# CS3520
# Programming in C++
# Strings and Statements

Ken Baclawski
Fall 2016

# Outline

- Review of Syllabus
- Example Program
  - Includes
  - Documentation
  - The main function
  - Variables
  - Standard I/O streams
- Coding style requirements
- Compiling and running programs
- Assignment #1
- Project ideas


© 2016 Ken Baclawski All Rights Reserved

# Review of Syllabus


© 2016 Ken Baclawski All Rights Reserved

# Office Hours

- Professor Office Hours
  - Tuesday 15:30-16:30
  - Friday 12:00-13:00
  - or by appointment
  - 342 WVH
- Starting 9 September 2016
- Ending 6 December 2016.
- TA Office Hours
  - Thursdays from 15:00 to 16:00
  - 462 WVH.
- No office hours or classes 11 and 25 November 2016.


© 2016 Ken Baclawski All Rights Reserved

# Textbook

**Accelerated C++ Practical Programming by Example**

by Andrew Koenig and Barbara E. Moo

Addison Wesley, 2000

ISBN: 0-901-70353-X

There will be a reading assignment for each class.

There will be additional material on topics missing from the textbook, such as threads, graphics, lambda expressions, type inference, and testing


© 2016 Ken Baclawski All Rights Reserved

# Prerequisites

- Previous experience with C++ *not* required
- Programming experience *is* required
  - Control Structures (for, while, if, switch)
  - Classes
  - Methods
  - Arrays
  - Hashing
  - Sorting
  - Graph Searching


© 2016 Ken Baclawski All Rights Reserved

# Course Organization and Grading

- 10 Programming Assignments (40%)
- Team Project (20%)
- Mid-Term Exam (15%) 18 October 2016 1.5 hours
- Final Exam (25%) during exam week 2 hours
  - The Mid-Term and Final exams will be an open-book/open-notes exams.
  - Laptops are permitted at the exams with prior approval.
- Submit deliverables to Blackboard.
  - The due time is 11:59pm, the default for Blackboard.
- The grade will be reduced for the following:
  - Late assignment (1 point out of 100 for each hour)
  - Late Mid-Term Exam (1 point out of 100 for each minute)
  - Late Final Exam (1 point out of 100 for each minute)
- Extensions
  - Individual extensions are never given (except certain medical reasons)
  - Class extensions will be given for good reasons that are discussed in class


© 2016 Ken Baclawski All Rights Reserved

# Grading Scale

| Numerical Grade | Letter Grade |
|---|---|
| 93.333-100 | A |
| 90-93.332 | A- |
| 86.667-89.999 | B+ |
| 83.333-86.666 | B |
| 80-83.332 | B- |
| 76.667-79.999 | C+ |
| 73.333-76.666 | C |
| 70-73.333 | C- |
| 66.667-69.999 | D+ |
| 63.333-66.666 | D |
| 60-63.332 | D- |


© 2016 Ken Baclawski All Rights Reserved

# Class Schedule

- 9/9 Introduction to the Course
- 9/13 Strings and Statements; Project Ideas
- 9/16 Control Structures; Safety and Robustness
- 9/20 Vectors; Functional Requirements
- 9/23 Classes and Functions Part 1; Trees
- 9/27 Classes and Functions Part 2; The Call Stack
- 9/30 Iterators; UML Class Diagrams
- 10/4 Library Algorithms Part 1; UML Class Diagrams
- 10/7 Library Algorithms Part 2; Computational Complexity
- 10/11 Maps and Sets; Random Number Generation
- 10/14 Threads; Mid-Term Review
- 10/18 Mid-Term Exam
- 10/21 Graphics
- 10/25 Lambda Expressions; Graphs
- 10/28 Testing
- 11/1 Generic Programming; Turing Completeness
- 11/4 Pointers and Memory Management
- 11/8 Copying, Assigning and Moving
- 11/15 Conversions
- 11/18 Polymorphism
- 11/22 Smart Pointers
- 11/29 Object-Oriented Design; Design Patterns
- 12/2 Selected Project Presentations
- 12/6 Review for Final exam
- TBD Final exam


