Проект

по дисциплине

Введение в разработку программного обеспечения



‘’Игра Сапёр на с++”

Выполнили : Волковец Сергей - разработчик

Папыльченко Ян - Документация

Полуянов Андрей - Тестировка

Преподаватель : Ромыш Александра Сергеевна

**Содержание :**

1. **Техническое задание**
   1. Область применения
   2. Основные компоненты игры
   3. Gameplay
   4. Управление
2. **Сценарий использования (Use Case)**
3. **Сценарий тестирования (Test Case)**
4. **Отчёт об ошибках(Bug Report)**
5. **Руководство для пользователя**
6. **Написание программы и её работа**

6.1 Необходимые знания

6.2 Подробное описание функций и методов

**1.Техническое задание**

Цель : создание игры “Сапёр” , где игрок выбирает клетки полей, чтобы не попасть на мины.Проект представляет собой консольное приложение , которое запускает игру “Сапёр” написанную на языке с++ с подключением таких библиотек как : <SFML/Graphics.hpp> и <SFML/Audio.hpp> .

Используется пространство имён sf .

* 1. Область применяя :

Применяется в развлекательных целях и развивает стратегическое и логическое мышление пользователя.

* 1. Основные компоненты игры :

Игровое поле (10х10) , мины , флажки .

* 1. Gameplay:

После запуска игры пользователь выбирает клетку и нажимает(левая кнопка мыши) 1 раз (открытие 1 ячейки) или 2 раза (открытие как выбранной , так и нескольких соседних ячеек). Суть игры не попасть на мину . При попадании на ячейку с миной игра заканчивается.

* 1. Управление :

• Левый click - открытие 1 ячейки

• Левый doubleclick - открытие нескольких ячеек

• Правый click - установка флажка (на ячейку , где предположительно находится мина

**2.Сценарий использования (Use case):**

1) Запуск программы

2) Генерация и отображение поля

3) Игрок выбирает ячейку

4) Игрок использует клики для открытия поля и установки флажков

5) Если игрок открыл поле и не попал на мину - победа

6) Если игрок попал на мину - игра завершается

**3.Сценарий тестирования (Test Case)**

1)Запуск игры.

Ожидаем запуск игры и отображение поля .

Успешно проведено

1. Раскрытие поля.

Ожидаем , что при выборе ячейки 1 щелчком она раскроется , а двойным щелчком раскроется область.

Успешно проведено

1. Завершение игры.

Ожидаем завершение игры при попадании на мину.

Успешно проведено

1. Победа .

Ожидаем вывод победа , при успешном раскрытии всего поля .

Успешно проведено

**4.Отчёт об ошибках(Bug Report)**

First bug :

При первом клике, при раскрытии зоны могла попасться мина, что приводило к мгновенному проигрышу

Статус: Исправлено

Теперь при раскрытии области и нахождении в ней бомбы , ячейка с бомбой не открывается .

gridLogic[nx][ny] != 9

Second bug :

Ошибка в логике неправильный подсчёт мин вокруг заданной клетки в углах поля

Статус: исправлено

if (nx >= 1 && nx <= 10 && ny >= 1 && ny <= 10 && gridLogic[nx][ny] != 9)

int n = 0;

if (gridLogic[i][j] == 9) continue;

if (i + 1 <= 10 && gridLogic[i + 1][j] == 9) n++;

if (j + 1 <= 10 && gridLogic[i][j + 1] == 9) n++;

if (i - 1 >= 1 && gridLogic[i - 1][j] == 9) n++;

if (j - 1 >= 1 && gridLogic[i][j - 1] == 9) n++;

if (i + 1 <= 10 && j + 1 <= 10 && gridLogic[i + 1][j + 1] == 9) n++;

if (i - 1 >= 1 && j - 1 >= 1 && gridLogic[i - 1][j - 1] == 9) n++;

if (i - 1 >= 1 && j + 1 <= 10 && gridLogic[i - 1][j + 1] == 9) n++;

if (i + 1 <= 10 && j - 1 >= 1 && gridLogic[i + 1][j - 1] == 9) n++;

gridLogic[i][j] = n;

1. **Руководство для пользователя**

Ознакомится с правилами игры “ Сапёр ”.- I`m sorry boy)))

Запускаем программу и выбираем ячейку .

