

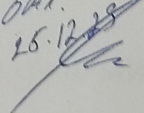
Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
по курсу «Программирование»
на тему «Разработка программы сложной структуры
методом нисходящего программирования»

Выполнил:
студент группы 22ВОА2
Нагаев Д. А.

Принял:
Митрохин М. А.

Омск.
15.12.22


Пенза 2023

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра «Вычислительная техника»

«УТВЕРЖДАЮ»
Зав. кафедрой ВТ
Митрохин М. А.

«_____» _____ 2023 г.

ЗАДАНИЕ
на курсовое проектирование по курсу

Программирование

Студенту Нагаеву Дмитрию Алексеевичу Группа 22ВОА2
Тема проекта: Разработка программы сложной структуры методом
нисходящего программирования

Исходные данные (технические требования) на проектирование
Наименование программы: Компьютерная игра «2048».

Задание. Программа должна выполнять следующие функции:

- Создание игрового поля.
- Генерация игровых объектов в виде квадратов со значениями 2 в 1-11 степенях.
- Появление при запуске игры меню с кнопками «Start» и «Exit».
- Вывод сообщения о победе или поражении.

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных — списков, структур, файлов.
3. Режим работы видеосистемы — текстовый/графический.
4. Устройство ввода информации — клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система — Windows.
8. Система программирования — C/C++ на базе среды Microsoft Visual Studio 2022.
9. Язык программирования — Си++.

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 « 9 » сентября 2023 г

Дата защиты проекта « ____ » _____ 2023 г

Руководитель Митрохин Максим Александрович

Задание получил « 9 » сентября 2023 г

Студент Князев Дмитрий Алексеевич

Содержание

Введение.....	5
1. Постановка задачи	6
2. Выбор решения.....	7
2.1 Определение необходимых модулей программы	7
2.2 Определение алгоритма программы	9
3. Описание разработки программы	12
4. Отладка и тестирование	14
5. Описание программы	16
6. Руководство пользователя.....	17
Заключение.....	19
Список использованных источников.....	20
Приложение А Листинг программы.....	21
Приложение А.1 Исходный файл «Main.cpp».....	21
Приложение А.2 Заголовочный файл «GameMenu.hpp».....	27
Приложение А.3 Заголовочный файл «RenderScreen.hpp».....	32
Приложение А.4 Заголовочный файл «Tiles.hpp».....	35
Приложение Б – Результаты работы программы (снимки экрана).....	41

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. Это позволило развиваться такому виду развлечений как видеоигры. Они имеют большое значение для людей, наряду с кинематографом и другими развлечениями. Они помогают людям узнать для себя что-то новое, расслабиться и весело провести время и просто дарят массу положительных эмоций. Как культура, игры не так давно обосновались, но плотно «врезались» в общество, уже каждый, по крайней мере, 3 игрок ассоциирует видеоигру, как маленькое произведение искусства. Целью этой курсовой работы является создание игры жанра головоломка, а также получение практических навыков написания и разработки приложений с использованием программной среды разработки Microsoft Visual Studio 2022.

1 Постановка задачи

Необходимо разработать игру «2048». В игре будут реализованы следующие правила:

1. При запуске игры на поле размером 4 на 4 появляется две плитки со значениями 2 или 4.
2. При нажатии на стрелки все плитки игрового поля в одну из 4 сторон. Если при перемещении две плитки одного номинала «налетают» одна на другую, то они превращаются в одну, номинал которой равен сумме соединившихся плиток.
3. За каждые объединённые плитки пользователю будут начисляться очки.
4. Основной целью игры является достижение плитки со значением 2048.
5. При заполнении всего игрового поля плитками и невозможности их объединения наступает конец игры.

Необходимо также реализовать меню, с помощью которого можно будет запустить игру и выйти из приложения.

Необходимо реализовать минималистичный и удобный интерфейс, чтобы у пользователя не возникало затруднений при использовании программы. В программе также будут использоваться сложные типы данных. Устройствами управления игрой является клавиатура. Все действия в игре должны осуществляться с помощью нажатия клавиш клавиатуры.

2 Выбор решения

2.1 Определение необходимых модулей программы

Разработанная программа состоит из четырёх модулей:

1. Main.cpp
2. Tiles.hpp
3. RenderScreen.hpp
4. GameMenu.hpp

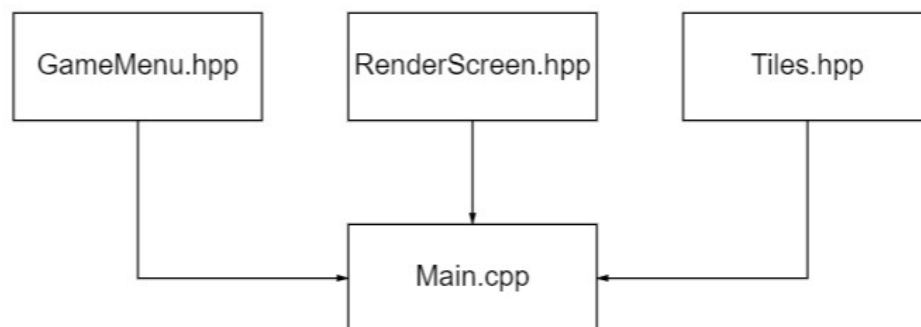
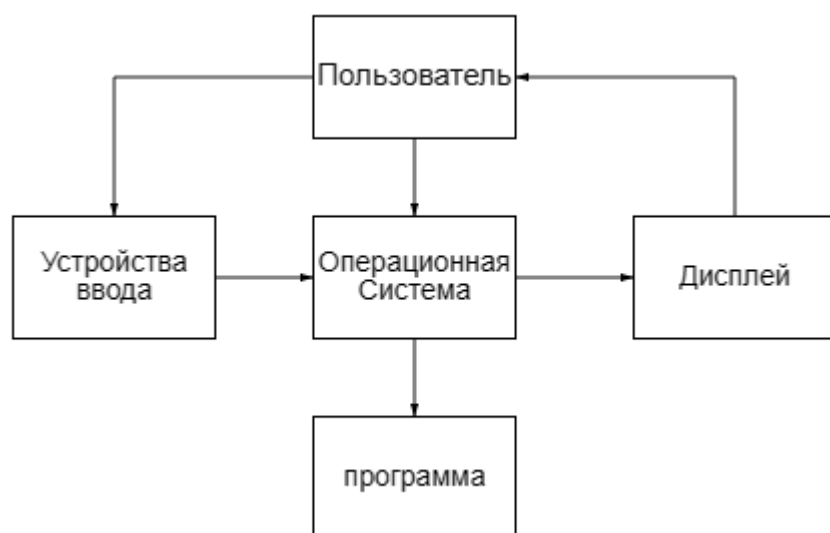


