CMPSC 121: Project 6

Jung-woo Sohn (jwsohn@ist.psu.edu)

November 9, 2017

1 Overview

In this project exercise, we will focus on the following two topics:

- Use of array and array indexing to access the data stored in the array
- Design of functions with input parameter(s) and/or a return value

On top of the previous Project files, specific implementations for this project are:

- Use array and associate index (from 0 to 3) to the corresponding city names.
- Modify `void navigate()` and `void investigate()` functions into `int navigate()` and `string investigate(int location)`. `navigate()` function returns the index of the city that the user selects and `investigate()` returns the investigation information as a string message. In addition, take out a part of `main()` function and make it into a function for simplicity.

An example output from a complete program will be similar to the following listing:

```
$ ./game
***** Welcome to WHERE IN THE WORLD IS CARMEN SANDIEGO *****

Your name, please: Foobar

Welcome, detective Foobar

You have 7 days and 12 hours to arrest a criminal.
You are currently in city index 0
Do you want to investigate? [y] y
She told me that she had always wanted to see Proms festival.
Please select your next destination:

1. New York
2. London
3. Cairo
4. Rio de Janeiro

Please enter your selection: 3

Your input is : 3
Your next destination is : Cairo
You are currently in city index 2
Do you want to investigate? [y] y
```

```
She checked out books related to pyramids and ancient mysteries.
Please select your next destination:

1. New York
2. London
3. Cairo
4. Rio de Janeiro

Please enter your selection: 1

Your input is : 1
Your next destination is : New York
You are currently in city index 0
Do you want to investigate? [y] y
She wanted to purchase a Portuguese dictionary.
Please select your next destination:

1. New York
2. London
3. Cairo
4. Rio de Janeiro

Please enter your selection: 4

Your input is : 4
Your next destination is : Rio de Janeiro
```

2 Instructions

1. Adding an array for city names

   - **Variable scope:** A question you might want to ask here is "To which extent do you need to make the city names (or variables in general) accessible?" A rule of thumb is that the variable scope should be small enough. In this project, we want to make `cities[]` array accessbile to `navigate()` and `investigate()` function but not to `main()` function.
   - Since `navigate()` and `invesigate()` are placed in `navigate.cpp`, we will make the variable scope of `cities[]` to the all functions inside `navigate.cpp`
   - To do this, you neeed to put the following array definition on top of `navigate.cpp`, below the `#include` directives and `using namespace` statements but above any function definitions.

     ```
     string cities[] = {"New York", "London", "Cairo", "Rio de Janeiro"};
     ```

2. Modification of `navigate()` function

   - **Return value:** Now `navigate()` will return an array index of the city that the user selects. For example, if the user select Cairo, `navigate()` will return an `int` value of 2. (Note that array index always starts from zero, not one!)
   - The function header is the place to specify the *type* of the return value. Change the header from `void navigate()` into `int navigate()`

- Notice that we do not need the `if - else if - else` statements any more to determine the city name corresponding to the number input by the user. When the user inputs a number, the corresponding city name can be obtained by accessing the array with the number. For example, if the user selects 2, the corresponding city name will be `cities[1]`. Again, note that the index is smaller by one since the array index starts from zero, not one as displayed in the menu printout.
- Take out all the `if - else if - else` statements.
- Change `destination` variable in the `cout` printout statements into `cities[menu - 1]`.
- The function header specification enforces return of an `int` value (City index). Put the return statment at the end of the function.

```
            return menu - 1;
```

3. Modification of `investigate()` function

   - **Input parameter and return value:** For this Project, `investigate()` function will take a city location as an `int` input parameter, and return the corresponding investigation information message as a `string` return value. For example, if the program passes 2, which is an index for `Cairo`, `investigate()` will return a corresponding message of "She checked out books related to pyramids and ancient mysteries."
   - Therefore, the function header will be as follows.

```
        string investigate(int criminalLocationIndex)
```

   - Similarly to associating index to the city names, we associate index to the corresponding invesitagion message as follows. For simplicity, a single sentence is assigned to a particular city.

```
    string investigation[] = {
        "She exchanged her money into dollars.",
        "She told me that she had always wanted to see Proms festival.",
        "She checked out books related to pyramids and ancient mysteries.",
        "She wanted to purchase a Portuguese dictionary." };
```

   - **Variable scope:** The `investigation[]` array variable does not need any access from outside of `investigate()` function. Thus, we will put the `investigation[]` array definition <u>inside</u> the `investigate()` function.
   - In the body of `investigate()` function, print out a message of "Do you want to investigate? [y] " and then get an keyboard input from the user.
   - If the user input `y` or `Y`, return the corresponding `investigation[]` message indexed by the input parameter of `int criminalLocationIndex`. If not, return an empty string of `""`.

