

Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «Игра виселица»

ОТЧ.
28.12.22


Выполнил:

студент группы 21ВО2:

Седов С. С.

Принял:

Митрохин М.А.

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра «Вычислительная техника»

«УТВЕРЖДАЮ»

Зав. кафедрой ВТ

Митрохин М. А.

«___» _____ 2022 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Седову Сергею Сергеевичу

Группа 21BO2

Тема проекта: Разработка программы сложной структуры методом нисходящего программирования

Исходные данные (технические требования) на проектирование

Наименование программы: Компьютерная игра «Виселица»

Задание. Программа должна выполнять следующие функции:

- Наличие меню.
- Возможность выбирать уровень сложности.
- Наличие графического интерфейса.
- Загрузка ранее сохраненных результатов и их выведение в меню.
- Сохранение результатов в файл.

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных — списков, структур, файлов.
3. Режим работы видеосистемы — текстовый/графический.
4. Устройство ввода информации — мышь.
5. Пользовательский интерфейс должен быть построен на основе меню.
6. Наличие меню.
7. Операционная система — Windows.
8. Система программирования — C/C++ на базе среды Microsoft Visual Studio 2019.
9. Язык программирования — C++.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Структура модулей программы, схема алгоритма программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

1. В соответствии с графиком выполнения курсового проекта.

2. _____

3. _____

Дата выдачи задания «1» _____ октября 2022

Дата защиты проекта «28» _____ декабря 2022

Руководитель _____ Митрохин М. А.

Задание получил _____ «1» _____ октября 2022

Студент _____ Седов Сергей Сергеевич

Содержание

Введение	4
1 Постановка задачи.....	5
2 Выбор решения.....	6
2.1 Определение необходимых модулей программы	6
3 Описание разработки программы	6
4 Отладка и тестирование	11
5 Описание программы.....	12
5.1 Описание функций программы	14
6 Руководство пользователя.....	32
Заключение	37
Список используемых источников	38
Приложение А.....	39
Приложение Б	49

Введение

С развитием технологий компьютеры все больше и больше вливаются в повседневную жизнь человека. Персональные компьютеры - это уже не просто машины для математических вычислений, их общедоступность значительно расширила область применения. И, конечно же, здесь не обошлось без игр.

Индустрия игр быстро набирала обороты, стремительно развивалась, завоевывая все большую и большую популярность у подрастающего поколения. И если первые игры отличались простотой дизайна и логики, то современная компьютерная игра - очень сложная программа, дело рук большого коллектива разработчиков. Так или иначе, компьютерные игры занимают не последнее место в жизни многих современных детей.

Целью этой курсовой работы является создание игры Виселица, а также получение практических навыков написания и разработки приложений с использованием программной среды разработки Microsoft Visual Studio 2019.

1 Постановка задачи

Необходимо разработать игру «Виселица». В игре будут реализованы классические правила:

1. Основной целью игры является отгадывание полного слова.
2. Игрок должен отгадывать буквы в загаданном слове.
3. Если игрок назвал неправильную букву, начинает вырисовываться виселица.
4. Игрок имеет всего 7 попыток на отгадывание полного слова.
5. Если игрок потратил все попытки и не отгадал слово, он проигрывает.

Необходимо также реализовать меню, с помощью которого можно будет запустить игру, выбрать уровень сложности и увидеть результат предыдущей игры.

Необходимо реализовать минималистичный и удобный интерфейс, чтобы у пользователя не возникало затруднений при использовании программы.

Программа должна быть многомодульной. Это облегчит обнаружение и исправление ошибок при ее отладке и тестировании.

В программе также будут использоваться сложные типы данных. Устройствами управления игрой является компьютерная мышь. Все действия в игре должны осуществляться с помощью нажатия кнопок мыши.

2 Выбор решения

2.1 Определение необходимых модулей программы

Разработанная программа состоит из следующих модулей:

1. mainwindow.h
2. play_widget.h
3. mainwindow.cpp
4. play_widget.cpp
5. ui_mainwindow.h
6. ui_playwidget.h
7. main.cpp

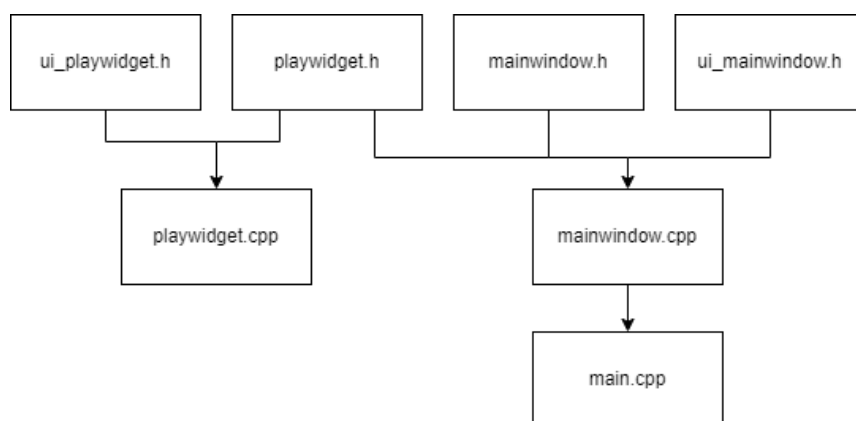


Рисунок 1 - Схема взаимодействия системы

playwidget.h, mainwindow.h – модули подключают все необходимые для работы программы библиотеки

playwidget.cpp – является основным модулем программы. Именно здесь происходит генерация процесса игры. В этом модуле осуществляется сохранение результатов игры.

mainwindow.cpp, main.cpp – В этих модулях происходит вывод главного меню, в котором выводится результат прошлой игры, выбор уровня сложности.

ui_mainwindow.h, ui_playwidget.h – В этих модулях содержатся функции для реализации графической составляющей игры.

3 Описание разработки программы

Для написания программы был использован язык программирования C++ и CSS, а также фреймворк Qt, с помощью которых очень удобно спроектировать графическое исполнение игры Виселица и обеспечить быстроедействие игры. В качестве среды разработки была использована программа Microsoft Visual Studio 2019.

Для начала было разработано меню игры «Виселица». Для ее отрисовки используются модули `ui_mainwindow.h` и `mainwindow.h`. Здесь пользователь может увидеть результат прошлой игры, а также выбрать уровень сложности игры.

При нажатии на элемент меню «Play», открывается окно выбора уровня сложности (от 1 до 3). После выбора уровня сложности, запускается сам процесс игры.

Проектирование игрового поля происходит с помощью методов из модуля `playwidget.h` и модуля `ui_playwidget.h`. С помощью фреймворка Qt и языка программирования CSS, мы выбираем оформление игры, а именно цвет и размер меню, клавиш, текста.

Функции меню прописаны в модуле `mainwindow.cpp`.

Основная логика игра написана в модуле `playwidget.cpp`.

В процессе игры игроку нужно отгадать слово, используя клавиатуру из игры. Нажатие на клавишу обрабатывается с помощью метода `connect` библиотеки Qt. В момент нажатия запускается функция `on_abc_clicked`, которая, в свою очередь, определяет есть ли нажатая буква в отгадываемом слове. Если игрок угадал букву, запустится функция `find_abc`, которая отобразит угаданную букву на форме, а клавиша на игровой клавиатуре закрасится и станет некликабельной. Если игрок не угадал букву, выбранная буква на игровой клавиатуре закрасится, станет некликабельной, а также уменьшится число попыток. Редактирование цвета и размера кнопок и текста происходит с помощью метода `SetStyleSheet` объектов Qt (например

QLabel, QPushButton). Параметры объекту задаются с помощью формального языка описания документов CSS.

