Минестерство образоваения Респубилики Беларусь

Учреждение образования

“Белорусский государственный технологический университет”

Факультет информационных технологий

Основы программной инженерии

**Отчет к проекту “Игра Змейка”**

**Выполнили**:

Студенты 1 курса 9 группы ПИ

Борисов Никита Андреевич

Даркович Денис Сергеевич

Ермоленко Станислав Андреевич

Володьков Дмитрий Александрович

Аманов Артур Русланович

**Проверяющий:**

Старший преподаватель

Архипенко Ольга Александровна

Мінск 2023

***НАВИГАЦИЯ***

* [***Введение***](#Введение)
* [***Описание режимов***](#Описание_режимов)
* [***Функции кода модуля Game.cpp***](#gamecpp)
* [***Функции кода модуля easyGameMode.cpp***](#ezgamemode)
* [***Заключение***](#заключение)

**Введение**

Проект "Игра Змейка" на C++ был разработан с целью создания классической аркадной игры с нововведениями . Игра была расширена с возможностью выбора режимов игры и сохранения личного рекорда. В проекте созданы два режима: "Hard" и "Easy". В этом отчете будет представлена информация о режимах, их функциональности и реализации.

**Описание режимов**

**- Режим 1: Easy**

- **Функциональность:**

- Управление змейкой с помощью клавиш.

- Змейка при выходе за границу поля появляется с противоположной стороны поля.

- Увеличение и уменьшение длины змейки при съедании пищи.

- Увеличение и уменьшение рекорда при съедании пищи.

- По мере увеличения счета, границы игрового поля начинают расширяться, для продолжения игры.

- Генерация пищи: обычный фрукт, дающий игроку +1 к рекорду и увеличивает змейку на 1 символ; специальный фрукт, который имеет размер 2x2, для отличия от остальных, и дающий бонус в виде +5 очков к рекорду и уменьшения размера змейки на 1 символ; мистический фрукт, который имеет 1 из 3 случайных бонусов: 1) + 1 очко к рекорду и увелечение змейки на 1 символ. 2) - 1 очко от рекорда и уменьшение змейки на 1 символ. 3) + 5 очков к рекорду.

- Отображение текущего счета и сохранение результата в таблицу лидеров.

**- Цель:**

- Собрать как можно больше фруктов и получить максимальный счет.

**- Режим 2: Hard**

**- Функциональность:**

- Управление змейкой с помощью клавиш.

- Врезание в стены приводит к смерти змейки и завершению игры.

- По мере увеличения счета, границы игрового поля начинают сужаться, создавая больший вызов для игрока.

- Увеличение длины змейки при съедании пищи.

- Увеличение рекорда при съедании пищи.

- Генерация пищи: 3 обычных фрукта, дающий игроку +1 к рекорду и увеличивает змейку на 1 символ;

- Отображение текущего счета и сохранение результата в таблицу лидеров.

**- Цель:**

- Собрать как можно больше фруктов и выжить как можно дольше, управляя змейкой на уменьшающейся области поля.

**Функции кода модуля Game.cpp**

**Библиотеки:**

**iostream (Input/Output Stream):**

iostream предоставляет основные средства для ввода и вывода в стандартные потоки (консоль).

Включает классы cin (стандартный ввод), cout (стандартный вывод) и cerr (стандартный вывод ошибок).

**fstream (File Stream)**:

fstream предоставляет классы для работы с файловыми потоками, что позволяет считывать данные из файлов и записывать данные в файлы.

Включает классы ifstream (ввод из файла), ofstream (вывод в файл) и fstream (ввод/вывод из/в файл).

**iomanip (Input/Output Manipulators):**

iomanip содержит манипуляторы для управления форматированием ввода и вывода.

Используется, например, для установки ширины полей вывода, управления точностью вывода чисел с плавающей запятой и других параметров форматирования.

**conio.h (Console Input/Output):**

conio.h предоставляет функции для работы с консольным вводом и выводом.

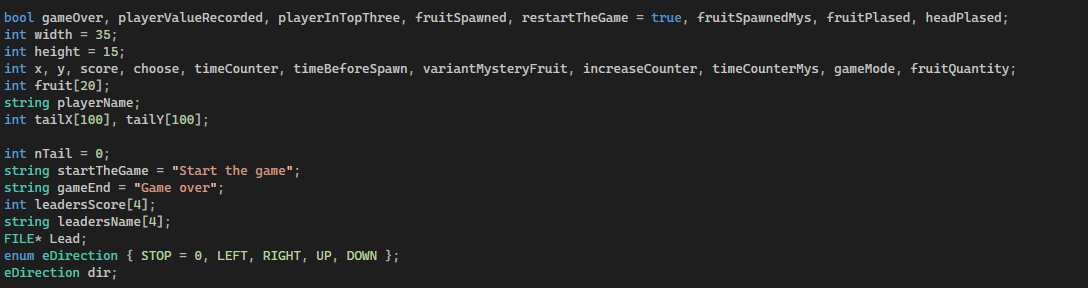
Включает функции, такие как getch(), которая считывает символы с клавиатуры без их отображения на экране, что может быть полезно для реализации интерактивных консольных приложений.

**windows.h:**

windows.h является заголовочным файлом для работы с API операционной системы Windows.

Включает различные функции, используемые для взаимодействия с операционной системой Windows, такие как функции работы с окнами, обработки сообщений, управления процессами и т. д.

**Определение переменных и инициализация**

****

Переменные состояния игры:

gameOver: Флаг, указывающий на завершение игры.

playerValueRecorded: Флаг, указывающий, был ли записан результат игрока.

playerInTopThree: Флаг, указывающий, находится ли игрок в топ-3 лучших результатов.

fruitSpawned: Флаг, указывающий, был ли создан плод.

restartTheGame: Флаг, указывающий, нужно ли перезапустить игру.

fruitSpawnedMys: Флаг, указывающий, был ли создан "мистический" плод.

fruitPlased: Флаг, указывающий, был ли размещен обычный плод.

headPlased: Флаг, указывающий, была ли размещена голова.

Переменные размерности поля и координат головы змеи:

width и height: Ширина и высота игрового поля.

x и y: Координаты головы змеи на игровом поле.

Переменные для управления игрой:

score: Текущий счет игрока.

choose: Переменная для выбора опций (вероятно, связанных с игровым меню).

timeCounter, timeBeforeSpawn: Счетчики времени для управления игровыми событиями, такими как появление фруктов.

variantMysteryFruit: Вариант "мистического" фрукта.

increaseCounter: Счетчик увеличения длины змеи.

timeCounterMys: Счетчик времени для "мистического" фрукта.

gameMode: Режим игры (возможно, различные уровни сложности).

fruitQuantity: Количество фруктов.

Массивы и строки:

fruit: Массив для хранения информации о фруктах.

playerName: Строка, содержащая имя игрока.

tailX и tailY: Массивы для хранения координат сегментов хвоста змеи.

Дополнительные переменные:

nTail: Длина хвоста змеи.

startTheGame и gameEnd: Строковые константы, представляющие фразы "Start the game" и "Game over".

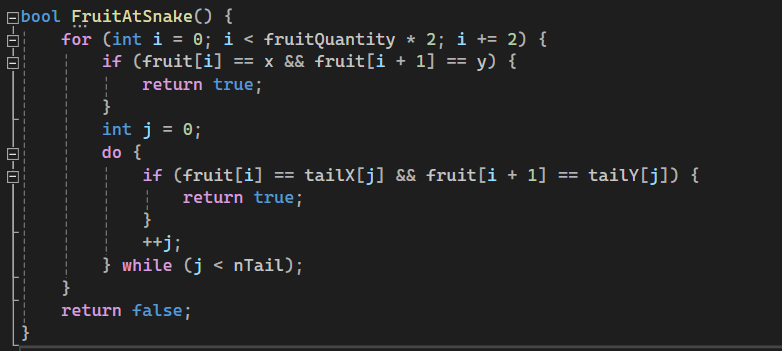
leadersScore и leadersName: Массивы для хранения информации о лучших результатах (рекордах) игры.

