CSCI2120 Introduction to Software Engineering
Assignment 2 – Number Guessing Game
Deadline: 31st March, 2013 23:59

Objective

The objective of this assignment is to practice writing object-oriented programs in C++ using inheritance.

Task

In this assignment, your task is to write a game to allow two players (human vs. computer or human vs. human) to guess a secret number that is in between 1 and 100 inclusively in text user interface. Your program should have the following steps.

Step 1: Display Game Menu

When your program starts, it first displays a menu with the game title and asks the player to press '1' for "Player vs. Computer", '2' for "Player vs. Player", or '3' for "Exit". All other inputs are ignored. For example, if the user presses 'q' or '4', nothing will happen. (Please use **_getch()** to get the input, which will not be echoed on the screen. Your program has to include **<conio.h>** to use this function.) An example of using **_getch()** is shown as follows:

```
char c;
c = _getch();
if(c == 'a')
    cout << "You entered the letter a" << endl;
```

After receiving a valid input, clear the screen by using **system("cls")**. (Your program has to include **<stdlib.h>** to use this function.) If the player presses '1' or '2', go to Step 2. If the player presses '3', exit the program.

Step 2: Ask for Player Name(s)

Your program has to ask the human player(s) to input the name(s), which may contain space characters. (Hint: use **getline(cin, str)** to read the input, where **str** is a string variable. You have to type **#include<iostream>** and **#include<string>** in the beginning of the code. If you use **getline(…)**, please avoid using "**cin >>**" in your program as they are incompatible with each other.)

If the player chose "Player vs. Computer", Player 1 will be a human player and Player 2 will be a computer player, whose name is "Computer". The name of Player 1 must contain at least one character and the name cannot be "Computer" (case sensitive). If the name is invalid, ask the player to input the name again. After receiving a valid name for Player 1, clear the screen and go to Step 3.

If the player chose "Player vs. Player", both Player 1 and Player 2 are human players. The names of both Player 1 and Player 2 must contain at least one character and Player 2 cannot have the same name (case sensitive) as Player 1 (in this case, the program only needs to ask Player 2 to input the name again). In this mode, the name "Computer" can be used. If the name is invalid, ask the player to input the name again. After receiving a valid name for Player 1, clear the screen and then ask Player 2 for input. After receiving a valid name for Player 2, clear the screen and go to Step 3.

Step 3: Start the Game

For each game, your program first randomly generates a secret number that is an integer between 1 and 100 inclusively. The two players (human or computer) take turns to guess the secret number and Player 1 is the first one to make a guess.

Before a player makes a guess, a hint will be given to indicate the range where the secret number is in. For example, in the beginning of the game, Player 1 will be given a hint that the range is in between 1 to 100 inclusively. Your program should convert the input of the user (in **`string`**) into an integer (in **`int`**) by using the **`atoi(...)`** function, which will return zero if the conversion is not successful. In some systems, you may have to type **`#include<stdlib.h>`** in the beginning of the code. An example of using this function is shown as follows:

```
string inputInStr;    // store the user input in string
int inputInInt;       // store the user input in integer

// get the user input and store it in inputInStr
getline(cin, inputInStr);

// try to convert inputInStr into an integer and store it in inputInInt
// if atoi(...) cannot convert inputInStr into an integer, it will return 0
inputInInt = atoi(inputInStr.c_str());
if(inputInInt == 0)
    cout << "Invalid input" << endl;
else
    cout << "Your input is " << inputInInt << endl;
```

If the input of the user cannot be converted to an integer, print an appropriate message and ask the player to input again. If the input of the user can be converted into an integer, check whether the integer is within the range. If not, print an appropriate message and ask the player to input again. If the input of the user is an integer within the range, check whether the guess is correct. If the guess is correct, go to Step 4. If the guess is not correct, display an appropriate message and wait for the player to press any key to proceed. Even if it is the computer's turn, the human player still needs to press any key to proceed to make sure that he/she can see the message. You can simply call **`_getch()`** for such a purpose. After the player pressed any key, clear the screen by using **`system("cls")`** and ask the next player to make a guess with the updated hint. For example, suppose the secret number is 32, and in the middle of the game, the

hint is “30 to 50”. If a player guesses 40, the updated hint to the next player is “30 to 39”. In this example, it is possible that the hint is “32 to 32” in one of the later rounds.