Если игрок потратил все свои попытки и не отгадал слово, выведется форма с сообщением о том, что он проиграл, а также правильный ответ выведется на форму. Если слово было угадано, на форму выведется соответствующее сообщение. По окончании игры вызывается функция lose/win, которые записывают результат игры в файл, а именно: отгадываемое слово, дата и время игры, число потраченных попыток, результат игры(победа/поражение). Работа с файлами происходит при помощи методов библиотеки Qt. Также на форме есть кнопка “Exit”, с помощью которой игрок может закрыть игру.

Иерархическая структура программы представлена на рисунке 2.

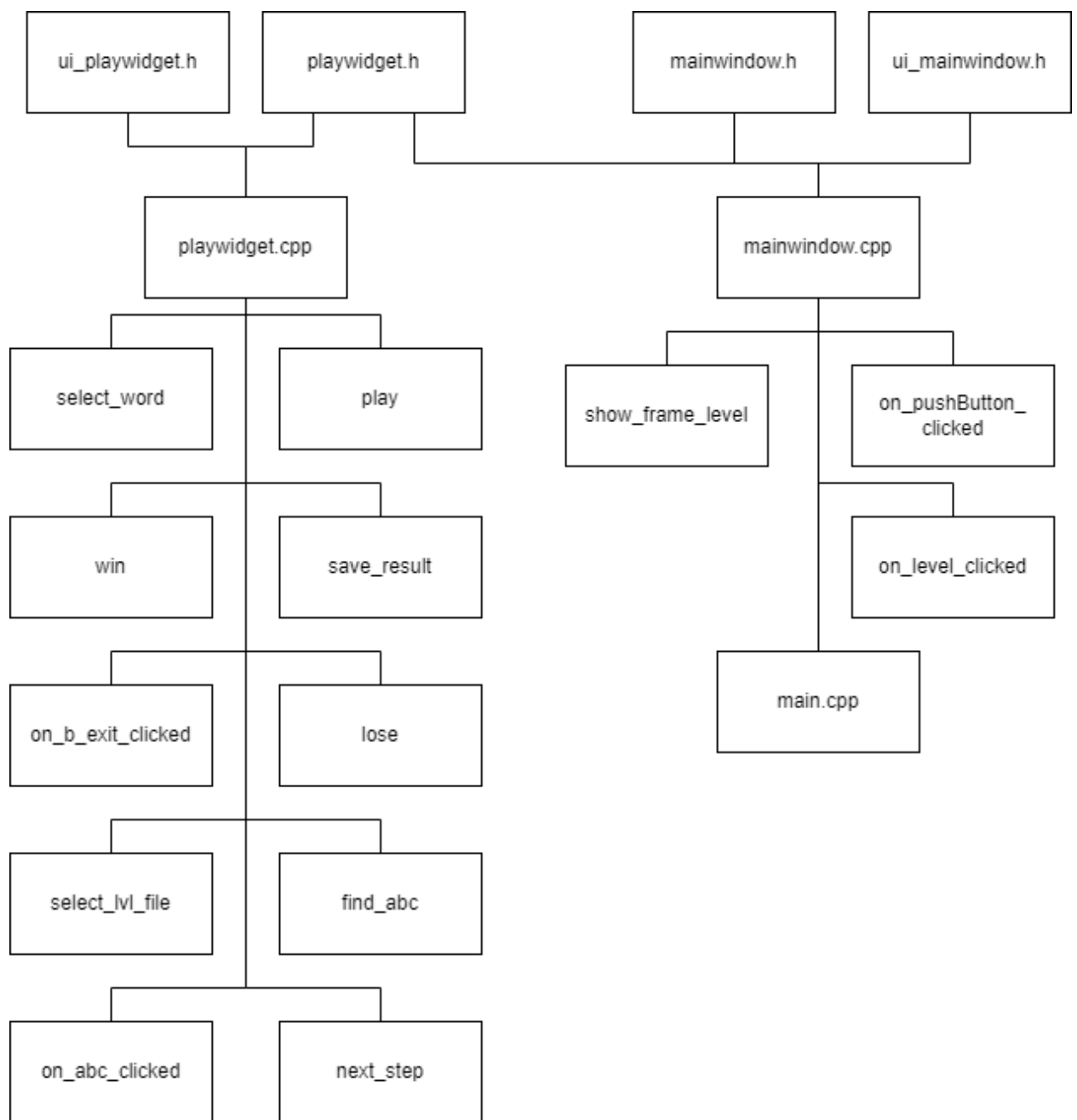


Рисунок 2 - Иерархическая структура программы

4 Отладка и тестирование

Согласно заданию, в качестве среды разработки была выбрана программа Microsoft Visual Studio 2019, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ. Для отладки программы использовались точки останова и пошаговое выполнение кода программы, анализ содержимого глобальных и локальных переменных, а также использование консоли и функции `qDebug` для проверки корректности работы той или иной функции.

В процессе отладки было выявлено и исправлено множество ошибок, например, некорректное открытие игры при нажатии кнопки «Play», проблемы, связанные с определением угаданной буквы, смены изображения, записью результатов в файл. Для решения проблем часто приходилось пользоваться различной документацией, в основном документаций к библиотеке Qt.

5 Описание программы

При запуске программы появляется меню, в котором указано название игры, результат предыдущей игры, а также имеет кнопку “Play”.

После нажатия на кнопку “Play” вызывается функция `on_pushButton_clicked`, которая выводит форму с выбором уровня сложности от 1 до 3.

После нажатия на кнопку уровня сложности вызывается функция `on_level_clicked`, после чего запускается сам игровой процесс.

Сначала программа, в зависимости от выбранного уровня сложности, открывает файл и берет оттуда случайное слово. Затем, в зависимости от количества букв в слове, создаются объекты `QLabel` на игровой форме. Каждому объекту достается одна буква от этого слова. В момент создания объекта ему сразу же присваиваются свойства, а именно: цвет текста, размер текста, цвет и размер самого объекта. Цвет текста внутри объекта становится таким же как и цвет самого объекта, чтобы буква отгадываемого слова сливалась с фоном этого объекта, и игрок не мог узнать содержимое объекта.

Далее начинается сам игровой процесс. Игроку необходимо нажимать на кнопки игровой клавиатуры для того, чтобы отгадать слово. При нажатии на кнопку игровой клавиатуры запускается функция `on_abc_clicked`. В ней происходит проверка является ли выбранная буква правильной (есть эта буква в отгадываемом слове или нет). Если игрок угадал, буква отобразится на форме (цвет буквы объекта перестанет сливаться с фоном), а сама кнопка станет темной и некликабельной. Если игрок не угадал букву, то на форме станет дорисовываться виселица, а сама кнопка, на которую нажал игрок, станет темной и некликабельной, также количество попыток уменьшится на 1.

Если у игрока кончатся попытки, а слово не будет отгадано, то запустится функция `lose`, которая отобразит ответ (отгадываемое слово) и выведет форму с сообщением о проигрыше, а также произойдет запись результатов игры в файл `result.txt` (отгадываемое слово, дата и время игры, количество потраченных попыток, результат игры: проиграл/выиграл).

