Chapter 1 – Introduction C++ Development with Qt SDK

Pengantar Bahasa C dan C++

Bahasa C merupakan bahasa pemrograman tingkat menengah. Pada tahun 1972 bahasa C pertama kali dirancang oleh **Dennis M Ritchie** di Bell Laboratories. Kemudian tahun 1978 Dennis dan Brian W. Kernighan mempublikasikan bahasa C melalui "The C Programming Language" sehingga bahasa C dikenal banyak orang. Selanjutnya pada tahun 1989, akhirnya bahasa C distandardisasi ANSI (American National Standard Institute) sehingga bahasa C menjadi bahasa pemrograman standar hingga saat ini dan bisa dibuat kompilernya pada beberapa platform yang berbeda-beda.

Bahasa C dikatakan sebagai bahasa pemrograman terstruktur, fungsional karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagian (*subroutine/ module*). Fungsi-fungsi selain fungsi utama disebut *subroutine/ module* dan ditulis setelah fungsi utama (**main**) atau diletakkan pada file pustaka (*library*). Jika fungsi-fungsi diletakkan pada file pustaka dan akan dipakai disuatu program, maka nama file *header*-nya harus dilibatkan dalam program menggunakan *preprocessor directive* **#include.**

Kemudian bahasa C dikembangkan oleh Bjarne Stroustrup at Bell Labs menjadi bahasa C++. Pada bulan Oktober 1985 munculah buku The C + + Programming Language yang membahas tentang bahasa pemrograman itu langsung dari penciptanya sendiri. Bahasa C++ mengalami dua tahap evolusi:

- Pertama, dirilis oleh **AT&T Laboratories**, dinamakan **cfront**. C++ versi ini hanya berupa kompiler yang menterjemahkan bahasa C++ menjadi bahasa C untuk dieksekusi
- Kedua, **Borland International Inc**. mengembangkan kompiler C++ menjadi sebuah kompiler yang mampu mengubah C++ langsung menjadi bahasa mesin (assembly). Tahun 1990, C++ mulai diarahkan ke pengembangan **pemrograman berorientasi obyek**

Beberapa keunggulan C++ dibandingkan dengan bahasa C adalah sebagai berikut:

1. **Object-oriented programming**
   Bahasa pemrograman ini sangat mendukung pemrograman berorientasi obyek yang melihat permasalahan secara obyek dan bukan prosedural.
2. **Portability**
   Kita dapat mengkompilasi C++ kode yang sama di hampir semua jenis komputer dan sistem operasi tanpa membuat perubahan apapun. C++ adalah bahasa pemrograman yang paling sering digunakan di dunia.
3. **Brevity**
   Karena bahasa C++ merupakan bahasa tingkat tinggi, maka bahasa yang ditulis dengan bahasa C++ termasuk ringkas dan pendek dibandingkan bahasa-bahasa sejamannya pada waktu itu.

Bahasa C++ termasuk bahasa pemrograman tua yang sudah mendukung berbagai macam kata kunci yang mampu menyingkat proses penulisan kode program.

4. **Modular programming**
   Tubuh program pada bahasa C++ dapat terdiri dari beberapa file source code yang disusun secara terpisah dan kemudian dihubungkan secara bersama-sama. Kemampuan ini jelas menghemat waktu karena tidak perlu mengkompilasi ulang aplikasi yang lengkap ketika membuat satu perubahan, tetapi hanya file yang berisi perubahan itu saja. Selain itu, karakteristik ini memungkinkan kita untuk menghubungkan kode C++ dengan kode yang dibuat oleh bahasa lain, seperti bahasa Assembly dan C dan dapat digunakan kembali (reuseable).
5. **C Compatibility**
   C++ sangat backward compatible dengan bahasa C, sehingga aplikasi / kode program yang ditulis dengan bahasa C dapat digabungkan dengan bahasa C++ dengan sangat mudah, bahkan hampir tidak perlu mengubah kodenya.
6. **Speed**
   Kode yang dihasilkan dari kompilasi C++ sangat efisien, karena C++ mendukung prinsip dualitas bahwa dia mendukung bahasa tingkat tinggi dan bahasa tingkat rendah sehingga dapat mengurangi ukuran hasil kompiliasi dari bahasa itu.

Qt Overview

Qt Framework sudah sejak lama digunakan untuk mengembangkan aplikasi lintas platform. Qt sendiri dibuat pada tahun 1996 oleh perusahaan dari swedia yang bernama Trolltech. Karena sifarnya yang lintas platform anda dapat membuat aplikasi yang berjalan diatas platform Windows, Linux, dan Mac. Dengan Qt kode yang sama dapat dijalankan pada target platform yang berbeda.

Qt Framework sudah didesain sedemikian rupa sehingga mudah digunakan oleh developer tanpa harus mengorbankan fleksibilitas dan efisiensi. Qt mendukung pengembangan dengan dua bahasa utama yaitu Object Oriented C++ dan Java.

Qt Framework memiliki koleksi class library yang lengkap dan konsisten didukung oleh dokumentasi yang komprehensif. Class library tersebut berisi semua function yang dibutuhkan untuk mengembangkan aplikasi. Qt Framework membantu mengurangi pekerjaan developer dengan meningkatkan produktifitas penulisan kode untuk pengembangan yang bersifat RAD (Rapid Application Development).

Qt in Use

Beberapa aplikasi ternama yang dikembangkan dengan menggunakan Qt Framework diantaranya Google Earth map application, Skype telephony application, VLC media player, KDE desktop environment, dan masih banyak lagi. Qt juga digunakan pada berbagai perangkat elektronik dan aplikasi industri, sebagai contoh adalah mobile transportation system yang dibuat oleh Volvo, MeVisLab digital imaging platform, dan RealFlow visual effect application pada industri dunia hiburan.

Licensing

Mulai versi 4.5 Qt mendukung 3 macam license yaitu GPL, LGPL, dan Commercial License.

- GPL adalah jenis license yang paling banyak digunakan untuk aplikasi open source dan free.
- LGPL mirip dengan GPL hanya mempunyai aturan yang lebih longgar, sehingga developer masih dapat mengembangkan aplikasi yang bersifat closed source.
- Commercial License digunakan untuk mengembangkan aplikasi yang closed source secara keseluruhan, dimana LGPL masih mempunyai aturan tertentu. Commercial License juga menyediakan layanan dukungan email dan beberapa komponen tambahan.

Nokia Qt SDK

Pada tahun 2008, Nokia mengakuisisi Trolltech untuk memperlancar strategi pengembangan aplikasi lintas platform. Saat ini strategi Nokia adalah memfokuskan teknologi pengembangan aplikasi mobile pada Qt sebagai single app development framework.

Untuk platform mobile Qt mendukung beberapa sistem operasi diantaranya Symbian S60, Maemo, Symbian^3, dan MeeGo. Sedangkan untuk platform desktop Qt mendukung sistem operasi Windows, Linux, dan Mac.

Dengan didukungnya pengembangan berbasis Symbian S60 dan Symbian^3 maka pengembang Qt dapat membuat aplikasi yang ditujukan untuk 80 juta pengguna symbian devices.

Nokia mempermudah pengembangan aplikasi mobile dengan menyediakan Nokia Qt SDK yang berisi class library, IDE (Qt Creator), dan Qt Simulator. Anda dapat mendownload paket Nokia Qt SDK pada alamat http://www.forum.nokia.com.

Instalasi software Nokia Qt SDK

Untuk instalasi Nokia Qt SDK, pastikan Anda telah mendownload softwarenya pada alamat http://www.forum.nokia.com/Develop/Qt/ . Setelah membuka halaman diatas, maka Anda dapat memilih untuk mendownload salah satu jenis software Nokia Qt SDK yang sesuai dengan keinginan. Pilihannya adalah:

- 32- or 64-bit Microsoft Windows XP Service Pack 2, Windows Vista, or Windows 7.
- 32-bit Ubuntu Linux 8.04 or later.

- 64-bit Ubuntu Linux 8.04 or later.
- Apple Mac OS X 10.6 or later (Beta).

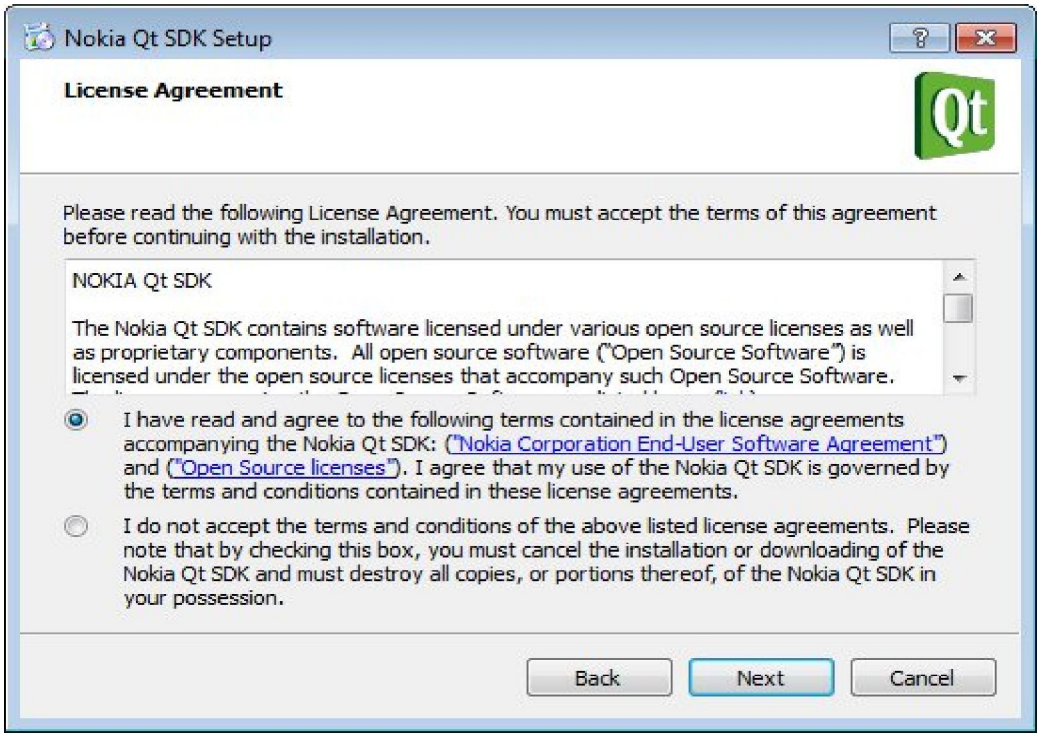
Semua installer tersedia versi online ataupun offlinenya. Pada contoh yang dibuat pada modul HOL ini, penulis mendownload versi untuk Windows 32 bit versi offline. Versi ini membutuhkan tempat diharddisk sebesar 935 MB. Sedangkan untuk instalasi penuh, membutuhkan ukuran sekitar 4 GB.

Setelah selesai didownload, maka akan terdapat file bernama **Nokia_Qt_SDK_Win_offline_v1_0_2_en.exe**. Langkah selanjutkan akan dilakukan instalasi software ini sebagai berikut:

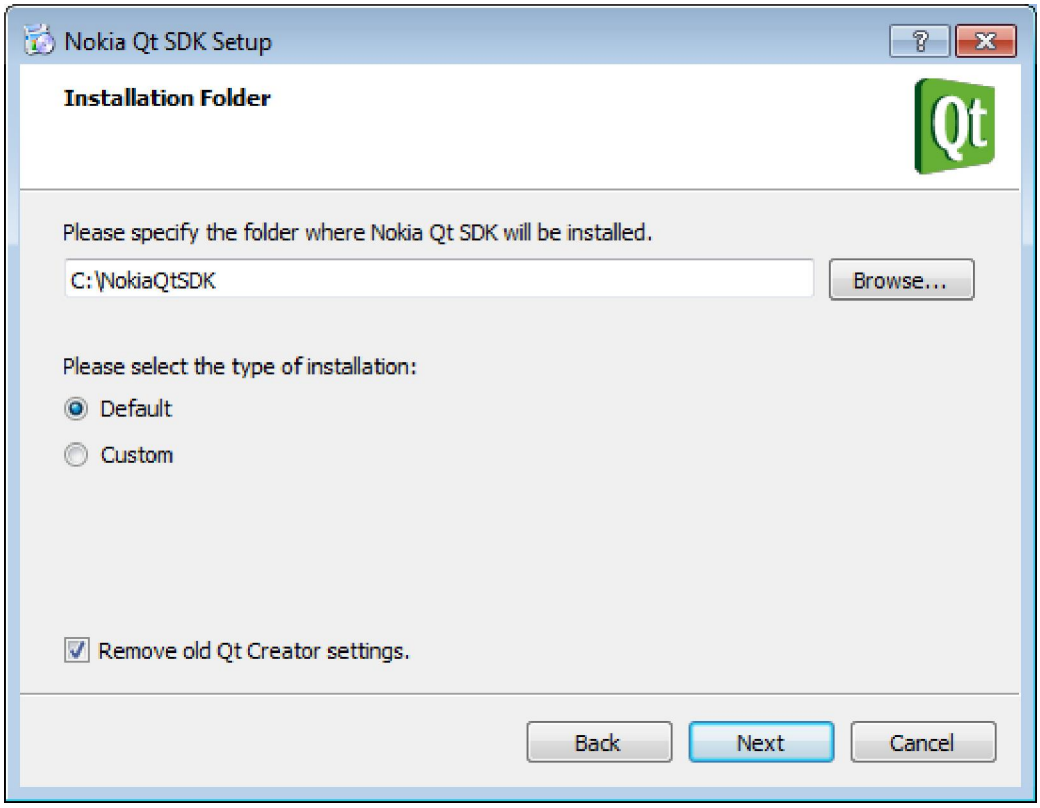
Labs 1. Instalasi Nokia Qt SDK

1. Double click file installer tersebut, sehingga muncul dialog sebagai berikut:

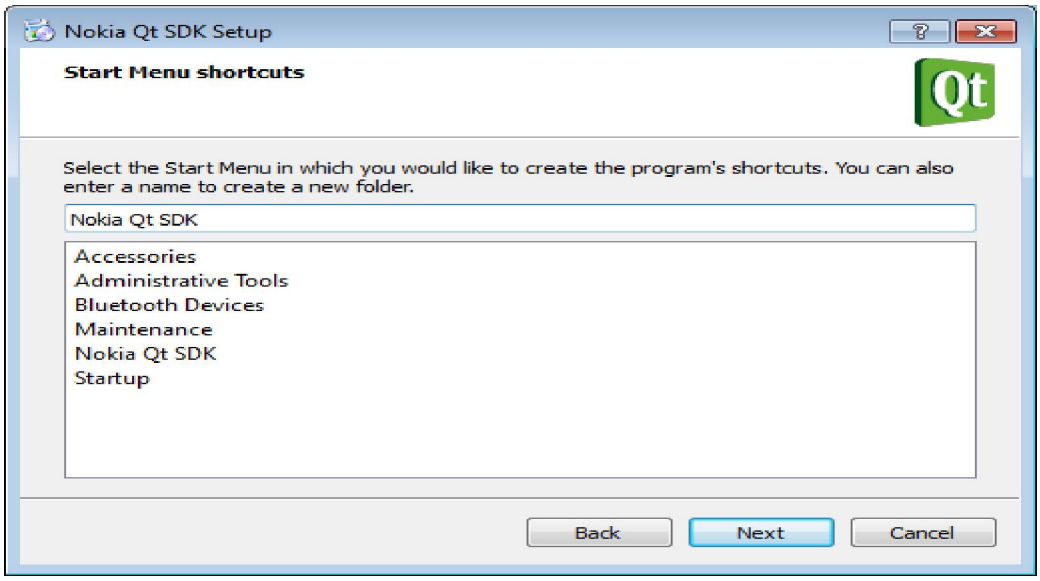
2. Kemudian, klik Next sehingga muncul kotak dialog berikutnya, yaitu License Agreement, dimana kita harus memilih opsi pertama dengan cara mengklik option buttonnya, sebagai berikut:



3. Kemudian kita akan diminta memasukkan lokasi instalasi tempat kita akan menginstall software ini. Defaultnya adalah C:\NokiaQtSDK. Lokasi ini dapat diganti sesuka hati, namun sangat disarankan diletakkan pada root drive folder, misalnya F:\NokiaQtSDK. Setelah itu kita harus memilih tipe instalasi, yaitu tipe Default atau Custom. Kita pilih Default.