Lead: Указатель на файл (вероятно, для сохранения/загрузки результатов).

Перечисление направлений:

enum eDirection: Перечисление для указания направлений движения змеи (STOP, LEFT, RIGHT, UP, DOWN).

**Функция, FruitAtSnake()**

****

Данная функция предназначена для проверки наличия фрукта в теле змеи.

Цикл по фруктам:

for (int i = 0; i < fruitQuantity \* 2; i += 2): Цикл проходит по всем фруктам в массиве fruit. Индекс i увеличивается на 2, так как каждый фрукт представлен парой координат (x, y), и i соответствует x-координате, а i + 1 - y-координате.

Проверка координат:

if (fruit[i] == x && fruit[i + 1] == y): Проверка, находится ли голова змеи в том же месте, что и фрукт. Если да, то функция возвращает true, указывая на то, что фрукт находится в голове змеи.

Проверка хвоста:

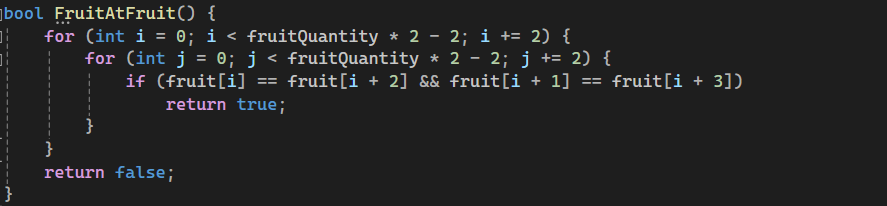
Затем следует вложенный цикл do-while, в котором проверяется, совпадают ли координаты фрукта с координатами сегментов хвоста змеи.

if (fruit[i] == tailX[j] && fruit[i + 1] == tailY[j]): Если есть совпадение с одним из сегментов хвоста, функция также возвращает true.

Возврат значения:

Если ни в голове, ни в хвосте не обнаружено совпадений, функция возвращает false, указывая на отсутствие фрукта в змее.

**Функция FruitAtFruit()**



Эта функция служит для проверки наличия пересечений между фруктами в игре.

Эта функция может быть полезна в игре для предотвращения ситуации, когда два или более фрукта находятся в одном и том же месте, что может быть нежелательным с точки зрения геймплея.

Двойной цикл по фруктам:

for (int i = 0; i < fruitQuantity \* 2 - 2; i += 2): Внешний цикл проходит по всем фруктам, кроме последнего, так как внутренний цикл сравнивает текущий фрукт с каждым фруктом после него.

for (int j = 0; j < fruitQuantity \* 2 - 2; j += 2): Внутренний цикл также проходит по всем фруктам, кроме последнего.

Проверка координат:

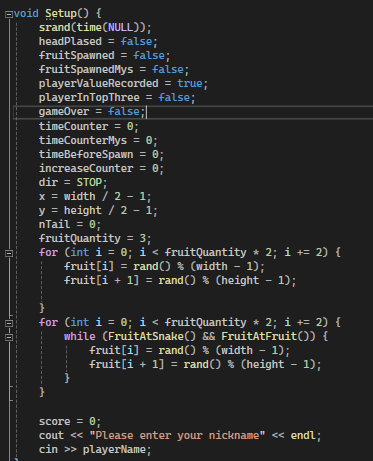
if (fruit[i] == fruit[i + 2] && fruit[i + 1] == fruit[i + 3]): Внутри вложенного цикла проверяется, совпадают ли координаты текущего фрукта с координатами каждого следующего фрукта в последовательности.

Если есть совпадение, функция возвращает true, указывая наличие пересечения между фруктами.

Возврат значения:

Если не обнаружено пересечений между фруктами, функция возвращает false.

**Функция Setup()**

****

Функция предназначена для настройки начальных условий перед началом игры.

Эта функция вызывается перед началом каждой новой игры и готовит все необходимые параметры и начальные условия для запуска игрового цикла.

Инициализация генератора случайных чисел:

srand(time(NULL));: Инициализация генератора случайных чисел на основе текущего времени. Это делается для того, чтобы каждый раз, когда программа запускается, генератор создавал новую последовательность случайных чисел.

Инициализация переменных:

Различные переменные, такие как headPlased, fruitSpawned, playerValueRecorded, и другие, инициализируются в соответствии с начальными значениями.

dir устанавливается в STOP, указывая на то, что змея в начале не двигается.

x и y устанавливаются в центр игрового поля.

nTail (длина хвоста) устанавливается в 0.

fruitQuantity устанавливается в 3 (количество фруктов).

Генерация координат фруктов:

Цикл for заполняет массив fruit случайными координатами фруктов. Каждый фрукт представлен парой координат (x, y).

Генерация происходит в пределах размеров игрового поля (width и height - 1).

Проверка пересечений:

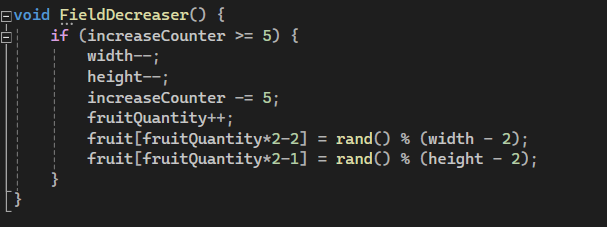
После генерации координат фруктов, используются циклы, чтобы убедиться, что новые фрукты не пересекаются с змеей (FruitAtSnake()) и между собой (FruitAtFruit()).

В случае пересечений фрукты генерируются заново.

Ввод имени игрока:

Пользователь приглашается ввести свой никнейм (playerName).

**Функция FieldDecreaser()**

****

Функция FieldDecreaser() выполняет уменьшение размеров игрового поля при достижении определенного условия.

Эта функция использоваться в игре для создания эффекта усложнения с течением времени, когда игровое поле становится меньше, что требует от игрока больше усилий для избегания столкновений и увеличивает сложность игры.

Условие уменьшения поля:

if (increaseCounter >= 5): Проверяет, достигло ли количество увеличений (increaseCounter) порогового значения 5. Если это условие выполняется, то выполняются действия по уменьшению поля.

Уменьшение размеров поля:

width--; и height--;: Уменьшение ширины и высоты игрового поля на единицу. Это сужает границы игрового поля.

Увеличение количества фруктов:

fruitQuantity++;: Увеличение переменной, отвечающей за количество фруктов.

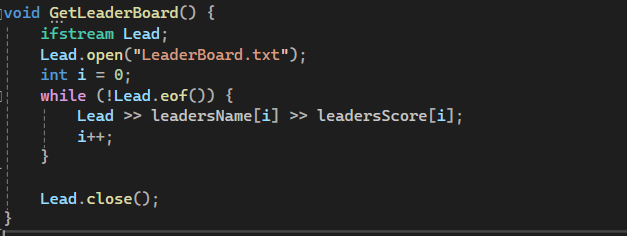
Генерация нового фрукта:

fruit[fruitQuantity\*2-2] = rand() % (width - 2); и fruit[fruitQuantity\*2-1] = rand() % (height - 2);: Генерация новых координат для нового фрукта. Координаты ограничиваются новыми размерами поля, чтобы новый фрукт не выходил за границы.

Сброс счетчика увеличений:

increaseCounter -= 5;: Уменьшение счетчика увеличений, чтобы следующее увеличение могло быть отслежено.

**Функция etLeaderBoard()**



Функция предназначена для считывания информации из файла "LeaderBoard.txt" и заполнения массивов leadersName и leadersScore данными о лидерах (игроках с наивысшими результатами).

Эта функция предполагает, что файл "LeaderBoard.txt" содержит данные в формате, где каждая строка представляет собой пару: имя игрока и его счет. Эти данные затем сохраняются в массивах leadersName и leadersScore для дальнейшего использования, например, для отображения таблицы лидеров в игре.

