

}

T8 - Layouts prin cod - minimal

QWidget \*window = new QWidget();

//atat v a afisa doar o fereastra mare, pt ca nu are parametrii

window->setWindowTitle("<My APP>");

QPushButton \*button1 = new QPushButton("1");

QPushButton \*button2 = new QPushButton("2");

QPushButton \*button3 = new QPushButton("3");

// QHBoxLayout \*hlayout = new QHBoxLayout; //layout orizontal

QVBoxLayout \*vlayout = new QVBoxLayout; //layout vertical

//hlayout->addWidget(button1);

//hlayout->addWidget(button2);

//hlayout->addWidget(button3);

// window->setLayout(hlayout);

vlayout->addWidget(button1);

vlayout->addWidget(button2);

vlayout->addWidget(button3);

window->setLayout(vlayout);

window->show();

return app.exec();

T9 - QGridLayout - minimal

QGridLayout \*layout = new QGridLayout();

QLabel \*label1 = new QLabel("Name");

QLineEdit \*txtName = new QLineEdit;

layout->addWidget(label1,0,0); //(widget, row, column) //param functiei

layout->addWidget(txtName,0,1);

QLabel \*label2 = new QLabel("Age");

QLineEdit \*txtAge = new QLineEdit;

layout->addWidget(label2,1,0); //(widget, row, column) //param functiei

layout->addWidget(txtAge,1,1);

QLabel \*label3 = new QLabel("Height");

QLineEdit \*txtHeight = new QLineEdit;

layout->addWidget(label3,2,0); //(widget, row, column) //param functiei

layout->addWidget(txtHeight,2,1);

QPushButton \*button = new QPushButton("Ok");

layout->addWidget(button,3,0,1,2);// (widget, row, column, how many rows we want it to span, how many collumns we want it to span)

window->setLayout(layout);

T10 - t10

splitters

Se selecteaza obiectele ui dorite si se alege din ribbon horizontal sau vertical splitter

ca sa revenim la pozitiile initiale dam click dr pe form - lay out - break

T11 - Dirs

QDir - directoare

#include <QDebug>

#include <QDir>

cand folosim un path hardcoded schimbam orientarea /-urilor

in mod normal (in windows, linux): C:\Users\Acasa\Desktop\Uni\Programare\TAP\Proiecte\Qt

in qt: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt"

qFileInfo tine informatia despre fisiere

#include <QCoreApplication>

#include <QDebug>

#include <QDir>

#include <QFileInfo>

#include <QString>

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

//QDir mDir("C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt");

//qDebug() << mDir.exists(); //returneaza true daca exista si false daca nu

/\*foreach(QFileInfo mItm , mDir.drives())// ne v a selecta toate drive-urile calculatorului

{

qDebug() << mItm.absoluteFilePath();

}\*/

// QDir mDir; // va lua dir curent

//QString mPath="C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/Dirs/exemplu";

/\* if(!mDir.exists(mPath)) // folosing mPath se va creea un director daca nu exista

{

mDir.mkpath(mPath); //creaza un director

qDebug()<<"Created!";

}

else

{

qDebug()<<"Already Exists!";

}

\*/

QDir mDir("C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt");

foreach(QFileInfo mItm , mDir.entryInfoList()){ // ne va afisa toate path-urile fisierelor din mDir

// qDebug() << mItm.absoluteFilePath(); // afiseaza path-ul lui mItm

if(mItm.isDir()) qDebug() << "Dir:" << mItm.absoluteFilePath(); //isDir verifica daca e diector

if(mItm.isFile()) qDebug() << "File:" << mItm.absoluteFilePath(); //isFile verifica daca e file

}

return a.exec();

}

A screenshot of a computer program

Description automatically generated

T12 – File

Qfile – interfata de citire si scriere in fisiere

#include <QCoreApplication>

#include <QFile>

#include <QString>

#include <QDebug>

#include <QTextStream> //interfata de scriere citire text

//functie scriere

void Write (QString Filename){

QFile mFile(Filename);

if(!mFile.open(QFile::WriteOnly | QFile::Text)){//verifica daca e writeonly si text si deschis

qDebug() << "Couldnt open";

return;

}

QTextStream out(&mFile); // referinta catre fisierul deschis

out<<"Hello World";

mFile.flush(); //

mFile.close();

}

void Read (QString Filename){

QFile mFile(Filename);

if(!mFile.open(QFile::ReadOnly | QFile::Text)){ //verifica daca e readonly si text si deschis

qDebug() << "Couldnt open";

return;

}

QTextStream in(&mFile); // referinta catre fisierul deschis // mai multe functii putem gasi in help

QString mText = in.readAll(); //citeste tot din fisierul text

qDebug() << mText;

mFile.close();

}

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

QString mFilename = "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/myfile.txt";

Write(mFilename);

Read(mFilename);

return a.exec();

}

T13 – File

(Proiectul este qmake nu cmake la crearea acestuia)