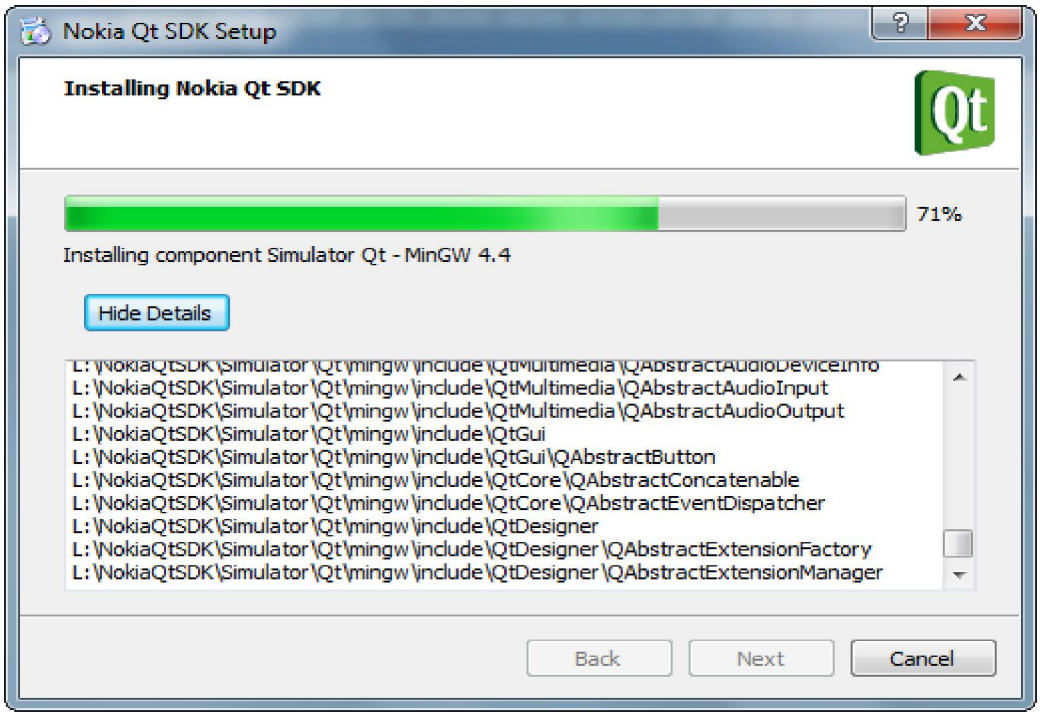


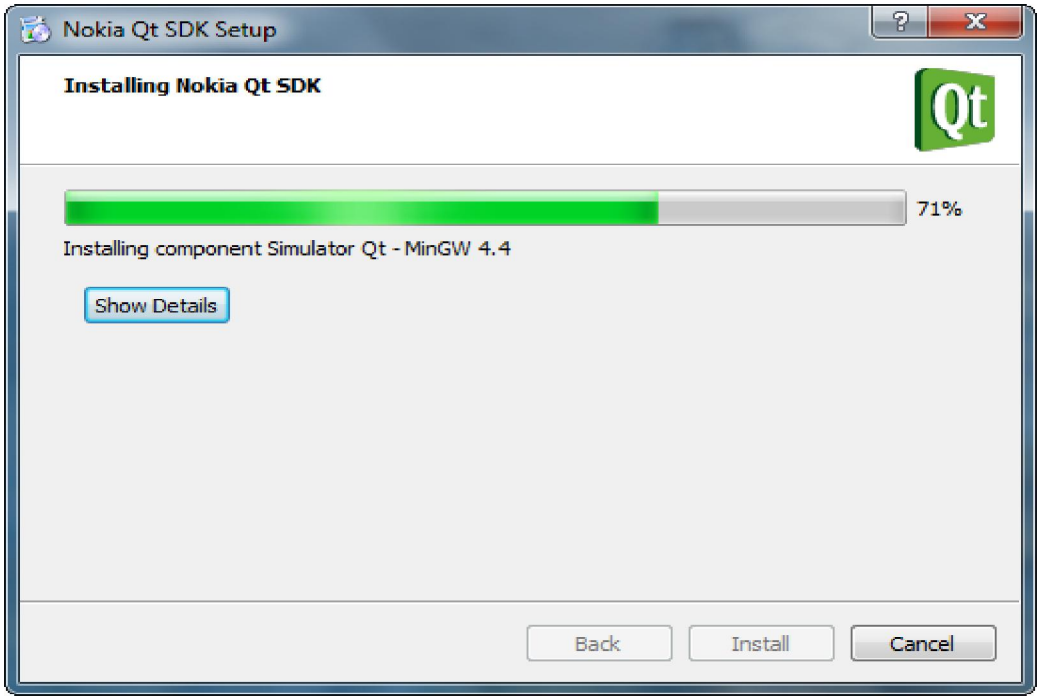
4. Setelah itu kita akan dihadapkan dengan dialog nama Start Menu Shortcut. Bagian ini kita biarkan apa adanya, lalu tekan Next.



5. Setelah itu proses instalasi akan dilakukan oleh Setup. Proses instalasi membutuhkan waktu yang cukup lama karena ukuran file instalasi dan data yang perlu diekstrak sangat besar. Kita harus menunggu mungkin sekitar 30 menit-an. Waktu ini dapat berbeda-beda pada setiap komputer. Pada mode ini kita dapat memilih apakah kita akan menampilkan Details atau tidak.

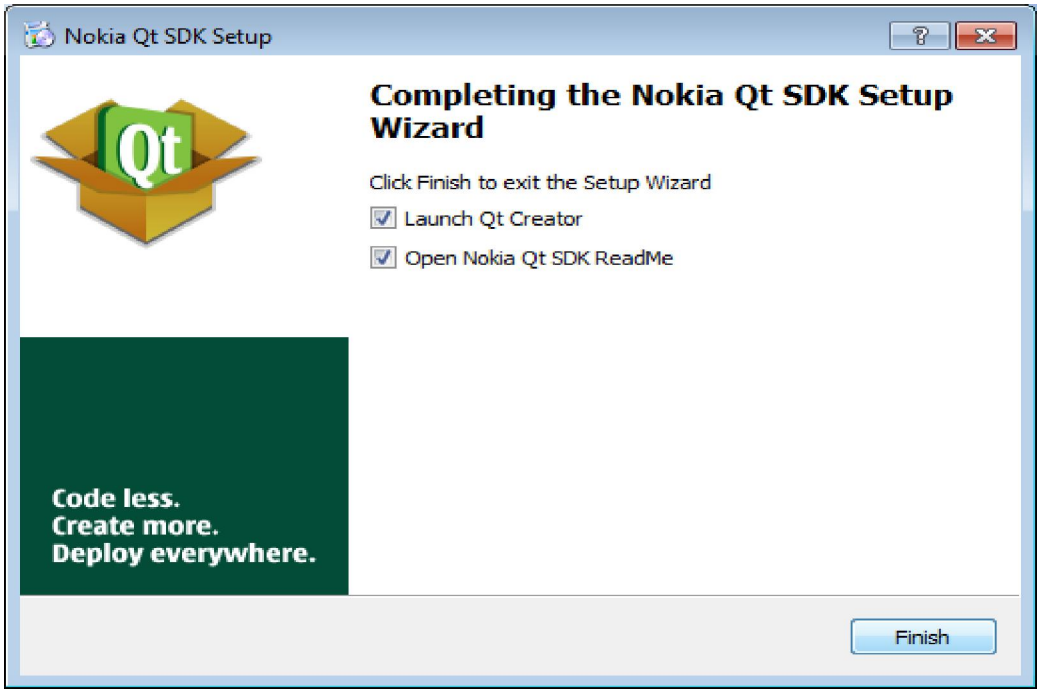
Untuk menampilkan Details, kita klik tombol Show Details, sedangkan untuk tidak menampilkannya, kita klik tombol Hide Details.



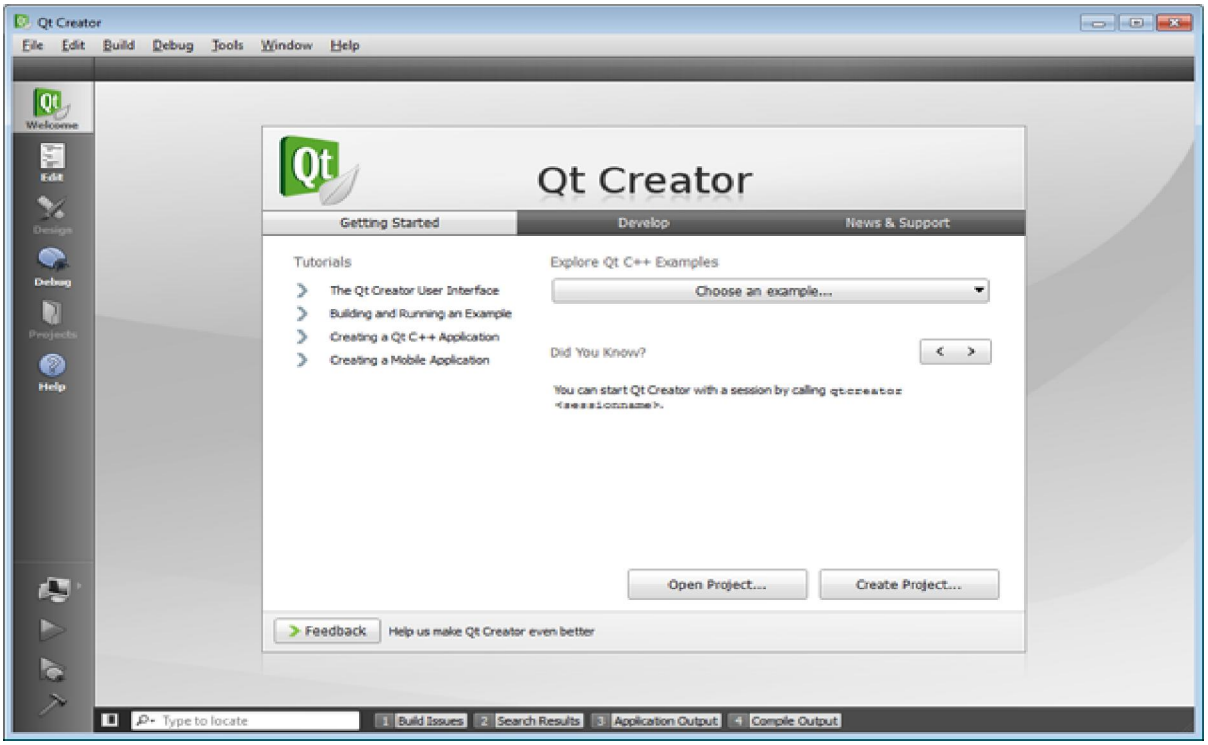


6. Setelah instalasi berhasil, maka akan muncul kotak dialog terakhir yang menginformasikan bahwa proses instalasi telah selesai. Kemudian kita centang checkbox Lauch Qt Creator untuk

menjalankan software ini pertama kalinya.



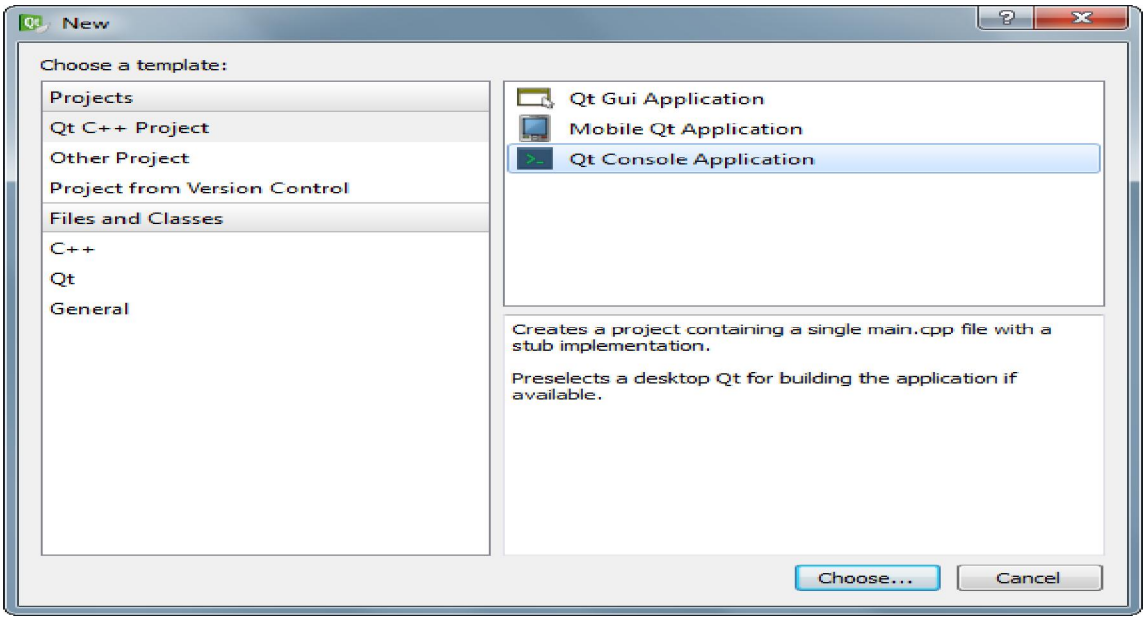
7. Setelah itu, program Qt Creator akan muncul dan kita dapat segera mencobanya.



