Pemrograman Qt 12 – Kalkulator Sederhana dengan QLineEdit dan Casting QString to int

Bismillahirrahmanirrahim.

Belajar pemrograman GUI sering kita awali dengan membuat aplikasi kalkulator. Kita menemukannya pada banyak tutorial. Hal itu wajar karena dalam 1 aplikasi kalkulator bisa terkumpul banyak elemen GUI dan semuanya harus dihubungkan. Oleh karena itu cocok sekali untuk latihan pemula. Kali ini saya akan membuat sebuah kalkulator yang sederhana sekali dengan memakai kelas QLineEdit dan teknik casting (konversi tipe data) dari QString ke int (dan sebaliknya). Semoga tulisan ini bermanfaat.

1. Spesifikasi Sistem

- Ubuntu 12.04
- Qt Creator 2.4.1
- Qt 4.8.0 (32 bit)

2. Daftar Kelas

- QLineEdit
- QIntValidator
- QString
- QPushButton

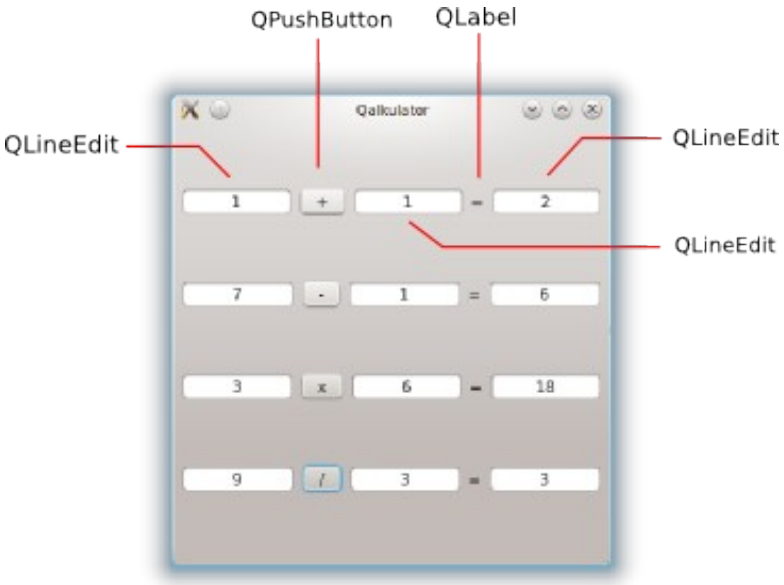
3. Daftar Method

- text() (untuk mengambil teks)
- setText() (untuk menampilkan teks)
- setAlignment (untuk mengatur perataan teks)

4. Arah Tulisan Ini

Aplikasi kalkulator yang akan dibuat bukan seperti yang ditemukan di Windows atau Ubuntu. Itu terlampau canggih. Saya hanya akan membuat satu jendela dengan empat baris elemen. Tiap-tiapnya adalah operasi +, -, x, dan /. Jadi, hanya ada 4 operasi matematika. Masukan dimasukkan pada satu QLineEdit dan satu QLineEdit di sebelahnya. Di tengahnya

ada tombol. Jika tombol diklik, maka dua masukan dioperasikan lalu dikeluarkan hasilnya pada QLineEdit keluaran. Ini gambarnya.



5. Kode

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QtGui>

namespace ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);

public:
    QString nilai_a, nilai_b, nilai_c;
    QString *piala_bergilir;
    QLineEdit *input_a;
    QLineEdit *input_b;
    QLineEdit *output;

    QLineEdit *input_a_kurang;
    QLineEdit *input_b_kurang;
    QLineEdit *output_kurang;

    QLineEdit *input_a_kali;
    QLineEdit *input_b_kali;
    QLineEdit *output_kali;

    QLineEdit *input_a_bagi;
    QLineEdit *input_b_bagi;
    QLineEdit *output_bagi;

    QLabel *sama_dengan;
    QLabel *sama_dengan_kurang;
    QLabel *sama_dengan_kali;
    QLabel *sama_dengan_bagi;

public slots:
    void jumlahkan();
    void kurangkan();
    void kalikan();
    void bagikan();

};