Рисунок 1 — Схема взаимодействия модулей

1. RenderScreen.hpp – модуль содержит в себе поле игры и все генерируемые изображения и надписи.
2. GameMenu.hpp – модуль содержит в себе код меню игры.
3. Tiles.hpp – модуль содержит функции для работы с плитками.
4. Main.cpp – главный модуль программы. Именно здесь происходит генерация меню игры. В этом модуле осуществляется запуск игрового процесса.

Ниже на рисунке 2 представлена схема данных программы

Рисунок 2 — Схема данных



2.2 Определение алгоритма программы

Алгоритм игрового процесса основан на простейших операциях и функциях:

- 1) Загрузить и включить меню при запуске игры;
- 2) Позволить игроку двигаться после запуска.
- 3) Считать двумерный массив, который представляет собой поле игры;
- 4) Загрузить на поле плитки;
- 5) Если игрок получил плитку номиналом 2048, то завершить игру.

Для представления принципа работы уровня игры ниже приведена его функциональная схема (рис.3).



Рисунок 3 — алгоритм, реализующий принцип работы игры

Алгоритм проверки на нажатие клавиш движения основан на простом условии, если есть свободное место или плитки с одинаковым номиналом

соприкасаются, то сдвинуть плитки в выбранное направление в зависимости от нажатой клавиши. Алгоритм движения представлен на (рис.4).

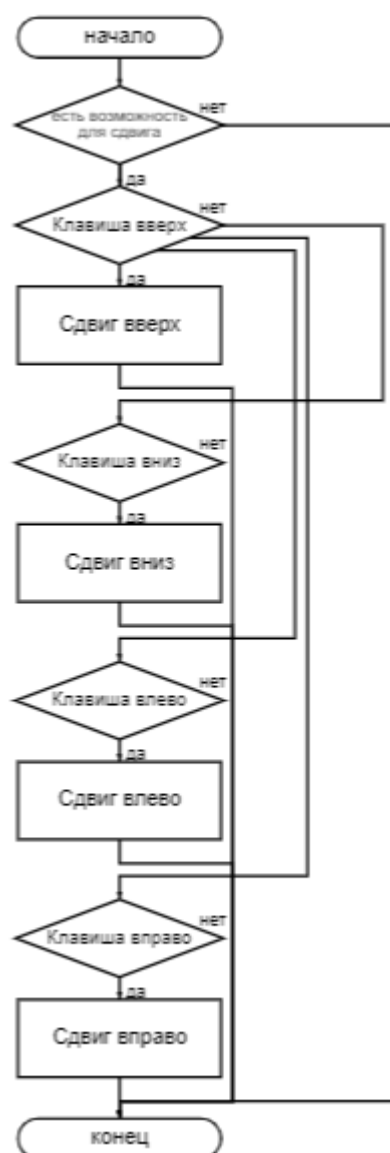


Рисунок 4 — алгоритм проверки на нажатия клавиш движения

Алгоритм объединения плиток очень простой и состоит из проверки условия, если плитки с одинаковым номиналом «налетают» друг на друга, то они объединяются, а сумма их номиналов прибавляется к счёту. Алгоритм объединения плиток приведен на (рис.5).

Рисунок 5 — алгоритм объединения плиток



3 Описание разработки программы

Для написания программы был использован язык программирования C++ а также библиотека SFML, с помощью которого очень удобно спроектировать исполнение игры головоломка и обеспечить быстроедействие игры. В качестве среды разработки была использована программа Microsoft Visual Studio 2022.

Для начала была разработаны меню, поле, движок игры «2048». Для работы используется модуль GameMenu.hpp, Main.cpp. Здесь пользователь может запустить игру и закрыть её.

При выборе элемента меню «Играть» начинаться сам процесс игры. Запускается игровая сессия. Проектирование игрового поля и плиток происходит с помощью методов из модулей RenderScreen.hpp и Main.cpp. С помощью библиотеки SFMLGraphics мы выбираем оформление игры. После нажатия клавиши Escape пользователь может выйти из игры, а при нажатии клавиши Enter – начать игру заново.

Движение игрока основано на перемещении в четыре разные стороны. Считывается клавиша, которая была нажата в функции keypressed. Если это клавиша направления (стрелочки), то заданное начинается движение со смещением по этому направлению.

В процессе, когда игроком будут объединены плитки, пользователь зарабатывает очки, которые равны сумме номиналов объединённых плиток. Информация о счёте выводится на экран.

В модуле Tiles.hpp содержится алгоритмы появления и движения плиток, если на поле остаётся свободное место или плитки с одинаковым номиналом соприкасаются, то происходит сдвиг плиток в выбранное направление и появляется новая плитка. Движение осуществляется при нажатии клавиш стрелок, считывание нажатия происходит с помощью функций keyPressed.

Для анализа программы необходимо выявить ее поведение (табл.1):

Таблица 1 — Анализ программы

Описание	Результат
Запуск программы	Включается меню с выбором уровней, включается фоновая музыка.
Управление плитками	Нажимаются клавиши движение, плитки сдвигаются(меняют координаты).
Появление плиток	Появление плиток в случайных местах поля после запуска уровня и осуществления сдвига плиток.
Проверка на поражение	Вывод сообщением о поражении и предложении попробовать снова.
Проверка на завершение игры	Вывод сообщения о победе в игре после получения плитки 2048.

4 Отладка и тестирование

Средой разработки была выбрана программа VisualStudio 2022. Программа содержит в себе все необходимые средства для успешной отладки и тестирования модулей программы, а также всей программы в целом. Данная среда разработки выпущена относительно недавно, что позволяет проводить необходимые операции над разработанной программой наиболее производительно и эффективно. Для выполнения отладки применялись средства Visual Studio, такие как: точки останова, анализ содержимого переменных, выполнение кода по шагам (по строкам), анализ памяти. В данном случае точка остановки была установлена после объединения плиток, что позволяет проверить получение очков счёта, представлен на (рис.6). Это поможет понять, какие значения были сгенерированы и какие проверки будут выполнены.