Если игрок отгадал слово, запускается функция `win` и выведется форма с сообщением о победе. Также произойдет запись результатов игры в файл `result.txt`.

Игрок в любой момент может завершить игру, нажав на кнопку “Exit”, которая вызовет функцию `quit()` и закроет приложение.

Дизайн для игры устанавливается в модулях `ui_mainwindow.h` и

ui_playwidget.h, а именно цвет, размер, содержимое объектов и самой формы.

5.1 Описание функций программы

5.1.1 Модуль mainwindow.cpp

5.1.1.1 Функция on_pushButton_clicked

Данная функция нужна для того, чтобы, при нажатии на кнопку “Play”, появилось окно с выбором уровня сложности.

```
void MainWindow::on_pushButton_clicked()
{
    show_frame_level();
}
```

5.1.1.2 Функция on_level_clicked

Данная функция нужна для того, чтобы определять какой уровень сложности выбрал игрок. Сначала, с помощью метода `sender()`, программа получает информацию об объекте, который отправляет сигнал (в нашем случае нажатие на кнопку), и помещает её в переменную `button_select_level`. Далее запускается функция `select_lvl_file()` из класса `Play_widget` модуля `playwidget.cpp`, в которую, в качестве параметра, передается текст из переменной `button_select_level` при помощи метода `text()`. Затем, с помощью метода `show()`, появляется игровое окно, а текущее меню, с помощью метода `close()`, закрывается.

```
void MainWindow::on_level_clicked()
{

    button_select_level = (QPushButton *)sender();

    play_wid->select_lvl_file(button_select_level->text());
    play_wid -> show();
    this -> close();
}
```

5.1.2 Модуль playwidget.cpp

5.1.2.1 Функция select_lvl_file

Функция select_lvl_file принимает в качестве параметра значение lvl (уровень сложности, который выбрал игрок). Затем запускается функция select_word, в которую передается параметр lvl.

```
void Play_widget::select_lvl_file(QString lvl)
{
    if(lvl=="1")
    {
        Play_widget::select_word(lvl);
    }
    if(lvl=="2")
    {
        Play_widget::select_word(lvl);
    }
    if(lvl=="3")
    {
        Play_widget::select_word(lvl);
    }
}
```

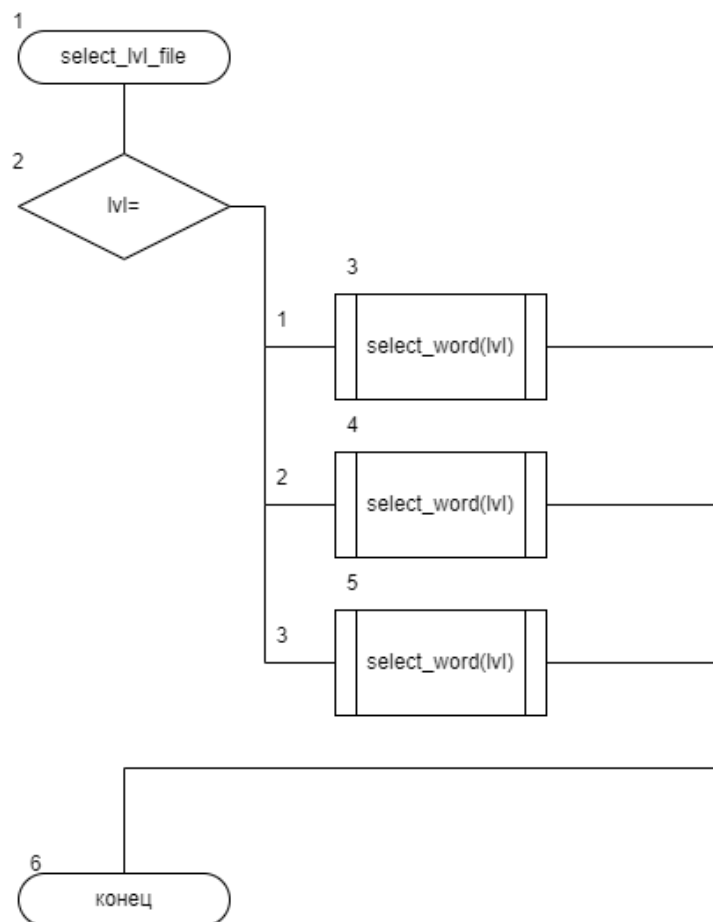


Рисунок 3 – блок-схема select_lvl_file

5.1.2.2 Функция on_b_exit_clicked

Данная функция вызывается при нажатии пользователем кнопки “Exit” на форме. Функция, с помощью метода quit(), закрывает приложение.

```

void Play_widget::on_b_exit_clicked()
{
    QApplication::quit();
}
  
```

5.1.2.3 Функция select_word

Функция в качестве параметра принимает значение lvl. Данная функция необходима для того, чтобы выбирать из файла случайное слово для игры в виселицу. Сам файл выбирается в зависимости от уровня сложности. Сначала в переменную path помещается путь до файла со словами для игры. Далее создается переменная file типа QFile. QFile нужен для ввода и вывода данных из файла. Затем создается переменная data типа QByteArray, которая будет хранить массив данных из файла. Далее происходит проверку на открытие файла на чтение, если файл не открыт, программа завершит свою работу. Затем в переменную stream передаются данные из data. Далее, с помощью цикла от 0 до случайного числа, которое не больше чем количество строчек в файле, в переменную result передается слово, которое нужно будет отгадать. После происходит закрытие файла и запуск функции play.

```
void Play_widget::select_word(QString lvl)
{
    srand((unsigned int)time(0));
    int n = 10;
    int random = rand()%n+1;
    QString path = "Resources/word_lvl/lvl_" + lvl + ".txt";
    QFile file(path);
    QByteArray data;
    if (!file.open(QIODevice::ReadOnly))
    {
        return;
    }
    QTextStream stream(&data);
    QString result;
    data = file.readAll();
    for (int i = 0; i < random; i++)
    {
        result = stream.readLine();
    }
}
```

```
}  
file.close();  
play(result); }
```



Рисунок 4 – блок-схема функции select_word

5.1.2.4 Функция play

Данная функция создает объекты QLabel и выводит их на форму. В качестве параметра передается значение result.

Сначала в переменную str передается значение result, затем создается массив cstr, размер которого равен длине str (количество букв в отгадываемом слове). Затем, с помощью цикла, происходит создание объекта QLabel, которому присваиваются следующие параметры: название объекта (для того, чтобы к этому объекту в дальнейшем можно было обратиться) – setObjectName(), стиль объекта (цвет, размер, цвет текста, размер текста) setStyleSheet(), минимальный и максимальный размеры setMaximumSize и setMinimumSize соответственно, расположение текста внутри объекта setAlignment, и, самое важное, задается текст (по одной букве из слова на каждый объект) setText(). Затем получившийся объект выводится на форму с помощью метода addWidget().

```
void Play_widget::play(QString result)
{
    str = result.toStdString();
    cstr = new char[str.size()];
    strcpy(cstr, str.c_str());
    label_word[strlen(cstr)];
    for(int i=0; i<strlen(cstr); i++)
    {
        label_word = new QLabel();
        label_word->setObjectName("label_letter_" +
std::to_string(i));
        label_word->setStyleSheet("QLabel { background-color: rgb(255,
255, 255); color: white; font-size: 18px; font-style: bold; border:
none;}");
        label_word->setMaximumSize(40,40);
        label_word->setMinimumSize(40,40);
        label_word->setAlignment(Qt::AlignCenter);
```

```
label_word->setText(QChar(cstr[i]));  
ui->horizontalLayout->addWidget(label_word);  
}  
}
```

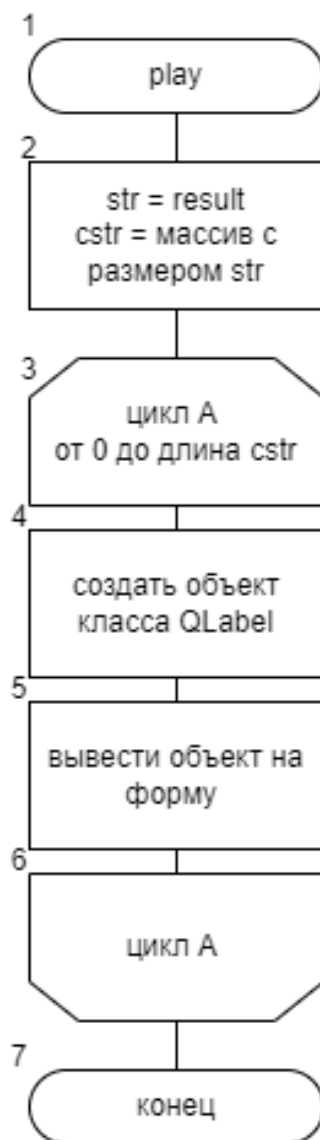


Рисунок 5 – блок-схема функции `play`

5.1.2.5 Функция on_abc_clicked

Данная функция обрабатывает сигнал, который отправляет объект при нажатие на него (в данном случае нажатие на кнопку игровой клавиатуры).

Сначала создается переменная `find` типа `bool`, которой присваивается значение `false`. Затем в переменную `button_abc` передается информация об объекте, который отправил сигнал. Затем, в цикле от 0 до длины `cstr` (длина отгадываемого слова), происходит проверка на совпадение текста из объекта, который испустил сигнал, с буквами из отгадываемого слова (`cstr`). Если условие выполняется, запускается функция `find_abc`, в которую передается параметр `i` (порядковый номер буквы внутри массива `cstr`), также переменная `find` принимает значение `true`. Иначе кнопка, на которую нажал пользователь, и, которая испустила сигнал, становится, с помощью метода `setDisabled`, некликабельной, а также, с помощью метода `setStyleSheet`, изменяется ее стиль (кнопка становится тёмной).

После цикла происходит проверка (`find = false?`). Если условие выполняется значение `num_of_attempts` (количество попыток) уменьшается на единицу, значения `num_of_score` (количество потраченных попыток) и `num_of_steps` (необходим для смены картинки) увеличиваются на единицу, а также запускается функция `next_step`. Далее, в этом же условии, происходит проверка на оставшиеся попытки, если их количество меньше или равно 0, то запускается функция `lose` и игра завершается.

```
void Play_widget::on_abc_clicked()
{
    bool find = false;
    QPushButton *button_abc = (QPushButton *)sender();

    for(int i=0; i<strlen(cstr); i++)
    {
        if(cstr[i]==button_abc->text())
        {
```

```

        find_abc(i);
    find = true;
    }
    else
    {
        button_abc->setDisabled(true);
        button_abc->setStyleSheet("QPushButton{ color: rgb(190,
190, 190); font-weight: 400; font-size: 20px; position: absolute;
width: 40px; height: 40px; background-color: rgb(100, 51, 51); border:
none; } ");
    }
}
if(find == false)
{
    num_of_attempts--;
    num_of_score++;
    num_of_steps++;
    next_step(num_of_steps);
    if(num_of_attempts<=0)
    {
        lose(strlen(cstr));
    }
}
}

```

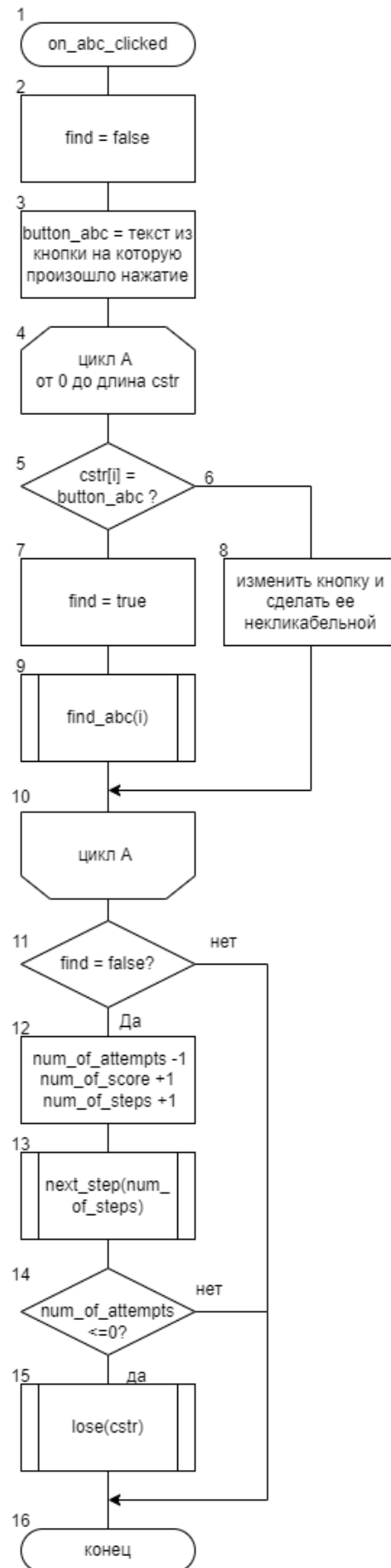


Рисунок 6 – блок-схема функции on_abc_clicked

5.1.2.6 Функция find_abc

Данная функция вызывается, если игрок угадал букву. Сначала в переменную `label_letter_name` передается название объекта, в котором находится угаданная буква. Затем в объект `label_letter_obj`, с помощью метода `findChild`, передается объект с именем `label_letter_name`. Далее происходит редактирование стиля объекта (найденная буква отображается на форме). Затем счетчик попыток `num_of_score` и счетчик отгаданных букв `score` увеличиваются на единицу. Далее происходит проверка: если счетчик отгаданных букв `score` равен количеству букв (длина `cstr`), то вызывается функция `win` и игра завершается.

```
void Play_widget::find_abc(int i)
{
    QString label_letter_name = QString("label_letter_%1").arg(i);
    QLabel *label_letter_obj = findChild<QLabel *>(label_letter_name);
    label_letter_obj->setStyleSheet("QLabel { background-color:
rgb(255, 255, 255); color: black; font-size: 18px; font-style: bold;
border: none;}");
    score++;
    num_of_score++;
    if(score == strlen(cstr))
    {
        win();
    }
}
```