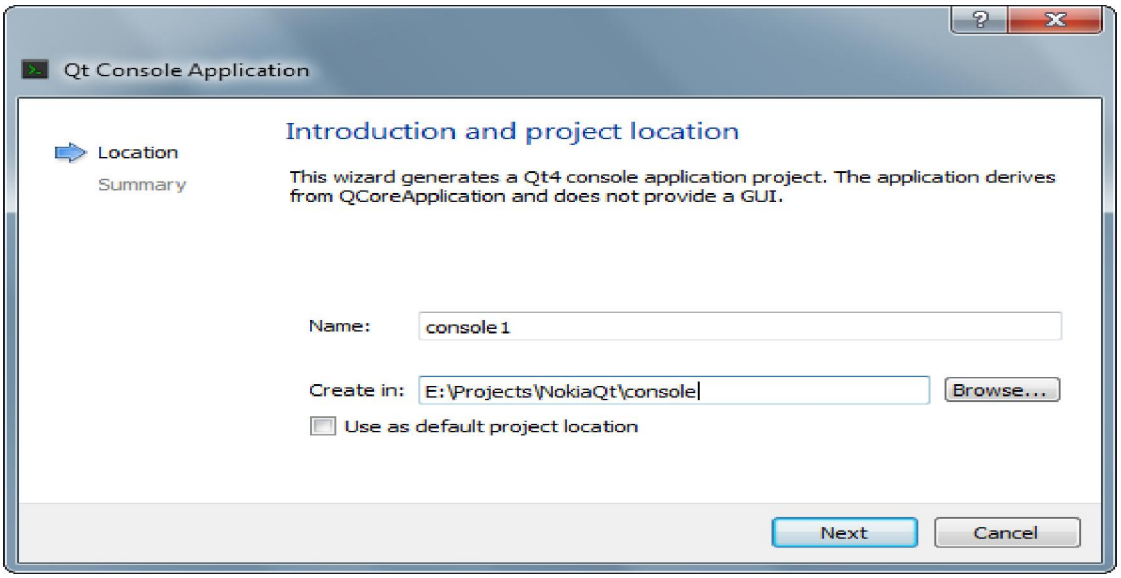
Program Console Pertama C++ dengan Qt Nokia SDK

Labs 2. Pembuatan Program Console Pertama

1. Untuk mencoba membuat program console pertama dengan Nokia Qt SDK, maka kita Buka menu File > New Project, dan pilih template project Qt C++ Project dan pilih Qt Console Application. Kemudian tekan tombol Choose...



2. Setelah itu kita isi Project Location dengan nama console1, dan isi juga direktori tempat kita menyimpan file-file project tersebut. Sangat disarankan bahwa kita membuat folder tersendiri untuk setiap project yang dibuat. Misalnya kita buat di E:\Projects\NokiaQt\Console dengan nama console1. Kemudian tekan Next.

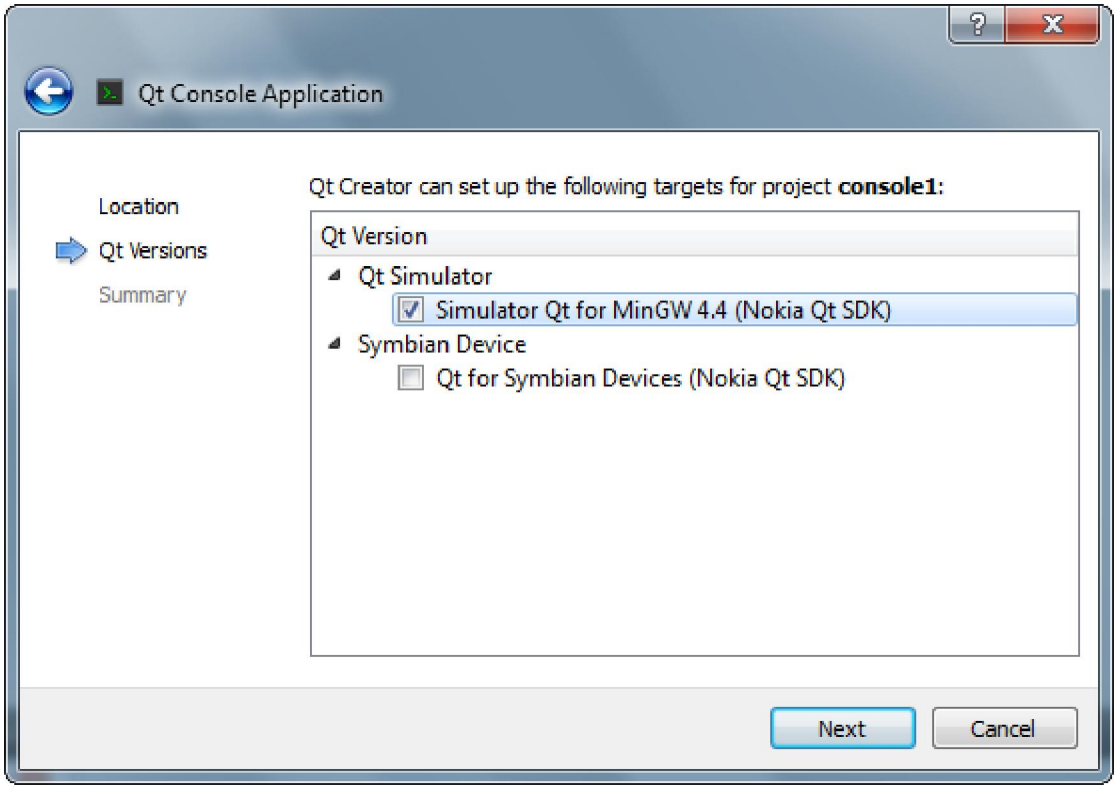


3. Kemudian kita akan memilih simulator yang akan digunakan untuk menjalankan program Qt Console yang akan dibuat. Pilihan Simulator ini sangat tergantung pada instalasi yang kita lakukan diawal, apakah kita menginstall semua Simulator atau tidak.

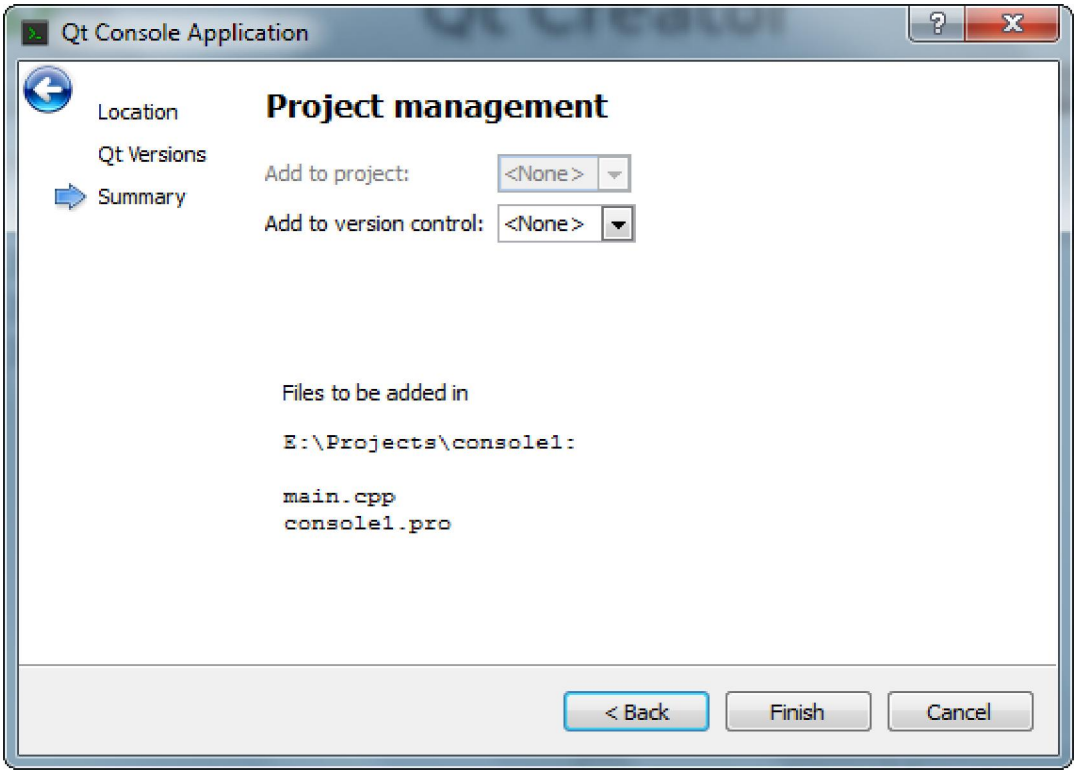
TIPS

Simulator yang lengkap terdiri dari:
Qt Simulator MingGW 4.4
Qt Simulator VS 2008
Qt Simulator pada Meemo

Kita juga dapat mengirimkan aplikasi yang kita buat itu ke sebuah perangkat mobile Symbian yang kita miliki.



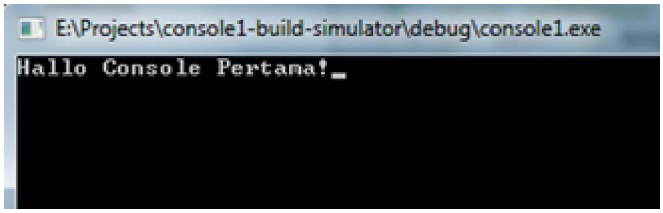
4. Setelah itu kita akan dihadapkan pada kotak dialog Project Management, dimana kita dapat memilih version control yang diinginkan. Namun untuk contoh pertama ini kita tidak perlu isi apapun, bagian ini kita biarkan seperti apa adanya saja. Kemudian tekan Finish.



5. Setelah semuanya sukses, maka selanjutnya adalah kita hanya perlu menulis kode program yang akan kita buat. Pada kertas source code main.cpp, kita tuliskan kode berikut ini, lalu jalankan dengan memilih menu Build > Run atau dengan shortcut CTRL + R.

```
1   #include <QtCore/QCoreApplication>
2   #include <iostream>
3
4   using namespace std;
5   int main(int argc, char *argv[])
6   {
7       QCoreApplication a(argc, argv);
8       cout<<"Hallo Console Pertama!";
9       return a.exec();
10  }
11
```

6. Setelah proses Build berhasil, maka akan muncul program aplikasi yang kita buat dalam bentuk Console application sebagai berikut:



Akhirnya aplikasi console pertama kita sudah jadi! Viola!

Analisa:

- Untuk menampilkan tulisan ke layar digunakan perintah **cout<<**
- Perintah cout<< harus menggunakan header khusus yaitu **iostream** sehingga perlu ditambahkan header baru pada bagian bawah **#include <QtCore/QtCoreApplication>**, yaitu dengan **#include <iostream>**
- Karena perintah cout termasuk dalam **namespace std**, maka tambahkan juga **bagian using namespace std;** pada bagian bawah **#include**
- **namespace std** berarti kumpulan fungsi-fungsi standard yang sudah disediakan oleh C++
- Bagian lainnya dibiarkan apa adanya saja dahulu.

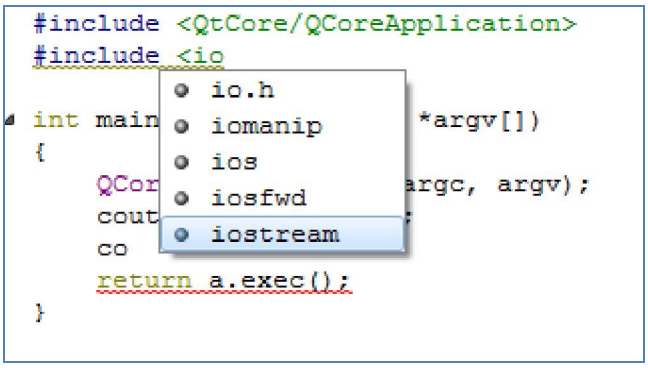
Fitur-fitur Nokia Qt SDK (Qt Creator)

An advanced C++ code editor

Code completion: menulis kode sedikit , QtCreator akan melengkapinya dengan cepat. Kita juga dapat menggunakan shortcut: Ctrl + Spasi untuk melakukan auto complete.

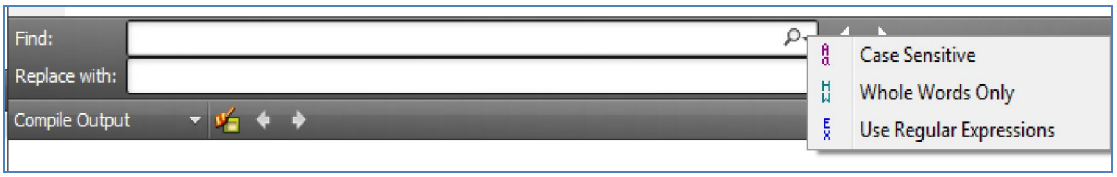
Labs 3. Percobaan Fitur Auto Complete

Buatlah sebuah project baru seperti pada langkah-langkah sebelumnya, beri nama yang berbeda untuk masing-masing project nya. Kemudian ketik **#include<**, maka otomatis akan muncul bantuan file header yang bisa digunakan untuk diikutsertakan pada program yang akan dibuat.



