CS 201 Homework 02
Computation

Sarah Carter

September 16, 2019

Source Code Link: `https://github.com/SarahFDAK/CS201.git`
This homework took approximately 10 hours to complete.

1 Design

This program took some trial and error, as well as researching errors I was receiving. The guidelines for the overall program were pretty clearly stated, so I just followed those in my overall design.

2 Post Mortem

I had a lot of issues with random errors that I had to look up and correct, as much of the syntax does not seem intuitive. Keeping each function simple is a challenge for me - I always want to overcomplicate them and make them do multiple things, so I have to keep that urge under control. This was challenging, but fun, and vectors seem like they will be exceedingly useful going forward. Also, maybe it's me overcomplicating again, but I feel like there were a number of topics in this homework that are not covered by lessons in class or in the book.

3 Answers to Questions

- A char is generally 1 byte (8 bits), an int is 4 bytes, and a double tends to be 8 bytes.
- A definition is a statement that creates a new name for a variable and sets aside memory for it.
- Initialization is defining a variable and giving it its initial value. Assignment is changing the variable's value later in the program.
- A conversion is safe if you are converting a from a smaller memory variable to a larger one, and therefore will not lose data in the conversion; i.e. converting an int to a double. A conversion is unsafe if there is the danger of narrowing, as in the case of converting a double to an int. If there are decimals on the double, they will be discarded when it is converted and any future usage of the variable will be inaccurate.
- A computation produces some output (such as a result) from an input.
- Statements are (slightly) more complex language used to execute expressions. An expression provides data and operators (e.g. a = b), and a statement adds punctuation and sequence to let the computer know in what order to execute expressions adn what output to provide.
- A constant expression is a programmer-defined constant used to keep the code more concise and readable. Defining a constant called "pi" instead of repeatedly using a typed-out value (3.14... etc) keeps the code more readable and reduces the chance of typos.
- An int can be used with all mathematical operators, whereas a string only works with assignment (=) and concatenation (+).

- String variables can receive pretty much any input provided, where as ints can only take integers - no decimals, no letters, no blanks.
- That vector definition creates a vector of characters with 26 elements. At this point, it will be blank.

4 Sample Output

Listing 1: Sample Program Output

```
Please enter a name: Wayne
Please enter a name: Sarah
Please enter a name: Caroline
Please enter a name: Gina
Please enter a name: Kristy
Please enter a name: Carrie
Please enter a name: Duane
Please enter a name: Bill
Please enter a name: Aaron
Please enter a name: Angus
Please enter the name you're looking for: Bill

The name Bill is in the list.

Here are the names in the order you entered them:
Wayne
Sarah
Caroline
Gina
Kristy
Carrie
Duane
Bill
Aaron
Angus
```

```
Here are the names in alphabetical order:
Aaron
Angus
Bill
Caroline
Carrie
Duane
Gina
Kristy
Sarah
Wayne
Program ended with exit code: 0
```

5 My Program

```
//
//  names.cpp
//  Sarah Carter
//  9/13/19
//  This program will build a vector containing ten names input by the user, then wi

#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

//Define prototypes
void InputNames(std::vector<std::string> & names);
bool DoesNameExist(const std::string & nameToFind, const std::vector<std::string> &
void PrintNames(const std::vector<std::string> & names);
void SortNames(std::vector<std::string> & names);

int main(int argc, const char * argv[]) {
    std::vector<std::string> names; //Define the vector "names."
    InputNames(names); //Call the InputNames function

    std::string nameToFind; //Define string variable
    std::cout << "Please enter the name you're looking for: ";
    std::cin >> nameToFind;
    std::cout << std::endl; //Add extra line for deliniation
    DoesNameExist(nameToFind, names); //Call function to see if name is in the vecto
    //Tell the user if the name is or isn't in the list.
    if(DoesNameExist(nameToFind, names)==true){
        std::cout << "The name " << nameToFind << " is in the list." << std::endl;
    }
    else{
        std::cout << "The name " << nameToFind << " is not in the list.\n";
```

```
    }
    std::cout << std::endl; //Add extra line for delination.
    //Print the list as the user entered it.
    std::cout << "Here are the names in the order you entered them: \n";
    PrintNames(names);
    std::cout << std::endl; //Add an extra line for deliniation.
    //Print the list in alphabetical order.
    std::cout << "Here are the names in alphabetical order: \n";
    SortNames(names);
    return 0;
}

//The InputNames function uses the names vector created in main() and requests names
void InputNames(std::vector<std::string> & names)
{
    for(int i = 0; i < 10; i++) //Runs statements 10 times
    {
        std::string name;
        std::cout << "Please enter a name: ";
        std::getline(std::cin, name);
        names.push_back(name); //Takes each entered name and pushes it to new elemen
    }
}
//Checks if a name entered by the user is in the created vector "names".
bool DoesNameExist(const std::string & nameToFind, const std::vector<std::string> &
{
    bool ans = false; //Sets the returned answer to a default of false.
    if(std::find(names.begin(), names.end(), nameToFind) != names.end()){
        ans=true; //If name is found, changes ans to true.
    }
    return ans;
}
//Print the list of names entered in the vector.
void PrintNames(const std::vector<std::string> & names){
    for(int i = 0; i < names.size(); i++){
        std::cout << names[i] << std::endl;
    }
}
//Sort the names into alphabetical order
void SortNames(std::vector<std::string> & names){
    std::sort(names.begin(),names.end());
    PrintNames(names); //Print the sorted names
}
```