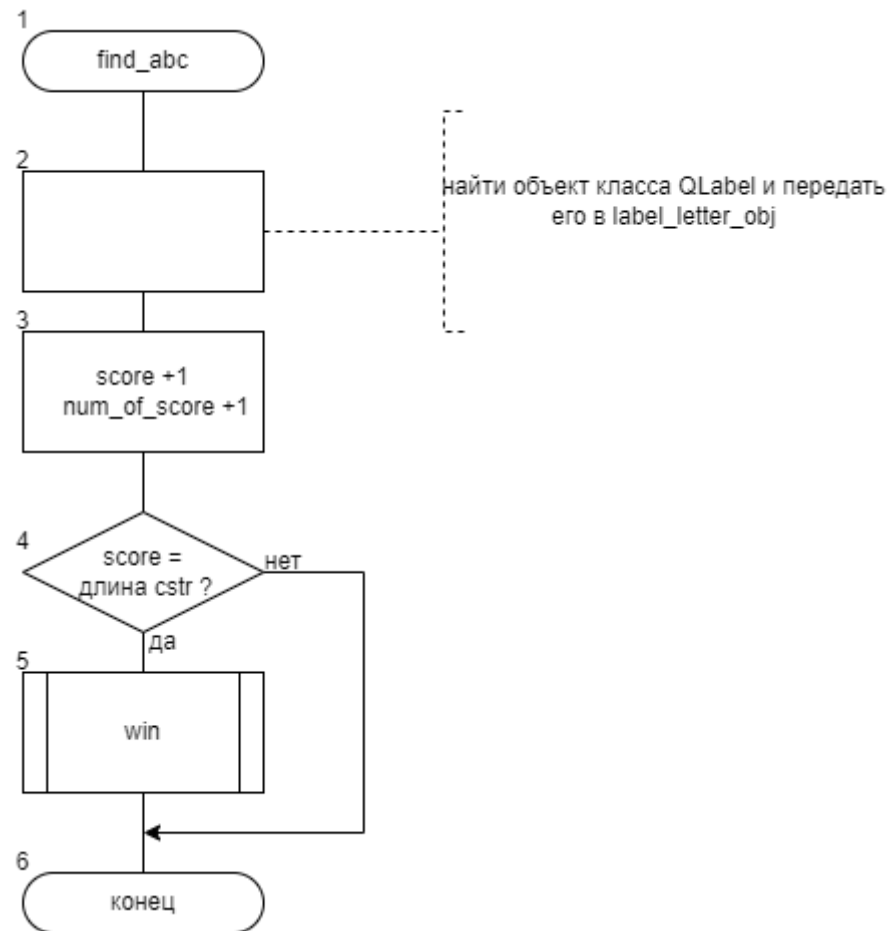


Рисунок 7 – блок-схема функции find_abc

5.1.2.7 Функция lose

Данная функция вызывается, если игрок потратил все попытки и не отгадал слово. В цикле от 0 до длины отгадываемого слова cstr происходит редактирование стиля всех объектов (буквы становятся видимыми), внутри которых расположены буквы отгадываемого слов. Далее запускается функция save_result, которая сохраняет результат игры, и выводится форма с информацией о проигрыше.

```

void Play_widget::lose(int cstr)
{
    game_result = "Lose";
    for(int i=0; i<cstr; i++)
    {
        QString label_letter_name = QString("label_letter_%1").arg(i);
    }
}
  
```

```

        QLabel *label_letter_obj = findChild<QLabel
*>(label_letter_name);
        label_letter_obj->setStyleSheet("QLabel { background-color:
rgb(255, 255, 255); color: black; font-size: 18px; font-style: bold;
border: none;}");
    }
    save_result(game_result);
    ui->key_frame->hide();
    ui->lose_frame->show();
}

```

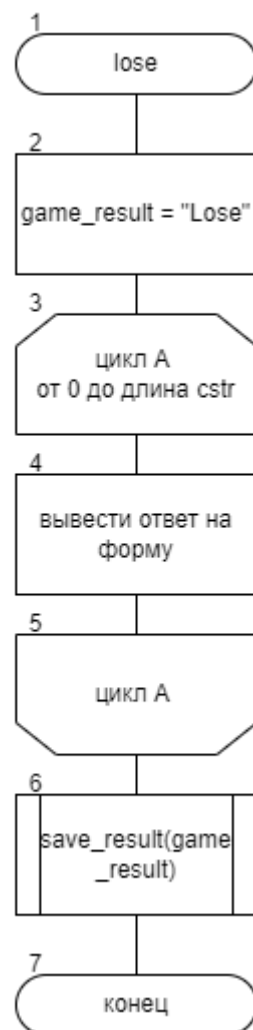


Рисунок 8 – блок-схема функции `lose`

5.1.2.7 Функция win

Данная функция вызывается, если игрок отгадал все буквы, не потратив все попытки. Сначала происходит сохранение результатов, с помощью функции `save_result`, а затем выводится форма с сообщением о победе.

```
void Play_widget::win()  
{  
    game_result = "Win";  
    save_result(game_result);  
    ui->win_frame->show();  
    ui->key_frame->hide();  
}
```

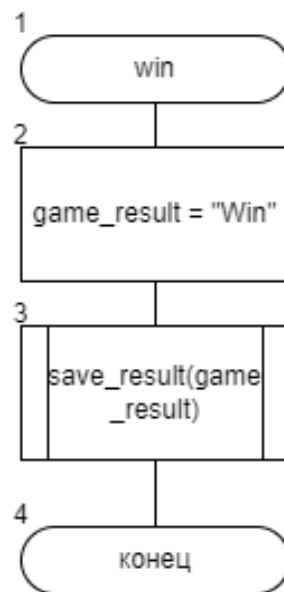


Рисунок 9 – блок-схема функции `win`

5.1.2.8 Функция save_result

Данная функция вызывается при проигрыше/победе игрока и нужна для записи результатов игры в файл. Сначала в переменную FILE_NAME передается путь до файла, в который происходит запись результатов игры. Далее файл открывается на запись QFile out(FILE_NAME). Затем создаются переменные result_try, result_word, result_time, result_result которым присваиваются значения, которые нужно сохранить. Затем, если проверка на открытие файла на запись выполняется, происходит непосредственно запись результатов. В конце файл закрывается с помощью метода close().

```
void Play_widget::save_result(QString game_result)
{
    static const char* const FILE_NAME =
"Resources/result/result.txt";
    QFile out( FILE_NAME );
    QString result_try = QString("Number of attempts:
%1\n").arg(num_of_score);
    QString result_word = QString("Guess word: %1\n").arg(cstr);
    QString result_time = QString("Date: " +
QDateTime::currentDateTime().toString("yyyy-MM-dd HH:mm:ss") + "\n");
    QString result_result = QString("Result: %1\n").arg(game_result);
    if( out.open( QIODevice::Append))
    {
        QTextStream stream( &out );
        stream << result_time;
        stream << result_word;
        stream << result_try;
        stream << result_result;
        stream << "\n\n";
        out.close();
    }
}
```



Рисунок 10 – блок-схема функции save_result

5.1.2.9 Функция save_result

Данная функция меняет изображение, если игрок ошибается. Сначала указывается путь до файла (изображение), затем это изображение выводится на форму. Какое изображение выведется зависит от количества ошибок игрока.

```
void Play_widget::next_step(int num_of_step)
{
    QString path_to_step =
QString("Resources/steps/step_%1.jpg").arg(num_of_step);
    ui->img_label->setPixmap(QPixmap(path_to_step));
}
```

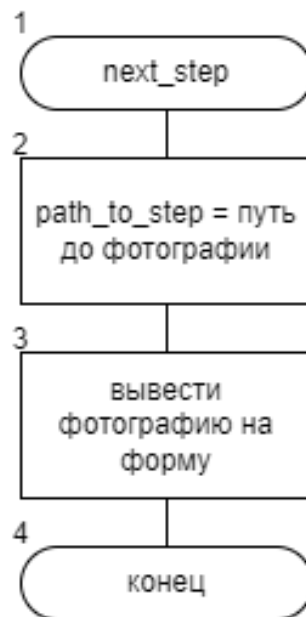


Рисунок 11 – блок-схема функции `next_step`

6 Руководство пользователя

При первом запуске игры появляется меню игры. Здесь указано название игры, результат предыдущей игры, а также кнопка “Play”.