© 2016 Ken Baclawski All Rights Reserved

# Assignments

1. 9/15 Strings and Statements
2. 9/22 Control Structures
3. 9/29 Vectors
4. 10/6 Classes and Functions
5. 10/13 Iterators and Library Functions
6. 10/27 Maps and Threads
7. 11/3 Lambda Expressions and Testing
8. 11/10 Generic Programming
9. 11/17 Pointers and Memory Management
10. 12/1 Polymorphism and Smart Pointers


© 2016 Ken Baclawski All Rights Reserved

# Team Project Due Dates

- Team formation Members of the team and an optional team name 0% No deadline
- Project topic and requirements Topic, use cases and use case descriptions 20% Due 9/26
- Project design UML class diagram 20% Due 10/10
- Project implementation Documented source code 50% Due 11/28
- Project report Document (in any format) and slide presentation 10% No deadline


© 2016 Ken Baclawski All Rights Reserved

# Required Software

- C++11 Compiler
  - Recommended: g++ using the std=c++11 option
- Standard Template Library (STL)
  - Normally included with the C++ compiler
- Boost Library (STL extensions)
- Doxygen and graphviz for documentation
- Make utility for building your programs
- Valgrind for memory debugging
- Simple DirectMedia Layer - SDL version 2.0.4 for graphics
- Optional: An IDE such as Eclipse
- Optional: UML drawing tool


© 2016 Ken Baclawski All Rights Reserved

# Example Program


© 2016 Ken Baclawski All Rights Reserved

# The twice program

- The first example program using only
- Illustrates the use of strings and standard I/O.
- Specification:
  - Read a line from the console.
  - Concatenate the line with an exclamation point.
  - Print the concatenation twice.
- Example input:

  Hello, world
- Example output:

  Hello, world!

  Hello, world!


© 2016 Ken Baclawski All Rights Reserved

# Includes

- C++ has a standard software library
  - A collection of modules
  - The modules are documented on the web
- Include each library module you wish to use
  - The `#include <...>` statement includes a module
- Prefix module names with a namespace prefix
  - The standard library prefix is `std::`


© 2016 Ken Baclawski All Rights Reserved

# Documentation

- Based on comments in your code
- Generated using doxygen
  - Also requires graphviz
- Everyone uses the same configuration file
- Internal comments are also required


© 2016 Ken Baclawski All Rights Reserved

# Documentation Blocks

- A block is a comment of the form `/** ... */`
  - Doxygen allows other forms such as `/*! ... */`
- The first part of the documentation (up to the first period) is the *brief documentation*.
  - Never put a period in the brief documentation!
- The `\namespace` directive specifies a namespace and the documentation for the namespace as a whole.
  - There are many other doxygen directives.
- A documentation block before a function, class or parameter is the documentation for that function, class or parameter.
  - Same as javadoc
- Doxygen documentation is required and will be checked.


© 2016 Ken Baclawski All Rights Reserved

# The main function

- This is the program entry point.
  - Each program has exactly one main function.
  - For the assignments and project put it in Main.cpp
- The command-line arguments are parameters of main
  - Can be omitted
  - Will be discussed later
- Returning ends the program
  - Return value is the status code
  - Normal (successful) status code is 0.


© 2016 Ken Baclawski All Rights Reserved

# Variables

- Variables
  - *Must* specify a type
  - Memory is allocated to hold the value
  - Variable name *must* be in camelcase and start with a lowercase letter
- String variables
  - Value is an ASCII string
- Initializations

| | |
|---|---|
| `string x;` | `empty string` |
| `string x = “hello”;` | `5 byte string` |
| `string x(“hello”);` | `5 byte string` |


© 2016 Ken Baclawski All Rights Reserved

# Standard I/O Streams

- The standard input/output streams
  - `std::cin` standard input
  - `std::cout` standard output
- The `std::getline` function reads the next line of input
- `endl` is a *stream manipulator*: write a newline and flush the buffer
- The operator << writes a variable (buffered write)
  - Example of ***operator overloading***
  - The value of this operator is the stream
  - The operator associates to the left
  - `cout << "Hi" << endl;` is the same as `(cout << "Hi") << endl;`


