**Программная реализация «Игры “Города”»**

**Описание классов**

– CityList (список городов; класс имеет: одно поле cities, которое представляет из себя контейнер из библиотеки STL – list, и 12 методов),

– Player (игрок; класс имеет: одно поле name типа string и 5 методов),

– ComputerPlayer (компьютерный игрок; класс является наследником класса Player и CityList; класс имеет 4 метода),

– UserInterface (интерфейс пользователя; класс имеет 9 методов),

– Game (игра; класс является наследником класса UserInterface; класс имеет 7 полей: 5 полей имеют пользовательский тип данных (объекты созданных ранее классов: CP – компьютерный игрок 1, CP2 – компьютерный игрок 2, P1 – игрок 1, P2 – игрок 2, G – список городов) и 2 поля: hod типа int и GOver типа bool, и 12 методов).

**Описание методов каждого из классов**

**Класс «Города»**

1. CityList() – конструктор по умолчанию.
2. CityList(string file) – метод, заполняющий список городами из файла.
3. list<string>::iterator beginCities() – метод, возвращающий итератор, указывающий на начало списка.
4. list<string>::iterator endCities() – метод, возвращающий итератор, указывающий на конец списка.
5. size\_t citiesCount() – метод, возвращающий размер списка.
6. void addCity(string c) – метод добавления нового города и проверки введённого города на правильность.
7. bool isCityFirst(string c) – метод проверки, был ли город ранее назван (проверка на уникальность).
8. bool cityExists(CityList\*C, string c) – метод проверки, реальный ли добавляемый город (есть в списке возможных городов).
9. string retrieveLastCity() – метод выдачи последнего названного города.
10. void print() – метод вывода текущего состояния списка городов.
11. void clearCities() – метод, очищающий список.
12. ~CityList() – деструктор.

**Класс «Игрок»**

1. Player() – конструктор по умолчанию.
2. Player(string name) – конструктор инициализации.
3. string printName() – метод выдачи имени.
4. void writeName() – метод задания имени.
5. ~Player() – деструктор.

**Класс ««Компьютерный игрок»**

1. ComputerPlayer() – конструктор по умолчанию.
2. string choiceOne(CityList\*C) – метод, алгоритм выбора города (случайным образом из списка доступных городов), если первым ходит компьютер.
3. string choice(CityList\* C, CityList\* G) – метод, алгоритм выбора города (случайным образом из списка доступных городов), если первым ходит игрок.
4. ~ComputerPlayer() – деструктор.

**Класс «Интерфейс пользователя»**

1. UserInterface() – конструктор по умолчанию.
2. static void printRules() – метод, отображающий инструкции к игре на экране.
3. static void printCommands() – метод, отображающий информацию о доступных консольных командах на экране.
4. static int printMenu() – метод, отображающий меню на экране и осуществляющий его работу.
5. static void printHod(int k) – метод, отображающий информацию о текущем ходе на экране.
6. static void printGame(CityList\* G) – метод, отображающий список названных городов на экране.
7. static void inputName(Player\* P) – метод ввода имени от игрока.
8. static void printWin(Player\*P) – метод, отображающий информацию о победителе на экране.
9. ~UserInterface() – деструктор.

**Класс «Игра»**

1. Game() – конструктор по умолчанию.
2. void start(CityList\* C) – метод начала игры.
3. void CPvsP1(CityList\* C) – метод раунда компьютер против игрока.
4. void P1vsCP(CityList\* C) – метод раунда игрок против компьютера.
5. void P1vsP2(CityList\* C) – метод раунда игрок против игрока.
6. void CPvsCP2(CityList\* C) – метод раунда компьютер против компьютера.
7. bool GameOver() const – метод завершения игры.
8. char getNextCharacter() – метод, получения следующей буквы из последнего названного города.
9. bool validateCity(CityList\* C, string city) – метод проверки правильности названного города текущим игроком.
10. bool validateFirstCity(CityList\* C, string city) – метод проверки правильности названного города текущим игроком, если пользователь ходит первым.
11. void clearing() – метод очистки игры.
12. ~Game() – деструктор.

**Инструкция, объясняющая, как использовать приложение**

1. Первым элементом цепочки может быть любой город.
2. Каждый следующий начинается с последней буквы предыдущего города (если название заканчивается на ь, ъ, ы - то берётся предпоследняя буква).
3. В цепочке не должно быть двух одинаковых городов.
4. Ввод городов производится с маленькой буквы.
5. Для вывода списка названных городов напишите «/Города»
6. Для окончания игры напишите «/Стоп».