Управление :

• Левый click - открытие 1 ячейки

• Левый doubleclick - открытие нескольких ячеек

• Правый click - установка флажка (на ячейку , где предположительно находится мина), при попадании на мину игра заканчивается - GAME OVER . YOU OUT((((

Удачной игры !!!!!!!

**6. Написание программы и её работа :**

**6.1 Необходимые знания :**

Для написания кода игры "Сапёр" на C++ с использованием библиотеки SFML, вам понадобятся следующие знания и навыки:

▎**1. Основы программирования на C++ :**

• **Синтаксис C++**: Знание базовых конструкций языка, таких как переменные, операторы, условия (if, else), циклы (for, while), функции и массивы.

• **Работа с указателями и массивами**: Понимание того, как работать с указателями и динамическими массивами, особенно в контексте двумерных массивов.

▎**2. Библиотека SFML :**

• **Графика**: Умение использовать SFML для работы с графикой, включая создание окон, загрузку текстур и отрисовку спрайтов.

• **Обработка событий**: Знание, как обрабатывать события ввода (например, нажатия клавиш и клики мыши)

▎**3. Алгоритмы и структуры данных** :

• **Рекурсия**: Понимание рекурсивных алгоритмов, таких как openCells, которые открывают соседние ячейки в игре.

• **Логические структуры**: Знание о том, как организовать данные (например, логическая сетка игры) для эффективной работы с ними.

▎**4. Основы разработки игр** :

• **Игровая механика**: Понимание основ игровой механики, таких как правила игры "Сапёр", обработка состояния игры (открытие клеток, установка флажков).

• **Цикл игры**: Знание о том, как организовать основной цикл игры, который обрабатывает события и обновляет состояние игры.

▎**5. Инструменты разработки :**

• **Среда разработки (IDE)**: Опыт работы с IDE, такой как Visual Studio , для написания и отладки кода.

• **Система контроля версий**: Опыт работы с системами контроля версий (например, Git) для управления кодом.

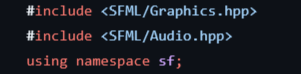
▎**6. Отладка и тестирование :**

• **Отладка кода**: Умение использовать инструменты отладки для нахождения и исправления ошибок в коде.

• **Тестирование**: Понимание важности тестирования кода для обеспечения его надёжности и корректности.

**6.2 Подробное описание функций и методов**

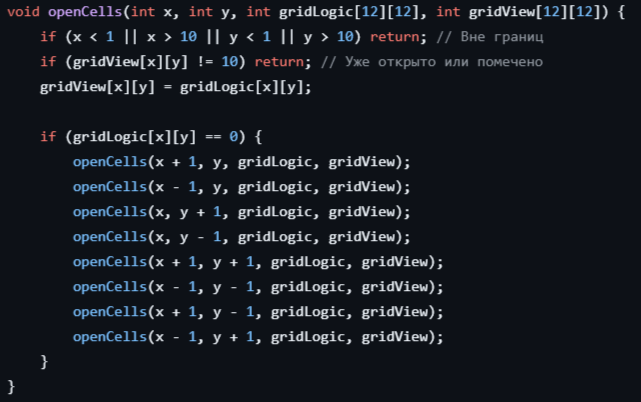
**1. Подключение библиотек**



• **SFML**: Библиотеки для работы с графикой и звуком

• **using namespace sf;** : Позволяет не указывать префикс sf:: перед классами и функциями из библиотеки SFML.

**▎2. Функция openCells**



**Параметры функции:**

• int x, int y: Координаты ячейки в сетке.

