

week 0 solutions

Generated by Doxygen 1.5.6

Tue Mar 24 17:14:11 2009

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	main.cpp File Reference	3
2.1.1	Detailed Description	3
2.1.2	Function Documentation	4
2.1.2.1	main	4
2.2	week0.hpp File Reference	5
2.2.1	Detailed Description	6
2.2.2	Function Documentation	7
2.2.2.1	babylonian	7
2.2.2.2	exp_approx	7
2.2.2.3	factorial	8
2.2.2.4	fib	8
2.2.2.5	green_crud	9
2.2.2.6	monty_hall	9
2.2.2.7	pi_approx	10
2.2.2.8	print_exp	10
2.2.2.9	quad_solve	11
2.2.2.10	rock_paper_scissors	11
2.2.2.11	statistics	11
2.2.2.12	to_lower	12

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

main.cpp (Main driver file, calls each function in turn)	3
week0.hpp (Week 0 solution file. Also serves as an example of proper programming style for this class. Pay close attention to the block comments for each function, which allows each to be used as a black box. Also, default values are used for most additional parameters) . .	5

Chapter 2

File Documentation

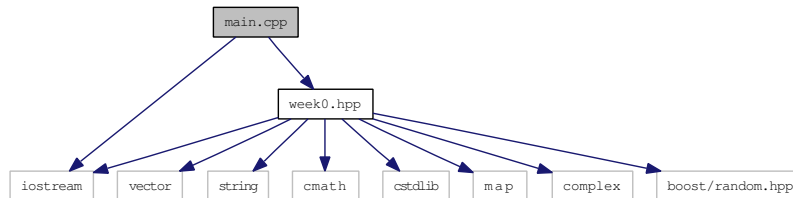
2.1 main.cpp File Reference

Main driver file, calls each function in turn.

```
#include <iostream>
```

```
#include <week0.hpp>
```

Include dependency graph for main.cpp:



Functions

- int [main](#) (int argc, char *argv[])

2.1.1 Detailed Description

Main driver file, calls each function in turn.

Author: D. Kevin McGrath

Created: 12 March 2009

Last Modified: 20 March 2009

Assignment: Chapter 2, Programming Project 12.

Chapter 2, Programming Project 11

Chapter 3, Programming Project 1.

Chapter 3, Programming Projects 10, 11, 12

EC. Chapter 3, Programming Projects 5, 6, 7, 13.

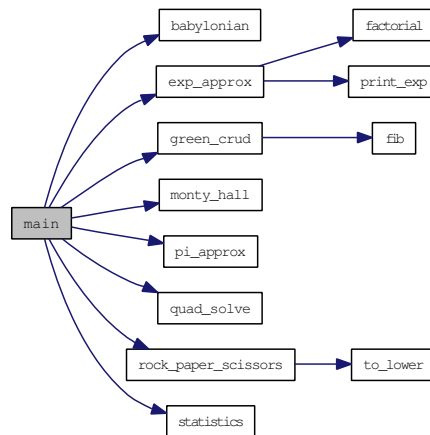
Definition in file [main.cpp](#).

2.1.2 Function Documentation

2.1.2.1 `int main (int argc, char * argv[])`

Definition at line 29 of file main.cpp.

Here is the call graph for this function:

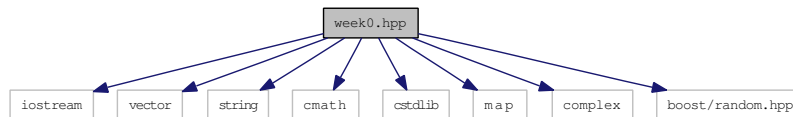


2.2 week0.hpp File Reference

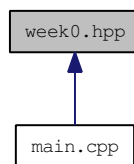
Week 0 solution file. Also serves as an example of proper programming style for this class. Pay close attention to the block comments for each function, which allows each to be used as a black box. Also, default values are used for most additional parameters.

```
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
#include <cstdlib>
#include <map>
#include <complex>
#include <boost/random.hpp>
```

Include dependency graph for week0.hpp:



This graph shows which files directly or indirectly include this file:



Functions

- double [babylonian](#) (double value, double tolerance=0.01)
Chapter 2, Programming Project 12: Babylonian algorithm.
- void [statistics](#) (int count=10, std::istream &in=std::cin, std::ostream &out=std::cout)
Chapter 2, Programming Project 11: statistics.
- std::string [to_lower](#) (std::string s)
convert strings to lower case
- void [rock_paper_scissors](#) (std::istream &in=std::cin, std::ostream &out=std::cout)
Chapter 3, Programming Project 1: Rock, Paper, Scissors.