#endif // MAINWINDOW_H
```

mainwindow.cpp

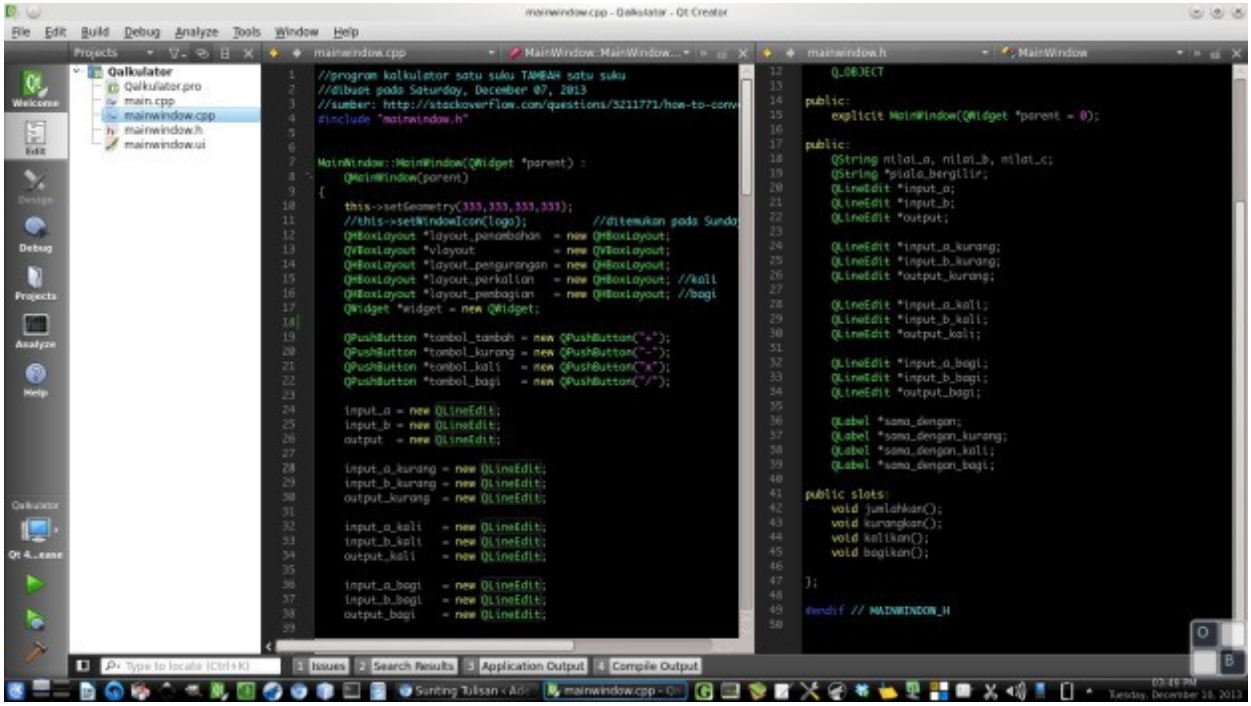
```
//program kalkulator satu suku TAMBAH satu suku
//dibuat pada Saturday, December 07, 2013
//sumber: http://stackoverflow.com/questions/3211771/how-to-convert-int-to-qstring
#include "mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
QMainWindow(parent)
{
this->setGeometry(333,333,333,333);
//this->setWindowIcon(logo); //ditemukan pada Sunday, December 08, 2013 06:27 AM
QHBoxLayout *layout_penambahan = new QHBoxLayout;
QVBoxLayout *vlayout = new QVBoxLayout;
QHBoxLayout *layout_pengurangan = new QHBoxLayout;
QHBoxLayout *layout_perkalian = new QHBoxLayout; //kali
QHBoxLayout *layout_pembagian = new QHBoxLayout; //bagi
QWidget *widget = new QWidget;

QPushButton *tombol_tambah = new QPushButton("+");
QPushButton *tombol_kurang = new QPushButton("-");
QPushButton *tombol_kali = new QPushButton("x");
QPushButton *tombol_bagi = new QPushButton("/");

input_a = new QLineEdit;
input_b = new QLineEdit;
output = new QLineEdit;

input_a_kurang = new QLineEdit;
input_b_kurang = new QLineEdit;
output_kurang = new QLineEdit;

input_a_kali = new QLineEdit;
input_b_kali = new QLineEdit;
output_kali = new QLineEdit;

input_a_bagi = new QLineEdit;
input_b_bagi = new QLineEdit;
output_bagi = new QLineEdit;

sama_dengan = new QLabel("=");
sama_dengan_kurang = new QLabel("=");
sama_dengan_kali = new QLabel("=");
sama_dengan_bagi = new QLabel("=");

//gantinya setInputMask yang kisruh
QIntValidator *valid = new QIntValidator(0,9);
//qintvalidator bisa menyaring supaya hanya angka integer saja yang masuk dalam
QLineEdit
//range-nya 0 - 999 dan bisa diubah-ubah sendiri
//ditemukan pada Saturday, December 07, 2013 10:07 PM