Tersedia tool find & replace

Tool find & replace dapat digunakan menggunakan menu CTRL + F, sehingga akan muncul kotak dialog seperti dibawah ini:

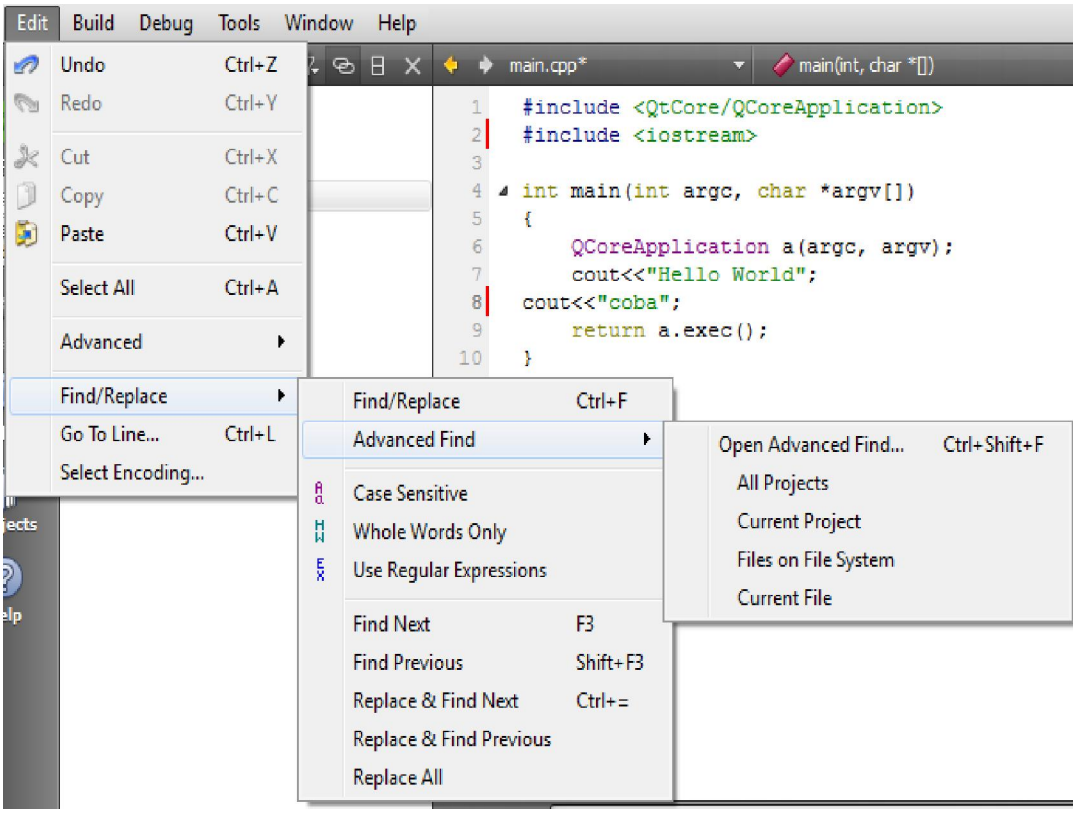


Kita dapat memilih model pencarian:

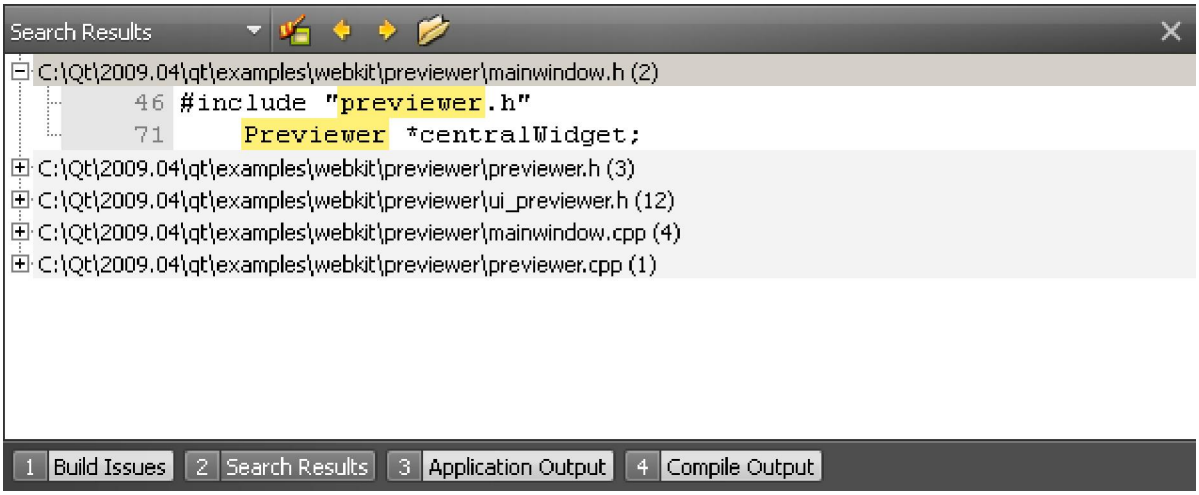
- **Case sensitive**: memperhatikan huruf besar dan kecil
- **Whole Words only**: mencari seluruh kata / kalimat sekaligus
- **Use Regular Expressions**: mencari berdasarkan pola regular expression yang khusus.

TIPS
Untuk mempelajari tentang Regular Expression, silahkan kunjungi http://www.regular-expressions.info/

Proses pencarian juga dapat dilakukan pada file tertentu saja yang terbuka, pada semua file pada project yang sedang aktif, pada folder tertentu pada sistem operasi yang terinstall, atau pada semua project yang sedang dibuka walaupun tidak aktif. Gambar berikut dapat menjelaskan tentang find and replace.



Contoh hasil Pencarian dapat dilihat pada gambar berikut:



Code Formatter dan Auto-Indent

QtCreator juga memungkinkan kita menulis kode dengan format yang baik dan standar, karena QtCreator akan otomatis melakukannya.

Labs 4. Percobaan Auto-indent

Ketiklah kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
cout<<"Hallo Dunia"<<endl;
cout<<"Percobaan bari kedua"<<endl;
    return a.exec();
}
```

Kemudian bloklah kedua bagian cout pada kode diatas, dan klik kanan mouse, pilih Auto-indent Selection seperti pada gambar berikut:

```
1   #include <QtCore/QCoreApplication>
2   #include <iostream>
3
4   int main(int argc, char *argv[])
5   {
6       QCoreApplication a(argc, argv);
7   cout<<"Hallo Dunia"<<endl;
8   cout<<"Percobaan bari
9       return a.exec();
10  }
11
```

Follow Symbol Under Cursor F2
Switch Between Method Declaration/Definition Shift+F2
Find Usages Ctrl+Shift+U
Rename Symbol Under Cursor Ctrl+Shift+R
Auto-indent Selection Ctrl+I
(Un)Comment Selection Ctrl+/
Add to Watch Window

maka baris kode tersebut akan tersusun secara rapi dan terindentasi dengan baik.

Display error and warning

QtCreator juga mampu menampilkan baris error dan warning seperti pada gambar berikut ini:

main.cpp* main(int, char *[])