- int `fib` (int n)
fibonacci sequence, iteratively
- double `green_crud` (double initial_mass, int days_passed, int maturation=5, std::ostream &out=std::cout)
Chapter 3, Programming Project 10: Green crud population.
- void `print_exp` (std::vector< double > values, std::ostream &out=std::cerr)
Formatted output for exponential series.
- double `factorial` (double n)
factorial, iteratively
- double `exp_approx` (double x, int terms=100, bool compare=false, std::ostream &out=std::cerr)
Chapter 3, Programming Project 11: Approximation to exponential.
- double `pi_approx` (int terms=10000, bool output=false, std::ostream &out=std::cout)
Chapter 3, Programming Project 12: pi approximation.
- void `monty_hall` ()
Chapter 3, Programming Project 13: The Monty Hall Problem.
- void `quad_solve` (double a, double b, double c)
Chapter 3, Programming Project 6: Quadratic equation solver.

2.2.1 Detailed Description

Week 0 solution file. Also serves as an example of proper programming style for this class. Pay close attention to the block comments for each function, which allows each to be used as a black box. Also, default values are used for most additional parameters.

Author: D. Kevin McGrath

Created: 12 March 2009

Last Modified: 20 March 2009

Style points:

- 1) All magic values are stored in variables, and comments are used to explain where they came from.
- 2) All variables are declared at TOP of each function. This allows type changes without hunting through the code to find the proper variable.
- 3) Block comments are used to describe each function, in doxygen style, including algorithm descriptions.
- 4) With the exception of break and continue, all if statements and loops use {} to delimit body.
- 5) Variables are named in meaningful ways, unless a parameter name is taken from the algorithm description.

Definition in file [week0.hpp](#).

2.2.2 Function Documentation

2.2.2.1 double `babylonian` (double *value*, double *tolerance* = 0.01)

Chapter 2, Programming Project 12: Babylonian algorithm.

The babylonian algorithm uses an iterative approach to calculating the square root of a value. Algorithm is as follows:

1. Make a guess at the answer.
2. Compute $r = \text{value} / \text{guess}$
3. $\text{guess} = (\text{guess} + r) / 2$
4. Repeat steps 2 and 3 unless consecutive values of guess are within tolerance.

Parameters:

value,: Value to take square root of.

tolerance,: Tolerance of consecutive guess. Defaults to 0.01.

Notice the use of a default parameter.

Definition at line 66 of file week0.hpp.

Here is the caller graph for this function:



2.2.2.2 double `exp_approx` (double *x*, int *terms* = 100, bool *compare* = false, std::ostream & *out* = std::cerr)

Chapter 3, Programming Project 11: Approximation to exponential.

Uses the taylor series expansion of e^x

Parameters:

x,: exponent

terms,: how many terms to include in expansion

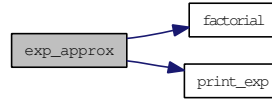
out,: Output stream to write to

The series expansion is

$$e^x = \sum_{n=0}^{\infty} x^n / n!$$

Definition at line 358 of file week0.hpp.

Here is the call graph for this function:



Here is the caller graph for this function:



2.2.2.3 double factorial (double n)

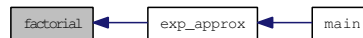
factorial, iteratively

Parameters:

n ,: value to take factorial of

Definition at line 332 of file week0.hpp.

Here is the caller graph for this function:



2.2.2.4 int fib (int n)

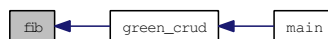
fibonacci sequence, iteratively

Parameters:

n ,: Term number in fibonacci sequence (0 offset)

Definition at line 264 of file week0.hpp.