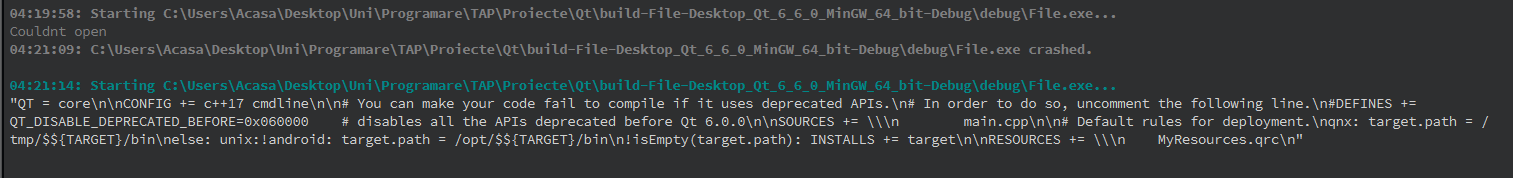
Resource files – collection of data that can be put into exec when its compiled and u can access that data at runtime

Se da Click dr in solution explorer -> add new -> qt -> resource file (asa se creeaza un nou resource file)

In fereastra de resurse, avem nevoie intai de un prefix, caruia ii putem adauga fisiere de resurse(alte .pro-uri , imagini , etc)

A screen shot of a computer code

Description automatically generated



T14 – GUI

Qlabel si Qt designer

La fel ca si java , avem tabela de proprietati;

Pentru Label-uri am putea ,

din proprietati : sa schimbam textul , sa deschidem un editor text tip word sau html

din cod ui->label->setText("<b>eyyy</b>"); si modificarile de tip bold etc se fac in sintaxe html

daca facem ambele , codul v a da override

T15 – GUI

Pushbutton

Pt a avea parte de o actiune la folosirea butonului dam click dr pe el -> go to slot -> clicked()

Acesta va genera o functie in care scrie

void Dialog::on\_pushButton\_clicked()

{

QMessageBox::information(this, "Title here", "Text here");

}

T16 – Gui

QLineEdit -> textbox din java

A white rectangular object with a white background

Description automatically generated

Echo mode (in proprietati) => vizualizarea a timpul scrierii

Putem avea normal (se vad literele in timp ce scrii)

Sau password – se vad stelute

Sau alternative

T17 – gui

Qcheckbox

A black text on a white background

Description automatically generated

Setarea starii checkbox-ului : ui->checkBox->setChecked(true);

Verificarea starii checkboxului : if( ui->checkBox-> isChecked())

T18 – Gui

Qradiobutton

A screenshot of a computer

Description automatically generated

Diferenta dintre radio button si checkbox este ca cu radio buttons poti allege o singura varianta pe cand cu checkbox-urile poti bifa mai multe

**void Dialog::on\_pushButton\_3\_clicked()**

**{**

**if(ui->radioButton1->isChecked()){**

**QMessageBox::information(this,ui->radioButton1->text(),"you like cats");**

**}**

**if(ui->radioButton2->isChecked()){**

**QMessageBox::information(this,ui->radioButton2->text(),"you like Dogs");**

**}**

**}**

T19 – Gui

ComboBox

A screenshot of a computer

Description automatically generated

T20 – Gui

Listwidget

Diferit de ListView

A white board with a white background

Description automatically generated

Folosit ca o lista de diferite variante

Fiecare varianta este un qlistwidgetobject cu numele item

QListWidgetItem \*itm =ui->listWidget->currentItem();

Acesta are diferite proprietati cum ar fi culoarea, alignment, foreground, background etc

//T21 Gui2

QTreeWidget

A screenshot of a computer

Description automatically generated

Vizualizare in tree, cum ar arata un set de fisiere

Se adauga in .h functii de adaugare root si copil



Apoi se implementeaza in .cpp ambele functii A screen shot of a computer code

Description automatically generated A screen shot of a computer code

Description automatically generated

Pentru Child I se da un nod root/parinte ca pointer

//T22 – continuare T21

//T23 MyActions

Actions

Clasa ce interactioneaza cu toolbar-ul si menubar-ul

Pot fi scrise : A screenshot of a computer

Description automatically generated

Daca dam dublu click pe una din actiuni de aici : A screenshot of a computer

Description automatically generated

apare

A screenshot of a computer program

Description automatically generated

Adaugarea de imagini se face din resurse (vezi tutorial 13)

Ca sa schimbam imaginea unei actiuni , la icon -> … -> selectam din imagine

Actiunile pot fi utilizate din toolbar precum si din file-ul nostrum

T24 – Gui3

QSlider si QprogressBar

Conectam manual semnalul slider la slotul progress bar

in cazul nostru avem valueChanged(int) din horizontal slider si setValue(int)

T25 – StatusBar

Se afla implicit in partea de sud a form-ului mainwindow

A white background with black border

Description automatically generated

Unde putem pune widget-uri

T25- Message

QMessageBox

Exista 3 tipuri de message box

Information A screenshot of a computer

Description automatically generated

Question A screenshot of a computer

Description automatically generated

Warning A screenshot of a computer

Description automatically generated

T27 –Timer

QTimer – clasa cu care putem declara un pointer

QTimer \*timer;

Si a il folosi

L-am folosit pt a afisa “Timer executed” dupa o durata de timp

timer = new QTimer(this);

connect(timer,SIGNAL(timeout()),this,SLOT(MySlot()));

timer->start(1000);