© 2016 Ken Baclawski All Rights Reserved

# The frame program

- Read a name from the standard input
- Enclose a greeting in a frame made of asterisks
- Write the frame to the standard output
- Derived from the program in the textbook


© 2016 Ken Baclawski All Rights Reserved

# Using standard input and output

- For user input, start with a *prompt*
- The prompt statement does not use `std::endl`
  - It does not end the line (good)
  - It does not flush (bad)
- However, `std::cout` is *tied* to `std::cin`
  - Using `std::cin` causes `std::cout` to be flushed
- The `tie` function of the ostream class can be used to tie other output streams to input streams


© 2016 Ken Baclawski All Rights Reserved

# The `const` specifier

- A variable declared to be const cannot be modified after it is initialized
- Useful feature for ensuring program correctness
- Also improves readability
- It will be essential for parameter passing
- In purely functional programming languages, all variables are const in this sense.


© 2016 Ken Baclawski All Rights Reserved

# String concatenation

- Strings can be concatenated with the + operator
- This is an example of overloading
- String literals are not concatenated with the + operator
  - Just place the two literals next to each other:

    “Hello, “ “World”


© 2016 Ken Baclawski All Rights Reserved

# Variable Initialization

- Variables can be initialized in two ways:
  - With an equal sign: `int x = 0;`
  - With an argument list: `int x(0);`
- If there are multiple arguments, then one cannot use an equal sign:

```
string fiveStars(5, '*');
```

  This initializes `fiveStars` to `"******"`

- There is another way to initialize that will be introduced later.


© 2016 Ken Baclawski All Rights Reserved

# Anonymous object

- An object initialized in an expression is anonymous
- These are temporary objects that will be destroyed after the expression is completed.


© 2016 Ken Baclawski All Rights Reserved

# Coding Style Requirements


© 2016 Ken Baclawski All Rights Reserved

# Purpose

- Companies usually require that all code conform to their coding style.
- In this course, all your code must follow the posted coding style.
  - In addition to the requirements, there are also optional recommendations
  - Based on the textbook and online examples of coding style guidelines
- Readability of your code is important, but the format you use is not specified.
  - Your IDE or text editor will usually have a default format.


© 2016 Ken Baclawski All Rights Reserved

# Some Requirements

- Each class will introduce some requirements.
- The requirements for variables have already been explained above.
- Here are some other requirements for today:
  - Write in English
  - The last line of a file must not be a partial line
  - Use Indentation
  - Use the string class, not C-style strings


© 2016 Ken Baclawski All Rights Reserved

# Compiling and Running Programs


© 2016 Ken Baclawski All Rights Reserved

# Compiling and Running Programs

- Each program *must* have a file named `Main.cpp`
  - It *must* only contain the main function.
- Each class *must* have an include file with `.h` extension.
  - This is the class interface.
  - No implementation code.
- Each class may have a source file
  - A source file *must* have the `.cpp` extension.
  - The file name base *must* be the class name.
- Compile with: `g++ -Wall -std=c++11 *.cpp -o main`
- Run program with: `./main`


© 2016 Ken Baclawski All Rights Reserved

# Redirecting I/O

- For testing purposes one can redirect to/from a file
- DOS formatting will have unexpected consequences
  - The end-of-line is the CR-NL combination
  - A line read from the file will end with CR
  - The CR character is the command to erase the previous line!

| | |
|---|---|
| `./main < infile.txt` | Input is from `infile.txt` |
| `./main > outfile.txt` | Output is to `outfile.txt` |
| `./main < infile.txt > outfile.txt` | Both input and output are redirected |


© 2016 Ken Baclawski All Rights Reserved

# Assignment


© 2016 Ken Baclawski All Rights Reserved

# Assignment #1

- Read 5 lines from the standard input
- Concatenate the lines in reverse separated by spaces
- Print the concatenated lines twice
- Example input and output on the next two slides


© 2016 Ken Baclawski All Rights Reserved

# Assignment #1 Example Input

Solving difficult problems
Algorithm efficiency
Parallelism
Memory access
Roundoff control
Source: Wikipedia Divide and Conquer


© 2016 Ken Baclawski All Rights Reserved

# Assignment #1 Example Output

Roundoff control Memory access Parallelism Algorithm efficiency Solving difficult problems