Открытие файла:

ifstream Lead;: Создается объект ifstream с именем Lead, предназначенный для считывания данных из файла.

Lead.open("LeaderBoard.txt");: Открывается файл "LeaderBoard.txt" для чтения.

Считывание данных из файла:

int i = 0;: Инициализация счетчика для индексации массивов leadersName и leadersScore.

while (!Lead.eof()) {: Цикл выполняется, пока не достигнут конец файла.

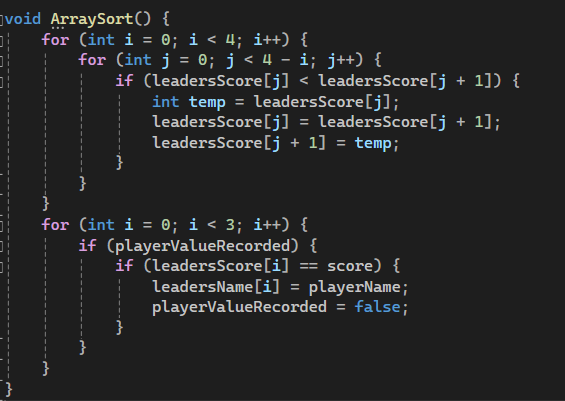
Lead >> leadersName[i] >> leadersScore[i];: Из каждой строки файла считываются имя игрока и его счет и сохраняются в соответствующие массивы.

i++;: Инкрементация счетчика для перехода к следующей записи в массивах.

Закрытие файла:

Lead.close();: Закрытие файла после того, как все данные были считаны.

**Функция ArraySort()**

****

Функция предназначена для сортировки массива leadersScore в порядке убывания и обновления массива leadersName в соответствии с новым порядком счетов.

Эта функция используется для обновления и сортировки данных в таблице лидеров после каждого завершенного раунда игры.

Сортировка:

Для сортировки массива leadersScore используется алгоритм сортировки.

Внешний цикл for (int i = 0; i < 4; i++) проходит по элементам массива.

Внутренний цикл for (int j = 0; j < 4 - i; j++) сравнивает пары элементов и меняет их местами, если текущий элемент больше следующего.

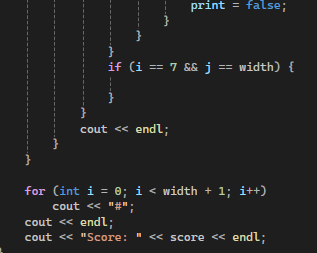
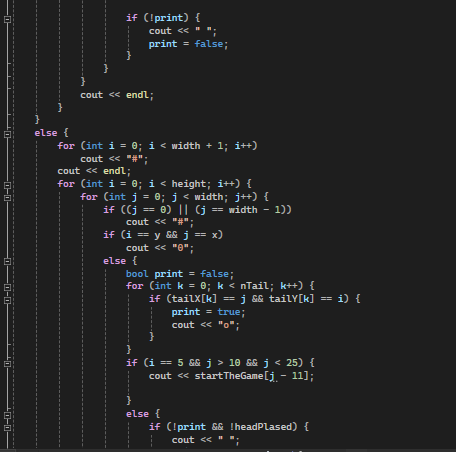
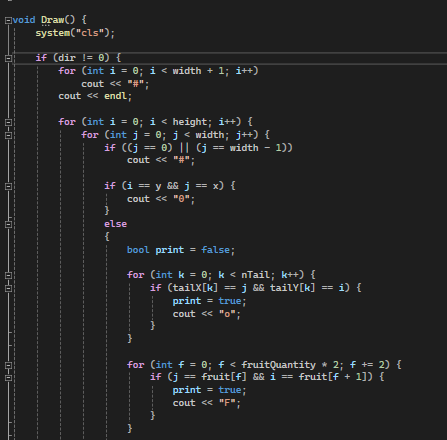
Обновление массива leadersName:

После сортировки массива leadersScore, следующий цикл for (int i = 0; i < 3; i++) обновляет массив leadersName с учетом нового порядка счетов.

Внутри цикла проверяется, были ли уже записаны данные об игроке (переменная playerValueRecorded).

Если запись еще не выполнена, и текущий элемент массива leadersScore совпадает со счетом игрока (leadersScore[i] == score), то имя игрока записывается в массив leadersName.

**Функция Draw()**



Функция отвечает за отрисовку игрового поля, змеи, фруктов и других элементов на экране консоли.

Функция визуализирует текущее состояние игры, отображая змею, фрукты и счет на экране консоли. В зависимости от значения dir она отображает различные элементы в соответствии с текущим состоянием игры.

Очистка экрана:

system("cls");: Эта команда очищает экран консоли. Помогает создать эффект анимации, когда новое состояние игры отображается поверх предыдущего.

Отрисовка границы поля:

В цикле for (int i = 0; i < width + 1; i++) отрисовывается верхняя граница поля.

Отрисовка элементов внутри поля:

Далее идет двойной цикл для отрисовки каждой ячейки игрового поля.

Если dir не равен нулю (змея движется), то отрисовываются элементы игрового поля (стены, змея, фрукты и т. д.).

Если dir равен нулю (игра еще не началась), отрисовывается начальный экран, где отображается сообщение "Start the game".

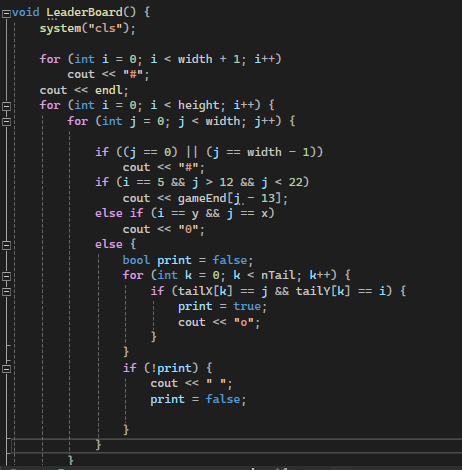
Отрисовка нижней границы поля:

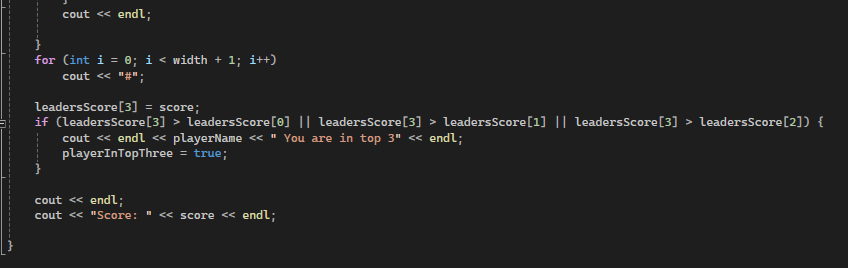
После цикла для отрисовки игрового поля, снова отрисовывается нижняя граница поля.

Отображение счета игрока:

cout << "Score: " << score << endl;: Отображение текущего счета игрока.

**Функция LeaderBoard()**

****

****

Функция отвечает за отрисовку экрана таблицы лидеров (Leaderboard) в консоли после завершения игры.

Эта функция предназначена для отображения таблицы лидеров после завершения игры и предоставляет игроку информацию о его месте в рейтинге.

Очистка экрана:

system("cls");: Эта команда очищает экран консоли перед отрисовкой таблицы лидеров.

Отрисовка верхней границы поля:

for (int i = 0; i < width + 1; i++) cout << "#";: Выводится верхняя граница поля.

Отрисовка элементов внутри поля:

Двойной цикл отрисовывает каждую ячейку таблицы лидеров.

Если i == 5 и j находится в определенном диапазоне, то выводится сообщение о завершении игры (gameEnd).

Если i == y && j == x, то отрисовывается голова змеи.

