CS 201 Homework 2

Solomon Himelbloom

September 21, 2020

- Repository Link: `https://github.com/techsolomon/cs201`
- Git Commits: `https://github.com/techsolomon/cs201/commits`
- This homework took approximately 8 hours to complete.

1 Design

This third homework assignment was designed with readability in mind, meaning that each file used multiple commits, functions were broken down into smaller parts, and various improvements were made after testing for optimal output. I once again utilized a monorepo to store four directories under the hw2 folder – the main program (names), two additional programs (money and rice), and a practice folder (for extra programs, such as a draft of scores). Every .cpp file starts with the author's name, the computer science class code, date file was created or last edited, and finally a short description. Every directory also has a Makefile.

2 Post Mortem

Upon building each C++ file and referencing notes from class, I tried my best to use the DRY (or don't repeat yourself) principles with avoidance of copying other parts of my program (such as cout statements) multiple times. In the future, I would like to work on a better understanding of when functions and multiple files are needed.

For example, additional clarification would be ideal when drafting a program that needs to be split into multiple files, functions, or header files, as this information proves beneficial for large-scale programs with increasing complexity. I'm sure that this skill comes with practice, and I look forward to refactoring specific areas of hw2 that will become more concise and succinct.

3 Answers to Questions

1. Typical sizes of a char, int, and double are 1 byte, 4 bytes, and 8 bytes respectively.
2. Definitions provide the compiler (and IDE) with the necessary information needed for machine code generation to be used later in the program.
3. Initialization makes sure that your function is ready to be used, whereas an assignment is a type of initialization.
4. Conversions are considered safe if the direction is from smaller to bigger, meaning that the type you are trying to convert to is smaller than your current type. This is important as you want a correct output that does not sacrifice any data precision during the conversion.
5. Computations generally tend to be thought of as calculations of numbers but can include far more. For example, one might use computations to include depictions of computer logic in their code and another person can use computations for combinations of variables for optimal memory storage.
6. Expressions specify specific computations whereas statements perform given actions about lines of code.
7. Aptly named, constant expressions are constant (or immutable) meaning they are expressions that cannot be changed.

8. Integers are defined as real numbers, whereas strings are depicted using text. This means that you can convert from int to string but not vice versa. You can also add multiple integers together but not multiple strings (without explicit text concatenation).

9. Strings allow for text manipulation, which is not possible with ints. Since strings are considered objects with their respective methods, strings and ints also have different length capacities before you enter an overflow.

10. The *vector char alphabet(26)* call initializes a vector of size 26 named alphabet. This means that we can eventually fill it with letters from A to Z.

4 Program 1

4.1 Sample Output/Screenshot

Listing 1: Sample Program Output

```
Please enter a name: Alice
Please enter a name: Bob
Please enter a name: Solomon
Please enter a name: Arielle
Please enter a name: Linda
Please enter a name: Brian
Please enter a name: Yale
Please enter a name: Edith
Please enter a name: Lottie
Please enter a name: Max

This prints the names that the user provided.
What name would you like to search for? Solomon
Solomon does exist in the list above.
Need a query? This allows for easy searchability of
   the names.
Learn the number of characters in each string.
```

```
Total Names: 10
This prints the names that the user provided.
```

4.2 Git Commit Messages

Date	Message
2020-09-22	add: searchName and adjust printed output
2020-09-22	add: print, search, encrypt, sort, and countNames function
2020-09-21	edit: user input/output
2020-09-21	add: vector testing loops
2020-09-20	add: hw2 names (initial files)

4.3 Source Code

```
// names.cpp
// Solomon Himelbloom
// 20 September 2020
// Names (and vectors) example for CS 201.

#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using std::cout;
using std::endl;
using std::cin;

void inputNames(std::vector<std::string> & names) {
    for (int i = 0; i < 10; i++) {
        std::string name;
        std::cout << "Please enter a name: ";
        std:getline(std::cin, name);
        names.push_back(name);
    }
}

bool doesNameExist(const std::string & nametoFind, const std::vector<std::string> & names) {
    for (int i = 0; i < names.size(); i++) {

    }

    return false;
}

void printNames(std::vector<std::string> & names) {
    cout << "This prints the names that the user provided." << endl;

    for (int i = 0; i < names.size(); i++) {

    }
}

void searchNames(std::vector<std::string> & names) {
    cout << "Need a query? This allows for easy searchability of the names." << endl;
}

void encryptNames(std::vector<std::string> & names) {
    cout << "COMING SOON: This encrypts the aformentioned names using a cipher." << endl;
}

void sortNames(std::vector<std::string> & names) {
    cout << "COMING SOON: Use computer science sorting algorithms to make your job easier!" << endl;
}

void countNames(std::vector<std::string> & names) {
    cout << "Learn the number of characters in each string." << endl;

    int totalChars = names.size();
    cout << "Total Names: " << totalChars << endl;
}

int main (int argc, char ** argv) {
    std::vector<std::string> names;
    std::string search_name;
    int name = 1;

    inputNames(names);
    cout << " " << endl;
    printNames(names);

    cout << "What name would you like to search for? ";
    cin >> search_name;

    if (name == 1) {
        cout << search_name << " does exist in the list above." << endl;
    }

    else {
        cout << search_name << " does not exist in the list above." << endl;
    }

    // cout << doesNameExist(search_name, names);

    searchNames(names);
    countNames(names);
    printNames(names);

    return 0;
}
```

5 Program 2

5.1 Sample Output/Screenshot

Listing 2: Sample Program Output

```
How many pennies do you have? 23
How many nickels do you have? 17
How many dimes do you have? 14
How many quarters do you have? 7
How many halfdollars do you have? 3
How many one-dollar coins do you have? 0

You have 23 pennies.
You have 17 nickels.
You have 14 dimes.
You have 7 quarters.
You have 3 half dollars.
You have 0 one-dollar coins.

The value of all your coins is: $5.73
```