//masking
input_a->setAlignment(Qt::AlignHCenter);
input_b->setAlignment(Qt::AlignHCenter);
```

```
output->setAlignment(Qt::AlignHCenter);
input_a->setValidator(valid);
input_b->setValidator(valid);

//masking
//validasi, bukan masking lagi
input_a_kurang->setAlignment(Qt::AlignHCenter);
input_b_kurang->setAlignment(Qt::AlignHCenter);
output_kurang->setAlignment(Qt::AlignHCenter);
input_a_kurang->setValidator(valid); //hanya integer yang boleh masuk
input_b_kurang->setValidator(valid);

//masking lagi
input_a_kali->setAlignment(Qt::AlignHCenter);
input_b_kali->setAlignment(Qt::AlignHCenter);
output_kali->setAlignment(Qt::AlignHCenter);
input_a_kali->setValidator(valid);
input_b_kali->setValidator(valid);

//tetap masking
input_a_bagi->setAlignment(Qt::AlignHCenter);
input_b_bagi->setAlignment(Qt::AlignHCenter);
output_bagi->setAlignment(Qt::AlignHCenter);
input_a_bagi->setValidator(valid);
input_b_bagi->setValidator(valid);

//layout bagian penambahan
layout_penambahan->addWidget(input_a);
layout_penambahan->addWidget(tombol_tambah);
layout_penambahan->addWidget(input_b);
layout_penambahan->addWidget(sama_dengan);
layout_penambahan->addWidget(output);

//layout bagian pengurangan
layout_pengurangan->addWidget(input_a_kurang);
layout_pengurangan->addWidget(tombol_kurang);
layout_pengurangan->addWidget(input_b_kurang);
layout_pengurangan->addWidget(sama_dengan_kurang);
layout_pengurangan->addWidget(output_kurang);

//layout bagian perkalian
layout_perkalian->addWidget(input_a_kali);
layout_perkalian->addWidget(tombol_kali);
layout_perkalian->addWidget(input_b_kali);
layout_perkalian->addWidget(sama_dengan_kali);
layout_perkalian->addWidget(output_kali);

//layout bagian pembagian
layout_pembagian->addWidget(input_a_bagi);
layout_pembagian->addWidget(tombol_bagi);
layout_pembagian->addWidget(input_b_bagi);
layout_pembagian->addWidget(sama_dengan_bagi);
layout_pembagian->addWidget(output_bagi);

//layout sebenarnya
vlayout->addLayout(layout_penambahan);
```

```
vlayout->addLayout(layout_pengurangan);
vlayout->addLayout(layout_perkalian);
vlayout->addLayout(layout_pembagian);
widget->setLayout(vlayout);
setCentralWidget(widget);

connect(tombol_tambah, SIGNAL(clicked()), this, SLOT(jumlahkan()));
connect(tombol_kurang, SIGNAL(clicked()), this, SLOT(kurangkan()));
connect(tombol_kali, SIGNAL(clicked()), this, SLOT(kalikan()));
connect(tombol_bagi, SIGNAL(clicked()), this, SLOT(bagikan()));
}

void MainWindow::jumlahkan(){
int casting_a, casting_b;
nilai_a = input_a->text(); //QString = isine QLineEdit
nilai_b = input_b->text();
casting_a = nilai_a.toInt(); //int = QString
casting_b = nilai_b.toInt(); //int = QString

casting_a = casting_a + casting_b;
QString recast_a = QString::number(casting_a);
output->setText(recast_a);
}

void MainWindow::kurangkan(){
int casting_a, casting_b;
nilai_a = input_a_kurang->text(); //QString = isine QLineEdit
nilai_b = input_b_kurang->text();
casting_a = nilai_a.toInt(); //int = QString
casting_b = nilai_b.toInt(); //int = QString

casting_a = casting_a - casting_b; //inti fungsi pengurangan
QString recast_a = QString::number(casting_a);
output_kurang->setText(recast_a);
}

void MainWindow::kalikan(){
int casting_a, casting_b;
nilai_a = input_a_kali->text(); //QString = isine QLineEdit
nilai_b = input_b_kali->text();
casting_a = nilai_a.toInt(); //int = QString
casting_b = nilai_b.toInt(); //int = QString

casting_a = casting_a * casting_b; //inti fungsi perkalian
QString recast_a = QString::number(casting_a);
output_kali->setText(recast_a);
}