Step 4: End the Game

Your program should print an appropriate message to show the name of the winner. After that, ask the player whether he/she wants to play another round by pressing ‘y’ or ‘n’. (Please use **`_getch()`**). If the answer is ‘y’, go to Step 3. If the answer is ‘n’, go to Step 1. All other inputs are ignored.

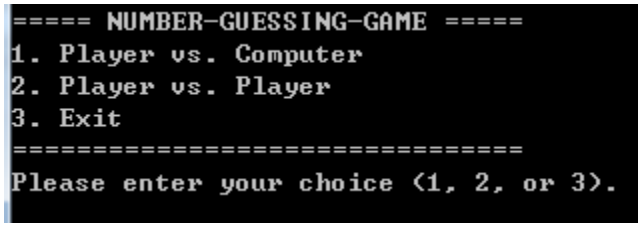
AI of Computer Player

The guess of the computer player is a random number within the range in the hint. (Hint: please note that **`rand() % 0`** will cause an error.)

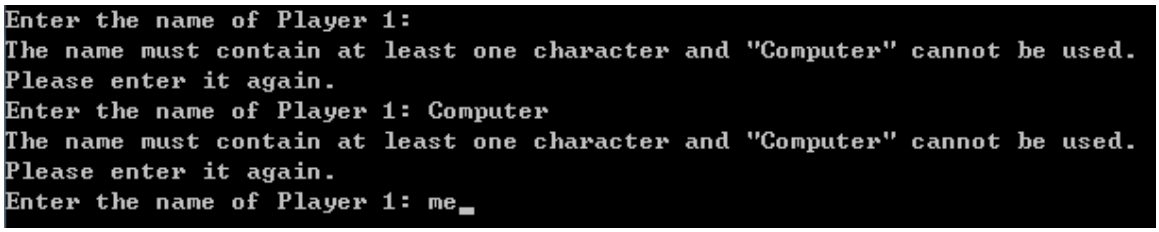
Screen Shots

The followings are some screen shots for your reference.

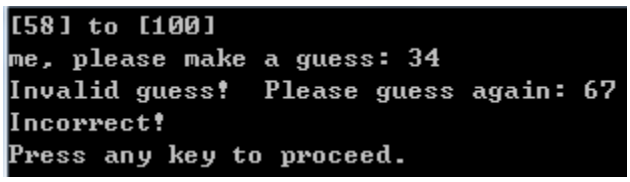
1. Menu:



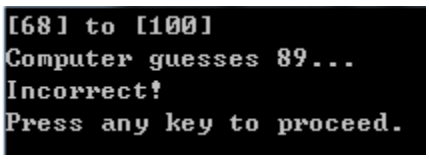
2. Input player’s name:



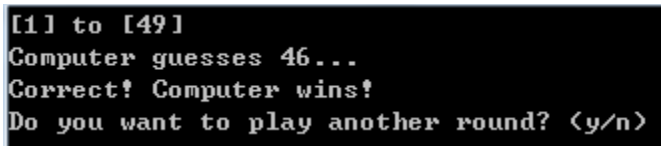
3. Give some hints, get a valid guess from the player and then give some messages:



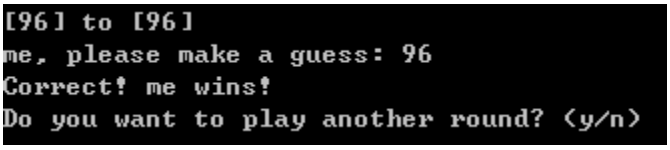
4. After the Computer player’s turn, give some messages:



5. If the guess of computer player is correct, show the winner:



6. If the guess of human player is correct, show the winner:



Class Design

Your program should have five classes, namely the **`NumberGuessingPanel`** class, the **`Player`** class, the **`HumanPlayer`** class, the **`ComputerPlayer`** class, and the **`NumberGuessingGame`** class and one **`main()`** function.