5.2 Git Commit Messages

Date	Message
2020-09-21	refactor: account for grammar and negative coins
2020-09-20	refactor: bankAccount and add: coinReceipt
2020-09-20	add: bankAccount function and total logic
2020-09-20	add: hw2 money sample output

5.3 Source Code

```cpp
// money.cpp
// Solomon Himelbloom
// 20 September 2020
// Money (coins to dollars) example for CS 201.

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

using namespace std;

void bankAccount(string coinType, float numberOfCoins) {
    cout << "How many " << coinType << " do you have? ";
}

void coinReceipt(float numberOfCoins, string coinType) {
    cout << "You have " << numberOfCoins << " " << coinType << "." << endl;
}

int main() {
    float pennies, nickels, dimes, quarters, half_dollars, one_dollars = 0;
    float total_coins = 0;

    bankAccount("pennies", pennies);
    cin >> pennies;

    bankAccount("nickels", nickels);
    cin >> nickels;

    bankAccount("dimes", dimes);
    cin >> dimes;

    bankAccount("quarters", quarters);
    cin >> quarters;

    bankAccount("halfdollars", half_dollars);
    cin >> half_dollars;

    bankAccount("one-dollar coins", one_dollars);
    cin >> one_dollars;

    // Print out the total number and type of each coin.
    cout << " " << endl;
    if (pennies == 1 || nickels == 1 || dimes == 1
    || quarters == 1 || half_dollars == 1 || one_dollars == 1) {
        coinReceipt(pennies, "penny");
        coinReceipt(nickels, "nickel");
        coinReceipt(dimes, "dime");
        coinReceipt(quarters, "quarter");
        coinReceipt(half_dollars, "half dollar");
        coinReceipt(one_dollars, "one-dollar coin");
    }

    else if (pennies == 0 || pennies > 1 || nickels == 0 || nickels > 1 ||
    dimes == 0 || dimes > 1 || quarters == 0 || quarters > 1 ||
    half_dollars == 0 || half_dollars > 1 || one_dollars == 0 || one_dollars > 1) {
        coinReceipt(pennies, "pennies");
        coinReceipt(nickels, "nickels");
        coinReceipt(dimes, "dimes");
        coinReceipt(quarters, "quarters");
        coinReceipt(half_dollars, "half dollars");
        coinReceipt(one_dollars, "one-dollar coins");
    }

    else {
        cout << "Please enter a positive number of coins. Thanks!" << endl;
    }

    // Convert number of coins to dollars.
    pennies *= 0.01;
    nickels *= 0.05;
    dimes *= 0.10;
    quarters *= 0.25;
    half_dollars *= 0.50;
    one_dollars *= 1.00;

    cout << " " << endl;
    total_coins = (pennies + nickels + dimes + quarters + half_dollars + one_dollars);
    cout << "The value of all your coins is: $" << total_coins << endl;

    return 0;
}
```

6 Program 3

6.1 Sample Output/Screenshot

Listing 3: Sample Program Output

```
What is your desired chess square? 9

SQUARE #1:
current square: 1
previous square total: 0

SQUARE #2:
current square: 2
previous square total: 1

SQUARE #3:
current square: 4
previous square total: 3

SQUARE #4:
current square: 8
previous square total: 7
```

```
SQUARE #5:
current square: 16
previous square total: 15

SQUARE #6:
current square: 32
previous square total: 31

SQUARE #7:
current square: 64
previous square total: 63

SQUARE #8:
current square: 128
previous square total: 127

SQUARE #9:
current square: 256
previous square total: 255

cs201     hw2 questions
At least 1,000 (grains of rice): SQUARE #10
At least 1,000,000 (grains of rice): SQUARE #21
At least 1,000,000,000 (grains of rice): SQUARE #31
Largest number (int): SQUARE #31
Largest number (float): SQUARE #1024
```

6.2 Git Commit Messages

Date	Message
2020-09-22 double tests	add: question limits; int
2020-09-21	add: double currentSquare, previousSquare, and user input
2020-09-21	add: square counting loop and move: practice/-names
2020-09-20	add: hw2 rice blank template files

6.3 Source Code

```cpp
// rice.cpp
// Solomon Himelbloom
// 20 September 2020
// Rice and chessboard (exponential growth) example for CS 201.

#include <iostream>
#include <math.h>
using std::cout;
using std::endl;
using std::cin;

double currentSquare (int square) {
    return pow(2, square - 1);
}

double previousSquare (int square) {
    double total = 0;
    for (int i = 1; i <= square; i++) {
        total = currentSquare(i);
    }

    return total;
}

int main() {
    double i, requested_square = 0;
    double square_number = 0;
    double current_square = 0;
    double previous_square = 0;

    cout << "What is your desired chess square? ";
    cin >> requested_square;
    cout << " " << endl;

    for (i = 0; i < requested_square; i++) {
        previous_square += current_square;
        square_number += 1;
        current_square = pow(2, square_number - 1);

        cout << "SQUARE #" << square_number << ":" << endl;
        cout << "current square: " << current_square << endl;
        cout << "previous square total: " << previous_square << endl;
        cout << " " << endl;
    }

    cout << "cs201 { hw2 questions" << endl;
    cout << "At least 1,000 (grains of rice): SQUARE #10" << endl;
    cout << "At least 1,000,000 (grains of rice): SQUARE #21" << endl;
    cout << "At least 1,000,000,000 (grains of rice): SQUARE #31" << endl;
    cout << "Largest number (int): SQUARE #31" << endl;
    cout << "Largest number (float): SQUARE #1024" << endl;
}
```