void MainWindow::bagikan(){
int casting_a, casting_b;
nilai_a = input_a_bagi->text(); //QString = isine QLineEdit
nilai_b = input_b_bagi->text();
casting_a = nilai_a.toInt(); //int = QString
casting_b = nilai_b.toInt(); //int = QString

casting_a = casting_a / casting_b; //inti fungsi pembagian
```

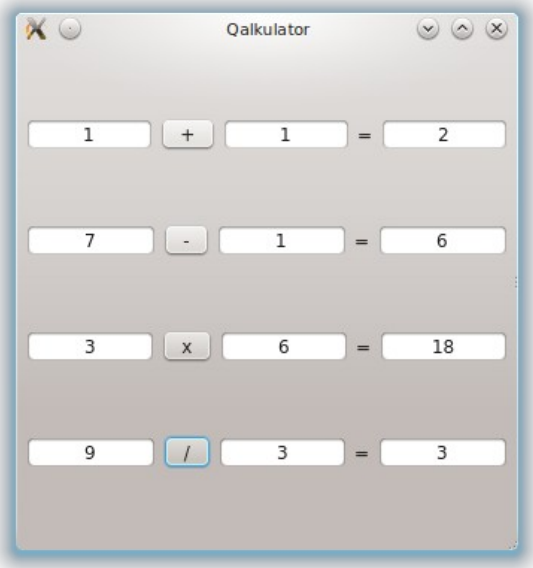
```
QString recast_a = QString::number(casting_a);
output_bagi->setText(recast_a);
}
```

6. Qt Creator dan Kode



7. Hasil

Demikianlah kalkulator kita.



8. Analisis

Yang perlu diperhatikan dalam program kalkulator ini hanya kode-kode dalam mainwindow.cpp saja. Di dalamnya ada pembuatan objek QLineEdit sebagai penerima input kita, ada validasi supaya hanya angka yang bisa dimasukkan, dan ada fungsi untuk mengonversi QString menjadi int dan sebaliknya untuk dioperasikan. Perlu diperhatikan juga bahwa logika yang dipakai dalam fungsi saya sangatlah sederhana. Masukan diambil sebagai QString, dikonversikan ke int, dioperasikan dengan sesama int dari masukan satunya, lalu hasilnya (int) dikonversikan lagi ke QString, lalu QString ini dikeluarkan. Hanya itu alur logikanya. Saya hanya mengingatkan pula bahwa konversi antartipe data itu namanya casting.

mainwindow.cpp

Ada beberapa hal yang penting di sini dari segi nonmatematik seperti validasi masukan berupa HANYA ANGKA dan perataan teks di dalam QLineEdit. Ada hal penting juga dalam perkara matematisnya karena kita memerlukan casting di sana. Jika Anda belum memahami bagaimana cara membangun GUI secara hard coding seperti kalkulator ini, silakan merujukke sini. Jika Anda belum paham bagaimana menghubungkan tombol dengan fungsinya (SIGNAL and SLOT), silakan merujuk ke sini.

1. Validasi Masukan

Kita harus tahu dulu bahwa sifat QLineEdit itu menerima semua teks baik angka maupun huruf. Bagaimana agar ia hanya bisa menerima angka saja? Jawabannya adalah validasi masukan. Ini dilakukan dengan method setValidator milik QLineEdit sendiri dan dibantu dengan kelas QIntValidator. Caranya dengan membuat objek QIntValidator bernama valid lalu memasukkan valid sebagai argumen pada setValidator. Hasilnya adalah QLineEdit hanya menerima masukan berupa angka saja sedangkan semua yang bukan angka tidak ikut masuk. Berikut kodenya.

```
QIntValidator *valid = new QIntValidator(0,9);