• int gridLogic[12][12]: Логическая сетка, где хранятся данные о количестве мин вокруг каждой ячейки.

• int gridView[12][12]: Сетка для отображения состояния ячеек (открыта/закрыта/помечена).

**Логика функции:**

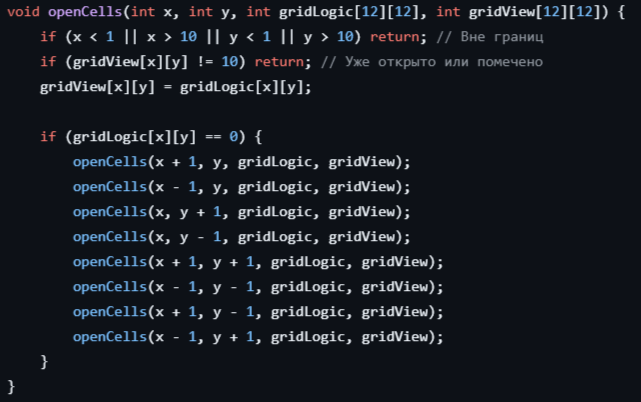
1. **Проверка границ**: Если координаты выходят за пределы сетки (от 1 до 10), функция завершает выполнение.

2025-01-19_17-44-04

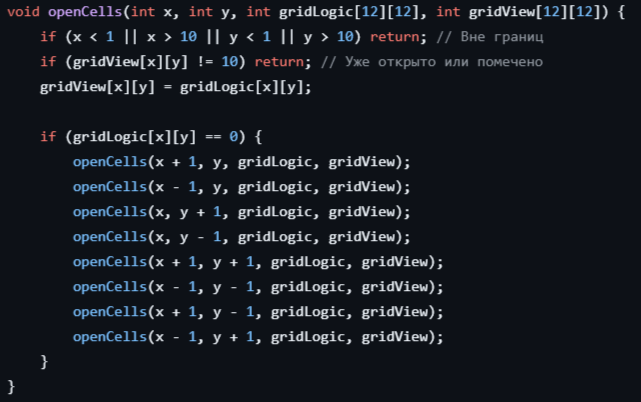
1. **Проверка состояния ячейки**: Если ячейка уже открыта или помечена (значение не равно 10), функция завершает выполнение.

2025-01-19_17-45-11

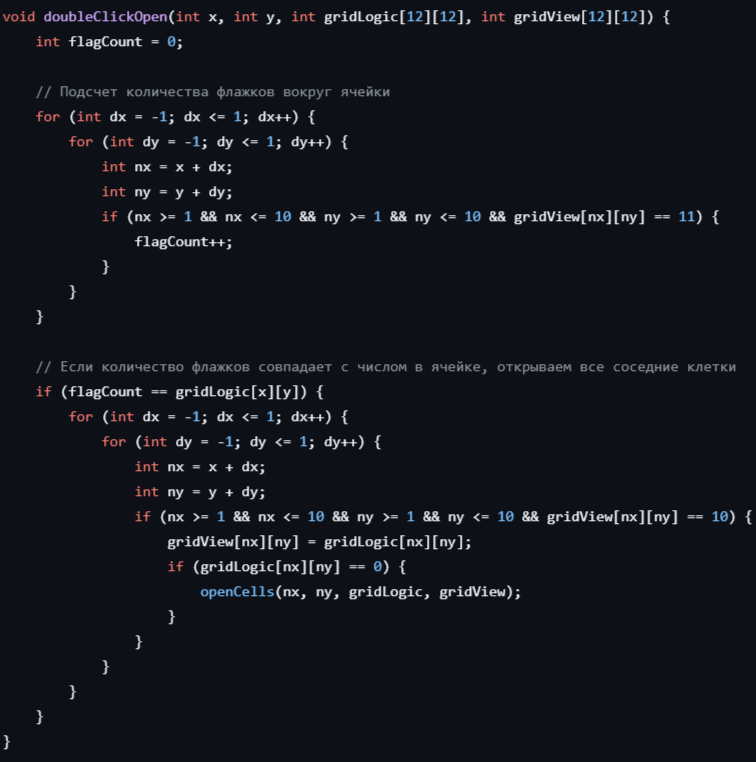
1. **Открытие ячейки**: Устанавливает значение ячейки в gridView равным значению из gridLogic.