4. Modification of `main()` function and `welcome.cpp`

   - In the `main()` function, we want to do a preliminary testing before we work on the detailed implementations of this hide-and-seek style gaming.
     - Repeat the "investigation - go to a new destination steps" for 3 times. Use of for loop is handy for this purpose.

  - In each loop. the investigation steps will include calling `investigate()` function and the "new destination" steps will include calling `navigate()` function.
  - Assume that the criminal (Carmen) will move from city index 1, 2, 3 for each loop.
  - When calling `investigate()` function, simply pass the current location of the criminal as the input parameter. In other words, we will not check whether the detective is correcly tracking the criminal.

- Since `investigate()` and `navigate()` functions are placed in a different file than `welcome.cpp`, there must be corresponding function prototypes defined in the header of `welcome.cpp`. Modify them accordingly.
- **Design of `welcome()` function** It is always a good idea to keep the length of an individual function short whenever possible. To keep `main()` function short and simple, we will take out the "welcoming" part of `main()` as a separate function.
  - The welcoming part can start from the beginning of printing the "Welcome to ..." message, include getting the name input, and end at displaying the remaining days and hours information.
  - The `welcome()` function does not need to have any input parameters, but we will design it such that it returns the name of the detective that the user enters. Therefore, the function header definion will be

    ```
    string welcome()
    ```

  - Most of the function body can be a simple copy-and-paste from the `main()` function. Note, however, that you need to return a string. Also use a correct synatax when calling `welcome()` from the `main()`.

3 Code templates

Refer to the following code templates for your information.

Listing 1: `navigate.cpp`

```
/* put proper #include directives and using namespace statements here */

/* definition for string cities[] goes here */

/* put proper function header for investigate() here */
{
    int menu = 0;

    /* print destination lists and get user input */

    cout << "Please select your next destination:" << endl;
    cout << endl;

    cout << "1. New York" << endl;
    cout << "2. London" << endl;
    cout << "3. Cairo" << endl;
    cout << "4. Rio de Janeiro" << endl;
    cout << endl;

    cout << "Please enter your selection: ";
```

```
    cin >> menu;

    while (menu < 1 || menu > 4)
    {
        cout << "You entered a wrong number." << endl;
        cout << "Please enter your selection: ";
        cin >> menu;
    }

    /* Printing out the menu and destination goes here. Take out the
       if - else - if else statements first. For destination print-outs,
       having cities[menu - 1] should be sufficient for now.

    Output example:

        Your input is : 2
        Your next destination is : Cairo
    */

    /* return the index for the destination city */
}

/* put proper function header for investigate() here */
{
    string investigation[] = {
        "She exchanged her money into dollars.",
        "She told me that she had always wanted to see Proms festival.",
        "She checked out books related to pyramids and ancient mysteries.",
        "She wanted to purchase a Portuguese dictionary." };

    /* Print out a message of "Do you want to investigate? [y] ".
       Then get a user input from keyboard here. */

    /* If the user enters 'y' or 'Y', return the corresponding
       string message of investigation[] with respect to the input parameter
       as the index. Otherwise, return an empty string. */
}
```

Listing 2: welcome.cpp

```
#include <iostream>
#include <string>

using namespace std;

/* put proper function prototypes here for the calling functions defined in
   navigate.cpp file */

/* put the proper function header for welcome() here */
{
    string name;
    int totalHours = 0, hoursLeft = 0, daysLeft = 0;

    /* print welcome screen. Get user name input from keyboard */
    cout << "***** Welcome to WHERE IN THE WORLD IS CARMEN SANDIEGO *****" << endl;
    cout << endl;

    cout << "Your name, please: ";
    cin >> name;

    /* print out the user name */

    cout << endl;
    cout << "Welcome, detective " << name << endl;
    cout << endl;

    /* print out time information */

    totalHours = 7 * 24 + 12;            // 7 and 1/2 days remaining
    daysLeft = totalHours / 24;
    hoursLeft = totalHours % 24;

    cout << "You have " << daysLeft << " days";
    cout << " and " << hoursLeft << " hours to arrest a criminal." << endl;

    /* put the proper return statement here so that welcom() returns the name
       of the detective as the return value */
}

int main()
{
    int cityIndex = 0;      // current location of the detective
    int criminalLocationIndex = 0;  // current location of the criminal (Carmen)

    string name = "";

    /* Call welcome() function here. Store the returned detective name. */


    /* Put a proper loop statement for repeating the following block 3 times */
    {
        ++criminalLocationIndex;      // For now, assume that Carmen moves from 1, 2, 3

        cout << "You are currently in city index " << cityIndex << endl;

        /* investigate on Carmen's destination */

        /* Call investigate() function here. Print out the returned
           investigation message associated with the criminal's location */


        /* select a destination and then update the location */

        /* Call navigate() function here. Update the current location
           of the detective with  the returned new city location index */
    }   // end of the loop

    return 0;
}
```