В противном случае проверяется, является ли текущая ячейка частью тела змеи, и если да, то отрисовывается сегмент хвоста "o". В противном случае отображается пробел.

Отрисовка нижней границы поля:

for (int i = 0; i < width + 1; i++) cout << "#";: Отрисовывается нижняя граница поля.

Определение места игрока в рейтинге:

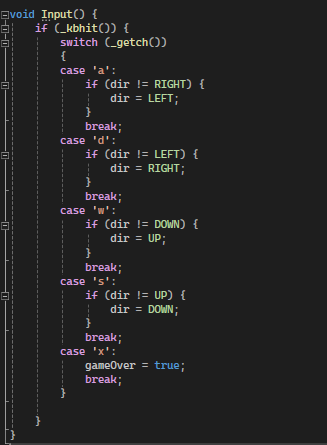
leadersScore[3] = score;: Последний элемент массива leadersScore обновляется с текущим счетом игрока.

Затем проверяется, превосходит ли счет игрока счеты в топ-3. Если да, то выводится сообщение, что игрок вошел в топ-3.

Вывод счета игрока:

cout << "Score: " << score << endl;: Выводится текущий счет игрока.

**Функция Input()**

****

Функция отвечает за обработку пользовательского ввода с клавиатуры во время игры.

Эта функция обрабатывает ввод пользователя и изменяет направление движения змеи в соответствии с нажатыми клавишами, а также позволяет завершить игру, если нажата клавиша 'x'.

Проверка наличия клавиш:

\_kbhit(): Эта функция возвращает ненулевое значение (true), если есть символ, ожидающий ввода с клавиатуры, и 0 (false) в противном случае. Таким образом, эта проверка определяет, была ли нажата клавиша.

Обработка нажатых клавиш:

\_getch(): Эта функция считывает символ с клавиатуры. Символ затем используется в конструкции switch для определения действия, соответствующего нажатой клавише.

Изменение направления змеи:

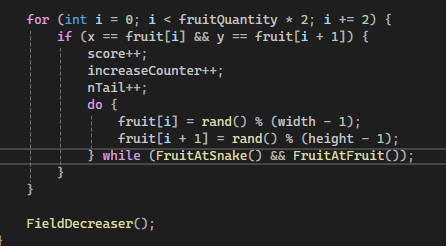
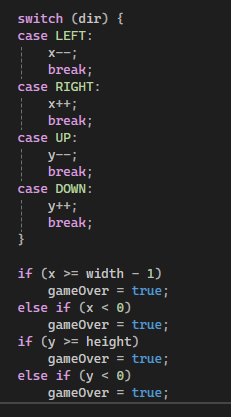
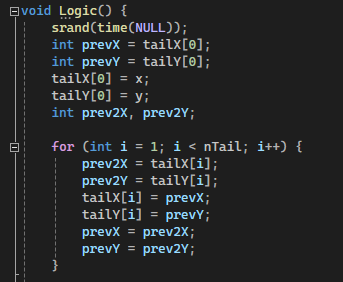
В зависимости от нажатой клавиши и текущего направления змеи (dir), происходит изменение направления.

Например, если нажата клавиша 'a' (влево), и змея не движется вправо, то dir устанавливается в LEFT, указывая, что змея должна двигаться влево.

Завершение игры:

Если нажата клавиша 'x', устанавливается флаг gameOver в true, что приведет к завершению игры.

**Функция Logic()**



Функция отвечает за логику игры, включая движение змеи, проверку условий завершения игры, обработку столкновений с хвостом и фруктами, Эта функция представляет собой центральную часть игровой логики, обеспечивая движение змеи, обработку столкновений и взаимодействие с фруктами и размерами поля.

а также управление размерами поля.

Обновление координат хвоста:

Сначала сохраняются текущие координаты головы змеи в переменные prevX и prevY.

Затем происходит обновление координат хвоста, где предыдущие координаты хвоста (prevX и prevY) копируются в следующий сегмент хвоста.

Обработка движения змеи:

В зависимости от текущего направления (dir), обновляются координаты головы змеи (x и y).

Проверка на столкновение с границами поля:

Если координата x выходит за правую или левую границу поля, или координата y выходит за верхнюю или нижнюю границу поля, устанавливается флаг gameOver, что приведет к завершению игры.

Проверка на столкновение с хвостом:

Происходит цикл, который проверяет, не столкнулась ли голова змеи с ее хвостом. Если такое столкновение обнаружено, устанавливается флаг gameOver.

Проверка на сбор фрукта:

Происходит цикл, который проверяет, не совпадают ли координаты головы змеи с координатами фрукта. Если они совпадают, выполняются следующие действия:

Увеличивается счет (score).

Увеличивается счетчик увеличения поля (increaseCounter).

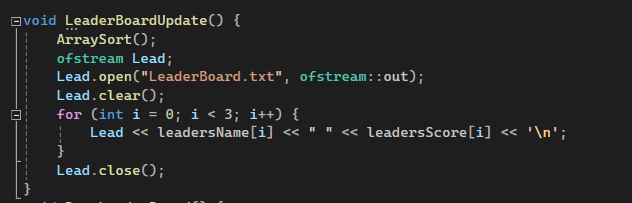
Увеличивается длина хвоста (nTail).

Генерируется новая позиция для фрукта с использованием цикла do-while, чтобы убедиться, что новый фрукт не пересекается с змеей или другими фруктами.

Управление размерами поля:

Вызывается функция FieldDecreaser(), которая уменьшает размеры игрового поля и увеличивает количество фруктов, если выполнено определенное условие (когда increaseCounter достигает 5).

**Функция LeaderBoardUpdate()**

****

Функция предназначена для обновления данных в файле "LeaderBoard.txt" с информацией о лидерах (таблица лидеров).

Эта функция обновляет файл "LeaderBoard.txt" с новой информацией о лидерах после каждого завершенного раунда игры, сохраняя только топ-3 игроков.

Сортировка массива лидеров:

Вызывается функция ArraySort(), которая сортирует массивы leadersName и leadersScore в порядке убывания счетов.

Открытие файла для записи:

ofstream Lead;: Создается объект ofstream (поток вывода в файл) с именем Lead.

Lead.open("LeaderBoard.txt", ofstream::out);: Открывается файл "LeaderBoard.txt" для записи. Флаг ofstream::out указывает, что файл будет открыт для записи, и его существующее содержимое будет удалено.

Очистка файла:

Lead.clear();: Очищает содержимое файла, чтобы затем записать в него обновленные данные.

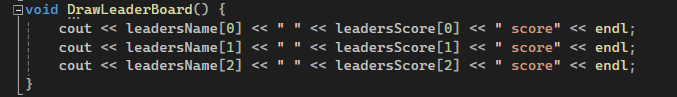
Запись данных в файл:

В цикле записываются данные о лидерах (только топ-3) в файл в формате "Имя Счет". Каждая запись разделяется символом новой строки '\n'.

Закрытие файла:

Lead.close();: Закрывает файл после того, как все данные были записаны.

**Функция DrawLeaderBoard()**

****

Функция предназначена для вывода информации о лидерах (таблицы лидеров) в консоль.

Функция просто выводит в консоль информацию о топ-3 лидерах, предоставляя игрокам представление о текущем состоянии таблицы лидеров.

Вывод данных о лидерах:

cout << leadersName[0] << " " << leadersScore[0] << " score" << endl;: Выводит в консоль информацию о первом лидере, включая имя и счет.

cout << leadersName[1] << " " << leadersScore[1] << " score" << endl;: Выводит информацию о втором лидере.

cout << leadersName[2] << " " << leadersScore[2] << " score" << endl;: Выводит информацию о третьем лидере.