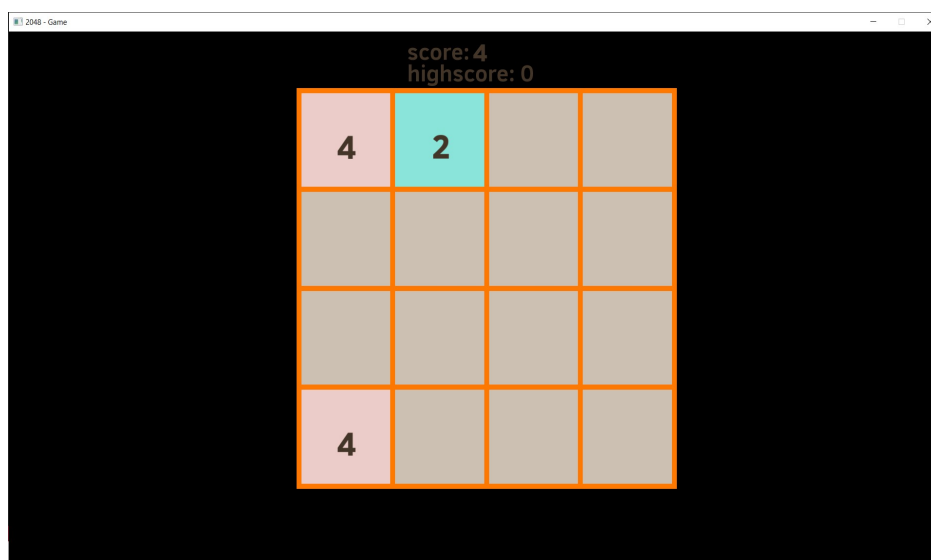


Рисунок 6 — Отладка программы

Тестирование проводилось в рабочем порядке, на всех этапах разработки программы, в том числе после завершения написания кода. Тестирование позволило выявить и исправить множество проблем и недочетов.

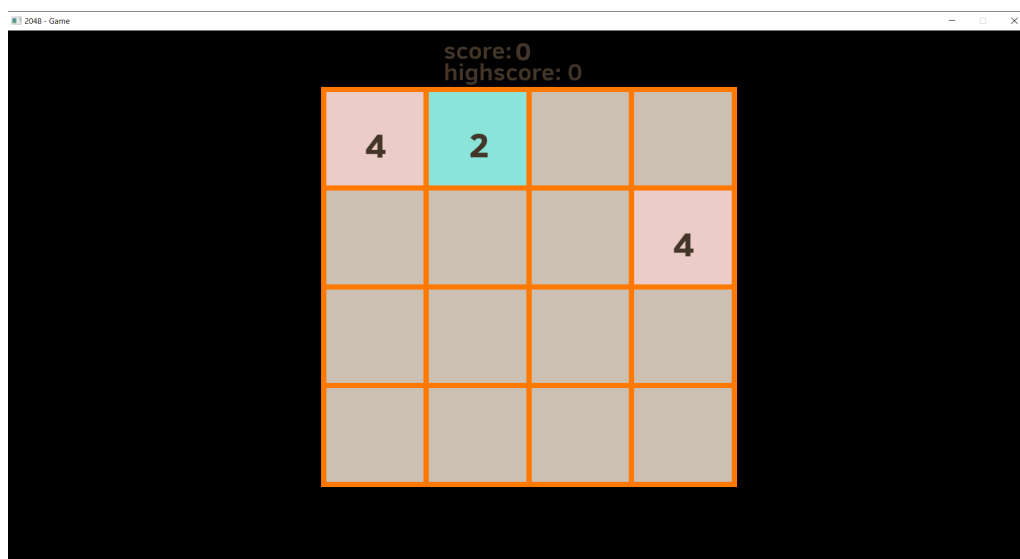


Рисунок 7 — Тестирование программы(появление плиток)

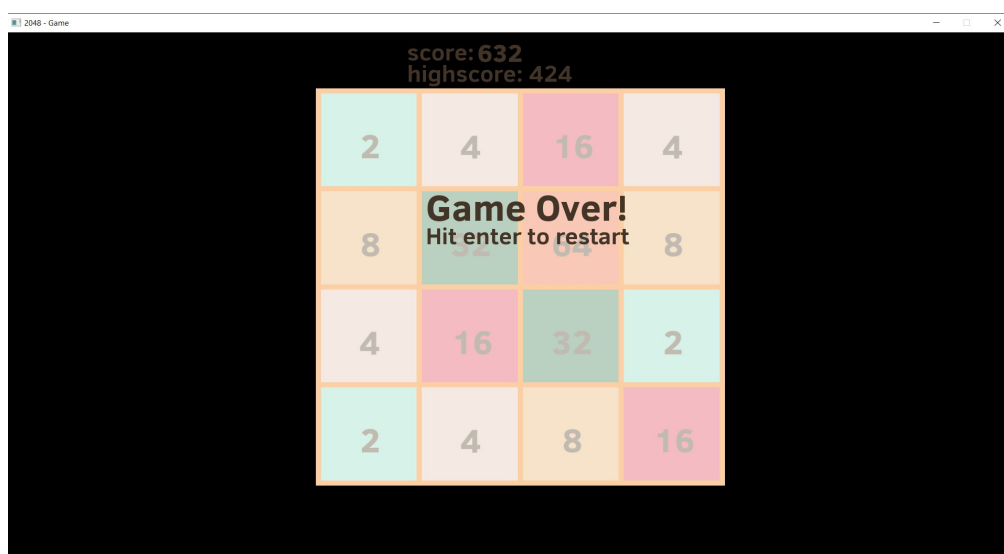


Рисунок 8 — Тестирование программы(поражение)

5 Описание программы

Данная программа представляет собой приложение на языке C++, которое позволяет запустить игру «2048».