**4)** **Рекурсивное открытие соседних ячеек**: Если текущее значение равно 0 (нет мин вокруг), функция рекурсивно вызывает себя для всех восьми соседних ячеек.



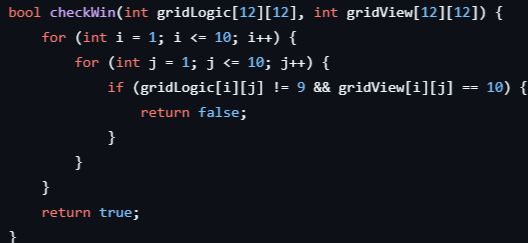
**▎3. Функция doubleClickOpen**

****

▎**Логика функции:**

1. **Подсчёт флажков**: Перебирает все соседние ячейки (всего 8) и считает количество флажков (значение 11) вокруг указанной ячейки.
2. **Проверка условия двойного клика**: Если количество флажков совпадает с числом в текущей ячейке (gridLogic[x][y]), это означает, что игрок правильно пометил все мины вокруг этой ячейки.
3. **Открытие соседних ячеек**: Перебирает все соседние ячейки и открывает их (обновляет значение в gridView). Если открытая ячейка содержит ноль, вызывается openCells, чтобы открыть все связанные с ней пустые ячейки.

**▎4. Функция checkWin :**



▎**Параметры функции**

• gridLogic[12][12]: Этот массив содержит логику игры. Он показывает, где находятся мины (значение 9) и сколько мин находится вокруг каждой ячейки (значения от 0 до 8).

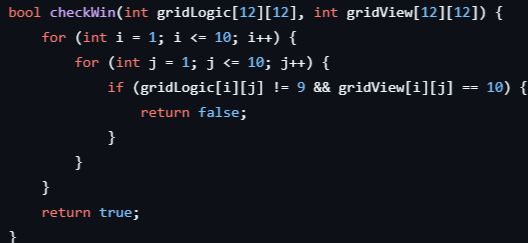
• gridView[12][12]: Этот массив хранит состояние ячеек, которые видит игрок. Значение 10 указывает, что ячейка закрыта, а другие значения могут указывать на открытые ячейки (например, количество мин вокруг или флаг мины).

▎**Пошаговый разбор:**

**1)Циклы по массивам**:

• Два вложенных цикла (for) проходят по всем ячейкам игрового поля от (1,1) до (10,10). Это означает, что функция проверяет каждую ячейку в пределах игрового поля.

**2)** **Проверка условий**:



Внутри циклов происходит проверка каждого элемента:

• Если значение в gridLogic[i][j] не равно 9 (то есть ячейка не содержит мину) и одновременно значение в gridView[i][j] равно 10 (то есть ячейка все еще закрыта), это означает, что игрок не открыл все безопасные ячейки.

• В этом случае функция возвращает false, указывая, что игрок еще не выиграл.

**3)Возврат значения**:

• Если после проверки всех ячеек не было найдено ни одной закрытой безопасной ячейки (то есть все безопасные ячейки открыты), функция завершает работу и возвращает true, что означает, что игрок выиграл.

**▎5. Функция main :**

**Создание окна приложения**

2025-01-19_18-07-27

• RenderWindow: Это класс из библиотеки SFML, который создает окно приложения.

• VideoMode(400, 400): Указывает размеры окна (ширина и высота в пикселях).