input_a->setValidator(valid);
input_b->setValidator(valid);
```

Maksud dari argumen (0,9) di atas adalah membuat objek bernama valid supaya hanya menerima maksimal 1 digit angka mulai awal = 0 sampai akhir = 9. Kalau diisikan (0,999) maka nanti objek valid bisa menerima mulai 0 sampai 999. Selain dari itu, tidak diterima. Di sinilah letak gunanya validasi masukan. Setelah menentukan jangkauan angka masukan, kita perlu memasangkan objek valid ke dalam QLineEdit dengan method setValidator di atas. Mudah, bukan?

2. Perataan (Alignment) Teks

Penting diperhatikan perilaku QLineEdit dalam menerima klik dan menempatkan kursor

pada awal masukan dari pengguna. Jenis perilaku paling umum (ada di semua OS) adalah ketika diklik pada posisi mana pun di sebuah LineEdit, maka kursor langsung berkedip-kedip pada sisi paling ujung kiri pada karakter urutan pertama. Ada kalanya perilaku sebuah LineEdit tidak wajar, entah peletakan kursornya tidak pada karakter urutan pertama, entah kursornya berubah bentuk, entah yang lainnya. Kita berusaha memberikan user experience yang terbaik dengan memberinya posisi kursor default pada karakter pertama tetapi pada perataan tengah untuk semua QLineEdit. Mengapa? Karena ini program kalkulator yang hanya menerima masukan 1 angka per QLineEdit. Sehingga perlu dibuat antarmuka yang sekali lihat sudah jelas angka-angkanya. Posisi perataan tengah adalah yang terbaik untuk hal ini. Hal ini (perataan tengah) dicapai dengan parameter Qt::AlignHCenter pada method setAlignment pada objek QLineEdit. Mudah, bukan? Berikut kodenya.

```
input_a->setAlignment(Qt::AlignHCenter);
input_b->setAlignment(Qt::AlignHCenter);
output->setAlignment(Qt::AlignHCenter);
```

Pilihan perataan dalam Qt ada banyak. Selain Qt::AlignHCenter, ada lagi Qt::AlignLeft (mode paling umum dipakai), Qt::AlignJustify, dan seterusnya. Ikuti saja code completion dari Qt Creator Anda.

3. Fungsi

Sebenarnya yang berhubungan langsung dengan aplikasi kalkulator ini selaku pemroses matematis hanyalah satu per satu fungsinya. Berikut keempat baris kodenya masing-masing.

```
casting_a = casting_a + casting_b;

casting_a = casting_a - casting_b;

casting_a = casting_a * casting_b;

casting_a = casting_a / casting_b;
```

Hanya saja, untuk melakukan operasi matematika pada program ini, tidak sesederhana itu. Harus ada mekanisme pengambilan masukan ke dalam variabel dahulu. Itu pun masih diperlukan casting ke int karena data yang diambil langsung dari QLineEdit selalu bertipe string (QString). Setelah casting, barulah fungsi matematis di atas bisa beraksi.

4. Casting

Konversi dari tipe data int ke string dan sebaliknya terjadi di sini. Berikut dua baris kodenya.

casting_a = nilai_a.toInt(); //int = QString
QString recast_a = Qstring::number(casting_a);

Baris pertama itu casting dari string (QString) ke int. Jadi, variabel nilai_a yang isinya berasal dari QLineEdit diubah jadi int lalu dimasukkan isinya ke casting_a. Variabel casting_a memang sengaja dibuat QString. Baris kedua itu casting dari int ke string (QString) yaitu dikembalikan ke tipe data semula. Tujuannya untuk mengeluarkan data ke QLineEdit kembali.

9. Kesimpulan

- Casting dari int ke QString bisa dilakukan dengan bentuk umum QString objek_string = QString::number(objek_integer);.
- Casting dari QString ke int bisa dilakukan dengan bentuk umum objek_int = objek_string.toInt();.
- Validasi masukan untuk menjamin hanya angka yang masuk dilakukan dengan membuat objek dari kelas QIntValidator.

10. Unduh Kode Sumber

Program kali ini bernama Qalkulator. Silakan unduh kode sumbernya dan bukalah pada Qt Creator Anda.

- Alamat: http://otodidak.freeserver.me/tarball/Qalkulator.tar.gz
- Ukuran: 17 KB

11. Referensi

http://stackoverflow.com/questions/3211771/how-to-convert-int-to-qstring

12. Tentang Dokumen Ini

Dokumen ini adalah versi PDF dari posting asli http://malsasa.wordpress.com/2013/12/11/pemrograman-qt-12-kalkulator-sederhana-dengan-qlineedit-dan-casting-qstring-to-int/. Dokumen ini ditulis dengan fonta Ubuntu 12pt. Dokumen ini disusun ulang dengan Libreoffice Writer 3.5. Dokumen ini selesai disusun pada 4 Maret 2014. Penulis mohon maaf jika terdapat kesalahan dalam dokumen ini.

13. Tentang Penulis

Penulis adalah warga Forum Ubuntu Indonesia . Penulis mendukung pendidikan perangkat lunak legal (terutama FOSS) untuk masyarakat. Penulis menyediakan buku-buku panduan Linux untuk pemula maupun ahli untuk diunduh secara gratis[1]. Penulis bisa dihubungi via SMS di nomor 0896 7923 7257.

1 http://malsasa.wordpress.com/pdf