Here is the caller graph for this function:



2.2.2.5 double green_crud (double *initial_mass*, int *days_passed*, int *maturation* = 5, std::ostream & *out* = std::cout)

Chapter 3, Programming Project 10: Green crud population.

Calculates the green crud population based on the observation that they grow in a fashion identical to the fibonacci sequence.

Parameters:

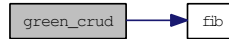
initial_mass,: Initial mass, in pounds, of green crud.

days_passed,: Amount of time we are interested in.

out,: Output stream to write to

Definition at line 294 of file week0.hpp.

Here is the call graph for this function:



Here is the caller graph for this function:



2.2.2.6 void monty_hall ()

Chapter 3, Programming Project 13: The Monty Hall Problem.

The problem can be stated as follows:

=====

Suppose you're on a game show and you're given the choice of three doors. Behind one door is a car; behind the others, consolation prizes. The car and the goats were placed randomly behind the doors before the show. The rules of the game show are as follows: After you have chosen a door, the door remains closed for the time being. The game show host, Monty Hall, who knows what is behind the doors, now has to open one of the two remaining doors, and the door he opens must have a goat behind it. If both remaining doors have goats behind them, he chooses one randomly. After Monty Hall opens a door with a goat, he will ask you to decide whether you want to stay with your first choice or to switch to the last remaining door.

Imagine that you chose Door 1 and the host opens Door 3, which has a goat. He then asks you "Do you want to switch to Door Number 2?" Is it to your advantage to change your choice?

=====

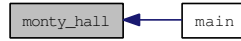
This function simulates the Monty Hall Problem, and outputs the probability of winning if you

- a) don't switch doors
- b) switch doors

The answer may just surprise you...

Definition at line 454 of file week0.hpp.

Here is the caller graph for this function:



2.2.2.7 double pi_approx (int terms = 10000, bool output = false, std::ostream & out = std::cout)

Chapter 3, Programming Project 12: pi approximation.

Parameters:

terms,: Number of terms in expansion

out,: Output stream to write to

The series expansion is given as

$$\pi = 4 \sum_{n=0}^{\infty} (-1)^n / (2n + 1)$$

Definition at line 398 of file week0.hpp.

Here is the caller graph for this function:



2.2.2.8 void print_exp (std::vector< double > values, std::ostream & out = std::cerr)

Formatted output for exponential series.

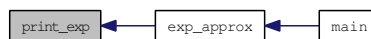
Parameters:

values,: Double vector of partial sums

out,: Output stream to write to

Definition at line 314 of file week0.hpp.

Here is the caller graph for this function:



2.2.2.9 void quad_solve (double a, double b, double c)

Chapter 3, Programming Project 6: Quadratic equation solver.

Parameters:

a,: coefficient of x^2

b,: coefficient of x

c,: constant term

This function prints the roots of the equation $ax^2 + bx + c = 0$

Definition at line 508 of file week0.hpp.

Here is the caller graph for this function:

**2.2.2.10 void rock_paper_scissors (std::istream & in = std::cin, std::ostream & out = std::cout)**

Chapter 3, Programming Project 1: Rock, Paper, Scissors.

Parameters:

in,: Input stream to read from

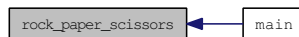
out,: Output stream to write to

Definition at line 184 of file week0.hpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**2.2.2.11 void statistics (int count = 10, std::istream & in = std::cin, std::ostream & out = std::cout)**

Chapter 2, Programming Project 11: statistics.

Again, using default parameters. Just in case we want to read from a file or output to a file.

Parameters:

count,: Number of values to read in

in,: Input stream to read from

out,: Output stream to write to

Definition at line 101 of file week0.hpp.

Here is the caller graph for this function:



2.2.2.12 std::string to_lower (std::string s)

convert strings to lower case

Parameters:

s,: argument to convert to lower case

Definition at line 167 of file week0.hpp.

Here is the caller graph for this function:

