Санкт-Петербургский Государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

# Курсовая работа по курсу «Программирование»

# Тема работы: «Разработка приложения с графическим интерфейсом на языке программирования С++ с использованием библиотеки QT»

Студент: Баратынский Александр

Преподаватель: Пышкин Евгений Валерьевич

2012 г.

# Образ и границы проекта

В качестве курсовой работы по программированию предлагается реализация игры «Виселица». Приложение должно быть реализовано на языке программирования C++ с использованием библиотеки QT. Также, данное приложение должно быть с графическим интерфейсом.

Основные моменты игры:

Суть игры состоит в том, чтобы отгадать предоставленное слово за определенное количество попыток. Графический интерфейс должен выглядеть следующим образом: в верхней части экрана должен появиться набор символов, определяющий количество букв в слове, снизу же должен быть набор кнопок с буквами на них. Пользователь должен нажимать на букву, которую выбрал, она будет зачеркиваться, и, если он ее угадал, то буква должна появиться сверху, если же нет, то справа появится виселица. По мере прохождения, если пользователь плохо отгадывает буквы, на виселице вырисовывается человечек. Победа в игре присуждается если пользователь угадал слово, поражение – если на виселице полностью вырисовался человечек.

Также, я рассматриваю возможность сделать игру на нескольких языках.

# Техническое задание

Реализовать игру, описанную в первом пункте, на языке программирования С++ с использованием библиотеки QT.

2 этапа разработки игры:

1. Реализация проекта в консольном режиме
2. Последующая интеграция кода с библиотекой QT. Реализация графического интерфейса.

Требования к проекту:

Главное меню должно содержать следующие пункты:

1. Начать новую игру
2. Продолжить игру ( Если пользователь не закончил предыдущую игру)
3. Статистика игр
4. Смена языка
5. Выход из игры

Сценарии выполнения программы:

1. Начать новую игру

При нажатии на эту кнопку выводится меню, где игроку предлагается выбрать одну из предложенных тем (страны, спорт, культура и т.д. и т.п.). После выбора темы, подгружается соответствующий файл. Из этого файла случайным образом выбирается слово. Далее начинается игра. После окончания игры, результаты заносятся в файл для статистики (скорее всего XML файл) или, если игра не была закончена, то текущее состояние игры сохранится в другой XMLфайл и, если пользователь пожелает, следующая игра начнется с этого же момента.

1. Продолжить игру

При нажатии на эту кнопку продолжится прошлый сеанс.

1. Статистика игры

При нажатии на эту кнопку на экран выводится процентное соотношение выигранных и проигранных игр.

1. Выбор языка

Здесь можно выбрать язык игры.

1. Выход из игры

# Требования к консольному режиму

1. Выбор языка
2. На экран должна выводиться последовательность \*, показывающая количество букв в слове. По мере отгадывания, буквы появляются вместо звездочек.
3. Должно выводиться число, показывающее число оставшихся попыток.

# Требования к графическому интерфейсу

1. Пункты меню в виде кнопок
2. Поле игры.
3. Вместо числа попыток, должна отображаться виселица
4. Остальные пункты идентичны.

# Системная архитектура

## Классы консольного приложения

В консольном приложении будут реализованы следующие классы:

1. class BasicWord

{

protected:

char letter;

string word;

public:

BasicWord();

char mistakes[6];

string getWord();

friend class Library;

};

Класс BasicWord отвечает за представление слова в целом.

1. class DictionaryWord : public BasicWord

{

public:

DictionaryWord();

void setWord(string dword);

};

Класс DictionaryWord – производный от BasicWord, отвечает за представление словарного слова.

1. class UserWord : public BasicWord

{

void enterLetter();

public:

UserWord();

void makeUserWord(DictionaryWord dword);

void putMistake(int curMisses);

bool checkLetter(DictionaryWord dword);

void putLetter(DictionaryWord dword);

};

Класс UserWord – производный от BasicWord, отвечает за представление конструируемого пользователем слова.

1. class Library

{

int randNum;

int var;

DictionaryWord \*word;

int numberOfWords(ifstream &in);

DictionaryWord\* takeWord(ifstream &in);

public:

Library();

~Library();

DictionaryWord\* takeWordOfLang();

int chooseTheme();

};

Класс Library отвечает за взятие слова из текстового файла.

1. class Man

{

void createMan();

void drawMan();

public:

char man[6];

int maxMisses;

int curMisses;

Man();

void makeMan();

};

Класс Man отвечает за конструирование человечка на виселице.

1. class Game

{

Man \*man;

Library \*lib;

DictionaryWord \*dword;

UserWord \*uword;

bool isWon();

bool isLose();

bool playAgain();

void reset();

~Game();

public:

Game();

int play();

};

Класс Game отвечает за сам игровой процесс.

class Test

{

DictionaryWord \*word;

UserWord \*uword;

Game \*game;

Man \*man;

DictionaryWord\* takeTheThirdWord();

bool test1();

bool test2();

bool test3();

bool test4();

bool test5();

bool test6();

public:

Test();

~Test();

bool fullTest();

};

Процесс тестирования приложения:

1. Метод test1() проверяет было ли взято то слово, которое планировалось взять.
2. Метод test2() проверяет работоспособность метода enterLetter() класса UserWord.
3. Метод test3() проверяет работоспособность метода checkLetter(DictionaryWord word) класса UserWord.
4. Метод test4() проверяет, вставилась ли угаданная буква на нужное место.
5. Метод test5() проверяет метод isWon() класса Game.
6. Метод test6() проверяет метод isLose() класса Game.

Функциональность консольного приложения

1. В данном консольном приложении реализовано взятие случайного слова из файла. Причем пользователь может выбрать язык слов.
2. Затем появляется изображение виселицы, составленное из определенных символов.
3. Ниже виселицы появляется маска слова.
4. Затем пользователю предлагается ввести букву, которая, по его мнению, должна быть в слове.
5. Если буква есть в слове, она появляется, если же нет – то на виселице появляются части тела человечка, реализованные в классе Man.
6. Игра заканчивается, когда заканчивается количество попыток угадать букву или же, когда пользователь отгадывает слово.
7. Далее пользователю предлагается начало новой игры.

## Классы графического приложения

На данный момент у меня недостаточно знаний, чтобы написать классы для графического интерфейса. Этот раздел будет дорабатываться по ходу изучения библиотеки QT.