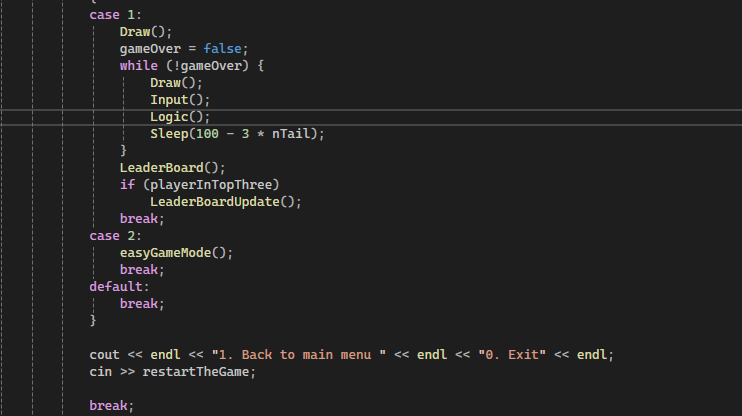
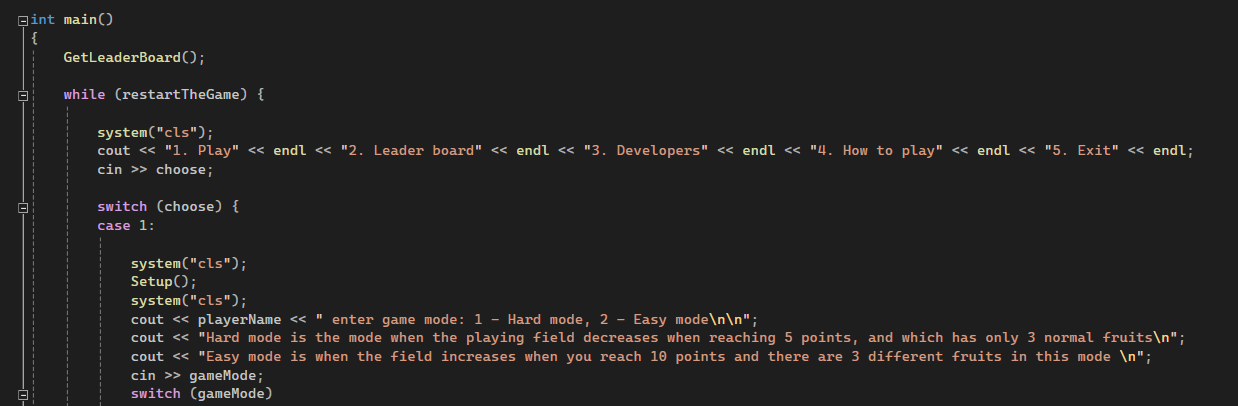
Формат вывода:

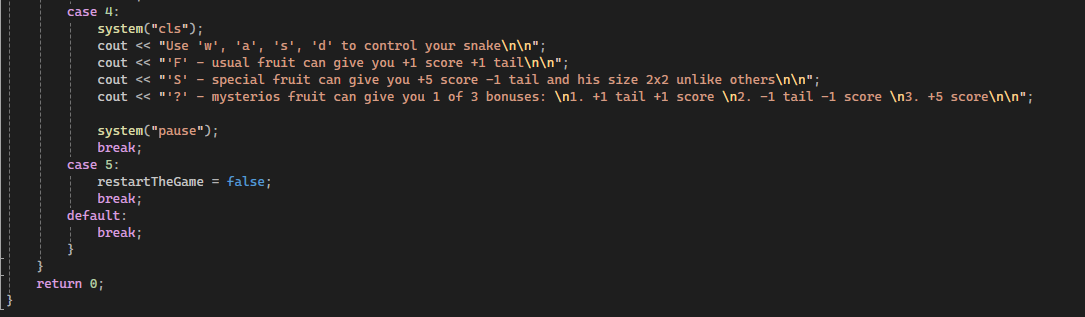
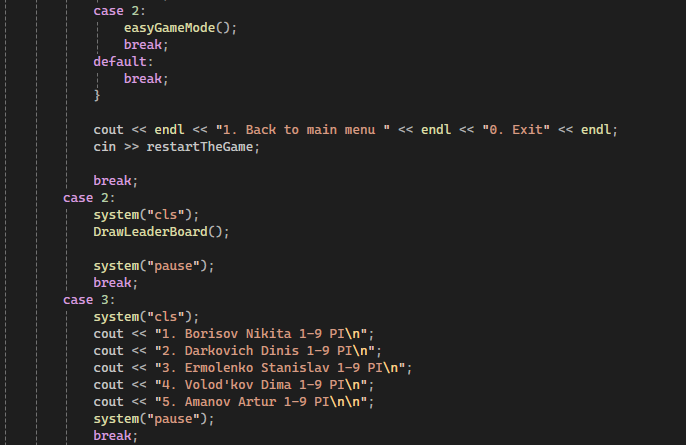
Каждая строка выводится в формате "Имя Счет score", где leadersName[i] - имя лидера, leadersScore[i] - его счет, а "score" - текстовая метка, указывающая на то, что следует ожидать информацию о счете.

Завершение строки:

endl: Добавляет символ новой строки, чтобы следующий вывод начинался с новой строки.

**Функция main()**

****



Представляет собой функция консольное приложение на C++, реализующее простую игру в змейку с различными режимами и возможностью просмотра таблицы лидеров.

Код использует различные функции для управления игровым процессом, отображением, обработкой ввода и логикой игры.

Главная функция main():

GetLeaderBoard(): Загружает данные таблицы лидеров из файла при запуске программы.

while (restartTheGame): Запускает бесконечный цикл, который будет выполняться до тех пор, пока флаг restartTheGame установлен в true.

Отображение главного меню:

Пользователю предлагается выбор между разными опциями: игра, таблица лидеров, информация о разработчиках, инструкция по игре и выход из программы.

Игровой процесс:

При выборе опции "Play" (choose == 1), игроку предоставляется выбор между сложным и легким режимами.

Если выбран сложный режим (gameMode == 1), запускается игра в змейку (Draw(), Input(), Logic()) с дополнительными функциями для отображения лидеров и обновления таблицы лидеров.

Если выбран легкий режим (gameMode == 2), запускается соответствующая функция easyGameMode().

Отображение информации о разработчиках, инструкции по игре и выход:

При выборе соответствующих опций выводится информация о разработчиках, инструкция по управлению змейкой и игре, либо программа завершает выполнение.

Цикл завершается и программа возвращается к началу, если restartTheGame установлен в true.

**Функции кода модуля easyGameMode.cpp**

**Библиотеки:**

**iostream (Input/Output Stream):**

iostream предоставляет основные средства для ввода и вывода в стандартные потоки (консоль).

Включает классы cin (стандартный ввод), cout (стандартный вывод) и cerr (стандартный вывод ошибок).

**fstream (File Stream)**:

fstream предоставляет классы для работы с файловыми потоками, что позволяет считывать данные из файлов и записывать данные в файлы.

Включает классы ifstream (ввод из файла), ofstream (вывод в файл) и fstream (ввод/вывод из/в файл).

**iomanip (Input/Output Manipulators):**

iomanip содержит манипуляторы для управления форматированием ввода и вывода.

Используется, например, для установки ширины полей вывода, управления точностью вывода чисел с плавающей запятой и других параметров форматирования.

**conio.h (Console Input/Output):**

conio.h предоставляет функции для работы с консольным вводом и выводом.

