

W3101.002 Intro to C++

Joshua Reich

`reich@cs.columbia.edu`

Department of Computer Science
Columbia University

Session 4



Review

- ▶ Functions
- ▶ Dealing w/ Errors
- ▶ OOP: structs and classes

Intro to Template Functions

- ▶ Problem: Let's say you don't know what type your function will be acting on.
Or maybe you need to do essentially the same thing with several types but don't want to write a separate function for each.
- ▶ Solution: Use a template!

CODE: `template.cc`

TEXT: *Accelerated C++* chapter 8

Automatic Class Generation

Template Classes allows us to autogenerate a new class based on input to the template!

- ▶ how to declare a template class
- ▶ using our new class
- ▶ why this is so powerful

CODE: `template_class.cc`

The Basics of Separate Compilation

- ▶ what is separate compilation? (robot juggling analogy)
- ▶ headers
- ▶ mechanics
- ▶ object (.o) files
- ▶ header guards: `#ifndef GUARD_file_name_h`

CODE: `make_example`

TEXT: *Accelerated C++* section 4.3

make: a useful tool

Now that we have several different files, wouldn't it be nice to have something to automate the process of compiling them?

- ▶ `makefile`
- ▶ `make`
- ▶ `targets`
- ▶ `dependencies`
- ▶ `procedure`
- ▶ `special targets - clean`

WEB:

<http://www.eng.hawaii.edu/Tutor/Make/index.html>

CODE: `make_example/makefile`



NDEBUG

No includes necessary for any preprocessor facilities!

- ▶ `ifndef NDEBUG ... endif`
- ▶ **use flag** `g++ -D NDEBUG filename.cc`

CODE: `ndebug.cc`

Preprocessor Variables & Macros

▶ Preprocessor Variables

- ▶ preprocessor variables are replaced before compilation by the compiler based on its own internal information
- ▶ `__FILE__`
- ▶ `__LINE__`
- ▶ `__TIME__`
- ▶ `__DATE__`

▶ Macros

- ▶ macros are entire statements that are replaced before compilation!
- ▶ you can give macros input but you don't need to prototype it since symbols are just replaced!
- ▶ macros can be very useful, but use them sparingly.



Preprocessor Variables & Macros: Resources

CODE: `preprocessor.cc`

CODE: `macro.cc`

CODE: `combination.cc`

WEB:

[http://stlplus.sourceforge.net/stlplus/docs/
debug_hpp.html](http://stlplus.sourceforge.net/stlplus/docs/debug_hpp.html)

assert

`assert` is a special preprocessor macro that only compiles when `NDEBUG` is off. If the condition in `assert` is false then the program will stop immediately and inform the user. This is a great safety feature to use!

TEXT: *C++ PRIMER* p.221-222

Debugging a multi-file project

- ▶ compile program using `-g` flag
- ▶ to run in UNIX type `gdb progname`
- ▶ to run in emacs type `M-x gdb`
- ▶ breakpoint (b)
- ▶ run (r)
- ▶ step (s)
- ▶ until (u)
- ▶ print (p)
- ▶ set
- ▶ backtrace (bt)
- ▶ continue (c)
- ▶ next (n)



Debugging a multi-file project: Resources

- ▶ CODE: `gdb_test/`
- ▶ WEB: <http://www.gnu.org/software/gdb/gdb.html>
- ▶ WEB:
<http://www.eecs.tufts.edu/comp/15/devref/compiling.html>
- ▶ TUTORIAL: <http://www.cs.cmu.edu/~gilpin/tutorial/>
- ▶ TUTORIAL:
<http://www.cs.princeton.edu/~benjasik/gdb/gdbtut.html>