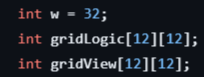
• "Minesweeper!": Заголовок окна.

**5.1Инициализация генератора случайных чисел:**

2025-01-19_18-09-14

• srand(time(0)): Инициализирует генератор случайных чисел текущим временем. Это позволяет получать разные последовательности случайных чисел при каждом запуске программы.

**5.2 Объявление переменных**



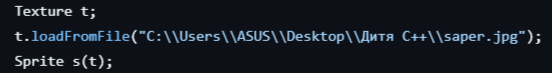
• int w = 32; : Ширина и высота каждой ячейки в пикселях.

• int gridLogic[12][12]; : Логическая сетка, где хранятся данные о количестве мин вокруг каждой ячейки.

• int gridView[12][12]; : Сетка для отображения состояния ячеек (открыта/закрыта/помечена).

**5.3 Загрузка текстуры фона**

saper

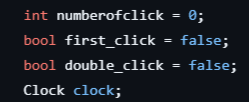


• Texture t;: Создает объект текстуры.

• t.loadFromFile(...): Загружает изображение из файла, которое будет использоваться в качестве фона или для клеток игры.

• Sprite s(t); : Создает спрайт на основе загруженной текстуры.

**5.4 Объявление переменных для кликов**



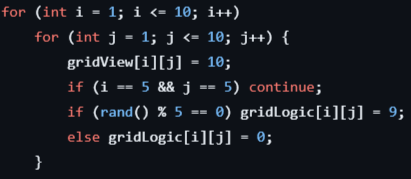
• int numberofclick = 0; : Эта переменная отслеживает количество кликов пользователя.

• first\_click: Флаг для отслеживания первого клика мыши.

• double\_click: Флаг для отслеживания двойного клика.

• Clock clock; : Создает объект часов, который будет использоваться для отслеживания времени между кликами.

**5.5 Инициализация сеток**



• **Двойной цикл**: Этот цикл инициализирует двумерные массивы gridView и gridLogic, представляющие игровое поле.

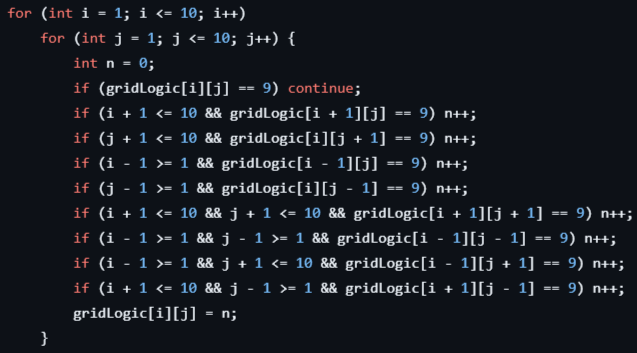
• gridView[i][j] = 10; : Устанавливает значение 10 для каждой ячейки в массиве gridView, что может означать, что ячейка закрыта.

• if (i == 5 && j == 5) continue; : Пропускает центральную ячейку (5,5), чтобы в ней не было мины.

• if (rand() % 5 == 0): С вероятностью 1 из 5 устанавливает значение 9 в массиве gridLogic, что, вероятно, означает наличие мины в этой ячейке.

• Если мина не установлена, значение устанавливается в 0.

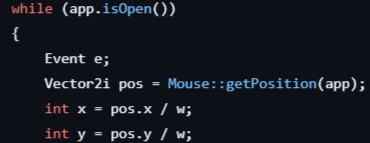
**5. 6 Подсчёт мин вокруг ячеек** :



• Подсчёт **соседних мин**: Этот код проходит по всем ячейкам и подсчитывает количество мин, находящихся вокруг каждой ячейки. Если ячейка содержит мину (gridLogic[i][j] == 9), она пропускается.

• Переменная n увеличивается на единицу каждый раз, когда находится мина в соседней ячейке. В конце цикла значение n присваивается текущей ячейке gridLogic[i][j].

