CS 202 Homework 00
Report

Justyn Durnford

January 19, 2020

Source Code Link: `https://github.com/Yaboi-Gengarboi/cs202/tree/master/Homework%200`

This homework took approximately 6 hours to complete.

1 Design

Since I will be working with C-strings, I decided to include the cstdlib header for my program, where I will use the strtod method as instructed. However, for the C++ method, I will create a string with the C-string as an argument and use the stod method defined in the string header. Since I will also be printing and getting information from the console, I will be using the iostream's cout and cin stream tools.

Of course, passing invalid input into these methods would throw errors, so I could use a try-catch structure in my main function with the stdexcept header so that I can catch these errors.

2 Post Mortem

For the Temperature project, the practice mostly aligned with the theory, though there was 1 issue that I did not anticipate.

The method strtod does not throw exceptions.

I had originally just used a try-catch structure to detect an invalid_argument exception that is thrown by the method stod, which worked fine. However, the C method strtod, upon an unsuccessful conversion, strtod instead simply returns 0.0, which is nice if we don't want to throw an error, but in my case, I do. In order to catch this, I instead had to check if the C-string was "0.0" using the strcmp method if strtod returned 0.0;

Furthermore, I created a global vector that would contain any C-strings that were not valid entries. In both the c_ctof and cpp_ftoc, when they detect an invalid string, they add it to the vector. Later, in the main program, if one of the C-strings is in the vector, it will simply print that the C-string is invalid.

Overall not terribly difficult, just a bit more complex than I thought it would be.

3 Answers to Questions

- Software has the most utility where calculations are present or there are tasks that need to be monitored or controlled. This means software would be great for complex calculations, measuring bodily responses such blood concentration of a chemical, or video games.
- There is no easy answer as to what a software developer looks like. There are rarely uniforms or specific dress codes for developers in many companies.
- My computer has a total memory capacity of 1,048 gigabytes, 117 of which being the main OS memory and the remaining 931 gigabytes being disk memory.
- Five applications of computer programs can include document formatting, deriving and integrating equations, tracking employee clock-ins and clock-outs, developing websites and developing game software, the latter of which I am extremely interested in due to their complexity and entertainment.

  I've always loved to analyze games I play to learn more about how they work, what I enjoy about them, and create ideas on how to improve them. I always love having discussions with other players and making lists of ideas to change mechanics and characters to make the game more fun, competitive and enjoyable.
- A hexadecimal digit represents 4 binary digits (half a byte, a nibble) while an octal digit represents 3 binary digits.

4 Temperature Output

Listing 1: temperature_main Output

```
Celcius to Fahrenheit:
45.0 -> 113
```

```
-12.0 -> 10.4
67 -> 152.6
ERROR: bsrb is not a valid input.
Fahrenheit to Celcius:
45.0 -> 7.22222
-12.0 -> -24.4444
67 -> 19.4444
ERROR: bsrb is not a valid input.
```

5 Fraction Output

Listing 2: fraction_test Output

```
f1: 0 / 1 = 0
f2: 5 / 2 = 2.5
f3: 7 / 3 = 2.33333
f4: 1 / 9 = 0.111111
f0: 1 / 1 = 1
```