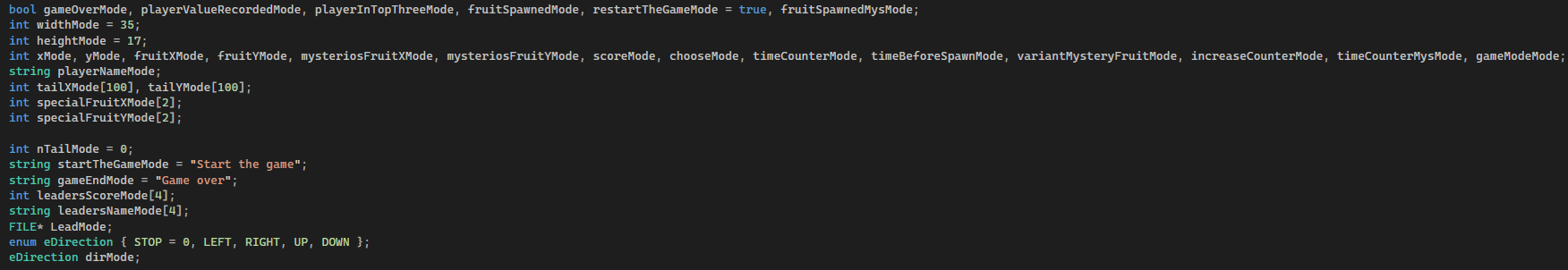
Включает функции, такие как getch(), которая считывает символы с клавиатуры без их отображения на экране, что может быть полезно для реализации интерактивных консольных приложений.

**windows.h:**

windows.h является заголовочным файлом для работы с API операционной системы Windows.

Включает различные функции, используемые для взаимодействия с операционной системой Windows, такие как функции работы с окнами, обработки сообщений, управления процессами и т. д.

**Определение переменных и инициализация**

****

Переменные состояния игры:

gameOverMode: Флаг, указывающий на завершение игры.

playerValueRecordedMode: Флаг, указывающий, был ли записан результат игрока.

playerInTopThreeMode: Флаг, указывающий, находится ли игрок в топ-3 лучших результатов.

fruitSpawnedMode: Флаг, указывающий, был ли создан плод.

restartTheGameMode: Флаг, указывающий, нужно ли перезапустить игру.

fruitSpawnedMysMode: Флаг, указывающий, был ли создан "мистический" плод.

Переменные размерности поля и координат головы змеи:

widthMode и heightMode: Ширина и высота игрового поля.

xMode и yMode: Координаты головы змеи на игровом поле.

Переменные для управления игрой:

scoreMode: Текущий счет игрока.

chooseMode: Переменная для выбора опций (вероятно, связанных с игровым меню).

timeCounter, timeBeforeSpawn: Счетчики времени для управления игровыми событиями, такими как появление фруктов.

variantMysteryFruitMode: Вариант "мистического" фрукта.

increaseCounterMode: Счетчик увеличения длины змеи.

timeCounterMysMode: Счетчик времени для "мистического" фрукта.

gameModeMode: Режим игры (возможно, различные уровни сложности).

fruitQuantityMode: Количество фруктов.

Массивы и строки:

fruitMode: Массив для хранения информации о фруктах.

playerNameMode: Строка, содержащая имя игрока.

tailXMode и tailYMode: Массивы для хранения координат сегментов хвоста змеи.

Дополнительные переменные:

nTaiModel: Длина хвоста змеи.

startTheGameMode и gameEndMode: Строковые константы, представляющие фразы "Start the game" и "Game over".

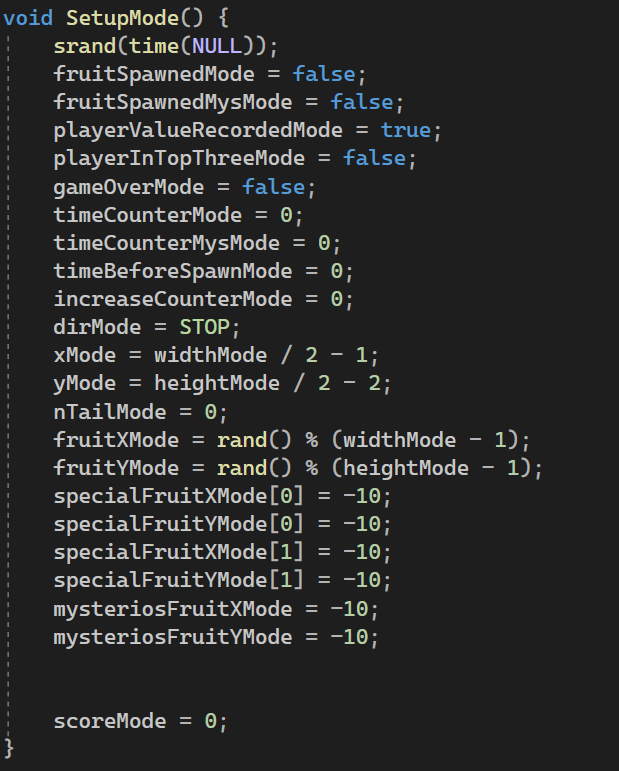
leadersScoreMode и leadersNameMode: Массивы для хранения информации о лучших результатах (рекордах) игры.

LeadMode: Указатель на файл (вероятно, для сохранения/загрузки результатов).

Перечисление направлений:

enum eDirection: Перечисление для указания направлений движения змеи (STOP, LEFT, RIGHT, UP, DOWN).

**Функция SetupMode()**

****

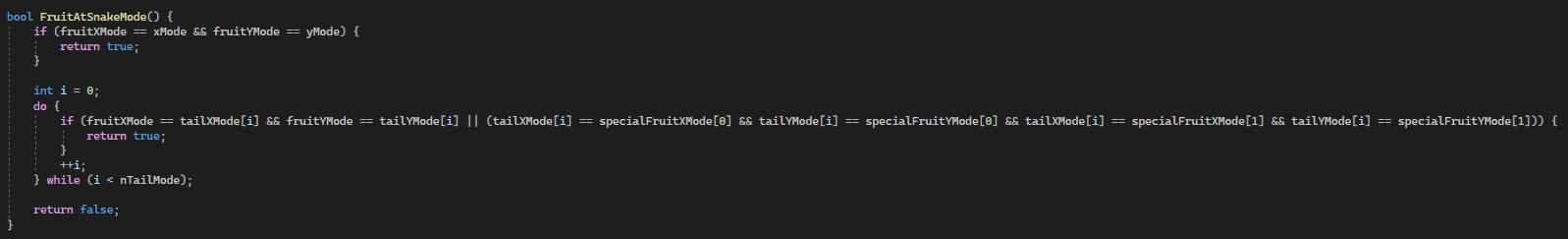
Данная функция `SetupMode()` выполняет инициализацию переменных и устанавливает начальные значения для игрового режима "Mode". Вот разбор того, как она работает:

Эта функция выполняет следующие действия:

1. Инициализирует генератор случайных чисел с использованием текущего времени в качестве семени (srand(time(NULL))), чтобы обеспечить случайность при генерации случайных чисел в дальнейшем.
2. Устанавливает начальные значения для различных переменных и флагов:
   * fruitSpawnedMode и fruitSpawnedMysMode устанавливаются в false для обозначения того, что фрукты еще не появились на игровом поле.
   * playerValueRecordedMode устанавливается в true для обозначения того, что значение игрока было записано.
   * playerInTopThreeMode устанавливается в false для обозначения того, что игрок не находится в тройке лучших результатов.
   * gameOverMode устанавливается в false для обозначения того, что игра не завершена.
   * timeCounterMode и timeCounterMysMode устанавливаются в 0 для отслеживания времени в игре.
   * timeBeforeSpawnMode устанавливается в 0 для обозначения времени до появления нового фрукта.
   * increaseCounterMode устанавливается в 0 для отслеживания количества увеличений поля.
   * dirMode устанавливается в STOP для обозначения отсутствия движения игрока.
   * xMode и yMode устанавливаются в середину игрового поля для начальной позиции игрока.
   * nTailMode устанавливается в 0 для обозначения отсутствия хвоста у игрока.
   * fruitXMode и fruitYMode устанавливаются в случайные координаты в пределах ширины и высоты игрового поля соответственно.
   * specialFruitXMode[0], specialFruitYMode[0], specialFruitXMode[1] и specialFruitYMode[1] устанавливаются в -10 для обозначения отсутствия особых фруктов на игровом поле.
   * mysteriosFruitXMode и mysteriosFruitYMode устанавливаются в -10 для обозначения отсутствия таинственного фрукта на игровом поле.
   * scoreMode устанавливается в 0 для обнуления счета игрока.