Roundoff control Memory access Parallelism Algorithm efficiency Solving difficult problems


© 2016 Ken Baclawski All Rights Reserved

# Assignment #1 Submitting and Grading

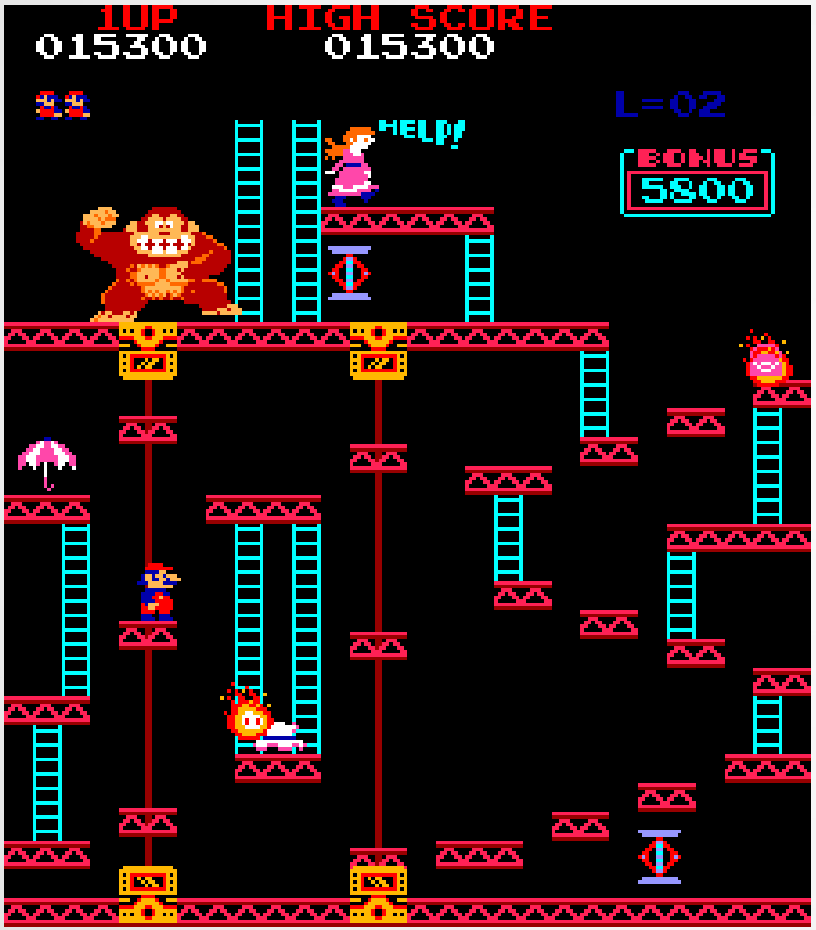
- All code is in the main function
- Submit `Main.cpp`
- No command-line arguments
- Grading:
  - Compile with no errors or warnings (20%)
  - Correct execution on test data (30%)
  - Documentation (20%)
  - Correct style (30%)