6 temperature_main.cpp

```cpp
//temperature_main.cpp
//Justyn P. Durnford
//Created on 1/15/2020
//Last Updated on 1/16/2020
//https://github.com/Yaboi-Gengarboi/cs202/tree/master/Homework%200/Temperature

#include <string>
using std::string;
using std::stod;
using std::to_string;

#include <vector>
using std::vector;

#include <cstring>
using std::strtod;
using std::strcmp;

#include <iostream>
using std::cout;
using std::endl;

#include <stdexcept>
using std::invalid_argument;

vector<const char*> invalid;

//Takes the given C-string, cstr, which
//represents a temperature in Celcius
//and converts it to Fahrenheit using the
//strtod method in the header cstdlib.
double c_ctof(const char* cstr)
{
  double return_value = 0.0;

  return_value = strtod(cstr, nullptr);

  //strtod never throws exceptions, it returns
  //0.0 if the conversion was unsuccessful.
  //Therefore, if 0.0 is the result, we will need
  //to use the strcmp method to check if the
  //original C-String was not 0.0.
  if (return_value == 0.0 && strcmp(cstr, "0.0") != 0) //Unsuccessful conversion
  {
    //Adds the invalid C-String to the vector
    //that will print the invalid arguments in
    //main().
    invalid.push_back(cstr);
  }
  else
  {
    //Modifies the double from celcius
    //to fahrenheit.
    return_value *= 9.0 / 5.0;
    return_value += 32;
  }

  return return_value;
}

//Takes the given C-string, cstr, which
//represents a temperature in Fahrenheit
//and converts it to Celcius using the
//stod method in the header string.
double cpp_ftoc(const char* cstr)
{
  double return_value = 0.0;

  try
  {
    //Creates a temporary C++string that
    //stod uses.
    string str(cstr);
    return_value = stod(str);

    //Modifies the double from fahrenheit
    //to celcius.
    return_value -= 32;
    return_value *= 5.0 / 9.0;
  }
  catch (invalid_argument & ia) //Invalid conversion
  {
    //Adds the invalid C-String to the vector
    //that will print the invalid arguments in
    //main().
    invalid.push_back(cstr);
  }

  return return_value;
}

int main(int argc, char* argv[])
{
  size_t vec_size = 0;
  double result = 0.0;

  if (argc < 2) //Not enough arguments
  {
    cout << "Not enough arguments entered. Enter at least 2." << endl;
    return 1;
```

```cpp
  }

  cout << "Celcius to Fahrenheit:" << endl;
  for (int i = 2; i < argc; ++i)
  {
    result = c_ctof(argv[i]);
    if (invalid.size() > vec_size)
    {
      cout << "ERROR: " << argv[i] << " is not a valid input." << endl;
      ++vec_size;
    }
    else
      cout << argv[i] << " -> " << result << endl;
  }

  invalid.clear();
  vec_size = 0;

  cout << "Fahrenheit to Celcius:" << endl;
  for (int i = 2; i < argc; ++i)
  {
    result = cpp_ftoc(argv[i]);
    if (invalid.size() > vec_size)
    {
      cout << "ERROR: " << argv[i] << " is not a valid input." << endl;
      ++vec_size;
    }
    else
      cout << argv[i] << " -> " << result << endl;
  }

  return 0;
}
```

7 Fraction.hpp

```cpp
//Fraction.hpp
//Justyn P. Durnford
//Created on 12/14/2019
//Last Updated on 1/18/2020
//https://github.com/Yaboi-Gengarboi/cs202/tree/master/Homework%200/Fraction

/*
This class allows an "exact" representation of the quotient
of two integers by storing them and allowing the use of the
individual integers or the actual result.
A Fraction can be constructed by default, with two individual
integers, a C-array or an std::array.
Fraction objects are also capable of arithmetic with integers
other Fraction objects.
A Fraction can be represented as an std::array with the to_arr
method or an std::string with the to_str method.
Note that any method that can assign or otherwise modify
_denominator will never allow it to be set to 0 and will result
in _denominator remaining at its default value of 1.
*/

#ifndef FRACTION_HPP
#define FRACTION_HPP
```

```cpp

#include <array>
#include <string>

class Fraction
{
  int _numerator = 0;
  int _denominator = 1;

public:

  //Constructors.
  Fraction();
  Fraction(int numer, int denom);
  Fraction(int f_arr[2]);
  Fraction(std::array<int, 2> f_arr);

  //Destructor.
  ~Fraction();

  //Returns _numerator.
  int get_numerator() const;

  //Returns _denominator.
  int get_denominator() const;

  //Sets _numerator to numer.
  void set_numerator(int numer);

  //Sets _denominator to denom.
  void set_denominator(int denom);

  //Returns the integer division of the fraction.
  int int_result() const;

  //Returns the decimal division of the fraction.
  double double_result() const;

  //Adds num onto the fraction.
  void add(int num);

  //Subtracts num onto the fraction.
  void subtract(int num);

  //Multiplies num onto the fraction.
  void multiply(int num);

  //Divides num onto the fraction.
  void divide(int num);

  //Adds frac onto the fraction.
  void add(const Fraction& frac);

  //Subtracts frac onto the fraction.
  void subtract(const Fraction& frac);

  //Multiplies frac onto the fraction.
  void multiply(const Fraction& frac);

  //Divides frac onto the fraction.
  void divide(const Fraction& frac);

  //Returns an std::array representation of the fraction.
  std::array<int, 2> to_arr() const;

  //Returns an std::string representation of the fraction.
  std::string to_str() const;
};

#endif //#ifndef FRACTION_HPP
```

