Lingkungan Pengembangan

Gunakan Toolchain GCC 4.8 atau yang lebih baru. Jika belum punya, silakan download dari repositori OS masing-masing.

Untuk Windows, terdapat beberapa alternatif:

TDM GCC: http://tdm-gcc.tdragon.net/

MinGW: http://mingw-w64.org/doku.php/download

Pelajari cara melakukan *linking* terhadap library pada GCC secara umum. Pengetahuan mengenai linking pada GCC diperlukan untuk menggunakan berbagai macam *library*. Petunjuk sederhana mengenai linking dapat dilihat pada halaman https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_C_libraries.html.

CppCheck

Static analysis tool untuk C/C++. CppCheck bukan digunakan untuk memeriksa error pada sintaks, melainkan untuk memeriksa bug yang umum terjadi, misalnya akses *out of bounds* dan lain-lain.

https://github.com/danmar/cppcheck

http://cppcheck.sourceforge.net/manual.pdf

Build source code dengan menggunakan make. Setelah selesai melakukan *build source code* CppCheck, periksa dengan menjalankan

```
cppcheck .
```

Hasilnya adalah sebagai berikut:

```
Checking hello.cpp ...
1/3 files checked 33% done
Checking hello_test.cpp ...
2/3 files checked 66% done
Checking main.cpp ...
3/3 files checked 100% done
```

CMake

Kakas untuk membantu proses *build.* Pada tugas besar ini, CMake hanya diperlukan untuk melakukan *build* Google Test. CMake dapat pula diunduh dari repositori OS masing-masing.

https://cmake.org/

Perintah sederhana untuk menggunakan CMake adalah sebagai berikut:

```
Mkdir build
cd build
cmake ..
make
sudo make install
```

Perintah tersebut akan membuat direktori baru untuk menyimpan hasil generate dari CMake, dan melakukan build dengan makefile yang sudah digenerate pada hasil tersebut.

Google Test

Unit test framework untuk C++. Google Test dapat di-*build* dari *source* dengan bantuan *CMake*. Anda dipersilakan untuk melakukan *build framework* tersebut sebagai *shared library* atau *static library.*

https://github.com/google/googletest

Setelah melakukan build Google Test, compile contoh kelas yang sudah disediakan dengan perintah

```
g++ hello.cpp hello_test.cpp -o main -pthread -lgtest_main -lgtest
```

Kemudian jalankan executable **main**. *Output* akan serupa dengan contoh di bawah.

```
Running main() from gtest_main.cc
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from HelloTest
[ RUN      ] HelloTest.GetTextMethod
hello_test.cpp:11: Failure
      Expected: "Hello World"
To be equal to: hello.GetText()
      Which is: "Hello World!"
[  FAILED  ] HelloTest.GetTextMethod (1 ms)
[----------] 1 test from HelloTest (1 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (1 ms total)
[  PASSED  ] 0 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] HelloTest.GetTextMethod
```

Doxygen

http://www.stack.nl/~dimitri/doxygen/download.html

Doxygen juga bisa didapat dari repository OS masing-masing.

Contoh penggunaan doxygen dapat dilihat pada contoh file yang diberikan, **kamus.h** di Minggu ke 5 pada Olympia, dan pada tautan di atas.

Periksa instalasi Doxygen dengan membangkitkan dokumentasi untuk contoh file yang diberikan. Hasilnya akan terletak pada folder **docs/html**.

Perintah sederhana untuk membangkitkan dokumentasi:

```
doxygen Doxyfile
```

Dengan **Doxyfile** merupakan *file* konfigurasi doxygen.