CMPSC 121: Project 8

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

November 9, 2017

1 Overview

As a continuation of project 7, we will implement additional game features using proper *function design*. The following features will be added to our game:

- Arresting the criminal[1]

1.1 Arresting the criminal

In order to arrest the criminal, we will use the following rules.

- Keep `arrestCounter` variable in `functions.cpp` and use it to determine if the detective captured the criminal or not.
  - Start with the `arrestCounter` set to a positive number. In our code, the default is set to 3.
  - Whenever the detective successfully select the same destination where the criminal is, decrement `arrestCounter` by one.
  - When `arrestCounter` reaches zero, print out a message that the detective captured the criminal and end the game
  - (For project 9) If `arrestCounter` is still postive but the remaining hours stored in `hours` variable has reached zero, print out a message that the detective failed to capture the criminal. End the game.
- In order to implement these features, design of the folllowing functions will be needed:

  In `functions.cpp`: (Should be ready if you have successfully completed Project 7)
  - `int getArrestCounter();`
  - `void decrementArrestCounter();`
  - `void resetArrestCounter(int);`

  In `navigate.cpp`:
  - `void arrest(string detectiveName);`
- Finally, modifications on `main()` function in `welcome.cpp` are needed.

An example output from the working program will be similar to the listing below:

[1] NOTE: Checking the remaining hours will be implemented in the next Project (Project 9). However, for the purpose of convenience, some routines in this project handles the remaining hours.

```
$ ./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 New York.

Do you want to investigate around? [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: 2

Your input is : 2
Your next destination is : London
You are currently in London.

Do you want to investigate around? [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: 4

Your input is : 4
Your next destination is : Rio de Janeriro
You are currently in Rio de Janeriro.

Do you want to investigate around? [y] y

Well, I can contact you if I happen to see any person like him... or her.

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 Cairo.

Do you want to investigate around? [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: 2

Your input is : 2
Your next destination is : London

****************************************
          There goes the criminal!
****************************************

Congratulations, Detecive Foobar!
You have successfully arrested the criminal.
We greatly appreciate your effort.

You have 7 days and 12 hours left.
```

2 Instructions

2.1 Function scoping: Low-level features vs. high-level features

First of all, make it sure that you implemented all the functions in `functions.cpp` as specified in the Listing 1 of `fheaders.h`. If not, go back to Project 7 and complete `functions.cpp` source code first.

One design feature that should be noted is that the functions defined in `functions.cpp` can be categorized as low-level functions; consider them as the simplest parts such as bolts and nuts that will be built up into a machine.

Similarly, you con consider the functions defined in `navigate.cpp`, or outside of `functions.cpp`, as high-level functions that utilize basic, low-level parts from `functions.cpp` and implement useful features. In other words, consider them as parts such as engines, brakes, exhaust, and so on that are built on top of nuts and bolts.

Also note that the important variables for gameplay are *hidden* inside `functions.cpp`. When you want to access them, you need to rely on `get...` or `set...` functions.

To summarize, you need to consider *what* functions to put *where*. For low-level features, use `functions.cpp`. For high-level features, use `cpp` files other than `functions.cpp`. Be sure to think in terms of scope — both for variables and functions.

Listing 1: Function prototypes defined in `fheaders.h`

```
/* collection of function header prototypes. By including this fheader.h. the
   code in the cpp file will have access to all the functions specified here.
   The details of the functions are in functions.cpp file.
 */

using namespace std;

string getCityName(int);
int nextRandomCityIndex(int currentLocation);

int getCurrentLocation();
void setCurrentLocation(int);

int getCriminalLocation();
void setCriminalLocation(int);
void setNextCriminalLocationRandom();

int getArrestCounter();
void decrementArrestCounter();
void resetArrestCounter(int);

int getHours();
void setHours(int);
```

2.2 Implementation of `arrest()` function in `navigate.cpp`

The function prototype for `arrest()` is:

- `void arrest(string detectiveName);`

As seen in the prototype, `arrest()` takes the name of the detective as the input parameter.

- In case *the current location of the detective is the same as current location of the criminal,* print out “Congratulations!” message that the detective successfully arrested the criminal. Be sure to print out the name of the detective as well.
- If not, print out a message that the detective failed to arrest the criminal. Print out the name of the detective as well.

Use the following code template for your information.