The `NumberGuessingPanel` Class

It should have at least three **private data members** to record the following data:

1. the secret number (integer)
2. the upper bound of the range in the hint (integer)
3. the lower bound of the range in the hint (integer)

It should have at least four **public member functions** to perform the following actions:

1. `void reset()`
   - reset the secret number and the range in the hint
2. `int getUpperBound()`
   - get the upper bound
3. `int getLowerBound()`
   - get the lower bound
4. `int guess(int input)`
   - allow a player to make a guess specified in the **`input`** parameter
   - the return value should indicate one of the following cases:
     i. the guess is invalid (i.e. out of the range)
     ii. the guess is valid but incorrect
     iii. the guess is valid and correct

The `Player` Class

It is the super class of the **`HumanPlayer`** and the **`ComputerPlayer`** classes. It should have at least one **protected data member** to record the following data (a protected data member can be accessed by the subclasses of this class)

1. the name of the player (string)

It should have at least two **public member functions** to perform the following actions:

1. `string getName()`
    - get the name of the player
2. `virtual int guess(NumberGuessingPanel *panel)`
    - this function will be overridden in the **`HumanPlayer`** and the **`ComputerPlayer`** subclasses and should not be invoked directly

The `HumanPlayer` Class

It is a subclass of the **`Player`** class. It should have one **constructor**:

1. `HumanPlayer(string name)`
    - set the name of the player to be the **`name`** parameter

It should have at least one **public member function** to perform the following action:

1. `int guess(NumberGuessingPanel *panel)`
    - override a function in the super class **`Player`**
    - allow a human player to input a guess through console until the input is valid (i.e., can be converted into a number and within the range)
    - the return value should indicate one of the following cases:
        i. the guess is incorrect
        ii. the guess is correct

The `ComputerPlayer` Class

It is a subclass of the **`Player`** class. It should have one **constructor** to do the following:

1. `ComputerPlayer()`
    - set the name of the player to be "Computer"

It should have at least one **public member function** to perform the following action:

1. `int guess(NumberGuessingPanel *panel)`
    - override a function in the super class **`Player`**
    - make a guess by generating a random number within the range in the hint
    - the return value should indicate one of the following cases:
        i. the guess is incorrect
        ii. the guess is correct

The `NumberGuessingGame` Class

It should have at least four **private data members** to record the following data:

1. a pointer to the first player (a pointer to a **`Player`** object)
    - it will not be changed once the game starts
2. a pointer to the second player (a pointer to a **`Player`** object)
    - it will not be changed once the game starts
3. a pointer to the current player (a pointer to a **`Player`** object)
    - it will be changed when switching players during the game
4. a pointer to the number guessing panel (a pointer to a **`NumberGuessingPanel`** object)

- it will not be changed once the game starts

It should have one **constructor** to do the following:

1. `NumberGuessingGame(Player *player1, Player *player2)`
    - create the **`NumberGuessingPanel`** object
    - initialize all data members

It should have at least one **public member function** to perform the following actions:

1. `void start()`
    - reset the number guessing panel
    - ask the current player to make a guess
    - switch the current player
    - display the winner

The main() Function

The **`main()`** function should perform the following:

- initialize the random number generator by the current time (i.e., **`srand((unsigned int)time(NULL))`**, you have to type **`#include<time.h>`** in the beginning of the code.)
- display the game menu
- create the **`Player`** objects
- create the **`NumberGuessingGame`** object and ask it to start the game
- ask if the player wants to play another round

The marks in "Class Design" (see Assessment Scheme) depend on how you implement the above design. If you can strictly follow the above design and implement it correctly, you can get full mark in "Class Design". If you wish, you may use your own design. However, your design will be marked according to object-oriented software engineering principles, such as information hiding and encapsulation.

Assessment Scheme

Correctness: 70%
Class Design: 20%
Program readability (e.g. indentation, comments, meaningful variable names, etc.): 10%

Please note that your TA will NOT help to debug your program. If your program fails to compile, you will get a very low score. Please also be reminded that your TA will do duplicate checking carefully. Consequences of confirmed plagiarism cases may be failing of the course and receiving a demerit.

Testing Platform

Your program will be tested in the following platform:

- Windows 7
- Visual Studio 2010

Submission

Please follow the following steps to submit your work.

Step 1: If you only have one .cpp source file, please save your source file as <your_student_id>_*assg2.cpp*. (e.g. 1155012345_assg2.cpp). If you have more than one source files, please put them into a compressed file (zip, rar, or 7z) and name the compressed file as <your_student_id>_*assg2.zip*. (or .rar or .7z) (e.g. 1155012345_assg2.zip)

Step 2: Go to CU eLearning (http://elearn.cuhk.edu.hk/) and login.

Step 3: Go to "Assignment Submission" category.

Step 4: Upload your file prepared in Step 1.

Resubmissions are allowed. But only the latest one will be counted. Late submissions within three days can only receive 70% of the marks. Late submissions more than three days will not be graded.