При выборе кнопки «Играть» запускается уровень игры. Доступны следующие действия:

1. Сдвинуть вправо «→»
2. Сдвинуть влево «←»
3. Сдвинуть вверх «↑»
4. Сдвинуть вниз «↓»
5. Начать сначала «enter»
6. Выйти из игры «escape»

Пользователь может нажать кнопку «Играть», после чего ему предоставляется управление игрой. Затем пользователь может начать перемещение, нажимая клавиши движения.

После запуска игры происходит появление двух плиток с номиналом 2 или 4. Также после сдвига плиток появляется плитка в случайном свободном месте номиналом 2 или 4.

Если игрок доходит до конца игры или проигрывает, игра выводит соответствующее сообщение и предоставляет выбор:

1. Начать заново.
2. Выйти из игры.

6 Руководство пользователя

Вид окна меню при запуске приведен на рис 9.



Рисунок 9 — вид окна меню

При выборе действия «Start» – запуститься игра на рис.10.

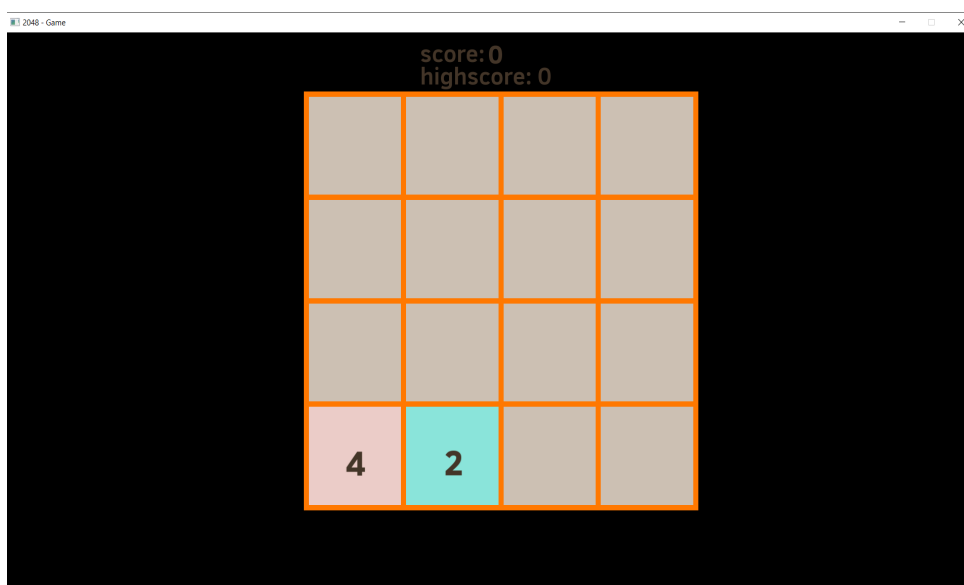


Рисунок 10 – выбор кнопки «Играть»

Далее представлено появление плитки после сдвига на рис.11.

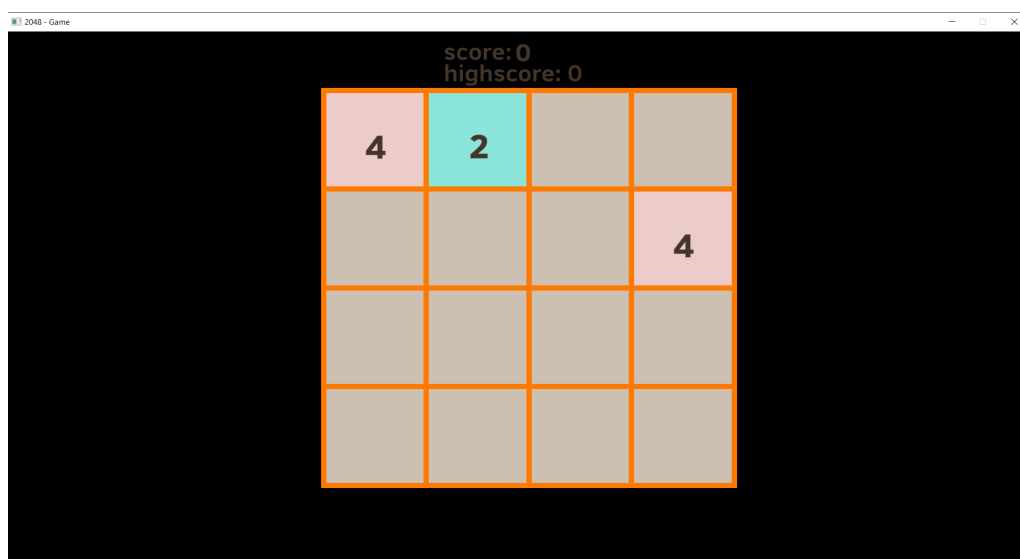


Рисунок 11 — появление плитки после сдвига

После поражения высвечивается окно с выбором, представлено на рис.12.

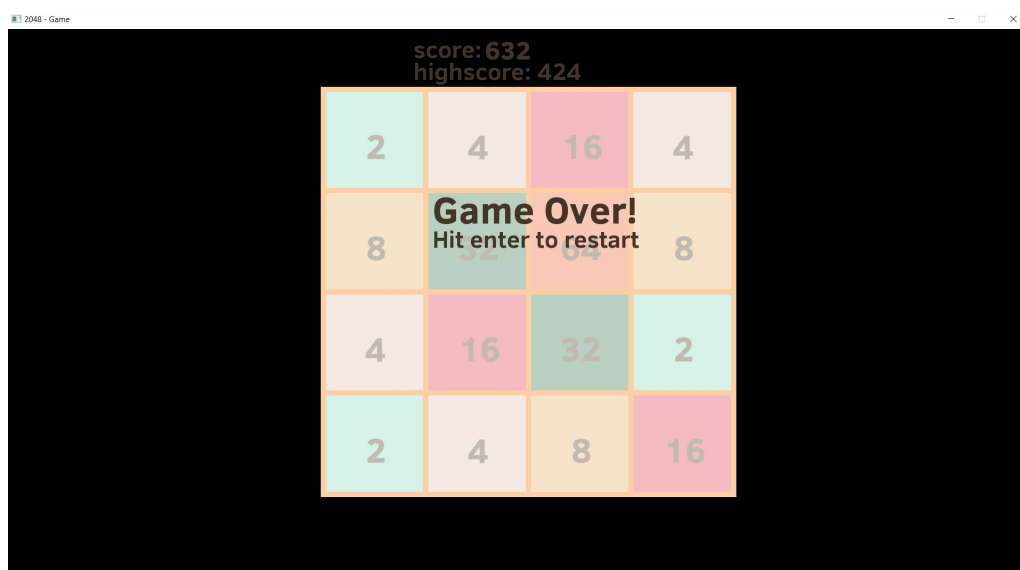


Рисунок 12 — вывод окна поражения с выбором

Заключение

В ходе выполнения курсовой работы были получены и расширены навыки разработки многомодульных программ на языке C++, освоены функции работы с библиотекой SFML, способы интерпретации событий от клавиатуры. Освоены на практике в большом объеме функции для работы с изображениями. Закреплены приемы работы в компиляторе Visual Studio 2022, отточены навыки программирования на языке C++.

В данной курсовой работе была написана программа, используемая для запуска игры «2048». Соответственно, программа не требует большого количества системных ресурсов, благодаря, в том числе использованию языка C++, обеспечивает быструю обработку информации.

Программу в будущем можно улучшать, добавлять новые функции. В целом, поставленная в начале курсового проекта цель была достигнута. В программе выполняются все необходимые функциональные требования.

Список использованных источников

1. Электронный ресурс. habr.com Официальный сайт. – URL: <https://habr.com/ru/articles/710100/> (дата обращения 19.10.23)
2. Электронный ресурс. sfml-dev.org Официальный сайт. – URL: <https://www.sfml-dev.org/tutorials/2.5/> (дата обращения 20.10.23)
3. Электронный ресурс. Learn.microsoft.com Официальный сайт. – URL: <https://learn.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170> (дата обращения 25.10.23)
4. Электронный ресурс. habr.com Официальный сайт. – URL: <https://habr.com/ru/companies/skillfactory/articles/588356/> (дата обращения 28.10.23)
5. Электронный ресурс. habr.com Официальный сайт. – URL: <https://habr.com/ru/articles/537682/> (дата обращения 29.10.23)

Приложение А

Листинг программы

Приложение А.1

Исходный файл «Main.cpp»

```
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <map>
#include <deque>
#include <utility>
#include <sstream>
#include <string>
#include <stdlib.h>
#include "GameMenu.hpp"
#include "RenderScreen.hpp"
#include "Tiles.hpp"

using namespace std;
using namespace sf;

// функция настройки текста
void InitText(Text& mtext, float xpos, float ypos, String str, int size_font =
60,
        Color menu_text_color = Color::White, int bord = 0, Color border_color =
Color::Black);

// Функция перехода к игре
void GameStart()
{
    srand(time(NULL));

    // Загрузка шрифта
    if (!font.loadFromFile("ClearSans.ttf")) {
        printf("Font fail\r\n");
    }

    // Определение цвета для ячеек на основе их значения
    numberColours[0] = Color(204, 192, 179, 255);
    numberColours[2] = Color(138, 228, 218, 255);
    numberColours[4] = Color(235, 204, 200, 255);
    numberColours[8] = Color(242, 177, 121, 255);
    numberColours[16] = Color(231, 55, 99, 255);
```

```

numberColours[32] = Color(46, 124, 95, 255);
numberColours[64] = Color(246, 94, 59, 255);
numberColours[128] = Color(55, 27, 114, 255);
numberColours[256] = Color(237, 204, 97, 255);
numberColours[512] = Color(137, 200, 80, 255);
numberColours[1024] = Color(27, 197, 63, 255);
numberColours[2048] = Color(39, 14, 46, 255);

// фон поля
gridBackgroundRect.setFill-color(gridBGColour);
gridBackgroundRect.setPosition(Vector2f(590, 110));

// счёт
scoreText.setFont(font);
scoreText.setStyle(Text::Bold);
scoreText.setFill-color(textColour);
scoreDecoratorText.setFill-color(textColour);

// плитки
tileText.setFont(font);
tileText.setCharacterSize(64);
tileText.setStyle(Text::Bold);
tileText.setFill-color(textColour);

// отображение игры
gameOverBackground.setFill-color(Color(250, 248, 239, 175));
gameOverBackground.setPosition(Vector2f(590, 110));
gameOverText.setStyle(Text::Bold);
gameOverText.setFill-color(textColour);
gameOverText.setPosition(Vector2f(800, 300));

reset();

bool moveDone = false;
bool scheduledNumberAdd = false;
Clock clock;
clock.restart();
Time addTimeout = milliseconds(75);

while (window.isOpen()) {
    Event event;
    while (window.pollEvent(event)) {

```

```

        if (event.type == Event::Closed) {
            window.close();
        }
        if (event.type == Event::KeyPressed && !moveDone
&& !scheduledNumberAdd) {
            unsigned char tilesMoved = 0;
            if (Keyboard::isKeyPressed(Keyboard::Up))
            {
                tilesMoved = moveUp();
            }
            else if
            (Keyboard::isKeyPressed(Keyboard::Right)) {
                tilesMoved = moveRight();
            }
            else if
            (Keyboard::isKeyPressed(Keyboard::Down)) {
                tilesMoved = moveDown();
            }
            else if
            (Keyboard::isKeyPressed(Keyboard::Left)) {
                tilesMoved = moveLeft();
            }
            else if
            (Keyboard::isKeyPressed(Keyboard::0)) {
                showGameOver = true;
            }
            else if
            (Keyboard::isKeyPressed(Keyboard::Return)) {
                reset();
                continue;
            }
            else if
            (Keyboard::isKeyPressed(Keyboard::Escape)) {
                window.close();
            }
            moveDone = true;
            if (tilesMoved > 0) {
                scheduledNumberAdd = true;
                clock.restart();
            }
        }
        if (event.type == Event::KeyReleased &&
moveDone) {
            printf("Emtpies:          %zd\r\n",
getEmptyCells().size());

```

```

getMovesLeft());

printf("Available      Merges:      %d\r\n",

over!\r\n\r\n");

        if (isGameOver()) {
            printf("\r\n\r\nGame

            showGameOver = true;
        }
        moveDone = false;
    }
}

        if (scheduledNumberAdd && clock.getElapsedTime() >

addTimeout) {
            addNumber();
            scheduledNumberAdd = false;
        }
        renderScreen();
    }

}

int main() {
    window.setCursorVisible(false);
    float width = VideoMode::getDesktopMode().width;
    float height = VideoMode::getDesktopMode().height;

    RectangleShape background(Vector2f(width, height));
    Texture texture_window;
    if (!texture_window.loadFromFile("menubg.jpg")) return 2;
    background.setTexture(&texture_window);

    Font font;
    if (!font.loadFromFile("ClearSans.ttf")) return 3;
    Text Titul;
    Titul.setFont(font);
    // Текст с названием игры
    InitText(Titul, 480, 50, L"Cyber2048", 150, Color(237, 147, 0), 3);
    // Название пунктов меню
    String name_menu[]{ L"Start", L"Exit" };
    // Объект игровое меню
    game::GameMenu mymenu(window, 950, 350, 4, name_menu, 100, 120);
    // Установка цвета элементов пунктов меню
    mymenu.setColorTextMenu(Color(237, 147, 0), Color::Red, Color::Black);
    // выравнивание по центру пунктов меню

```

```

mymenu.AlignMenu(2);

while (window.isOpen())
{
    Event event;
    while (window.pollEvent(event))
    {
        if (event.type == Event::KeyReleased)
        {
            // События выбора пунктов меню
            // нажати на клавиатуре стрелки вверх
            if (event.key.code == Keyboard::Up)
            {
                mymenu.MoveUp();
            }

            // нажати на клавиатуре стрелки вниз
            if (event.key.code == Keyboard::Down)
            {
                mymenu.MoveDown();
            }

            // нажати на клавиатуре клавиши Enter
            if (event.key.code == Keyboard::Return)
            {
                // Переходим на выбранный пункт меню
                switch (mymenu.getSelectedMenuNumber())
                {
                    {
                        case 0:GameStart(); break;
                        case 1:window.close(); break;

                    }
                }
            }
        }
    }

    window.clear();
    window.draw(background);
    window.draw(Titul);
    mymenu.draw();
    window.display();
}

return 0;
}

```

```
// функция настройки текста
void InitText(Text& mtext, float xpos, float ypos, String str, int size_font,
             Color menu_text_color, int bord, Color border_color)
{
    mtext.setCharacterSize(size_font);
    mtext.setPosition(xpos, ypos);
    mtext.setString(str);
    mtext.setFillColor(menu_text_color);
    mtext.setOutlineThickness(bord);
    mtext.setOutlineColor(border_color);
}
```

Приложение А.2

Заголовочный файл «GameMenu.hpp»

```
#pragma once
#include <SFML/Graphics.hpp>

namespace game {

class GameMenu
{
    float menu_X;                // Координаты меню по X
    float menu_Y;                // Координаты меню по Y
    int menu_Step;               // Расстояние между пунктами меню
    int max_menu;                // Максимальное количество пунктов
меню

    int size_font;               // Размер шрифта
    int mainMenuSelected;        // Номер текущего пункта меню
    sf::Font font;               // Шрифт меню
    // Динамический массив текстовых объектов названий пунктов меню
    sf::Text* mainMenu;
    sf::Color menu_text_color = sf::Color::White;    // Цвет пунктов
меню
    sf::Color chose_text_color = sf::Color::Yellow; // Цвет выбора
пункта меню
    sf::Color border_color = sf::Color::Black;      // Цвет обводки
текста пунктов меню

    // Настройка текста пунктов меню
    // Параметры: ссылка на текстовый объект, текст, координаты текста
    void setInitText(sf::Text& text, sf::String str, float xpos, float
ypos);

    // Ссылка на графическое окно
    sf::RenderWindow& mywindow;

public:
    // Конструктор
    // Параметры: ссылка на графическое окно, координаты игрового меню
по x и y
    // количество пунктов меню, массив названий пунктов меню, размер
шрифта, шаг между пунктами меню
    GameMenu(sf::RenderWindow& window, float menux, float menuy,
            int index, sf::String name[], int sizeFont = 60, int step =
80);
```

```

~GameMenu()
{
    delete[] mainMenu;
}

void draw();           // Рисуем меню

void MoveUp();         // Перемещение выбора меню вверх

void MoveDown();       // Перемещение выбора меню вниз

// Цвет элементов игрового меню
void setColorTextMenu(sf::Color menColor, sf::Color ChoColor,
    sf::Color BordColor);

void AlignMenu(int posX); // Выравнивание положения меню

int getSelectedMenuNumber() // Возвращает номер выбранного элемента
меню
{
    return mainMenuSelected;
}

};

}

// метод настройки текстовых объектов пунктов игрового меню
void game::GameMenu::setInitText(sf::Text& text, sf::String str, float xpos,
float ypos)
{
    text.setFont(font);           // шрифт
    text.setFillColor(menu_text_color); // цвет
    text.setString(str);          // текст
    text.setCharacterSize(size_font); // размер шрифта
    text.setPosition(xpos, ypos);  // координаты текстового объекта
    text.setOutlineThickness(3);    // толщина контура обводки текста
    text.setOutlineColor(border_color); // цвет контура обводки текста
}

// Выравнивание пунктов меню по левому по правому по центру
void game::GameMenu::AlignMenu(int posX)
{
    float nullx = 0;
    for (int i = 0; i < max_menu; i++) {
        switch (posx)

```

```

        {
            case 0:
                nullx = 0; // выравнивание по правому краю от установленных
координат
                break;
            case 1:
                nullx = mainMenu[i].getLocalBounds().width; // по левому
краю
                break;
            case 2:
                nullx = nullx = mainMenu[i].getLocalBounds().width / 2; //
по центру
                break;
        }
        mainMenu[i].setPosition(mainMenu[i].getPosition().x - nullx,
mainMenu[i].getPosition().y);
    }

}
// Конструктор
game::GameMenu::GameMenu(sf::RenderWindow& window, float menux, float menuy,
    int index, sf::String name[], int sizeFont, int step)
    :mywindow(window), menu_X(menux), menu_Y(menuy), size_font(sizeFont),
menu_Step(step)
{
    // Загрузка шрифта
    if (!font.loadFromFile("ClearSans.ttf")) exit(32);
    max_menu = index; // Количество элементов меню
    mainMenu = new sf::Text[max_menu]; // Динамический массив пунктов меню
    // Выстраиваем элементы меню
    for (int i = 0, ypos = menu_Y; i < max_menu; i++, ypos += menu_Step)
setInitText(mainMenu[i], name[i], menu_X, ypos);
    mainMenuSelected = 0; // Задаём начальное положения выбранного пункта меню
    // цвет выбранного пункта меню
    mainMenu[mainMenuSelected].setFillColor(chose_text_color);
}
// перемещение выбора меню вверх
void game::GameMenu::MoveUp()
{
    mainMenuSelected--;
    // подсвечиваем выбранный пункт меню
    if (mainMenuSelected >= 0) {
        mainMenu[mainMenuSelected].setFillColor(chose_text_color);
        mainMenu[mainMenuSelected + 1].setFillColor(menu_text_color);
    }
}