Рисунок 12 – меню

Затем, после нажатия на кнопку “Play” игрок может выбрать уровень сложности от 1 до 3.

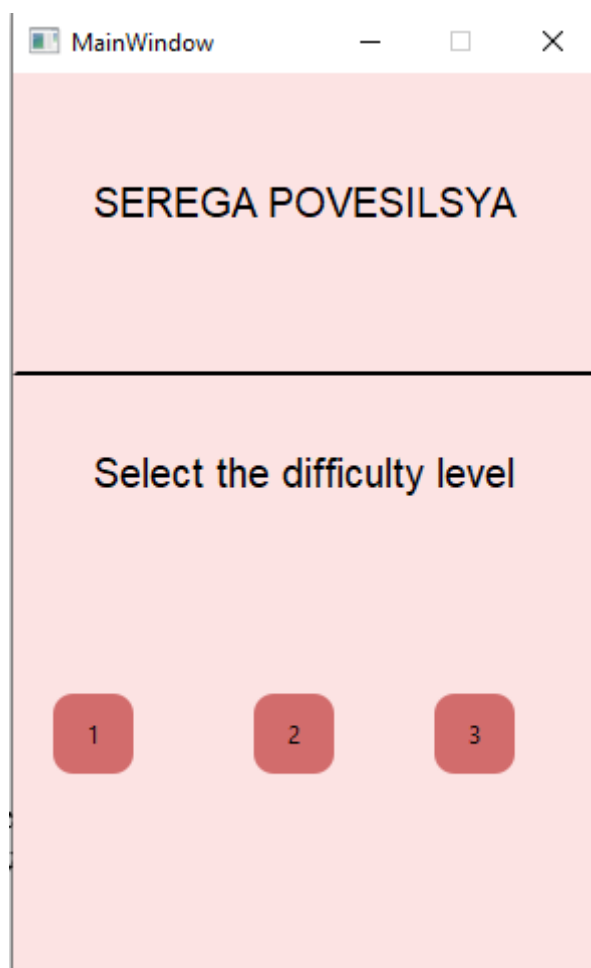


Рисунок 13 – выбор уровня сложности

После выбора уровня сложности запускается сам игровой процесс. Перед игроком на форме расположены: блок с виселицей, которая будет дорисовываться при нажатии неправильной буквы, блок с пустыми окнами (буквы от загаданного слова находятся внутри этих окон), блок с игровой клавиатурой, кнопка “Exit”. При нажатии на кнопку “Exit” игра закроется.

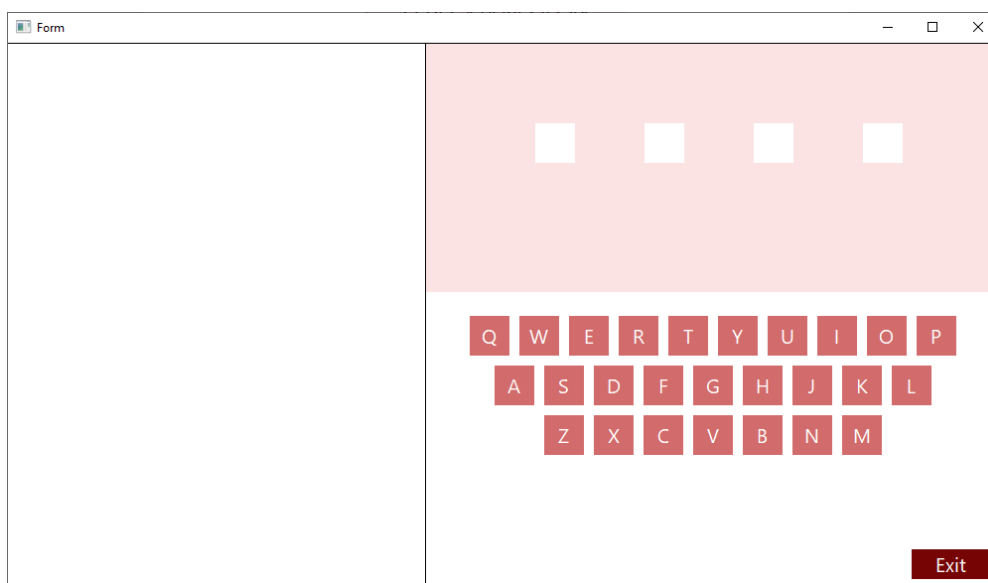


Рисунок 14 – игровой процесс

Затем, отгадывая слово, использованные буквы будут закрашиваться, и, если игрок нажал неправильную букву, будет вырисовываться виселица.

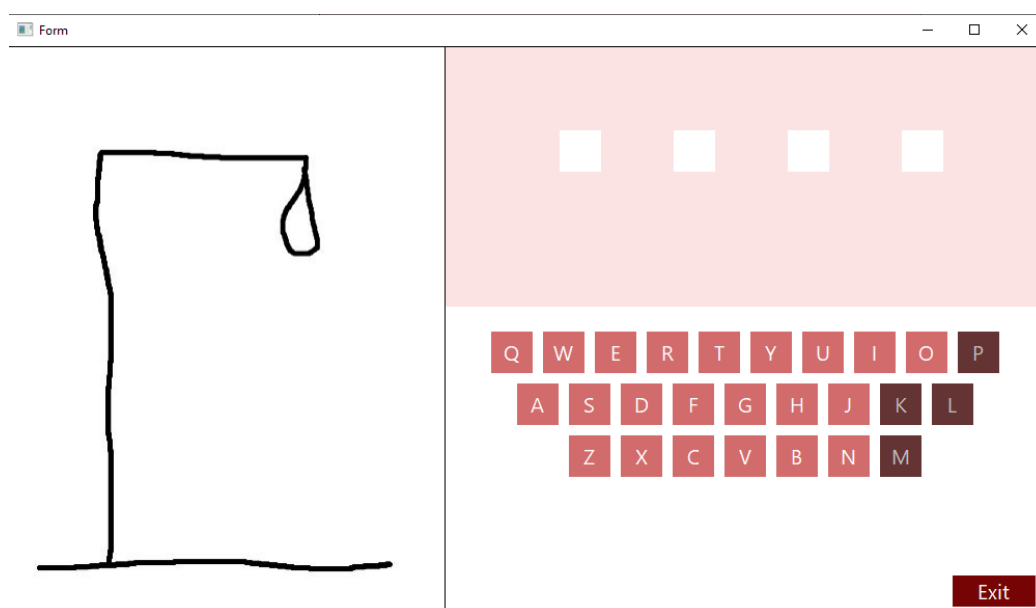


Рисунок 15 – дорисовывание виселицы, при нажатии неправильных букв

Если игрок потратит все свои попытки (всего их 7) и не отгадает слово, выведется форма с сообщением о проигрыше, а также отобразится отгадываемое слово.