```
1   #include <QtCore/QCoreApplication>
2   #include
3
4   int main(int argc, char *argv[])
5   {
6       QCoreApplication a(argc, argv);
7       cout<<"Hello World";
8       co
9       return a.exec();
10  }
11
```

expected token `;' got `return'

Build Issues

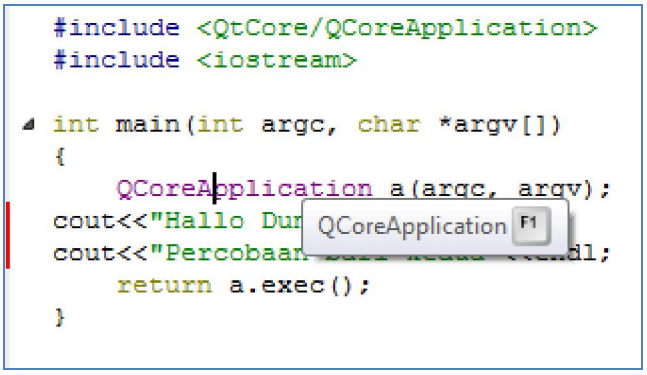
In member function 'void TextFinder::on_findButton_clicked()':	textfinder.cpp
'previouslyFound' was not declared in this scope	textfinder.cpp 57
'QMessageBox' has not been declared	textfinder.cpp 58
'QMessageBox' has not been declared	textfinder.cpp 61
'QMessageBox' has not been declared	textfinder.cpp 61
'QMessageBox' has not been declared	textfinder.cpp 61
'QMessageBox' has not been declared	textfinder.cpp 63
'previouslyFound' was not declared in this scope	textfinder.cpp 70

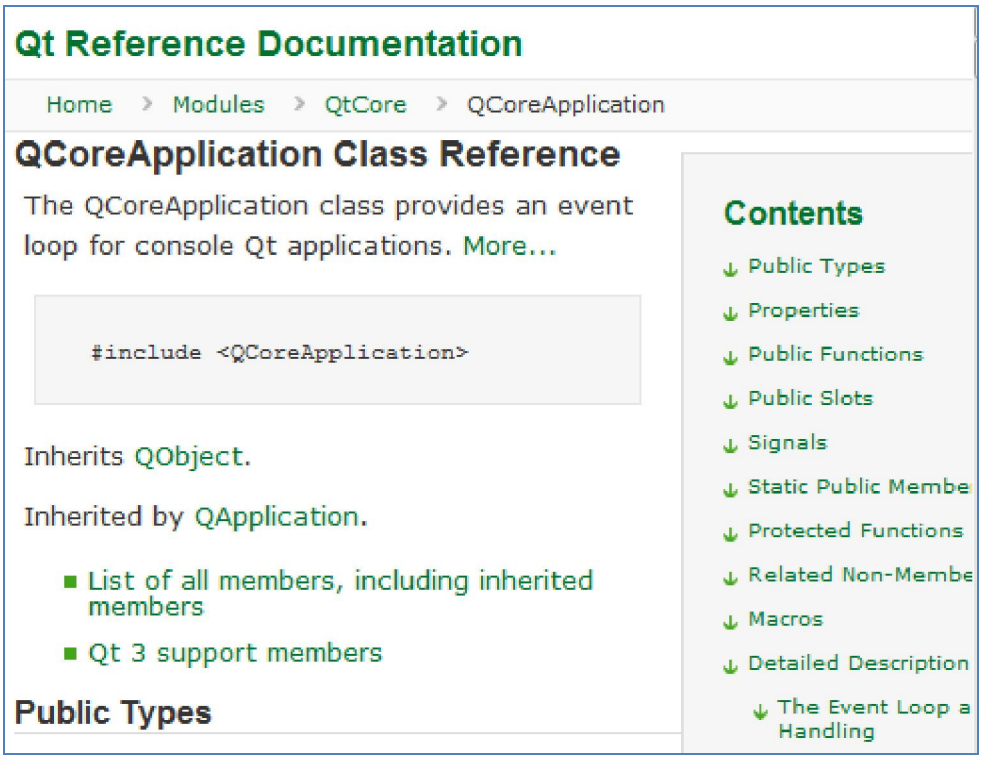
Selain itu QtCreator memungkinkan kita untuk menjelajahi semua class, fungsi dan simbil yang telah kita

buat atau kita gunakan. QtCretor juga menyediakan kita help yang cerdas berdasarkan konteks pada class, fungsi, dan symbol yang kita gunakan. Caranya arahnya kursor ditengah-tengah tulisan yang akan kita cari tahu helpnya, lalu tekan F1.

Labs 5. Percobaan Penggunaan Help F1

Dekatkan kursor pada kata QcoreApplication, maka akan muncul sebuah tips bertuliskan QcoreApplication F1. Saat itulah kita harus menekan tombol F1 sehingga akan muncul tampilan help dari class QcoreApplication seperti pada gambar berikut:





Fitur Integrated Help ini membuat pencarian keyword yang cepat, full text search, indexing dan bookmark pada hasil pencarian. QtCreator juga memiliki kemampuan indexing dan search pada koleksi-

koleksi dokumen help secara simultan. Dan terakhir, dokumentasi class-class dan aplikasi dapat disimpan secara offline maupun dicari secara online.

Beberapa fitur QtCreator lainnya adalah:

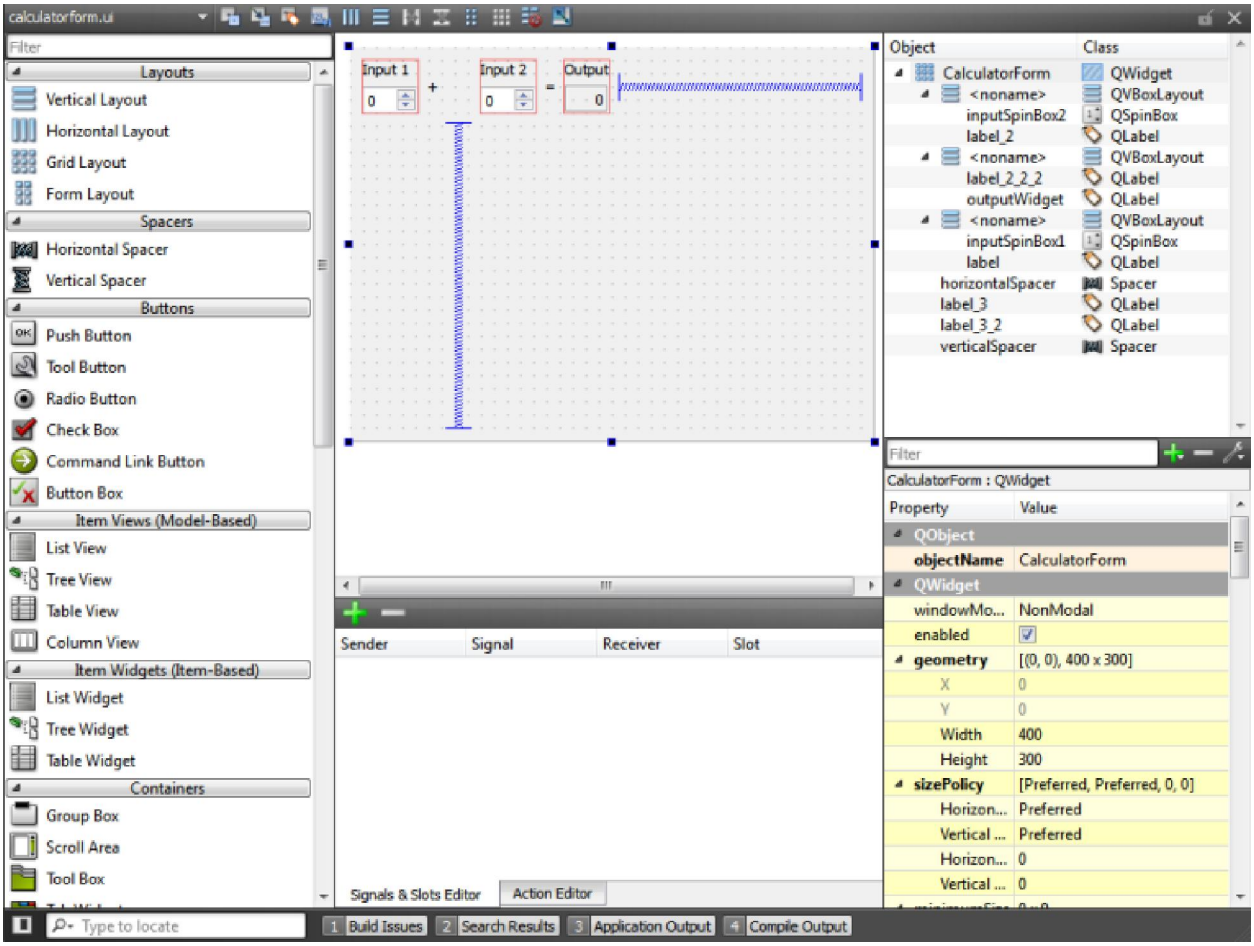
- **Project creator wizard.**

  Qt Creator mendukung pembuatan project dengan berbasis wizard sehingga memudahkan kita dapat membuat project secara terpisah dan terstruktur.

  - Seperti pada latihan 1 sebelumnya, maka kita sudah pernah membuat sebuah project pada QtCreator. Nah, QtCreator C++ project dapat berisi:
    - File-file yang dikelompokkan secara bersama
    - Proses Build yang terkustomisasi secara khusus untuk project tersebut
    - Form dan resource files yang diikutsertakan dalam project tersebut
    - Semua setting untuk menjalankan aplikasi dalam project tersebut

- **Integrated GUI designer.**

  Desain interface dapat dibuat dengan cepat dengan cepat dengan fasilitas drag and drop. GUI Designer dapat melakukan kustomisasi widget atau memilih widget standard yang ada dan dapat melakukan preview secara real time, dan hasil preview sama dengan yang didesain. Pada hasil output, kita dapat menentukan apakah akan dihasilkan kode C++ atau Java dari prototipe antarmuka yang dibuat.



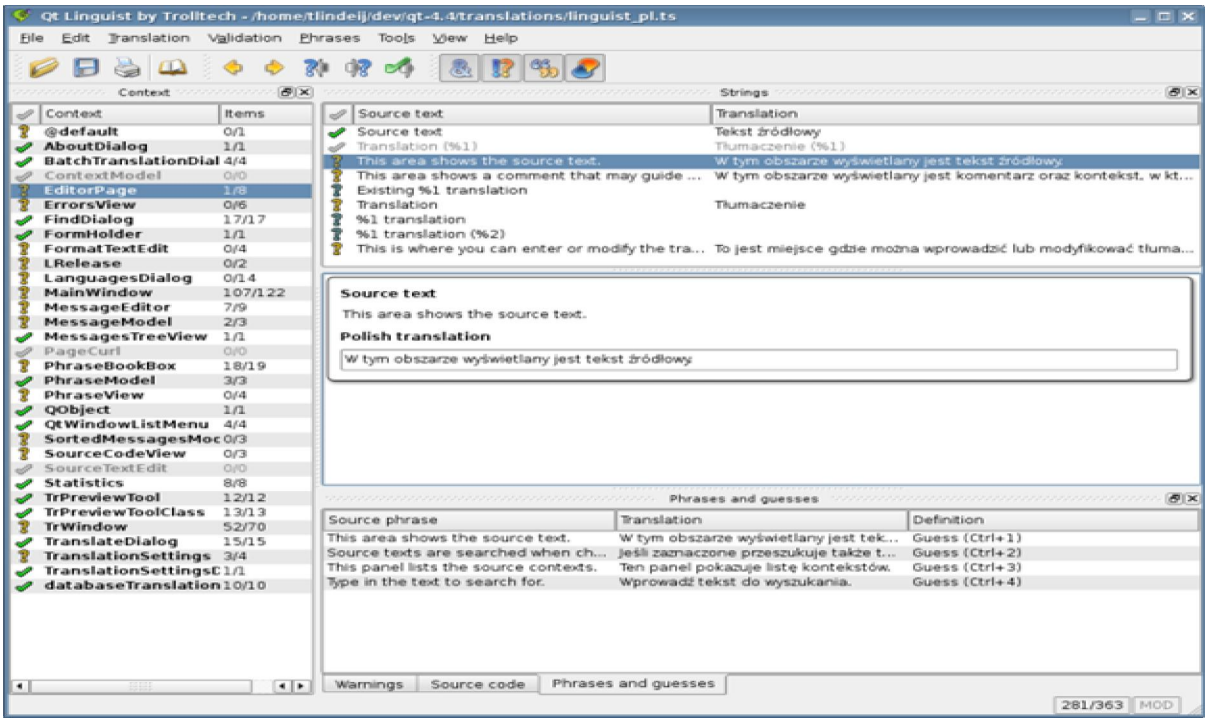
QtCreator juga dapat mengintegrasikan Qt Designer dengan Visual Studio atau Eclipse IDE. Beberapa fitur integrasi tersebut adalah:

- Menyediakan wizards untuk membuat Qt projects dan classes baru langsung pada VS dan Eclipse
- Dapat secara otomatis build setup untuk Qt Meta-Object Compiler, User Interface Compiler, dan Resource Compiler
- Dapat melakukan import dan export dari Qt Project and Project Include files
- Integrated Qt resource management pada VS dan Eclipse
- Integrated Qt documentation pada VS dan Eclipse
- Debugging extensions for Qt data types pada VS dan Eclipse

- **International Translation**

International Translation dilakukan dengan mengumpulkan dan menyajikan semua teks pada

User Interface untuk seorang penerjemah ke dalam sebuah aplikasi sederhana bernama **QtLinguist**. QtLinguist bersifat language and font-aware, cepat untuk menambahkan bahasa baru untuk aplikasi yang ada dengan alat penggabungan yang cerdas , mendukung karakter Unicode, dapat berpindah-pindah antara bahasa kanan-ke-kiri dan kiri-ke-kanan pada saat runtime, dan dapat mendukung campuran beberapa bahasa dalam satu dokumen aplikasi. Berikut adalah contoh tampilan dari QtLinguist yang dijalankan pada sistem operasi Linux.



- **Integrated Debugger**

  QtCreator mendukung beberapa debugger terintegrasi, yaitu:

  - GNU Symbolic Debugger (**gdb**)
  - Microsoft Console Debugger (**CDB**)
  - Internal Java Script debugger

  Contoh tampilan debugging adalah:

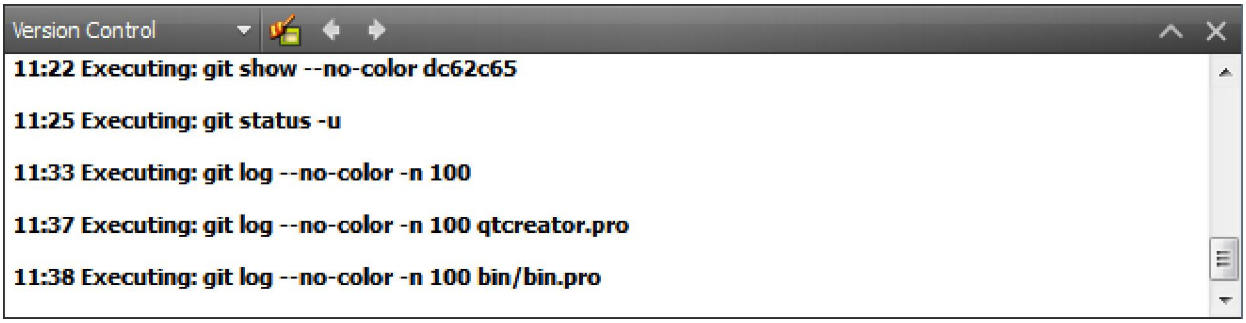
```
23          QTextStream in(&inputFile);
24          QString line = in.readAll();
25          inputFile.close();
26
27          ui.textEdit->setPlainText(line);
28      }
29
```

- **Version Control Systems**

  QtCreator dapat mendukung berbagai VCS yang terintegrasi demi keamanan source code yang telah kita buat. Beberapa VCS yang didukung adalah:

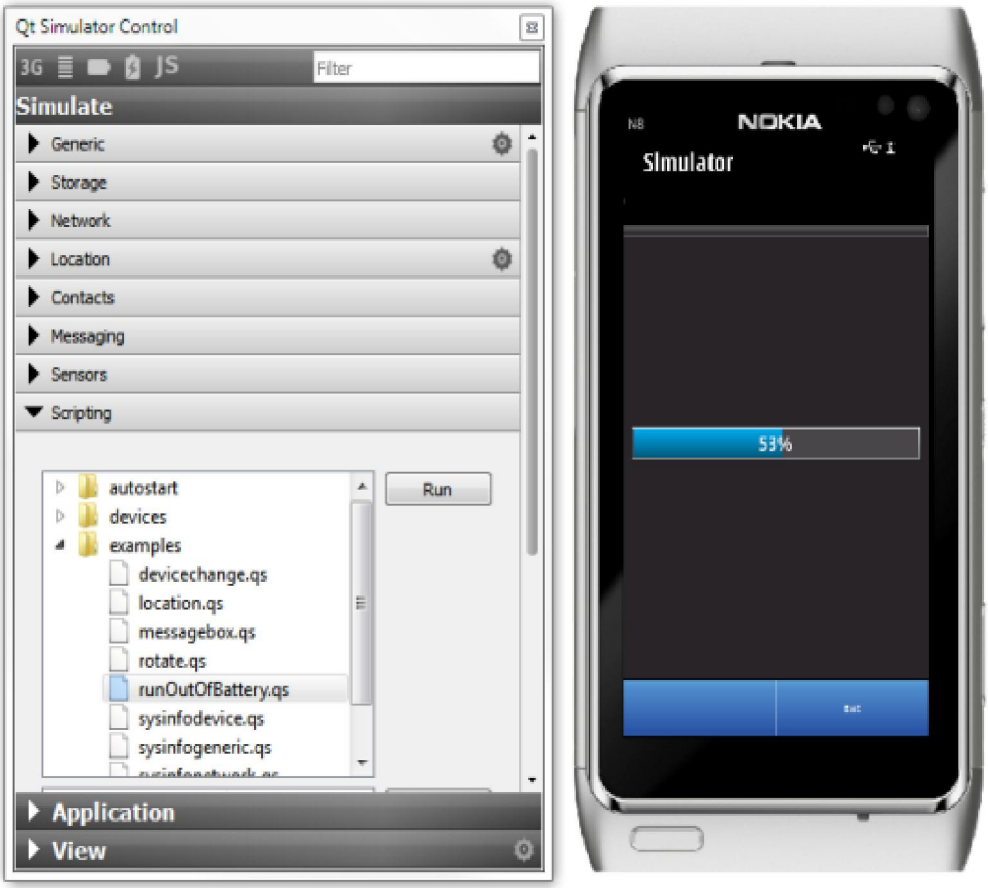
  - Git (http://git-scm.com/)
  - Subversion (http://subversion.apache.org/)
  - Perforce (http://www.perforce.com/)
  - CVS (http://savannah.nongnu.org/projects/cvs)
  - Mercurial (http://mercurial.selenic.com/)

  Contoh tampilan penggunaan git adalah:



- **Integrated Simulator**

  Sementara ini baru mendukung Nokia Qt Simulator.



- **Cross Platform Compiler**

  QtCreator menyederhanakan proses build untuk project pada platform yang berbeda dan dapat mengotomatiskan proses Makefile generation . Hal ini tentu dapat mempersingkat baris informasi yang diperlukan untuk menciptakan setiap Makefile. Dan terakhir, qmake yang digunakan juga dapat untuk menghasilkan proyek untuk Microsoft Visual studio tanpa memerlukan pengubahan pada file project yang telah dibuat.

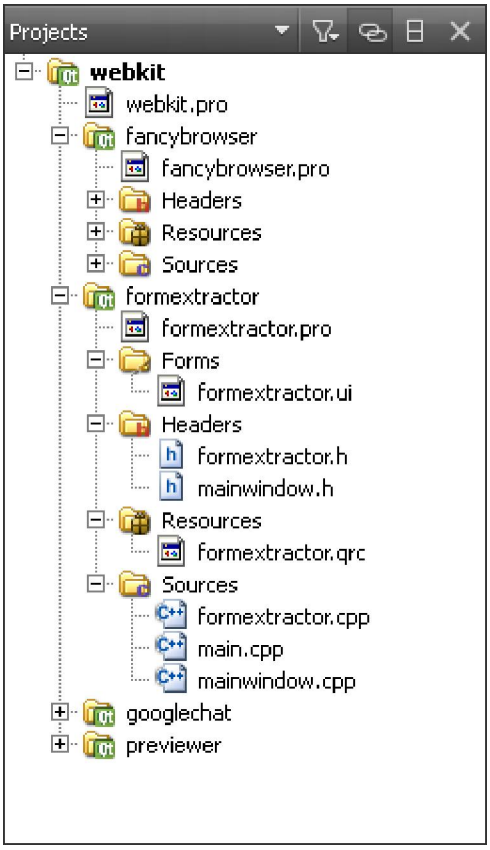
QtCreator Integrated Development Environment

Struktur IDE QtCreator dapat dilihat pada tampilan berikut ini:

Pada mode **Welcome** mode, maka kita dapat melakukan:

- Membuka tutorials dan example projects
- Melihat tips dan hints penggunaan Qt Creator
- Membuat dan membuka project
- Mengirim feedback ke development team
- Membuka recent sessions dan projects
- Membaca berita dari Qt labs
- Meminta bantuan / support

Bagian Project Contents dapat berisi semua file project yang telah dibuat seperti pada gambar berikut ini:

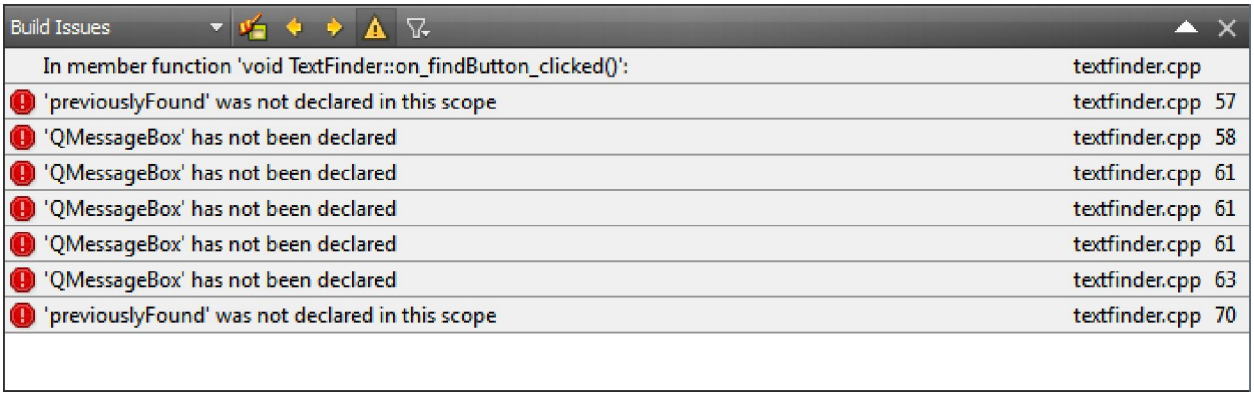


Kita dapat memilih content pada sidebar menu berikut:

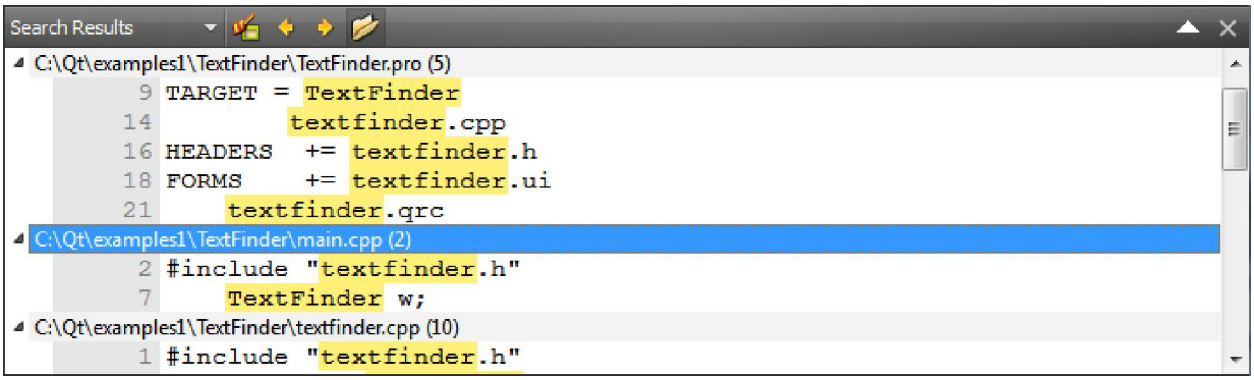
- **Projects** untuk menampilkan semua project pada sesi yang aktif
- **File System** menampilkan isi file pada sebuah direktori pada sistem operasi yang diinstall
- **Bookmarks** menampilkan semua bookmark yang ada pada sesi aktif
- **Open Documents** menampilkan file yang terbuka

Sedangkan pada bagian output pada bagian bawah editor kita dapat melihat output untuk:

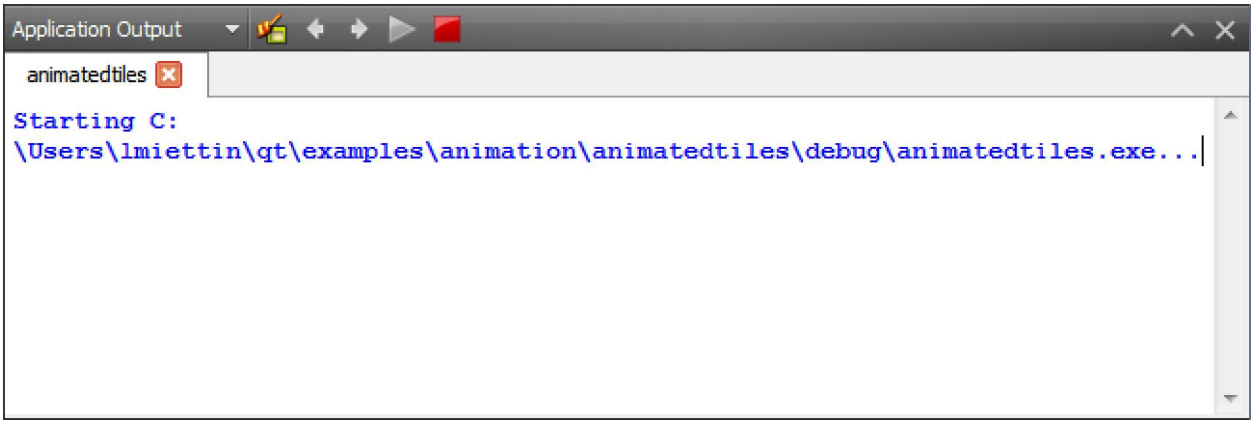
- **Build Issues**: menampilkan hasil dari proses build



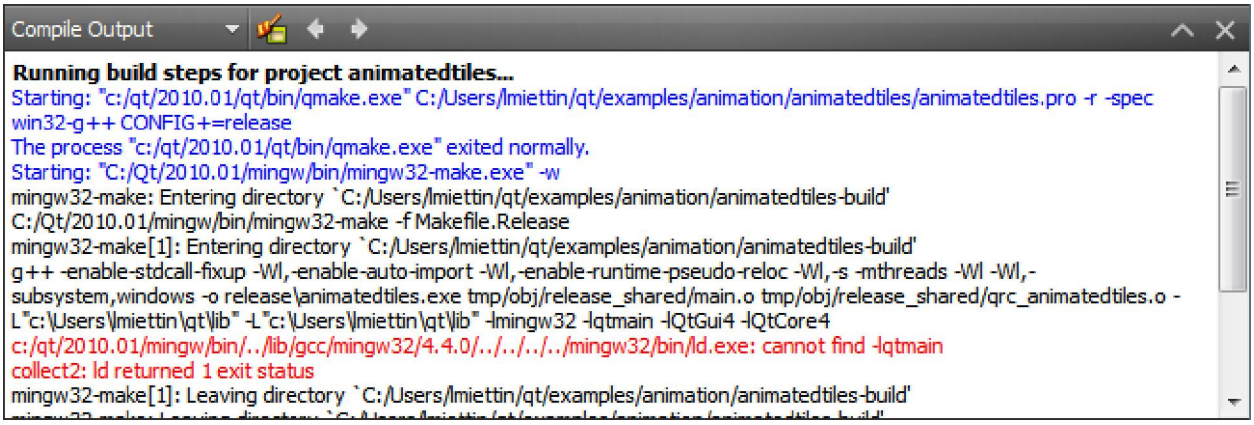
- **Search Results**: menampilkan hasil pencarian



- **Application Output**: menampilkan hasil dari aplikasi yang kita buat



- **Compile Output**: menampilkan hasil proses kompilasi



Struktur Program C++

Program Bahasa C/C++ tidak mengenal aturan penulisan di kolom/baris tertentu, jadi bisa dimulai dari kolom/baris manapun. Namun demikian, untuk mempermudah pembacaan program dan untuk keperluan dokumentasi, sebaiknya penulisan program di bahasa C/C++ diatur sedemikian rupa sehingga mudah dan enak dibaca.

Berikut adalah struktur dasar program yang dibuat dengan bahasa C++:

```
#include <header>
using namespace std;
int main(int argc, char *argv[])
{
  deklarasi variabel;
  deklarasi konstanta;
  perintah âperintah;
  //komentar
  return 0;
}
```

Penjelasan :

 # include

adalah salah satu pengarah preprocessor directive yang tersedia pada C++. Preprocessor selalu dijalankan terlebih dahulu pada saat proses kompilasi terjadi. Bentuk umumnya:

```
# include <nama_file>
```

Bagian tersebut tidak diakhiri dengan tanda semicolon, karena bentuk tersebut bukanlah suatu bentuk pernyataan, tetapi merupakan preprocessor directive. Baris tersebut menginstruksikan kepada kompiler untuk menyisipkan file lain dalam hal ini file yang berakhiran .h (file header) yaitu file yang berisi C++ standard library. Pada C++ ekstensi .h tidak dituliskan.

Beberap contoh pengikutsertaan berkas adalah:

- **#include <iostream>** : diperlukan pada program yang melibatkan objek **cout** dan **cin**
- **#include <conio>** : diperlukan bila melibatkan **clrscr()**, yaitu perintah untuk membersihkan layar dan fungsi **getch()** untuk menerima sembarang input keyboard dari user.
- **#include <iomanip>** : diperlukan bila melibatkan **setw()** yang bermanfaat untuk mengatur lebar dari suatu tampilan data.
- **#include <math>** : diperlukan pada program yang menggunakan operasi **sqrt()** yang bermanfaat untuk operasi matematika kuadrat.

using namespace std;

Semua elemen standard C + + library dinyatakan dalam apa yang disebut namespace, namespace tersebut bernama **std**. Jadi artinya untuk mengakses semua fungsionalitas std kita menuliskan bahwa kita menggunakan namespace std.

int main ()

Program C++ terdiri dari satu atau lebih fungsi, dan di antara salah satunya harus ada fungsi main dan hanya boleh ada satu main pada tiap program C++. Setiap program C++ akan dan pasti akan memulai eksekusi programnya pada fungsi main ini, meskipun main bukan fungsi yang pertama ditulis di program.

Melihat bentuk seperti itu dapat kita ambil kesimpulan bahwa batang tubuh program utama berada didalam fungsi main(). Berarti dalam setiap pembuatan program utama, maka dapat dipastikan seorang pemrogram menggunakan minimal sebuah fungsi.

Tanda { dan pada akhir program terdapat tanda }. Tanda { harus ada pada setiap awal dari sebuah fungsi dan tentu saja harus diakhiri dengan tanda }. Tanda ini digunakan untuk menunjukkan cakupan(scope) dari sebuah fungsi, dimana untuk menunjukkan fungsi ini dimulai dan berakhir.

Komentar

Komentar tidak pernah dicompile oleh compiler. Dalam C++ terdapat 2 jenis komentar, yaitu:
1 : `/* Komentar anda diletakkan di dalam ini bisa mengapit lebih dari satu baris */`
2 : `// Komentar anda diletakkan disini ( hanya bisa sebaris )`

Programmer sering sekali memasukkan komentar di dalam code agar program lebih mudah dibaca. Komentar juga membantu orang lain untuk membaca dan mengerti isi dari code. Komentar tidak menyebabkan komputer melakukan suatu instruksi ketika program dijalankan.

Tanda Semicolon (;)

Tanda semicolon **" ; "** digunakan untuk mengakhiri sebuah pernyataan. Setiap pernyataan harus diakhiri dengan sebuah tanda semicolon.

return 0

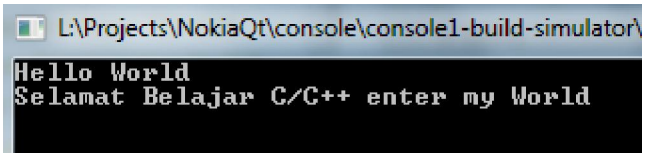
Pernyataan return menyebabkan fungsi utama untuk menyelesaikan kegiatannya lalu mengembalikan hasil dari fungsi utama. Kode kembalian biasanya angka 0 atau 1. Jika angka yang dikembalikan 0 berarti program berakhir dengan tidak ada error, sedangkan jika 1 maka program berakhir dengan error.

Labs 6. Struktur Program C++

Untuk lebih jelasnya silahkan coba ketik program berikut pada project baru.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    cout<<"Hello World"<<endl;
    cout<<"Selamat Belajar C/C++ ";
    cout<<"enter my World";
    return a.exec();
}
```

Kemudian jalankan dengan menekan tombol Run (CTRL + R)



Analisa

- Tampilan Hello World diakhiri dengan tanda enter baru kemudian dilanjutkan dengan tulisan berikutnya yaitu Selamat Belajar C/C++ enter my World. Artinya perintah endl merupakan perintah untuk memberi tanda enter.
- Sedangkan untuk tulisan Selamat Belajar C/C++ dan tulisan enter my World yang pada source code terpisah dengan perintah cout, pada tampilan hasil program tetap sama dan tidak ada enter diantaranya. Hal ini karena tidak ada perintah untuk menampilkan enter diantara kedua kalimat tersebut. Penulisan pada kode tidak akan mempengaruhi hasil output program.

TIPS

Selain cara penulisan diatas, kita juga dapat menuliskan kode C++ seperti berikut ini:

```
int main () { cout << " Hello World! "; cout << " I'm a C++ program "; return 0; }
```

Atau seperti berikut ini:

```
1 int main ()
2 {
3   cout <<
4     "Hello World!";
5   cout
6     << "I'm a C++ program";
7   return 0;
8 }
```

Namun hasilnya akan sama saja.

Hello World! I'm a C++ program

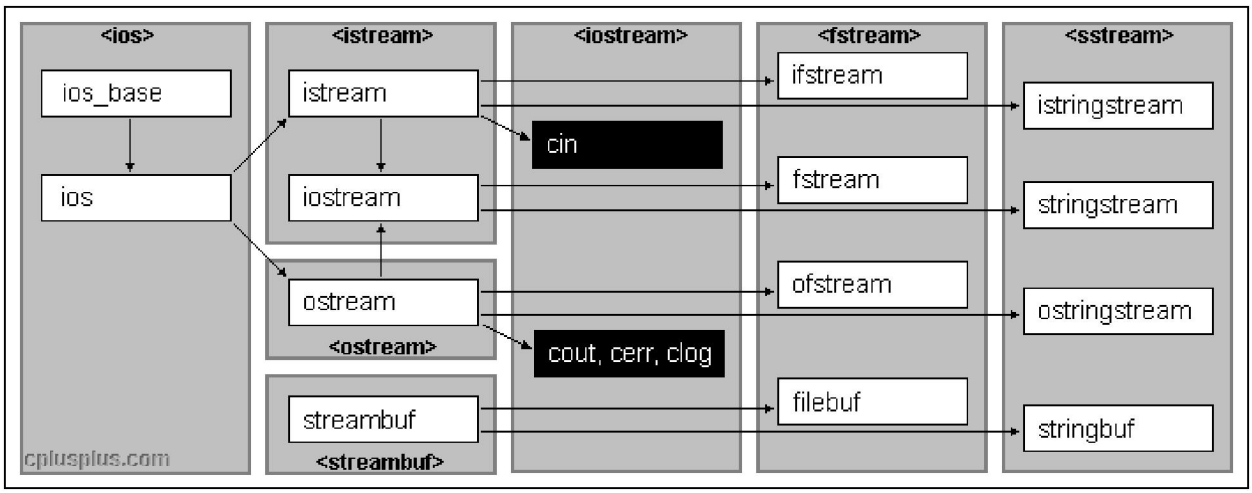
Jadi biasakanlah menuliskan kode program C++ dengan baik dan terformat rapi sehingga memudahkan pembacaan.