```

```

    }
    else
    {
        mainMenu[0].setFillColor(menu_text_color);
        mainMenuSelected = max_menu - 1;
        mainMenu[mainMenuSelected].setFillColor(chose_text_color);
    }
}

// перемещение выбора меню вниз
void game::GameMenu::MoveDown()
{
    mainMenuSelected++;
    // подсвечиваем выбранный пункт меню
    if (mainMenuSelected < max_menu) {
        mainMenu[mainMenuSelected - 1].setFillColor(menu_text_color);
        mainMenu[mainMenuSelected].setFillColor(chose_text_color);
    }
    else
    {
        mainMenu[max_menu - 1].setFillColor(menu_text_color);
        mainMenuSelected = 0;
        mainMenu[mainMenuSelected].setFillColor(chose_text_color);
    }
}

// рисуем элементы меню в графическом окне
void game::GameMenu::draw()
{
    // перебираем для отрисовки существующие текстовые объекты пунктов меню

    for (int i = 0; i < max_menu; i++) mywindow.draw(mainMenu[i]);
}

// назначение цвета элементам пунктов меню
void game::GameMenu::setColorTextMenu(sf::Color menColor, sf::Color ChoColor,
sf::Color BordColor)
{
    menu_text_color = menColor;    // цвет пунктов меню
    chose_text_color = ChoColor;    // цвет выбранного пункта меню
    border_color = BordColor;    // цвет контура пунктов меню

    for (int i = 0; i < max_menu; i++) {
        mainMenu[i].setFillColor(menu_text_color);
        mainMenu[i].setOutlineColor(border_color);
    }
}

```

```
    }  
  
    mainMenu[mainMenuSelected].setFillColor(chose_text_color);  
}
```

Приложение А.3

Заголовочный файл «RenderScreen.hpp»

```
#pragma once
#include <SFML/Graphics.hpp>
#include <map>
#include <deque>
#include <utility>
#include <sstream>
#include <string>
#include <stdlib.h>

using namespace sf;
using namespace std;

unsigned short score = 0;
unsigned short highScore = 0;
unsigned short grid[4][4];
bool showGameOver;

Font font;
Color windowBGColour(0);
Color gridBGColour(255, 120, 0, 255);
Color textColour(67, 53, 40, 255);
map<int, Color> numberColours;

float padding = 10.f;
float gridSize = (800.f - (padding * 2.f));
float tileSize = (gridSize - (5 * padding)) / 4;

RenderWindow window(VideoMode::getDesktopMode(), "2048 - Game", Style::Close |
Style::Titlebar);
RectangleShape gridBackgroundRect(Vector2f(gridSize, gridSize));
RectangleShape gameOverBackground(Vector2f(gridSize, gridSize));
RectangleShape gridTile(Vector2f(tileSize, tileSize));
Text gameOverText("Game Over!", font, 70);
Text gameWinText("Congratulatings\nYou win!", font, 100);
Text scoreText(to_string(score), font, 48);
Text scoreDecoratorText("", font, 45);
Text tileText;
```

```

void renderScreen() {
    window.clear(windowBGColour);

    window.draw(gridBackgroundRect);

    // Отрисовка поля
    for (int ix = 0; ix < 4; ix++) {
        for (int iy = 0; iy < 4; iy++) {
            gridTile.setPosition(sf::Vector2f(600 + (ix * (tileSize +
padding)), 120 + (iy * (tileSize + padding))));
            gridTile.setFillColor(numberColours.find(grid[ix][iy]) -
>second);
            window.draw(gridTile);

            if (grid[ix][iy] > 0) {
                tileText.setString(to_string(grid[ix][iy]));
                sf::Vector2f textPosition = (gridTile.getPosition() +
(gridTile.getSize() / 2.f));
                textPosition.x -= tileText.getGlobalBounds().width /
2.f;
                textPosition.y -= (tileText.getGlobalBounds().height +
padding) / 2.f;
                tileText.setPosition(textPosition);
                window.draw(tileText);
            }
        }
    }

    // ведение счёта
    scoreText.setString(to_string(score));
    scoreText.setPosition(Vector2f(980 - scoreText.getGlobalBounds().width,
10));
    window.draw(scoreText);
    scoreDecoratorText.setString("score:");
    scoreDecoratorText.setPosition(Vector2f(scoreText.getPosition().x -
scoreDecoratorText.getGlobalBounds().width - 10, 10));
    window.draw(scoreDecoratorText);
    scoreDecoratorText.setString("highscore: " + to_string(highScore));
    scoreDecoratorText.setPosition(Vector2f(scoreDecoratorText.getPosition().x, 52));
    window.draw(scoreDecoratorText);

    // вывод поражения
    if (showGameOver) {
        window.draw(gameOverBackground);
    }
}

```

```
        window.draw(gameOverText);
        scoreDecoratorText.setString("Hit enter to restart");

        scoreDecoratorText.setPosition(Vector2f(gameOverText.getPosition().x,
gameOverText.getPosition().y + gameOverText.getLocalBounds().height + padding * 2));
        window.draw(scoreDecoratorText);
    }

    window.display();
}
```