Таким образом, функция SetupMode() инициализирует переменных и устанавливает начальные значения для игрового режима "Mode", готовя его к началу новой игры.

**Функция, FruitAtSnakeMode()**

****

Данная функция предназначена для проверки наличия фрукта в теле змеи.

Проверка координат:

if (fruitXMode == xMode && fruitYMode == yMode): Проверяет, совпадают ли координаты фрукта (fruitXMode и fruitYMode) с координатами головы змеи (xMode и yMode). Если координаты совпадают, функция возвращает значение true, указывая на столкновение головы змеи с фруктом.

Проверка хвоста:

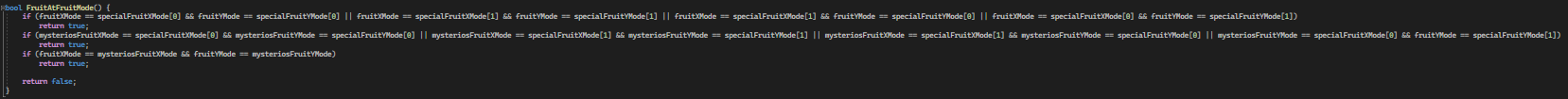
Если координаты фрукта не совпадают с координатами головы змеи, функция проверяет каждый элемент хвоста змеи.

Если координаты фрукта совпадают с координатами любого элемента хвоста змеи (tailXMode[i] и tailYMode[i]), или если координаты элементов специального фрукта (specialFruitXMode[0] и specialFruitYMode[0], specialFruitXMode[1] и specialFruitYMode[1]) совпадают с координатами хвоста змеи (tailXMode[i] и tailYMode[i]), функция также возвращает значение true для указания на столкновение головы змеи с фруктом.

Возврат значения:

Если ни одно из условий не выполняется, функция возвращает значение false, указывая на отсутствие столкновения головы змеи с фруктом.

**Функция FruitAtFruit()**

****

Эта функция с именем FruitAtFruitMode() проверяет, находится ли фрукт в позиции особого фрукта или таинственного фрукта в игре.

Эта функция может быть полезна в игре для предотвращения ситуации, когда два или более фрукта находятся в одном и том же месте, что может быть нежелательным с точки зрения геймплея.

Проверка координат фрукта с координатами особого фрукта:

Проверяет, совпадают ли координаты фрукта (fruitXMode и fruitYMode) с координатами особого фрукта:

Если координаты фрукта совпадают с первыми координатами особого фрукта (specialFruitXMode[0] и specialFruitYMode[0]), или совпадают со вторыми координатами особого фрукта (specialFruitXMode[1] и specialFruitYMode[1]), или совпадают первая координата с второй координатой особого фрукта и вторая координата с первой координатой особого фрукта, функция возвращает значение true.

Проверка координат таинственного фрукта с координатами особого фрукта:

Затем функция проверяет, совпадают ли координаты таинственного фрукта

(mysteriosFruitXMode и mysteriosFruitYMode) с координатами особого фрукта:

Если координаты таинственного фрукта совпадают с первыми координатами особого фрукта, или совпадают со вторыми координатами особого фрукта, или совпадают первая координата таинственного фрукта с второй координатой особого фрукта и вторая координата таинственного фрукта с первой координатой особого фрукта, функция возвращает значение true.

Проверка координат фрукта с координатами особого фрукта:

Затем функция проверяет, совпадают ли координаты фрукта с координатами таинственного фрукта (fruitXMode == mysteriosFruitXMode и fruitYMode == mysteriosFruitYMode).

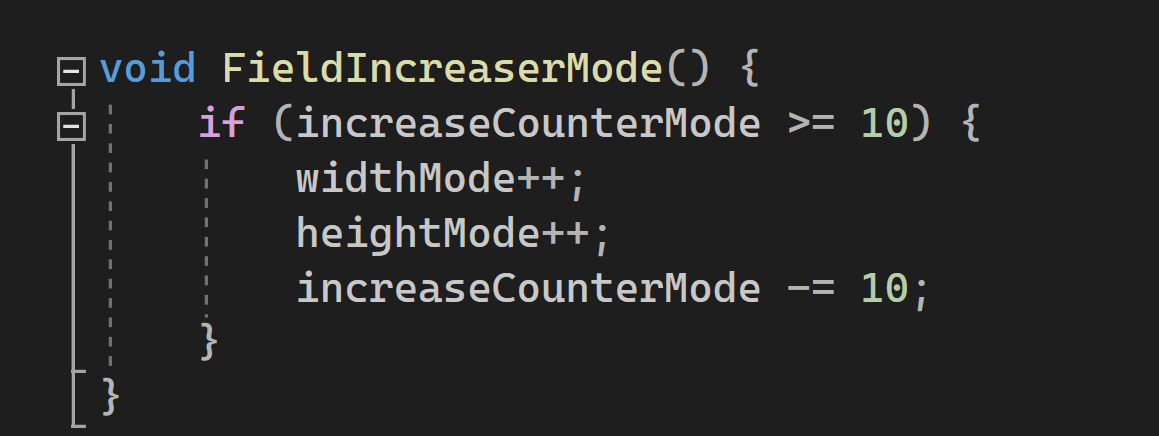
Если координаты совпадают, функция возвращает значение true.

Возврат значения:

Если ни одно из условий не выполняется, функция возвращает значение false, указывая на отсутствие столкновения фрукта с особым фруктом или таинственным фруктом.

Таким образом, эта функция проверяет наличие столкновения фрукта с особым фруктом или таинственным фруктом в игре.

**Функция FieldIncreaserMode()**

****

Данная функция FieldIncreaserMode() увеличивает размер игрового поля

Проверка:

Проверяет, достиг ли счетчик увеличения (increaseCounterMode) значения 10 или больше.

Выполнение действий:

Если счетчик увеличения больше или равен 10, выполняются следующие действия:

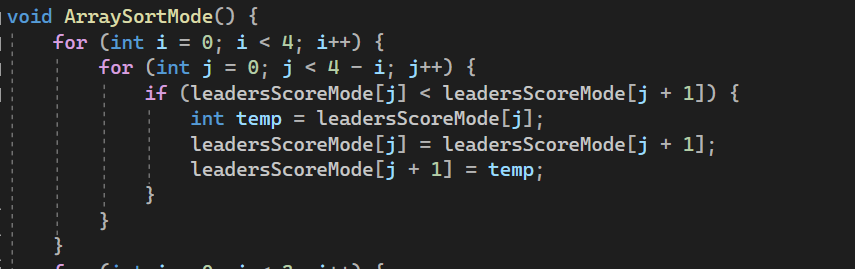
Увеличивается ширина игрового поля (widthMode) на 1.

Увеличивается высота игрового поля (heightMode) на 1.

Значение счетчика увеличения уменьшается на 10 (increaseCounterMode -= 10).

Таким образом, функция FieldIncreaserMode() увеличивает размер игрового поля в режиме "Mode", если счетчик увеличения достигает или превышает значение 10.