Struktur Input Output ke layar pada C++

Pada bahasa C++ untuk memproses input dan output digunakan standard library **<iostream>** yang harus diincludekan pada bagian preprocessor directive. Terdapat 4 jenis object stream, yaitu:

- **cin>>** untuk menerima input
- **cout<<** untuk mengeluarkan output biasa dilayar
- **clog<<** untuk mengeluarkan output ke dalam file log
- **cerr<<** untuk mengeluarkan output ke dalam stream error

Berikut adalah bagan dari **<iostream>**:



Struktur Output ke layar pada C++

Pada bahasa C++ cara untuk menampilkan output adalah dengan menggunakan standard output yang bernama stream **cout**.

cout digunakan dengan tanda hubung (conjunction) dengan *insertion operator*, yang ditulis dengan cara << (dua tanda "lebih kecil" berurutan). Contohnya adalah sebagai berikut:

```
1 cout << "Output sentence"; // prints Output sentence on screen
2 cout << 120;               // prints number 120 on screen
3 cout << x;                 // prints the content of x on screen
```

Kita harus membedakan antara sebuah kalimat/kata dengan sebuah indentifier atau dengan sebuah angka ketika kita akan menginputkan sesuatu ke layar. Pada contoh diatas, untuk mengouputkan kata/kalimat, maka kita harus menggunakan tanda " (petik ganda), sedangkan untuk mengouputkan angka 120, maka kita langsung menuliskan angka tersebut pada cout. Sedangkan pada bagian terakhir untuk menampilkan nilai suatu indentifier x, kita juga langsung meletakkannya setelah perintah cout.

```
1 cout << "Hello";  // prints Hello
2 cout << Hello;    // prints the content of Hello variable
```

Sedangkan pada contoh berikutnya diatas, kita lihat bahwa "Hello" dan Hello tanpa tanda kutip memiliki perbedaan. Hello dengan tanda kutip berarti mengoutpukan kata Hello kelayar, sedangkan Hello tanpa tanda kutip berarti mengoutputkan nilai indentifier Hello.

TIPS
Mengenai identifier akan dibahas di bab 2

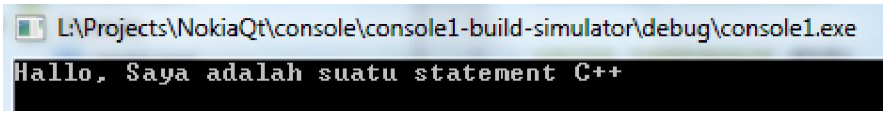
Labs 7.1 Penggunaan cout dan tanda <<

Buatlah project baru dengan QtCreator dan tuliskan kode berikut ini lalu jalankan!

```
#include <QtCore/QCoreApplication>
#include <iostream>
```

```
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    //pertama
    cout << "Hallo, " << "Saya adalah " << "suatu statement C++";
    return a.exec();
}
```

Hasil:



Analisa

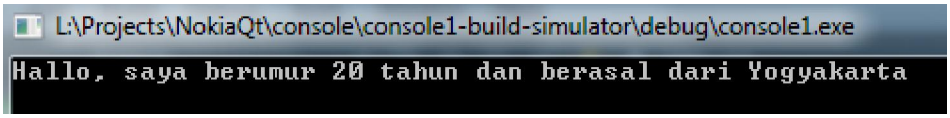
- Tanda konjungsi `<<` pada cout dapat digunakan untuk menggabungkan output ke layar, sehingga hasilnya berupa penggabungan antara "Hallo, " dan "Saya adalah ", dan "suatu statement C++"

Labs 7.2 Penggunaan cout dan tanda <<

Buatlah project baru dengan QtCreator dan tuliskan kode berikut ini lalu jalankan!

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    //kedua
    int umur = 20;
    string kota = "Yogyakarta";
    cout << "Hallo, saya berumur " << umur << " tahun dan berasal dari " << kota;
    return a.exec();
}
```

Hasil



Analisa

- Pada program diatas, terdapat dua buah identifier, yang bertama bernama **umur** dan yang kedua bernama **kota**. Kedua identifier tersebut berisi suatu nilai yang berjenis data tertentu. Pada contoh diatas, umur berisi nilai angka (numerik) bulat, sedangkan kota berisi nilai kalimat (yang biasa disebut string). String adalah kumpulan karakter-karakter yang digabungkan menjadi satu dan ditandai dengan tanda kutip ganda.
- **cout** dapat digunakan untuk menggabungkan antara kalimat yang dituliskan (string) dengan

identifier berjenis apapun (dalam hal ini numerik dan string).

TIPS
Tentang String akan dibahas lebih lanjut pada bab bab selanjutnya.

Labs 7.3 Penggunaan cout dan tanda <<

Buatlah project baru dengan QtCreator dan tuliskan kode berikut ini lalu jalankan!

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    //ketiga
    cout << "Ini kalimat.";
    cout << "Ini kalimat lainnya.";
    return a.exec();
}
```

Hasil:

Analisa

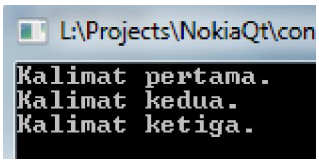
- Pada program ketiga, akan muncul dilayar berupa tulisan “Ini kalimat.Ini kalimat lainnya.”. Dan perhatikan bahwa diantara kedua kalimat tersebut tidak ada sama sekali batas enter. Mereka menyatu, walaupun pada program ditulis **cout** dua kali dan berada pada baris terpisah. Mengapa hal ini terjadi? Karena **cout** hanya digunakan untuk menampilkan output yang harus ditampilkan saja, dan di dalam kalimat yang ditampilkannya tidak ada perintah / karakter untuk enter.

Labs 7.4 Penggunaan cout dan tanda <<

Buatlah project baru dengan QtCreator dan tuliskan kode berikut ini lalu jalankan!

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    //keempat
    cout << "Kalimat pertama.\n";
    cout << "Kalimat kedua.\nKalimat ketiga.";
    return a.exec();
}
```

Hasil



Analisa

- Program diatas mirip dengan program sebelumnya namun terdapat tanda enter antara kalimat satu dengan dua dan tiga. Perhatikan bahwa untuk menampilkan enter, digunakan tanda "\n" (tanpa tanda kutip). Tanda enter **\n** merupakan escape character yang khusus merepresentasikan enter. Walaupun terlihat bahwa \n berarti dua karakter, namun seluruh **escape character** dianggap satu karakter saja.

TIPS

Berikut adalah daftar escape characters pada bahasa C++:

For this	Use this	Setting x to:	Printing x will yield:
'	\'	"Don\'t do that"	Don't do that
"	\"	"She said \"hi\""	She said "hi"
?	\?	"Who are you\?"	Who are you?
\	\\	"Backslash: \\"	Backslash: \
[newline]	\n	"1\n2"	1 2
[horizontal tab]	\t	"1\t2"	1 2
[backspace]	\b	"12\b3"	13
[16 bit unicode]	**u**xxxx	"Katakana a: \u30A1"	Katakana a: ア
[32 bit unicode]	**U**xxxxxxxx	"Katakana a: \U000030A1"	Katakana a: ア
[numeric escape sequence.]	**x**nn	"Printing the ESC character: \x1Bzyxwu"	"Printing the ESC character: yxwu"

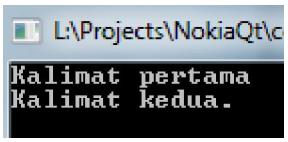
Labs 7.5 Penggunaan cout dan tanda <<

Buatlah project baru dengan QtCreator dan tuliskan kode berikut ini lalu jalankan!

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
```

```
    QCoreApplication a(argc, argv);
    //kelima
    cout << "Kalimat pertama" << endl;
    cout << "Kalimat kedua." << endl;
    return a.exec();
}
```

Hasil



Analisa

- Pada program diatas, antara kalimat pertama dan kedua juga terdapat enter, namun kali ini tanda enter tidak menggunakan **\n** melainkan **endl**. Tanda **endl** juga merupakan tanda enter yang artinya **end line**. Jadi selain menggunakan escape character \n, kita dapat menggunakan endl untuk tanda enter.

Output pada C++ juga dapat diatur secara manual agar sesuai dengan harapan kita. Untuk mengatur bentuk output, kita dapat include-kan <iomanip>. Library ini singkatan dari Input Output Manipulator yang dapat digunakan untuk memanipulasi tampilan (output) maupun inputan. Yang perlu dilakukan adalah dengan menambah header file pada preprocessor directive. Tambahkan:

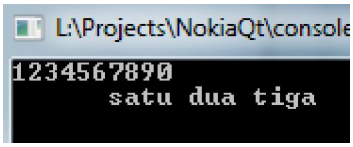
#include <iomanip>

Labs 8.1 Mengatur lebar suatu tampilan dengan setw() bagian 1

Buatlah project baru dan tuliskan kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
#include <iomanip>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    cout<<"1234567890"<<endl;
    cout<<setw(10)<<"satu"<<" dua"<<" tiga";
    return a.exec();
}
```

Hasil



Analisa

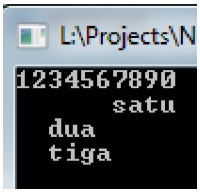
- Perhatikan bahwa tampilan baris pertama sengaja dituliskan angka 1-0 yang artinya terdapat 10 karakter. Kemudian pada baris kedua, dituliskan perintah **setw(10)** yang berarti mengatur lebar output pertama, yaitu "satu" sebanyak 10 karakter. Dan perhatikan bahwa tulisan "satu" persis berada tepat dibawah karakter 7890 (4 buah karakter). Berarti setw(10) berarti akan mengatur lebar output "satu" dalam 10 karakter, karena tulisan "satu" hanya berjumlah 4 karakter, maka 6 karakter sisanya dari depan, akan dituliskan dalam bentuk spasi kosong. Selanjutnya kita outputkan tulisan " dua" dan " tiga" yang persis seperti apa adanya. Hal ini karena perintah setw() hanya digunakan pada output setelah perintah itu saja, yaitu "satu". Jika kita ingin mengatur tampilan "dua" dan "tiga", maka berikan perintah setw() untuk masing-masing output. Ubahlah kode diatas menjadi seperti latihan berikutnya.

Labs 8.2 Mengatur lebar suatu tampilan dengan setw() bagian 2

Buatlah project baru dan tuliskan kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
#include <iomanip>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    cout<<"1234567890"<<endl;
    cout<<setw(10)<<"satu"<<endl;
    cout<<setw(5)<<"dua"<<endl;
    cout<<setw(6)<<"tiga"<<endl;
    return a.exec();
}
```

Hasil:



Analisa

- Perhatikan hasil program diatas, terlihat bahwa tulisan "satu" karena berjumlah 4 karakter terdapat 6 karakter spasi didepannya (karena kita setw(10)). Sedangkan tulisan "dua" karena berjumlah 3 karakter, maka terdapat 2 spasi didepannya (karena setw(5)). Terakhir tulisan "tiga" karena berjumlah 4 karakter dan kita setw(6), maka terdapat 2 spasi didepannya.

Labs 8.3 Mengatur lebar suatu tampilan dengan setw() bagian 3

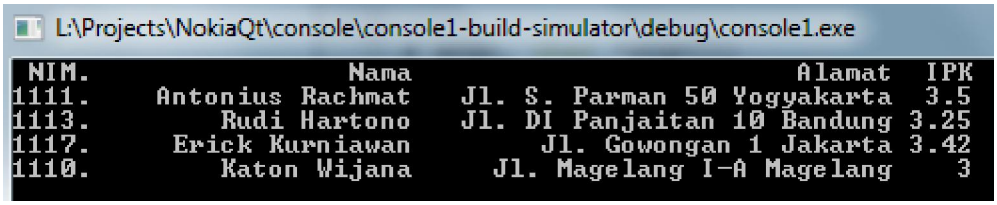
Buatlah project baru dan tuliskan kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
#include <iomanip>
```

```
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

cout<<setw(5)<<"NIM."<<setw(20)<<"Nama"<<setw(30)<<"Alamat"<<setw(5)<<"IPK"<<endl;
    cout<<setw(5)<<"1111."<<setw(20)<<"Antonius Rachmat"<<setw(30)<<"Jl. S. Parman 50
Yogyakarta"<<setw(5)<<"3.5"<<endl;
    cout<<setw(5)<<"1113."<<setw(20)<<"Rudi Hartono"<<setw(30)<<"Jl. DI Panjaitan 10
Bandung"<<setw(5)<<"3.25"<<endl;
    cout<<setw(5)<<"1117."<<setw(20)<<"Erick Kurniawan"<<setw(30)<<"Jl. Gowongan 1
Jakarta"<<setw(5)<<"3.42"<<endl;
    cout<<setw(5)<<"1110."<<setw(20)<<"Katon Wijana"<<setw(30)<<"Jl. Magelang I-A
Magelang"<<setw(5)<<"3"<<endl;
    return a.exec();
}
```

Hasil:



Analisa

- Pada contoh diatas, kita mencoba membuat tabel dengan mengeset setw() yang berbeda-beda sehingga pada outputnya tetap terlihat bahwa hasil yang ditampilkan tetap rapi dan rata kanan, karena sifat setw() selalu menambah spasi didepan kalimat yang hendak dioutputkan jika jumlah karakternya lebih kecil dari jumlah setw() yang diinputkan.

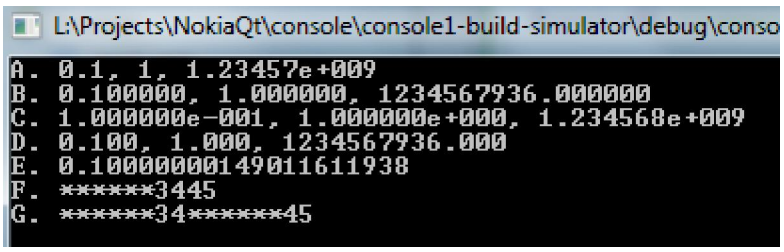
Labs 9. Penggunaan Berbagai Fungsi pada iomanip

Buatlah project baru dan tuliskan kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
#include <iomanip>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    const float satu = 0.1;
    const float dua   = 1.0;
    const float tiga   = 1234567890.0;
    cout << "A. "                    << satu << ", " << dua << ", " << tiga << endl;
    cout << "B. " << fixed           << satu << ", " << dua << ", " << tiga << endl;
    cout << "C. " << scientific      << satu << ", " << dua << ", " << tiga << endl;
    cout << "D. " << fixed << setprecision(3) << satu << ", " << dua << ", " << tiga
<< endl;
    cout << "E. " << setprecision(20) << satu << endl;
    cout << "F. " << setw(8) << setfill('*') << 34 << 45 << endl;
```

```
    cout << "G. " << setw(8) << 34 << setw(8) << 45 << endl;
    return a.exec();
}
```

Hasil:



Analisa:

- **const** digunakan untuk mendeklarasikan / menuliskan sebuah konstanta (suatu data yang sifatnya konstan dan tidak berubah).
- **setfill()** digunakan untuk mengisi spasi kosong dengan karakter sesuai dengan yang diinputkan oleh setfill. Pada contoh diatas digunakan karakter *, berarti karakter spasi kosong yang biasa digunakan pada setw() diganti dengan karakter *.
- **fixed**, **scientific** digunakan untuk mengatur tampilan data numerik.
- Sedangkan **setprecision** digunakan untuk mengatur presisi (tingkat ketelitian) berapa angka di belakang koma.

Standard Input pada C++

Untuk menerima input dari keyboard pengguna, pada C++ digunakan perintah obyek input stream, **cin>>**. Nama variabel bersifat bebas dan boleh berjenis data apapun juga, bisa numerik atau string (kalimat). Perintah cin menggunakan konjungsi / *extraction operator* bertanda **>>** setelah perintah cin.

