Отчет по 6 лабораторной работе

Постановка задачи: разработать консольное приложение на языке программирования C++, реализующее игру «Быки и коровы». Необходимо реализовать данное приложение с помощью системы классов.

Описание программной реализации:

В моей программе присутствуют три класса: *Computer, Player, Game*. В классе Game в качестве приватных полей (private) присутствуют объекты классов Computer и Player (возможность использовать функционал данных классов в классе Game). Функционал класса Game используется в функции main. Также есть отдельная функция **ent**, используемая только в функции main. Функция **main** реализует пользовательский интерфейс, предоставляя игроку выбор режимов

*В своей программе я использую строки (т.е. генерируемые компьютером числа и вводимые пользователем числа в программе это не тип int, а тип std::string).

Разберем каждый класс по отдельности. Начнем с класса **Computer**

Данный класс не имеет полей.

Присутствует два приватных метода: *proverka_duplicate, min*.

Метод **proverka_duplicate** возвращает тип bool: true – если новый сгенерированный символ совпадает с символом строки (у числа, которое генерирует компьютер, не должно быть повторяющихся цифр), иначе false (т.е. назначение: метод проверяет, содержится ли символ «c» в строке «s»). Сложность алгоритма O(n). Данный метод используется в публичном методе *generation_number*. В качестве параметров принимает константную ссылку на строку (const std::string&) и символ (тип char). Является константным методом.

Метод **min**. Возвращает тип int, принимает два параметра типа int. Использует тернарный оператор и возвращает наименьшее из двух чисел (используется в публичном методе *result*). Является константным методом.

Переходим к публичным методам. Их также два: *generation_number, result*.

Метод **generation_number** возвращает строковый тип (std::string), принимает один параметр типа int, является константным. Используется для генерации числа, заданной длины (принимаемый параметр), с неповторяющимися цифрами (используется метод proverka_duplicate). Для генерации используется библиотека random. Создаются отдельные цифры типа int, которые затем переводятся в символы (тип char). Присутствуют две проверки: первая цифра не ноль, все цифры различны (метод proverka_duplicate). Используется цикл do…while, который работает, пока число не будет нужной длины. Сложность алгоритма O(n) (в худшем случае O(n^2)). Возвращает готовую строку.

Метод **result** возвращает тип int, принимает два параметра: константные ссылки на строки. Является константным методом. Определяет количество коров и быков (сколько цифр угадано без совпадения с их позициями, сколько угадано цифр с совпадением позиций). Имеется одна проверка: совпадение длин чисел. Имеются две переменные count_cow (количество коров), count_bull (количество быков), два массива count_s1, count_s2 (количество элементов в массиве 10 (для цифр от 0 до 9), изначально заполнены нулями, s1 – для первой строки, s2 – для второй). С помощью цикла for проходимся по строкам: если и позиция, и символ совпадают, то увеличиваем количество быков; иначе идет заполнение элементов массивов (пример: строка s1 = “5641”, s2 =”5467”. Первый символ совпал, увеличили количество быков. Далее: 6 не равна 4, значит count_s1[6]=1, count_s2[4]=1. И так далее). Затем в следующем цикле for проходимся по всем элементам массивов. Каждый из элементов принимает значение: либо 0, либо 1. Далее приведу пример. Пример: s1=”5641”, s2=”5467”.

Count_s1: 0100101000

Count_s2: 0000101100

Выбираем минимальный элемент на i-ой позиции. В s1 есть символ 1, а в s2 нету. Значит в count_cow добавляем 0. 4, 6 есть и в s1, и в s2. Значит в count_cow добавляем две единицы. «7» нет в s1, значит добавляем 0. Итог: 1 бык, 2 коровы. Если количество быков равно длине числа, то выводим сообщение: “You win!!!” Иначе выводим количество коров и быков. Сложность алгоритма O(n). Возвращаемые значения нужны в методах класса Game.

Переходим к классу **Player**.

У данного класса также нет полей.

Приватные методы: *proverka_duplicate, proverka_number, to_number*.

Метод **proverka_duplicate**. Возвращает тип bool, принимает константную ссылку на строку, является константным методом. С помощью двух вложенных циклов for проверяет есть ли в строке повторяющиеся символы. Если есть возвращает true, иначе false. Сложность алгоритма O(n^2).

Метод **proverka_number**. Возвращает тип int, принимает константную ссылку на строку, является константным методом. С помощью цикла for проверяет, что каждый символ является цифрой («0123456789»). Если это не так, выводит сообщение об ошибке, возвращает значение 1. Иначе 0. Сложность алгоритма O(n).

Метод **to_number**. Возвращает тип int, принимает константную ссылку на строку, является константным методом. Переводит строку в число. Используется цикл for. Сложность алгоритма O(n). Возвращает полученное число.

Публичные методы: *choose_lenght, enter_number*.

Метод **choose_lenght**. Возвращает тип int, параметров не принимает. Является константным методом. Данный метод используется для того, чтобы пользователь выбрал длину числа. Используется цикл do…while (пользователь продолжает вводить, пока не введет корректное значение), метод proverka_number, метод to_number. Длина числа не может быть равной 0 и 1 (см пункт «**Крайние случаи. Выбор длины числа**»). Сложность алгоритма O(n). Возвращает длину строки.

Метод **enter_number**. Возвращает строковый тип, принимает тип int. Является константным методом. Используется для ввода пользователем числа, выбранной до этого длины. Используется цикл do…while (аналогично методу choose_lenght). Имеются проверки: совпадение длины числа, все символы – это цифры (метод proverka_number), число не может начинаться с 0, все цифры различны (метод proverka_duplicate). Сложность алгоритма O(n^2). Возвращает полученную строку.