8 Fraction.cpp

```
//Fraction.cpp
//Justyn P. Durnford
//Created on 12/14/2019
//Last Updated on 1/18/2020
//https://github.com/Yaboi-Gengarboi/cs202/tree/master/Homework%200/Fraction

#include "Fraction.hpp"

#include <array>
using std::array;

#include <string>
using std::string;
using std::to_string;

Fraction::Fraction() {/* Default values are 0 and 1. */ }

Fraction::Fraction(int numer, int denom)
{
  _numerator = numer;

  if (denom != 0)
    _denominator = denom;
}

Fraction::Fraction(int f_arr[2])
{
  _numerator = f_arr[0];

  if (f_arr[1] != 0)
    _denominator = f_arr[1];
}

Fraction::Fraction(array<int, 2> f_arr)
{
  _numerator = f_arr[0];

  if (f_arr[1] != 0)
    _denominator = f_arr[1];
}

Fraction::~Fraction() {/* Destructor */ }

int Fraction::get_numerator() const
{
  return _numerator;
}

int Fraction::get_denominator() const
{
  return _denominator;
}

void Fraction::set_numerator(int numer)
{
  _numerator = numer;
}

void Fraction::set_denominator(int denom)
{
  if (denom != 0)
    _denominator = denom;
}
```

```cpp

int Fraction::int_result() const
{
  return _numerator / _denominator;
}

double Fraction::double_result() const
{
  return ((1.0 * _numerator) / (1.0 * _denominator));
}

void Fraction::add(int num)
{
  num *= _denominator;
  _numerator += num;
}

void Fraction::subtract(int num)
{
  num *= _denominator;
  _numerator -= num;
}

void Fraction::multiply(int num)
{
  _numerator *= num;
}

void Fraction::divide(int num)
{
  if (num != 0)
    _denominator *= num;
}

void Fraction::add(const Fraction& frac)
{
  array<int, 2> frac_arr = frac.to_arr();
  int temp = 0;

  if (_denominator != frac_arr[1])
  {
    _numerator *= frac_arr[1];
    frac_arr[0] *= _denominator;

    temp = _denominator;
    _denominator *= frac_arr[1];
    frac_arr[1] *= temp;
  }

  _numerator += frac_arr[0];
}

void Fraction::subtract(const Fraction& frac)
{
  array<int, 2> frac_arr = frac.to_arr();
  int temp = 0;

  if (_denominator != frac_arr[1])
  {
    _numerator *= frac_arr[1];
    frac_arr[0] *= _denominator;

    temp = _denominator;
    _denominator *= frac_arr[1];
    frac_arr[1] *= temp;
  }

  _numerator -= frac_arr[0];
}

void Fraction::multiply(const Fraction& frac)
{
  _numerator *= frac.get_numerator();
}

void Fraction::divide(const Fraction& frac)
{
  _denominator *= frac.get_denominator();
}

array<int, 2> Fraction::to_arr() const
{
  array<int, 2> arr = { _numerator, _denominator };
  return arr;
}

string Fraction::to_str() const
{
  string str = "";
  str += to_string(_numerator);
  str += " / ";
  str += to_string(_denominator);
  return str;
}
```

9 fraction_test.cpp

```
//fraction_test.cpp
//Justyn P. Durnford
//Created on 1/18/2020
//Last Updated on 1/18/2020
//https://github.com/Yaboi-Gengarboi/cs202/tree/master/Homework%200/Fraction

#include "Fraction.hpp"

#include <array>
using std::array;

#include <iostream>
using std::cout;
using std::endl;

int main()
{
  Fraction f1;

  Fraction f2(5, 2);

  int f3_arr[2] = { 7, 3 };
  Fraction f3(f3_arr);

  array<int, 2> f4_arr = { 1, 9 };
  Fraction f4(f4_arr);

  Fraction f0(1, 0);

  cout << "f1: " << f1.to_str() << " = " << f1.double_result() << endl;
```

```cpp
  cout << "f2: " << f2.to_str() << " = " << f2.double_result() << endl;
  cout << "f3: " << f3.to_str() << " = " << f3.double_result() << endl;
  cout << "f4: " << f4.to_str() << " = " << f4.double_result() << endl;
  cout << "f0: " << f0.to_str() << " = " << f0.double_result() << endl;

  return 0;
}
```