Bentuk umum:

cin>>{nama variabel / string}

Contoh:

```
1 int age;
2 cin >> age;
```

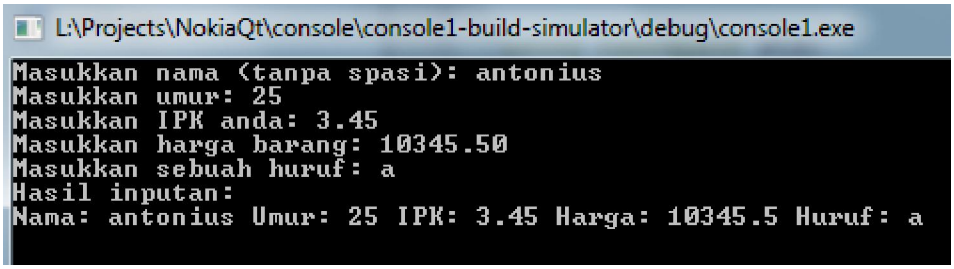
Pada contoh diatas, terdapat sebuah identifier bernama **age** dan bertipe **int** (numerik). Kemudian perintah cin digunakan untuk menampung suatu inputan dari keyboard yang berupa angka dan dimasukkan kedalam identifier bernama age tersebut.

Labs 10. Penggunaan cin

Buatlah project baru dan tuliskan kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    string nama;
    int umur;
    float ipk;
    double harga;
    char huruf;
    cout<<"Masukkan nama (tanpa spasi): ";cin>>nama;
    cout<<"Masukkan umur: ";cin>>umur;
    cout<<"Masukkan IPK anda: ";cin>>ipk;
    cout<<"Masukkan harga barang: ";cin>>harga;
    cout<<"Masukkan sebuah huruf: ";cin>>huruf;
    cout<<"Hasil inputan:"<<endl;
    cout<<"Nama: "<<nama<<" Umur: "<<umur<<" IPK: "<<ipk<<" Harga: "<<harga<<" Huruf:
"<<huruf;
    return a.exec();
}
```

Hasil



Analisa:

- Pada program diatas kita dapat menginputkan berbagai jenis data: data string (kalimat) pada identifier nama, data umur yang berupa numerik bulat, data IPK yang berupa numerik pecahan, data harga barang yang berupa data numerik dengan jumlah yang besar (sampai ribuan bahkan jutaan) dan bisa mengandung pecahan, dan terakhir sebuah data huruf yang berupa satu buah karakter saja.
- Semua jenis data tersebut dapat dimasukkan dari keyboard ke masing-masing identifier dengan menggunakan perintah **cin**. Setelah itu, program akan mengoutputkan semua data yang telah disimpan pada masing-masing identifier tersebut ke layar dengan perintah **cout** yang telah kita pelajari sebelumnya.

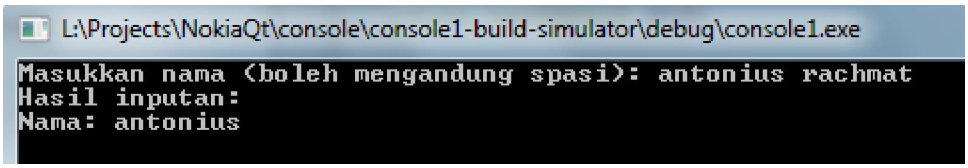
Labs 11. Penggunaan cin untuk kalimat yang mengandung spasi

Buatlah project baru dan tuliskan kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
```

```
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    string nama;
    cout<<"Masukkan nama (boleh mengandung spasi): ";cin>>nama;
    cout<<"Hasil inputan:"<<endl;
    cout<<"Nama: "<<nama<<endl;
    return a.exec();
}
```

Hasil:



Analisa

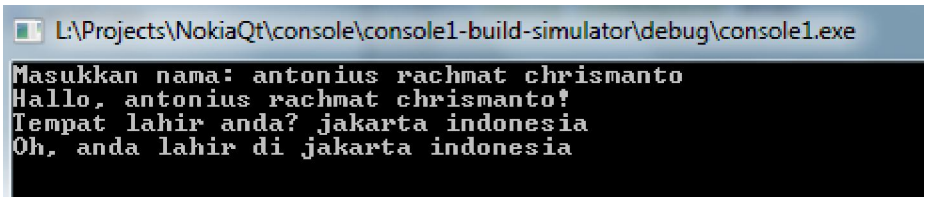
- Perhatikan program diatas, kita sudah menginputkan "antonius rachmat" dimana kalimat tersebut mengandung spasi diantaranya. Namun yang terjadi adalah bahwa cin hanya bisa menerima string sampai dengan spasi. Jadi string setelah spasi diabaikan. Hal ini terjadi karena sifat cin adalah menerima semua karakter sampai dengan terdapat sebuah karakter yang berjenis **blank space**. Blankspace character adalah karakter spasi, karakter enter, karakter backspace, dan karakter tab. Untuk menanggulanginya kita gunakan fungsi **getline** seperti pada contoh lab berikutnya.

Labs 12. Penggunaan cin untuk kalimat yang mengandung spasi

Buatlah project baru dan tuliskan kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    string nama;
    cout<<"Masukkan nama: ";
    getline(cin,nama);
    cout<<"Hallo, "<<nama<<"!"<<endl;
    cout<<"Tempat lahir anda? ";
    getline(cin,nama);
    cout<<"Oh, anda lahir di "<<nama;
    return a.exec();
}
```

Hasil:



Analisa

- Seperti yang sudah dijelaskan sebelumnya kita menggunakan function standard **getline**() yang menerima 2 buah parameter, yaitu stream input, dalam hal ini cin, dan sebuah identifier yang kita miliki, yaitu nama. Kita menggunakan identifier nama tersebut dua kali berturut-turut sehingga nilai dari nama yang pertama, yang berisi "antonius rachmat chrismanto" ditimpa dengan nilai dari nama yang kedua, sehingga hasil akhirnya, identifier nama berisi "jakarta indonesia".

Debugging pada QtCreator

Debugging adalah kegiatan menelusuri semua tingkah laku, isi data, alur kerja dari program yang dibuat ketika kondisi runtime. Kondisi runtime adalah kondisi ketika program sedang berjalan. Proses debugging bersifat realtime, dimana kita dapat dengan cepat mengetahui isi data dari identifier, isi memory, alamat memory, dan lain-lain. Harapan dari kegiatan debugging adalah kita dapat menemukan kesalahan program yang kita buat jika terdapat kesalahan / error.

Qt Creator tidak memiliki debugger. Lalu bagaimana caranya melakukan debugging? Caranya adalah dengan menggunakan plugin debugger yang bertindak sebagai antarmuka antara core Qt Creator dan debugger eksternal seperti GNU Debugger Simbolik (gdb), Microsoft Console Debugger (CDB), dan Javascript debugger.

Dalam mode debug kitadapat berinteraksi dengan debugger dalam beberapa cara, yaitu:

- Menuju ke baris program atau instruksi tertentu
- Menginterupsi jalannya program.
- Set breakpoint.
- Memeriksa isi stack pada memory.
- Memeriksa dan memodifikasi register dan isi memori pada saat debugging.
- Memeriksa dan memodifikasi register dan isi memori variabel lokal dan global.
- Memeriksa daftar library bersama yang dibuat.
- Membuat snapshot dari keadaan saat ini ketika program didebug.

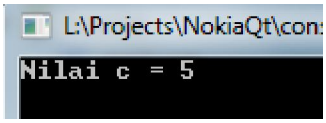
Labs 13. Debugging Simple Program

Buatlah project baru dan tuliskan kode berikut ini:

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
```

```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    int aa = 10;
    int b = 2;
    int c = aa/b;
    cout<<"Nilai c = "<<c;
    return a.exec();
}
```

Hasil:

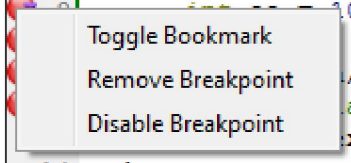


Langkah-langkah debugging:

1. Alihkan mode dari mode edit ke mode debug, dengan cara mengklik icon Debug pada sisi kiri Editor QtCreator.

2. Kemudian beri tanda set breakpoint pada kode program yang telah kita tuliskan sebelumnya dengan cara mengklik kanan baris yang akan diset breakpointnya seperti berikut ini:

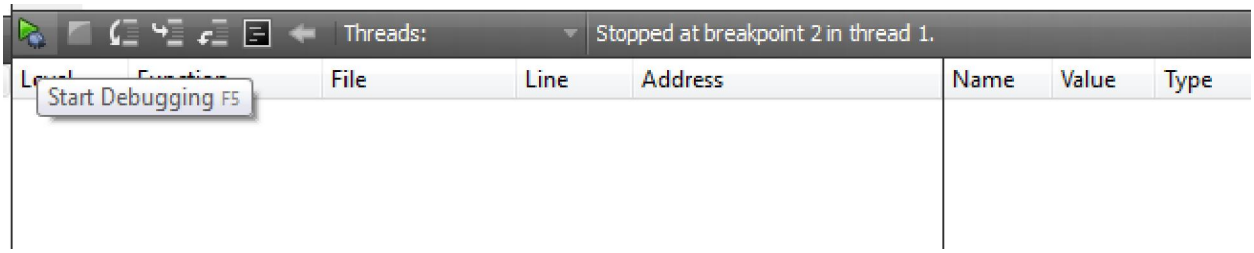
```
1   #include <QtCore/QCoreApplication>
2   #include <iostream>
3
4   using namespace std;
5
6   int main(int argc, char *argv[])
7   {
8       QCoreApplication a(argc, argv);
9       int aa = 10;
10      int b = 2;
11      int c = aa/b;
12      cout<<"Nilai c = "<<c;
13      return a.exec();
14  }
15
```



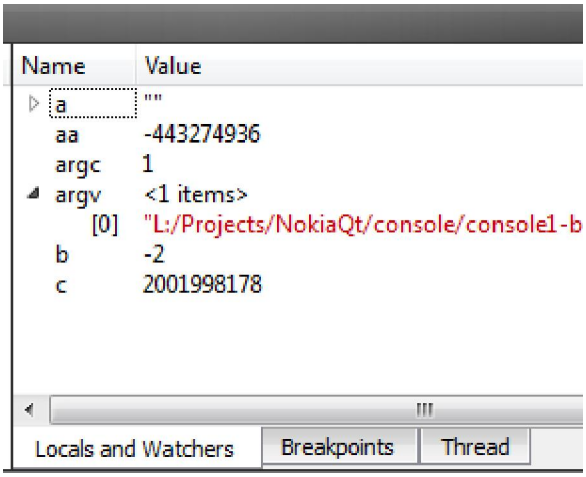
Sehingga menjadi sebagai berikut:

```
1   #include <QtCore/QCoreApplication>
2   #include <iostream>
3
4   using namespace std;
5
6   int main(int argc, char *argv[])
7   {
8       QCoreApplication a(argc, argv);
9       int aa = 10;
10      int b = 2;
11      int c = aa/b;
12      cout<<"Nilai c = "<<c;
13      return a.exec();
14  }
15
```

3. Kemudian klik icon start debugging pada bagian bagian bawah editor seperti berikut:

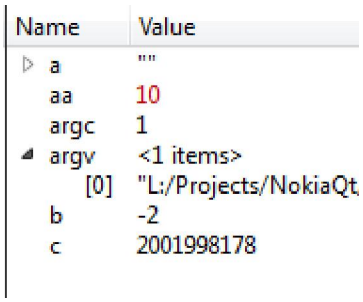


Atau tekan F5, sehingga akan muncul semua nilai-nilai aa, b, dan c.

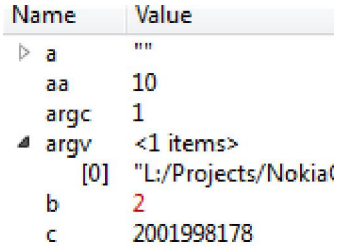


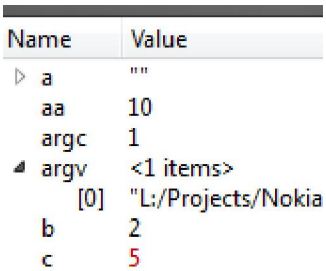
Pada tahap pertama debugging, ketika breakpoint mencapai breakpoint pertama, terlihat bahwa nilai identifier aa adalah -443274936, nilai b adalah -2, dan nilai c 2001998178. Tahap awal ini sering disebut dengan tahap inisialisasi. Tahap inisialisasi adalah tahap awal ketika kompiler bahasa C++ mengisi nilai untuk masing-masing identifier yang dideklarasikan. Nilai inisialisasi ini bersifat acak (random) namun sesuai dengan jenis datanya, yaitu numerik bulat (int).

4. Kemudian lanjutkan debugging dengan mengklik continue.



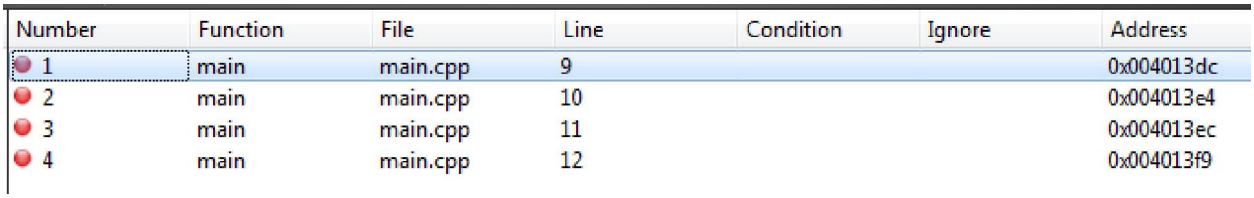
Terlihat bahwa aa bernilai 10 ketika breakpoint sudah mencapai breakpoint kedua. Kemudian dilanjutkan pada breakpoint ketiga dan keempat sebagai berikut:





Nilai b adalah 2 dan nilai c adalah 5 yang berasal dari 10 dibagi 2.

5. Selain untuk melihat nilai identifier, kita juga dapat melihat alamat memory masing-masing identifier yang sudah dideklarasikan dengan cara mengklik tab breakpoints sebagai berikut:



Number	Function	File	Line	Condition	Ignore	Address
1	main	main.cpp	9			0x004013dc
2	main	main.cpp	10			0x004013e4
3	main	main.cpp	11			0x004013ec
4	main	main.cpp	12			0x004013f9

Terlihat bahwa address masing-masing identifier berjarak 8 byte. Cara menghitungnya adalah dengan mengurangkan 0x00413e4 – 0x00413dc = 8.