Listing 2: `arrest()` in `navigate.cpp`

```
void arrest(string detectiveName)
{
    if ( (/* FILL IN THE BLANK with proper condition */ && getHours() > 0 )
    {
        cout << endl;
        cout << "****************************************" << endl;
        cout << "          There goes the criminal!         " << endl;
        cout << "****************************************" << endl;
        cout << endl;

        cout << "Congratulations, Detecive " << /* FILL IN THE BLANK */ << "!" << endl;
        cout << "You have successfully arrested the criminal." << endl;
        cout << "We greatly appreciate your effort." << endl;
        cout << endl;

    }
    else
    {
        cout << endl;
        cout << "########################################" << endl;
        cout << "          There goes the criminal?         " << endl;
        cout << "########################################" << endl;
        cout << endl;

        cout << "Alas, Detective " << /* FILL IN THE BLANK */ << "..." << endl;
        cout << "It is a failure. The criminal already left the city." << endl;
        cout << endl;
    }


    cout << "You have " << /* FILL IN THE BLANK */ << " days and ";
    cout << /* FILL IN THE BLANK */ << " hours left." << endl;
}
```

2.3 Modifications of `investigate()` function in `navigate.cpp`

Now `investigate()` function needs to check if `location` is equals to `getCriminalLocation()`, i.e. if detective is at the same location as criminal’s. If true, it prints out the clue message from `investigation[]`. If not, it prints out “I have not seen anything” type messages from `notSeen[]`. Note that `location` contains the number index of the corresponding city.

Listing 3: `investigate()` in `navigate.cpp`

```
string investigate(int location)
{
    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." };

    string notSeen[] = {
            "Sorry but I have not seen anybody close to your description.",
            "I might have seen the person but it was a long time ago. I do not remember anything.",
            "Nobody. You'd better visit somewhere else.",
            "Well, I can contact you if I happen to see any person like him... or her." };

    if (/* FILL IN THE BLANK */)    // if detective's location is the same as criminal's
    {
        setNextCriminalLocationRandom();
            // move the criminal to a different city using a random number index

        return investigation[getCriminalLocation()];
    }
    else
    {
        return notSeen[location];
    }
}
```

2.4 Modifications of `main()` function in `welcome.cpp`

The following features need implementations in `main()`

- Replace the `for` loop with a `while` loop with a proper loop exit condition. In the previous project, we simply repeated four loops and let the detective chase the criminal. Now, we will let the detective chase the criminal *until* the detective selects the correct destination for `arrestCounter` times.
- Thus, whenever the detective correctly selects the destination where the criminal is, decrement `arrestCounter` by one. This can be done by calling `decrementArrestCounter()` function with proper condition checking with `if`.
- Finally, when the detective exits out of the `while` loop (by reaching zero for `arrestCounter`), call `arrest()` function and display proper message dependent on whether the detective captures the criminal or not.[2]

Please refer to the following code template for more information.

Listing 4: Modified `welcome.cpp`

```
/*  Project 8: Description goes here
    Programmer: Your name (abc123@psu.edu)
```

[2]The failure case will be yet to be implemented in the later project since we do not check the hours left for the detective.

```
    Date: 01/01/2015
    (Optional) Assistance with cout statements provided by Rob the tutor
*/

/* FILL IN THE BLNK: Use proper #include directives */

using namespace std;

/* FILL IN THE BLNK: Put additional function prototypes, especially for those
 *    defined in navigate.cpp file */

string welcome()
{

    /* Details of the welcome() function are left out for convenience */

    /*      .
            .
            .
            .
     */
}


int main()
{
    string name = welcome();

    const int CRIMINAL_COUNT = 3;

    /* repeat until the counter reaches zero */
    resetArrestCounter(CRIMINAL_COUNT);

    while (/* FILL IN THE BLANK: Put proper loop-exit condition here */)
    {
        char input;
        int cityIndex;

        cityIndex = getCurrentLocation();

        cout << "You are currently in " << getCityName(cityIndex) << "." << endl;
        cout << endl;

        /* investigate on Carmen's destination */
        cout << "Do you want to investigate around? [y] ";
        cin >> input;

        if (input == 'y' || input == 'Y')
        {
            cout << endl;
            cout << investigate(/* FILL IN THE BLANK */) << endl;
            cout << endl;
        }
```

```
        else
        {
            cout << endl;
            cout << "Okay, skipping investigations." << endl;
            cout << endl;
        }

        /* select a destination and then update the location */
        cityIndex = /* FILL IN THE BLANK: which funcion to call? */

        setCurrentLocation(cityIndex);      // update with the new location

        /* In case correctly tracking the criminal, decrement the arrest counter */
        if (/* FILL IN THE BLANK: put a proper condition */)
        {
            /* FILL IN THE BLANK */
        }
    }

    /* check the condition for capturing Carmen. Result in arrest or lost */

    /* FILL IN THE BLANK: Call arrest() function properly here */

    return 0;
}
```