© 2016 Ken Baclawski All Rights Reserved

# Project Ideas


© 2016 Ken Baclawski All Rights Reserved

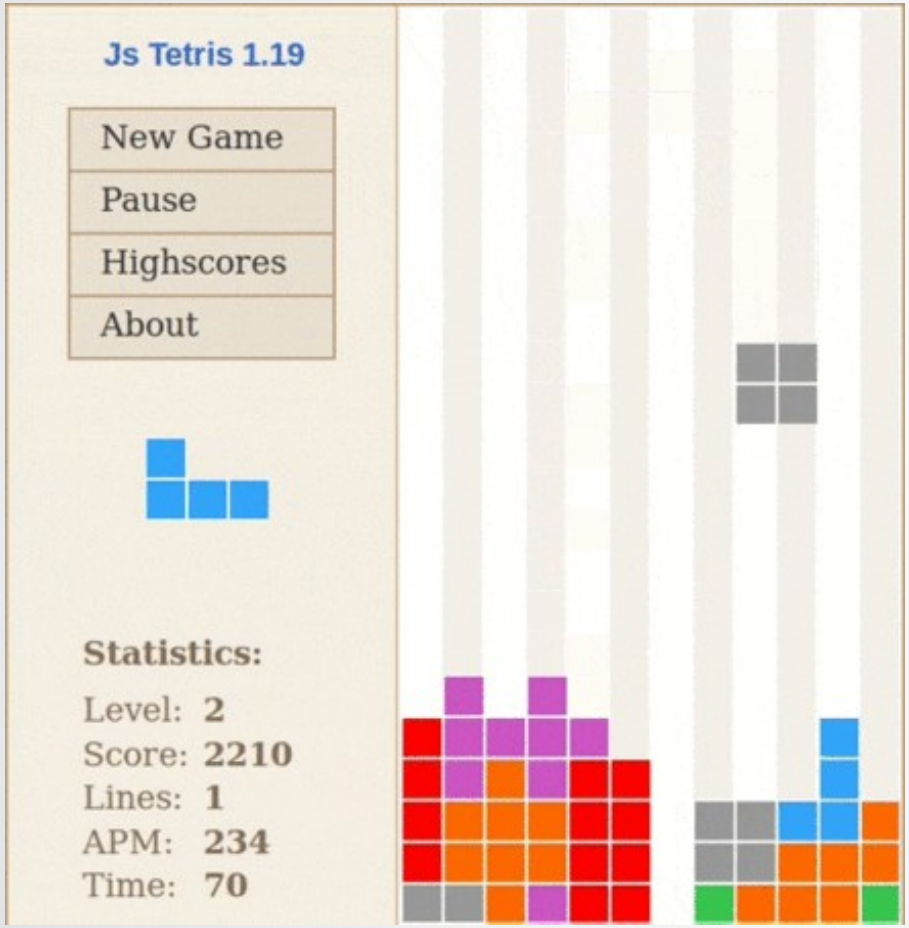
# Platform Game




- Characters climb or jump from platform to platform
- Can be multiplayer
  - Each player has a thread
  - Characters cannot be in the same place at the same time
  - Other behavior is possible


© 2016 Ken Baclawski All Rights Reserved

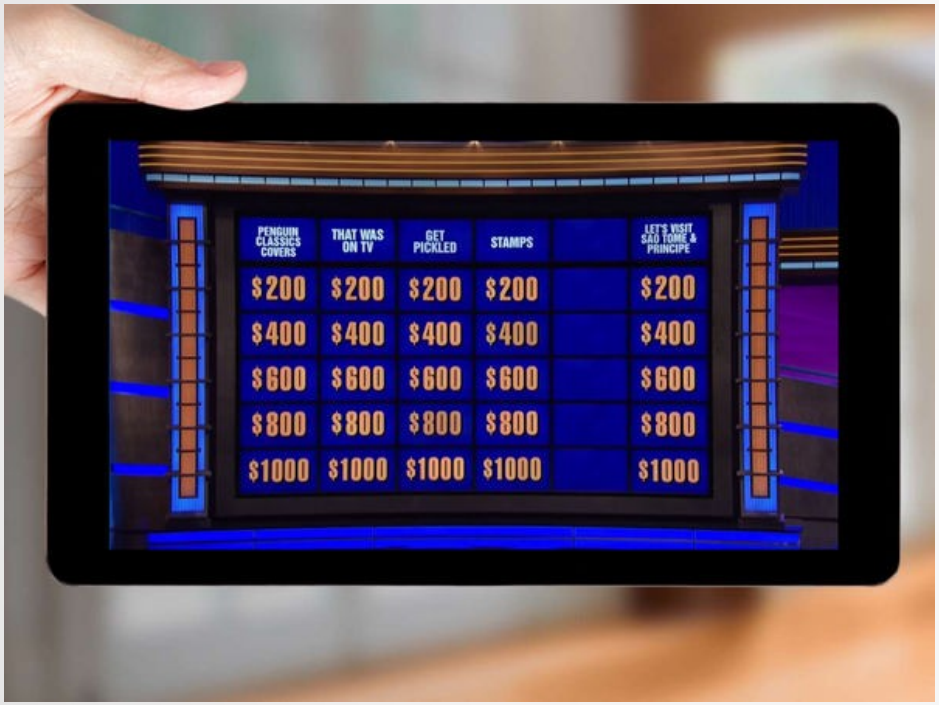
# Logic Game




- Single person game
- Pieces have logical constraints
- Pieces usually move
- May have time constraints