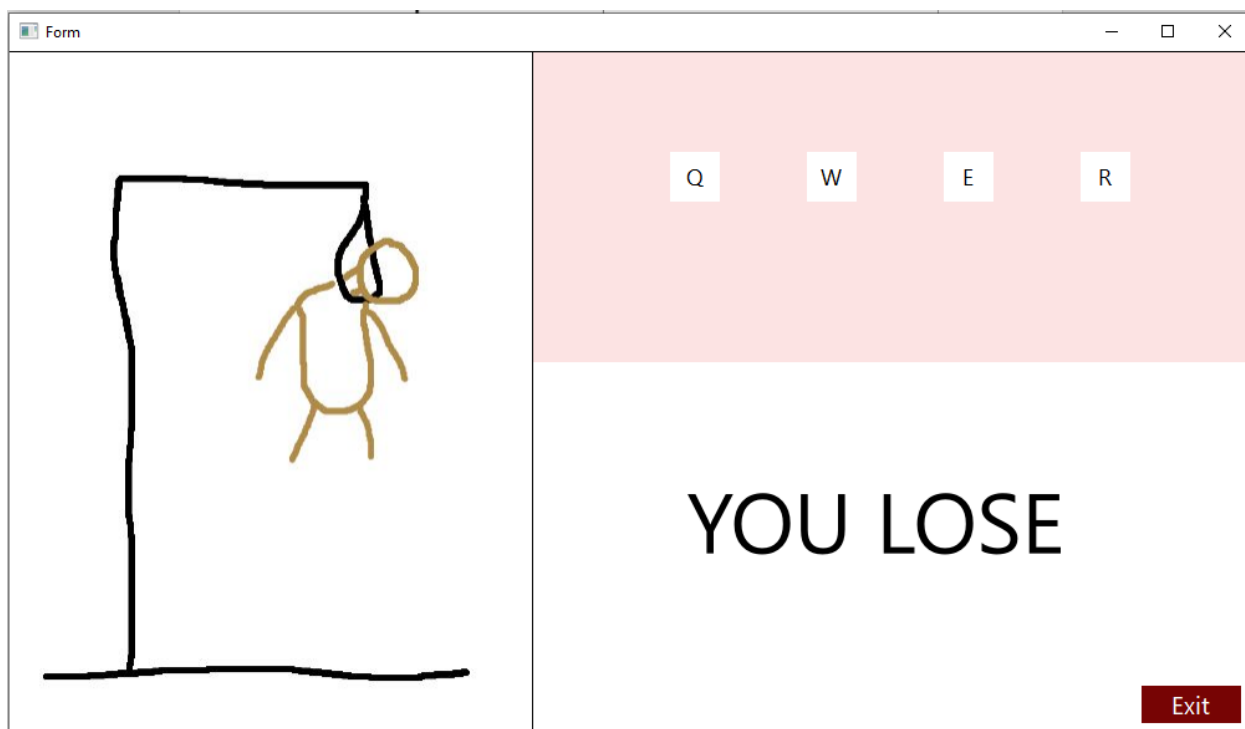


Рисунок 16 – проигрыш

Если игрок отгадал слово и не потратил все попытки, выведется форма с сообщением о победе.



Рисунок 17 – победа

Также в файле result.txt можно посмотреть результаты предыдущих игр.

```
1 Date: 2022-12-28 05:01:42
2 Guess word: ZXCX
3 Number of attempts: 7
4 Result: Win
5
6
7 Date: 2022-12-28 05:06:18
8 Guess word: ASDF
9 Number of attempts: 8
10 Result: Lose
11
12
13 Date: 2022-12-28 05:20:22
14 Guess word: QWER
15 Number of attempts: 7
16 Result: Lose
17
18
19 Date: 2022-12-28 05:21:54
20 Guess word: ASDF
21 Number of attempts: 5
22 Result: Win
23
24
25 Date: 2022-12-28 05:23:27
26 Guess word: PROJECT
27 Number of attempts: 9
28 Result: Lose
29
30
31
```

Рисунок 18 – файл result.txt

Заключение

При выполнении данной курсовой работы были улучшены базовые навыки программирования на языках C++, CSS, а также улучшена работа с фреймворком Qt.

Изучены основные возможности среды программирования Microsoft Visual Studio 2019 и Qt Creator. Получены навыки работы с Qt Creator, разработки интерфейса, отладки и тестирования программ. Были освоены приемы создания и подключения модулей программы. Получены навыки работы со сложными типами данных, такие как структуры и файлы.

В результате была разработана игра «Виселица», с приятным текстовым оформлением и простым для использования интерфейсом.

В дальнейшем игру можно улучшить, добавив больше функций и улучшив дизайн.

Список используемых источников

1. <https://github.com>
2. <https://wiki.qt.io/Main>
3. <https://ru.stackoverflow.com>
4. <https://habr.com/ru/all/>
5. <https://www.cyberforum.ru/qt/>

Приложение А

Основной код в модулях

Файл main.cpp

```
#include "mainwindow.h"
#include <QApplication>
#include <QLocale>
#include <QTranslator>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "test_qt_" + QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            a.installTranslator(&translator);
            break;
        }
    }
    MainWindow w;
    w.show();
    return a.exec();
}
```

Файл mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "play_widget.h"
#include <QFile>
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
```

```

{
    ui->setupUi(this);
    ui->level_frame->hide();
    play_wid = new Play_widget();
    connect(ui->play_lvl_1, SIGNAL(clicked()), this,
    SLOT(on_level_clicked()));
    connect(ui->play_lvl_2, SIGNAL(clicked()), this,
    SLOT(on_level_clicked()));
    connect(ui->play_lvl_3, SIGNAL(clicked()), this,
    SLOT(on_level_clicked()));
    QString last_result;
    QString path_res = "Resources/result/result.txt";
    QFile file_res(path_res);
    QByteArray data_res;
    if (!file_res.open(QIODevice::ReadOnly))
    {
        return;
    }
    QTextStream stream(&data_res);
    QString word = "1 1";
    data_res = file_res.readAll();
    while( !stream.atEnd())
    {
        word = stream.readLine();
        if(word == "Result: Lose" or word == "Result: Win")
        {
            last_result = word;
        }
    }
    ui->last_result->setText(last_result.section(":",1));
    file_res.close();
}

MainWindow::~MainWindow()
{

```

```

        delete ui;
    }

void MainWindow::show_frame_level()
{

    ui->skip_frame->hide();
    ui->level_frame->show();
}

void MainWindow::on_pushButton_clicked()
{
    show_frame_level();
}

void MainWindow::on_level_clicked()
{
    button_select_level = (QPushButton *)sender();
    play_wid->select_lvl_file(button_select_level->text());
    play_wid -> show();
    this -> close();
}

```

Файл playwidget.cpp

```

#include "play_widget.h"
#include "ui_play_widget.h"

#include <QDebug>
#include <QFile>
#include <QByteArray>
#include <QString>
#include <QLabel>
#include <string>
#include <fstream>
#include <time.h>

```

```
#include <QVector>
#include <QPoint>
#include <QTextStream>
#include <QDateTime>
```

```
int score = 0;
int num_of_attempts = 7;
int num_of_steps = 0;
int num_of_score = 0;
QString game_result = " ";
```