Приложение А.4

Заголовочный файл «Tiles.hpp»

```
#pragma once

#include <SFML/Graphics.hpp>
#include <map>
#include <deque>
#include <utility>
#include <sstream>
#include <string>
#include <stdlib.h>

using namespace std;
using namespace sf;

// возвращает координаты пустой плитки
vector<sf::Vector2i> getEmptyCells() {
    vector<sf::Vector2i> empties;
    for (unsigned char ix = 0; ix < 4; ix++) {
        for (unsigned char iy = 0; iy < 4; iy++) {
            if (grid[ix][iy] == 0) {
                empties.push_back(sf::Vector2i(ix, iy));
            }
        }
    }
    return empties;
}

// Добавляет случайную плитку в пустое пространство

void addNumber() {
    int newCellOptions[2] = { 2, 4 };
    vector<sf::Vector2i> empties = getEmptyCells();
    if (empties.size() == 0) {
        printf("No empties\r\n");
        return;
    }
    else {
        int randEmpty = rand() % empties.size();
        grid[empties.at(randEmpty).x][empties.at(randEmpty).y]
newCellOptions[rand() % 2]; =
```

```

    }
}

// Возвращает количество плиток, соприкасающихся с плитками одинакового значения
unsigned short getMovesLeft() {
    auto movesLeft = 0;
    for (unsigned char ix = 0; ix < 4; ix++) {
        for (unsigned char iy = 0; iy < 4; iy++) {
            if (grid[ix][iy] > 0) {
                if (ix + 1 < 4 && (grid[ix][iy] == grid[ix + 1][iy] ||
grid[ix + 1][iy] == 0)) {
                    movesLeft++;
                }
                if (iy + 1 < 4 && (grid[ix][iy] == grid[ix][iy + 1] ||
grid[ix][iy + 1] == 0)) {
                    movesLeft++;
                }
            }
        }
    }
    return movesLeft;
}

// проверка на поражение
bool isGameOver() {
    if (getEmptyCells().size() > 0 || getMovesLeft() > 0) {
        return false;
    }
    else {
        return true;
    }
}

// Перезапуск игры
void reset() {
    // новый счёт
    if (score > highScore) {
        highScore = score;
    }
    score = 0;

    for (int ix = 0; ix < 4; ix++) {
        for (int iy = 0; iy < 4; iy++) {
            grid[ix][iy] = 0;

```

```

        }
    }

    showGameOver = false;

    // Добавляет две начальных плитки
    addNumber();
    addNumber();
}

void combine(deque<int>& cellList) {
    if (cellList.size() <= 1) {
        return;
    }
    unsigned char index = 0;
    while (index < cellList.size() - 1) {
        int cell1 = cellList.at(index);
        int cell2 = cellList.at(index + 1);
        if (cell1 == cell2) {
            cellList.at(index) = cell1 * 2;
            if (cellList.at(index) == 2048)
            {
                window.draw(gameOverBackground);
                window.draw(gameWinText);
                scoreDecoratorText.setString("Hit enter to restart");

                scoreDecoratorText.setPosition(Vector2f(gameOverText.getPosition().x,
gameOverText.getPosition().y + gameOverText.getLocalBounds().height + padding * 2));
                window.draw(scoreDecoratorText);
            }
            score += cellList.at(index);
            cellList.erase(cellList.begin() + index + 1);
        }
        index++;
    }
}

// Перемещение плиток
unsigned char moveDown() {
    unsigned char tilesMoved = 0;

```

```

for (int ix = 0; ix < 4; ix++) {
    deque<int> cellList;
    for (int iy = 3; iy >= 0; iy--) {
        if (grid[ix][iy] != 0) {
            cellList.push_back(grid[ix][iy]);
        }
    }
    combine(cellList);

    unsigned short tileBefore;
    for (int iy = 3; iy >= 0; iy--) {
        tileBefore = grid[ix][iy];

        if (cellList.size() > 0) {
            grid[ix][iy] = cellList.front();
            cellList.pop_front();
        }
        else {
            grid[ix][iy] = 0;
        }

        if (grid[ix][iy] != tileBefore) {
            tilesMoved++;
        }
    }
}
return tilesMoved;
}

```

```

unsigned char moveUp() {
    unsigned char tilesMoved = 0;

    for (int ix = 0; ix < 4; ix++) {
        deque<int> cellList;
        for (int iy = 0; iy < 4; iy++) {
            if (grid[ix][iy] != 0) {
                cellList.push_back(grid[ix][iy]);
            }
        }
        combine(cellList);

        unsigned short tileBefore;

```

```

        for (int iy = 0; iy < 4; iy++) {
            tileBefore = grid[ix][iy];

            if (cellList.size() > 0) {
                grid[ix][iy] = cellList.front();
                cellList.pop_front();
            }
            else {
                grid[ix][iy] = 0;
            }

            if (grid[ix][iy] != tileBefore) {
                tilesMoved++;
            }
        }
    }
    return tilesMoved;
}

```

```

unsigned char moveLeft() {
    unsigned char tilesMoved = 0;

    for (int iy = 0; iy < 4; iy++) {
        deque<int> cellList;
        for (int ix = 0; ix < 4; ix++) {
            if (grid[ix][iy] != 0) {
                cellList.push_back(grid[ix][iy]);
            }
        }
        combine(cellList);

        unsigned short tileBefore;
        for (int ix = 0; ix < 4; ix++) {
            tileBefore = grid[ix][iy];

            if (cellList.size() > 0) {
                grid[ix][iy] = cellList.front();
                cellList.pop_front();
            }
            else {
                grid[ix][iy] = 0;
            }
        }
    }
}

```

```

    }

    if (grid[ix][iy] != tileBefore) {
        tilesMoved++;
    }
}
}
return tilesMoved;
}

unsigned char moveRight() {
    unsigned char tilesMoved = 0;

    for (int iy = 0; iy < 4; iy++) {
        deque<int> cellList;
        for (int ix = 3; ix >= 0; ix--) {
            if (grid[ix][iy] != 0) {
                cellList.push_back(grid[ix][iy]);
            }
        }
        combine(cellList);

        unsigned short tileBefore;
        for (int ix = 3; ix >= 0; ix--) {
            tileBefore = grid[ix][iy];

            if (cellList.size() > 0) {
                grid[ix][iy] = cellList.front();
                cellList.pop_front();
            }
            else {
                grid[ix][iy] = 0;
            }

            if (grid[ix][iy] != tileBefore) {
                tilesMoved++;
            }
        }
    }
    return tilesMoved;
}

```

Приложение Б

Результаты работы программы (снимки экрана)