**5.7 Основной игровой цикл :**



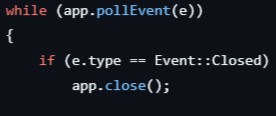
• Цикл работает до тех пор, пока окно приложения открыто (app.isOpen()).

• Event e;: Объект для хранения событий (например, нажатий клавиш или мыши).

• Vector2i pos = Mouse::getPosition(app) ;: Получает текущую позицию курсора мыши относительно окна приложения.

• int x = pos.x / w; и int y = pos.y / w;: Преобразует координаты мыши в индексы клеток сетки.

**5.8 Обработка событий**



• Цикл обрабатывает все события, которые произошли с момента последнего обновления.

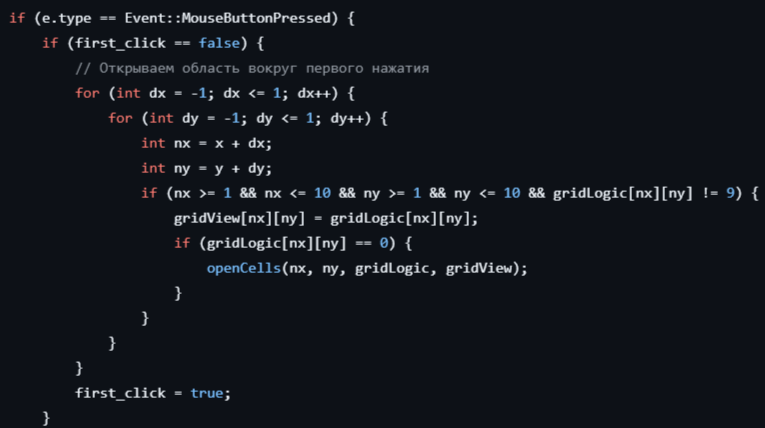
• Если событие — закрытие окна (Event::Closed), то приложение закрывается.

**5.9 Проверка нажатия кнопки мыши**

if (e.type == Event::MouseButtonPressed)

• Эта строка проверяет, произошло ли событие нажатия кнопки мыши. Если это так, выполняется следующий код.

**5.10 Обработка первого клика**



• **Проверка первого клика**: Если это первый клик (first\_click == false), выполняется код для открытия области вокруг нажатой ячейки.

• **Циклы** for: Эти вложенные циклы перебирают соседние ячейки вокруг текущей ячейки (x, y). dx и dy могут принимать значения -1, 0 и 1, что позволяет проверять все восемь соседних ячеек.

• **Проверка границ**: Условия if (nx >= 1 && nx <= 10 && ny >= 1 && ny <= 10) проверяют, находится ли соседняя ячейка в пределах игрового поля.

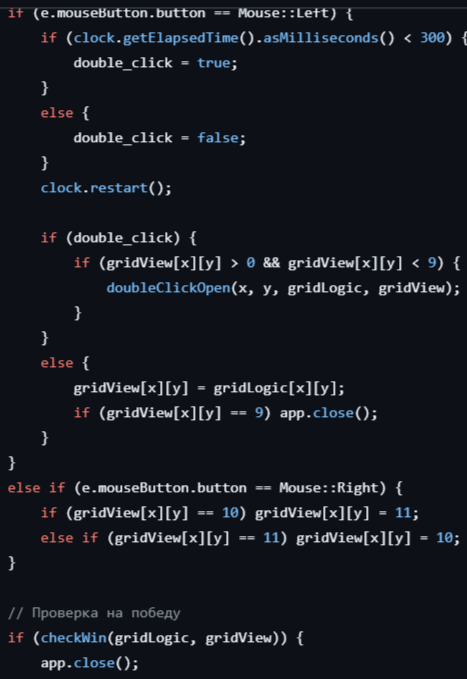
• **Проверка на мину**: Условие gridLogic[nx][ny] != 9 проверяет, не является ли соседняя ячейка миной.

