

W3101.002 Intro to C++

Joshua Reich

`reich@cs.columbia.edu`

Department of Computer Science
Columbia University

Session 3



Review

- ▶ Variables
- ▶ Datatypes
- ▶ Operators
- ▶ Containers
- ▶ Control Flow
- ▶ Random Numbers

What is a Function?



A *function* takes input and produces output (you could call this a functional definition ;-).

Some examples:

- ▶ `sqrt(9)`
- ▶ `name.size()`
- ▶ `sort(list)`
- ▶ `eat(veggies)`

Function Basics

- ▶ function components
 - ▶ parameter list (the input)
 - ▶ return value (the output) - any type of `void`
 - ▶ body (how the function turn input into output)
- ▶ simple parameters and/or returns
- ▶ example redux: exponentiation
- ▶ complicated parameters and/or returns - (e.g. `cin`)
- ▶ using references: `&`
- ▶ lvalues vs. rvalues
- ▶ speeding things up with `inline`

CODE: `exponentiation-redux.cc`, `vector`  `int.cc`  COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Functions as Input

- ▶ a function can be an input parameter in C++
- ▶ the parameter type will look confusing
- ▶ errors will look very strange
- ▶ setting default input - works with any parameter type
- ▶ we will come back and look at the example code after learning about structs.

CODE: `function_as_parameter.cc`

Basic Error Handling

- ▶ check logical conditions and use return values
 - ▶ integer codes
 - ▶ object states (e.g. `istream&`)
- ▶ use language level-facilities
 - ▶ `const`
 - ▶ `const_iterator`
 - ▶ cast positive variables to unsigned
- ▶ throw exceptions: `<stdexcept>`
- ▶ which option to use?

CODE: `const_parameter.cc`, `vectorPrint2.cc`,
`error_codes.cc`



Doing Things with Exceptions

- ▶ `try`
- ▶ `catch`
- ▶ `finally` doesn't exist (for you Java programmers)
- ▶ beware Visual Studio for it offers non-standard options
- ▶ avoid side effects when using exceptions
- ▶ `what ()`

CODE: `using_exceptions.cc`

TEXT: *C++ Primer* section 6.13

Motivation

- ▶ Consider sorting first name / last name pairs:
Any suggestions how we might do this with what we already know?
- ▶ But what if we needed to deal with the combination of several data types? (e.g. first name, last name, age, list of favorite foods, etc)

Motivation

- ▶ Consider sorting first name / last name pairs:
Any suggestions how we might do this with what we already know?
- ▶ But what if we needed to deal with the combination of several data types? (e.g. first name, last name, age, list of favorite foods, etc)

Structs and the OO Approach

- ▶ `struct`
- ▶ accessing data members with dot operator
- ▶ compare functions implement < comparison (p.64 in text)
- ▶ using `sort` on structs
- ▶ accessing object members through `->`

CODE: `struct.cc`, `sorting_struct.cc`

TEXT: *Accelerated C++* section 4.2

Classes: Expanding the OO Approach

- ▶ Classes vs. Struct
- ▶ Data protection - `private`, `public`
- ▶ Accessor functions
- ▶ Constructors
- ▶ Overloading operators and printing objects

CODE: `class.cc`, `constructor.cc`,
`overload_operators.cc`

TEXT: *Accelerated C++* chapter 9