Класс **Game**.

Приватные поля: объект класса Computer, объект класса Player; строка str_man, строка str_pc, длина строк (тип int).

Конструктор: изначально строки инициализируются как пустые, длина строк инициализируется 0.

Публичные методы: *enter_lenght, gen_number, attempt, Result*.

Метод **enter_lenght**. Использует метод класса Player: choose_lenght. Присваивает полученное значение полю **len** (длина строк). Сложность O(n).

Метод **gen_number**. Использует метод класса Computer: generation_number. Присваивает полученную строку полю str_pc. Сложность O(n) (в худшем случае O (n^2)).

Метод **attempt**. Использует метод класса Player: enter_number. Присваивает полученную строку полю str_man. Сложность O(n^2).

Метод **Result**. Использует метод класса Computer: result. Возвращает либо 0, либо 1 (как и метод result). Если 0 – значит пользователь угадал, иначе игра продолжается.

Функция **ent**. Возвращает строковый тип, принимает две константные ссылки на строки. Используется для выбора режима (два варианта). Используется цикл do_while (пользователь вводит, пока не введет корректное значение). Возвращает выбранное значение. Используется в main.

Основная функция **main**.

Создается объект класса Game. Через данный объект вызываются методы класса Game.

Пользователю необходимо выбрать режим: 1 – начать игру, 2 – завершить игру (выйти из игры). Для этого используется функция **ent**. Далее идет блок switch-case.

Игрок выбрал режим 1 (case 1:):

1. Пользователь выбирает длину числа (метод enter_lenght);
2. Вызывается метод для генерации числа, в соответствии с выбранной длиной;
3. Заходим в цикл do…while, который работает, пока пользователь не угадает число, или пока сам не решит завершить игру (используется метка flag2);
4. Пользователь выбирает режим: 1 – сделать попытку угадать число, 2 – завершить игру (снова функция ent). Внутренний switch-case:
    - Пользователь выбрал 1: вызывается метод attempt, метке flag2 присваивается значение метода Result (если 0, то цикл завершиться, иначе выйдет сообщение с количеством быков и коров). Если пользователь не угадал, то ему вновь предлагается выбрать режим (пользователь либо продолжает угадывать, либо завершает игру).
    - Пользователь выбрал 2: метке flag2 присваивается значение 0, происходит выход из цикла, и игра завершается.

Игрок выбрал режим 2 (case 2:): игра завершается.

Пример работы программы (вводятся корректные значения).

- «Please select the mode: 1 – start the game, 2 – end the game» (Пожалуйста выберите режим: 1 – начать игру, 2 – завершить игру/выйти из игры)
- Пользователь вводит 1
- «Choose the length of the number. The minimum number length is 2: » (Выберите длину числа. Минимальная длина числа равна 2)
- Пользователь вводит 3 (длина числа равна 3)
- «Stroka (pc): 591» (данную строку пользователь не должен видеть, но она нужна для проверки корректности работы программы. Компьютер сгенерировал число 591. Метод класса Computer: generation_number)
- «Please select the mode: 1 – an attempt, 2- end the game» (Пожалуйста выберите режим: 1 – сделать попытку, 2 – завершить игру)
- Пользователь вводит 1
- «Enter the number with non-repeating digits» (the length of the number is 3) (Введите число с неповторяющимися цифрами (длина числа равна 3))
- Пользователь вводит 519
- «Count cow = 2» (количество коров равно 2. Две совпавшие цифры не на своих местах: 1 и 9)

«Count bull = 1» (количество быков равно 1. Совпавшая цифра на своей позиции: 5)

- «Please select the mode: 1 – an attempt, 2- end the game»
- Пользователь вводит 1
- «Enter the number with non-repeating digits» (the length of the number is 3)
- Пользователь вводит 591
- «You win!!!» (Пользователь угадал число. Игра завершена)
  В случае если пользователь выбирает режим «2», то выходит сообщение «Thanks for playing!», и игра завершается.

Крайние случаи (попытка ввода некорректных значений).
Основные крайние случаи:

- **Выбор режима**: если пользователь попытается ввести некорректное значение, т.е. что-то отличное от «1», «2» (например «12», «h1j», «1hj», «2gh» и т.п.), то выходит сообщение: «Error please enter the correct value»

- **Выбор длины числа.**
  Минимальная длина числа равна 2, т.к. если допустить длину числа равной 1, то количество «коров» всегда будет равной 0 (пользователь просто вводит значения от 0 до 9). По этой причине я установил данное ограничение.

  Если пользователь вводит значение «1», то выходит сообщение: «Error. It is not interesting to play with the length of a number equal to 1! » (Ошибка. Не интересно играть с длиной числа равной 1!)

  Если пользователь вводит значение, в котором присутствуют не только цифры (например «2hjl», «23 45» - присутствует символ пробела, и т.п.), то выходит сообщение: «Error. Please enter the correct value (A space is also an incorrect value) »
- **Ввод числа (попытка угадать число)**
  Пользователь вводит числа, некорректной длины. Сообщение: «Incorrect number length» (Некорректная длина числа)

  Пользователь вводит число, в котором есть повторяющиеся цифры (пример: 122). Сообщение: «Error. There are duplicate numbers» (Ошибка. Есть повторяющиеся цифры)

  В числе не должно быть посторонних символов. Сообщение: «Error. Please enter the correct value (A space is also an incorrect value) »

Если пользователь вводит число, начинающееся с 0 (пример: длина числа равна 3, пользователь вводит «013»), то выходит сообщение: «Error. A number cannot start with 0» (Ошибка. Число не может начинаться с 0).

Большаков Владислав ФИИТ-2