© 2016 Ken Baclawski All Rights Reserved

# Trivia Games




- General knowledge questions
- Can be multiple-choice
- Questions can be categorized
- Multiplayer version
  - Each player has a thread
  - First to click is allowed to answer
  - Penalty for wrong answer


© 2016 Ken Baclawski All Rights Reserved

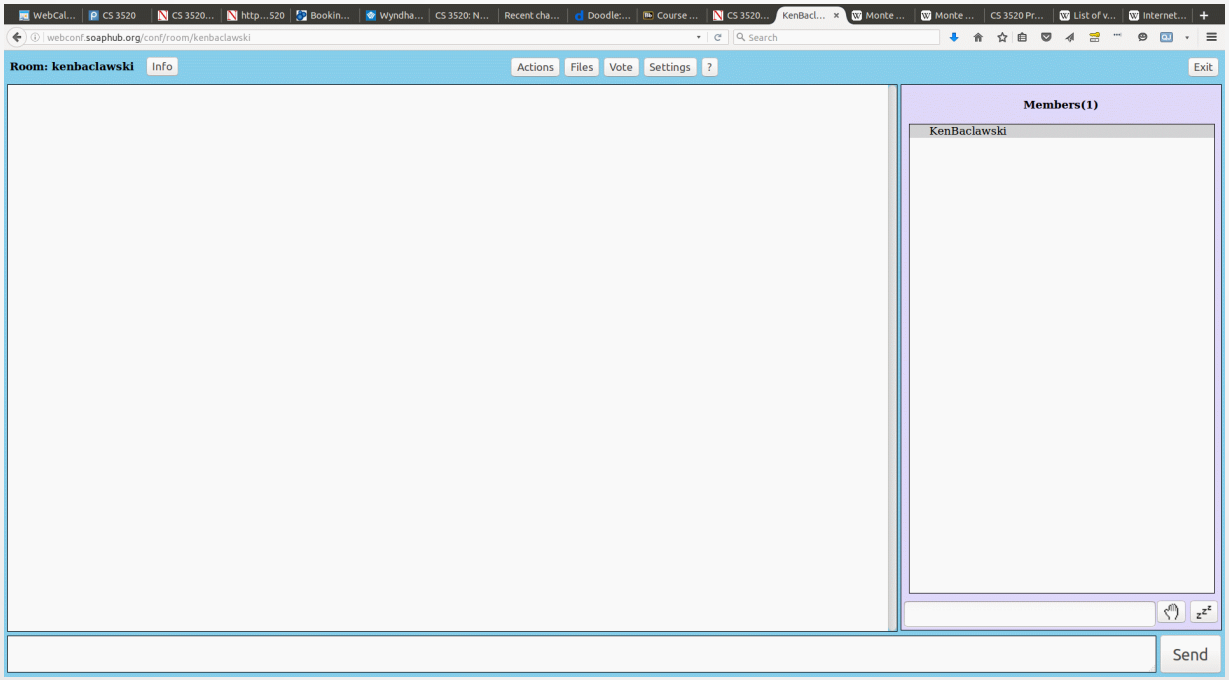
# Matching Game


- Single-person matching game
- Three-tile matching is the most popular


© 2016 Ken Baclawski All Rights Reserved

# Chat Room

- Always multiperson
- Each participant has a thread
- Each participant can enter text or images





© 2016 Ken Baclawski All Rights Reserved

# Polling Tool

- Always multiperson
- Each person has a thread
- Anyone can view the current state of a poll
- A person can vote in a poll
- Various constraints
  - Limited number of votes
  - Different types of votes (Yes, No, Maybe)


© 2016 Ken Baclawski All Rights Reserved

# Collaboration Tool

- Always multiperson
- Each person has a thread
- Anyone can view the current state of a document
- A person can check out a document, edit it and check it back in
- If another person wants to check out a document that is in use, then the person must wait.


© 2016 Ken Baclawski All Rights Reserved