• **Открытие ячейки**: Если ячейка безопасна (не содержит мину), ее значение из gridLogic копируется в gridView, чтобы отобразить ее состояние.

• **Рекурсивное открытие пустых ячеек**: Если открытая ячейка содержит значение 0 (gridLogic[nx][ny] == 0), вызывается функция openCells(nx, ny, gridLogic, gridView), которая рекурсивно открывает все связанные пустые ячейки.

• **Установка первого клика**: После обработки первого клика переменная first\_click устанавливается в true, чтобы последующие клики не выполняли этот блок кода.

**5.11 Обработка нажатия левой кнопки мыши**



**Проверка нажатия левой кнопки**:

• Если нажата левая кнопка мыши, выполняется следующий код.

**Проверка на двойной клик**:

• Сначала проверяется, прошло ли менее 300 миллисекунд с последнего нажатия.

• Если да, устанавливается переменная double\_click в true.

• В противном случае double\_click устанавливается в false.

**Сброс таймера**: Вызывается clock.restart(), чтобы сбросить таймер для следующей проверки двойного клика.

**Обработка двойного клика**:

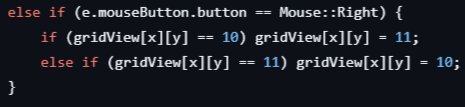
• Если это двойной клик и ячейка открыта (gridView[x][y] > 0 && gridView[x][y] < 9), вызывается функция doubleClickOpen(x, y, gridLogic, gridView), которая открывает все связанные ячейки.

**Обработка одиночного клика**:

• Если это не двойной клик, текущая ячейка открывается путем копирования значения из gridLogic в gridView.

• Если открыта ячейка с миной (gridView[x][y] == 9), приложение закрывается.

**5.12 Обработка нажатия правой кнопки мыши**



**Проверка нажатия правой кнопки**:

• Если нажата правая кнопка мыши, выполняется следующий код. • **Переключение состояния ячейки**:

• Если ячейка закрыта (gridView[x][y] == 10), она помечается как флажок (gridView[x][y] = 11).

• Если ячейка уже помечена (gridView[x][y] == 11), она возвращается в закрытое состояние (gridView[x][y] = 10).

**5.13 Проверка на победу**



**Проверка на победу**: После обработки всех кликов проверяется, выиграл ли игрок, вызвав функцию checkWin(gridLogic, gridView). **Закрытие основного приложения**:

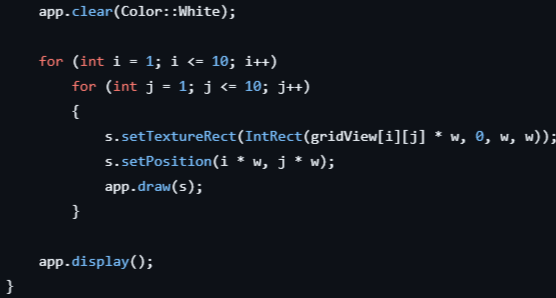
• Если игрок выиграл, основное приложение закрывается. **Создание окна поздравления**:

• Создаётся новое окно с поздравлением. Внутри этого окна обрабатываются события и отображается текст "Congratulations!" с использованием шрифта Arial.

**Цикл обработки событий окна поздравления**: В этом цикле обрабатываются события для нового окна.

• Если окно закрывается, оно закрывается.

**5.14 Отрисовка игры**



• app.clear(Color::White); : Очищает экран белым цветом перед отрисовкой нового кадра.

• Двойной цикл проходит по всем ячейкам сетки:

• Устанавливает прямоугольник текстуры спрайта в зависимости от состояния ячейки (gridView[i][j] \* w).

• Устанавливает позицию спрайта на основе индексов ячеек.

• Рисует спрайт на экране с помощью app.draw(s);.

• app.display();: Отображает все нарисованные объекты на экране.