```
Play_widget::Play_widget(QWidget *parent) :
```

```
    QWidget(parent),
    ui(new Ui::Play_widget)
```

```
{
```

```
    ui->setupUi(this);
```

```
    ui->lose_frame->hide();
```

```
    ui->win_frame->hide();
```

```
    connect(ui->b_q, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_w, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_e, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_r, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_t, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_y, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_u, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_i, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_o, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_p, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_a, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
    connect(ui->b_s, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
```

```

connect(ui->b_d, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_f, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_g, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_h, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_j, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_k, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_l, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_z, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_x, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_c, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_v, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_b, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_n, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
connect(ui->b_m, SIGNAL(clicked()), this, SLOT(on_abc_clicked()));
ui->img_label->setPixmap(QPixmap("Resources/steps/step_0.jpg"));
}

```

```

Play_widget::~Play_widget()
{
    delete ui;
}

void Play_widget::select_lvl_file(QString lvl)
{
    if(lvl=="1")
    {
        Play_widget::select_word(lvl);
    }
    if(lvl=="2")
    {
        Play_widget::select_word(lvl);
    }
    if(lvl=="3")
    {
        Play_widget::select_word(lvl);
    }
}

```

```

    }
}

void Play_widget::on_b_exit_clicked()
{
    QApplication::quit();
}

void Play_widget::select_word(QString lvl)
{
    srand((unsigned int)time(0));
    int n = 10; //кол во строк в файле
    int random = rand()%n+1;
    QString path = "Resources/word_lvl/lvl_" + lvl + ".txt";
    QFile file(path);
    // создаем объект класса QFile
    QByteArray data;
    // Создаем объект класса QByteArray, куда мы будем считывать данные
    if (!file.open(QIODevice::ReadOnly))
    {
        // Проверяем, возможно ли открыть наш файл для чтения
        return;
    }
    // если это сделать невозможно, то завершаем функцию
    QTextStream stream(&data);
    QString result;
    data = file.readAll();
    //считываем все данные с файла в объект data
    for (int i = 0; i < random; i++)
    {
        result = stream.readLine();
    }
    file.close();
    play(result);
}

```

```
}
```

```
void Play_widget::play(QString result)
```

```
{
```

```
    str = result.toStdString();
```

```
    cstr = new char[str.size()];
```

```
    strcpy(cstr, str.c_str());
```

```
    label_word[strlen(cstr)];
```

```
    for(int i=0; i<strlen(cstr); i++)
```

```
    {
```

```
        label_word = new QLabel();
```

```
        label_word->setObjectName("label_letter_" +  
std::to_string(i));
```

```
        label_word->setStyleSheet("QLabel { background-color: rgb(255,  
255, 255); color: white; font-size: 18px; font-style: bold; border:  
none;}");
```

```
        label_word->setMaximumSize(40,40);
```

```
        label_word->setMinimumSize(40,40);
```

```
        label_word->setAlignment(Qt::AlignCenter);
```

```
        label_word->setText(QChar(cstr[i]));
```

```
        ui->horizontalLayout->addWidget(label_word);
```

```
    }
```

```
}
```

```
void Play_widget::on_abc_clicked()
```

```
{
```

```
    bool find = false;
```

```
    QPushButton *button_abc = (QPushButton *)sender();
```

```
    for(int i=0; i<strlen(cstr); i++)
```

```
    {
```

```

        if(cstr[i]==button_abc->text())
        {
            find_abc(i);
            find = true;
        }
        else
        {
            button_abc->setDisabled(true);
            button_abc->setStyleSheet("QPushButton{ color: rgb(190,
190, 190); font-weight: 400; font-size: 20px; position: absolute;
width: 40px; height: 40px; background-color: rgb(100, 51, 51); border:
none; } ");
        }

    }
    if(find == false)
    {
        num_of_attempts--;
        num_of_score++;
        num_of_steps++;

        next_step(num_of_steps);

        if(num_of_attempts<=0)
        {
            lose(strlen(cstr));
        }
    }
}

void Play_widget::find_abc(int i)
{

    QString label_letter_name = QString("label_letter_%1").arg(i);

```

```

        QLabel *label_letter_obj = findChild<QLabel *>(label_letter_name);

        label_letter_obj->setStyleSheet("QLabel { background-color:
rgb(255, 255, 255); color: black; font-size: 18px; font-style: bold;
border: none;}");
        score++;
        num_of_score++;
        if(score == strlen(cstr))
        {
            win();
        }
    }

void Play_widget::lose(int cstr)
{
    game_result = "Lose";
    for(int i=0; i<cstr; i++)
    {
        QString label_letter_name = QString("label_letter_%1").arg(i);

        QLabel *label_letter_obj = findChild<QLabel
*>(label_letter_name);
        label_letter_obj->setStyleSheet("QLabel { background-color:
rgb(255, 255, 255); color: black; font-size: 18px; font-style: bold;
border: none;}");
    }

    save_result(game_result);
    ui->key_frame->hide();
    ui->lose_frame->show();
}

void Play_widget::win()

```

```

{
    game_result = "Win";
    save_result(game_result);
    ui->win_frame->show();
    ui->key_frame->hide();
}

void Play_widget::save_result(QString game_result)
{
    static const char* const FILE_NAME =
"Resources/result/result.txt";
    QFile out( FILE_NAME );

    QString result_try = QString("Number of attempts:
%1\n").arg(num_of_score);
    QString result_word = QString("Guess word: %1\n").arg(cstr);
    QString result_time = QString("Date: " +
QDateTime::currentDateTime().toString("yyyy-MM-dd HH:mm:ss") + "\n");
    QString result_result = QString("Result: %1\n").arg(game_result);
    if( out.open( QIODevice::Append))
    {
        QTextStream stream( &out );

        stream << result_time;
        stream << result_word;
        stream << result_try;
        stream << result_result;
        stream << "\n\n";
        out.close();
    }
}

```

```
void Play_widget::next_step(int num_of_step)
{
    QString path_to_step =
QString("Resources/steps/step_%1.jpg").arg(num_of_step);
    ui->img_label->setPixmap(QPixmap(path_to_step));
}
```

Приложение Б

Заголовок/mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QPushButton>
//new window
#include "play_widget.h"
QT_BEGIN_NAMESPACE
namespace Ui {
    class MainWindow;
}
QT_END_NAMESPACE
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    QPushButton *button_select_level;
    // added
public slots:
    void show_frame_level();
private:
    Ui::MainWindow *ui;

    //new window
    Play_widget *play_wid;
private slots:
    void on_pushButton_clicked();
    void on_level_clicked();
};
#endif // MAINWINDOW_H
```

Заголовок playwidget.h

```
#ifndef PLAY_WIDGET_H
#define PLAY_WIDGET_H
#include <QWidget>
#include <QLabel>
namespace Ui {
class Play_widget;
}
class Play_widget : public QWidget
{
    Q_OBJECT
public:
    explicit Play_widget(QWidget *parent = 0);
    ~Play_widget();
    void select_lvl_file(QString lvl);
    std::string str;
    char* cstr;
    QLabel *label_word;
signals:
    void mainwindow();
private slots:
    void on_b_exit_clicked();
    void on_abc_clicked();
    void play(QString res);
    void select_word(QString num_file);
    void find_abc(int i);
    void lose(int i);
    void win();
    void save_result(QString result);
    void next_step(int num_of_step);
private:
    Ui::Play_widget *ui;
};
#endif // PLAY_WIDGET_H
```