**Функция GetLeaderBoardMode()**

****

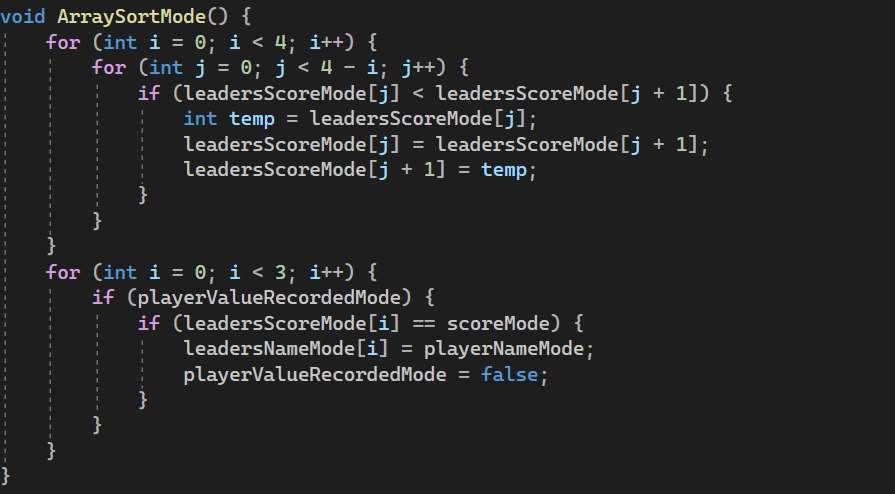
Данная функция GetLeaderBoardMode() используется для получения данных из файла "LeaderBoard.txt" и заполнения массивов leadersNameMode и leadersScoreMode с информацией об игроках и их результатами в режиме "Mode". Вот разбор того, как она работает:

Эта функция выполняет следующие действия:

1. Создает объект Lead типа ifstream для чтения данных из файла "LeaderBoard.txt".
2. Открывает файл "LeaderBoard.txt" с помощью метода open().
3. Инициализирует счетчик i для отслеживания индексов массивов leadersNameMode и leadersScoreMode.
4. В цикле while выполняет следующие действия до тех пор, пока не достигнут конец файла (!Lead.eof()):
   * Считывает данные из файла, используя оператор >>. Записывает имя лидера в leadersNameMode[i], а его счет - в leadersScoreMode[i].
   * Увеличивает счетчик i на единицу для перехода к следующему элементу массивов.
5. Закрывает файл с помощью метода close().

Таким образом, функция GetLeaderBoardMode() читает данные из файла "LeaderBoard.txt" и заполняет массивы leadersNameMode и leadersScoreMode информацией о лидерах и их счетах в режиме "Mode".

**Функция ArraySortMode()**

****

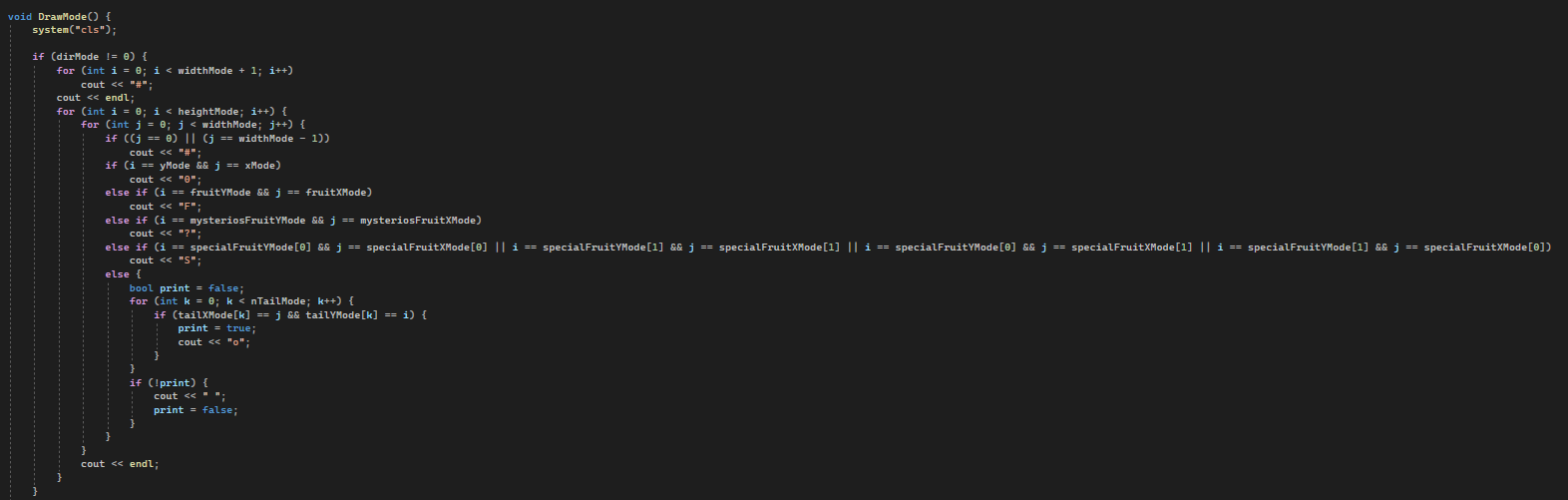
Данная функция ArraySortMode() используется для сортировки массива leadersScoreMode в убывающем порядке и обновления массива leadersNameMode с именем текущего игрока и его результатом в режиме "Mode". Вот разбор того, как она работает:

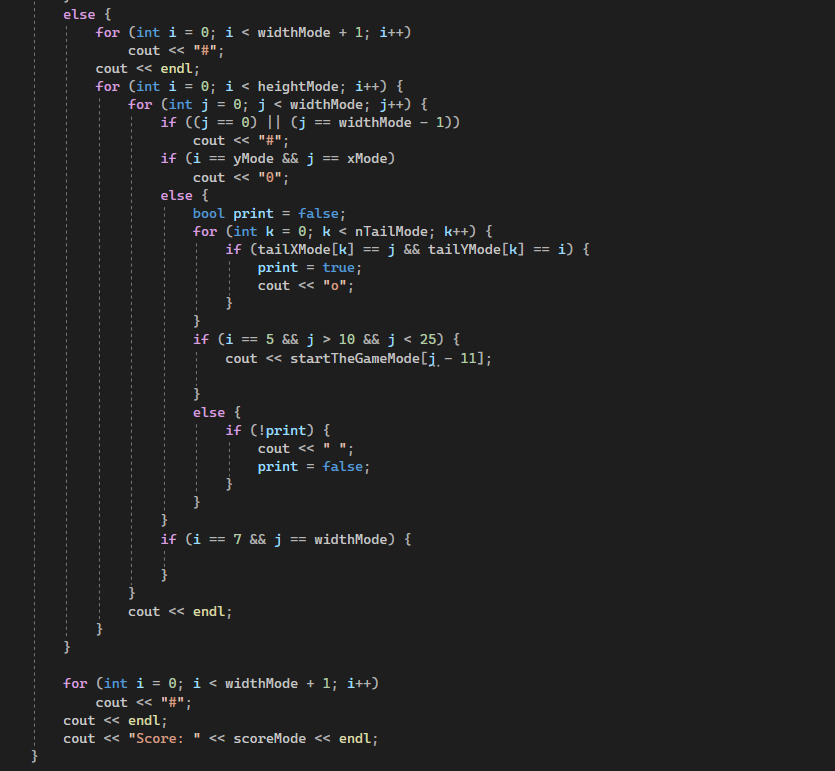
Эта функция выполняет следующие действия:

1. Внешний цикл for выполняется 4 раза для прохода через каждый элемент массива leadersScoreMode.
2. Внутренний цикл for выполняется для каждого элемента массива leadersScoreMode и сравнивает его с последующим элементом.
3. Если текущий элемент (leadersScoreMode[j]) меньше следующего элемента (leadersScoreMode[j + 1]), выполняется следующее:
   * Значения элементов меняются местами с помощью временной переменной temp.
4. После завершения сортировки массива leadersScoreMode в убывающем порядке, выполняется следующее:
   * Во втором цикле for (для i от 0 до 2) проверяется условие playerValueRecordedMode.
   * Если условие выполняется и текущий элемент массива leadersScoreMode равен scoreMode, выполняется следующее:
     + Значение playerNameMode присваивается элементу leadersNameMode[i], чтобы обновить имя текущего игрока.
     + Флаг playerValueRecordedMode устанавливается в false, чтобы указать, что значение игрока было записано.

Таким образом, функция ArraySortMode() сортирует массив leadersScoreMode в убывающем порядке и обновляет массив leadersNameMode с именем текущего игрока и его результатом в режиме "Mode".

**Функция DrawMode()**





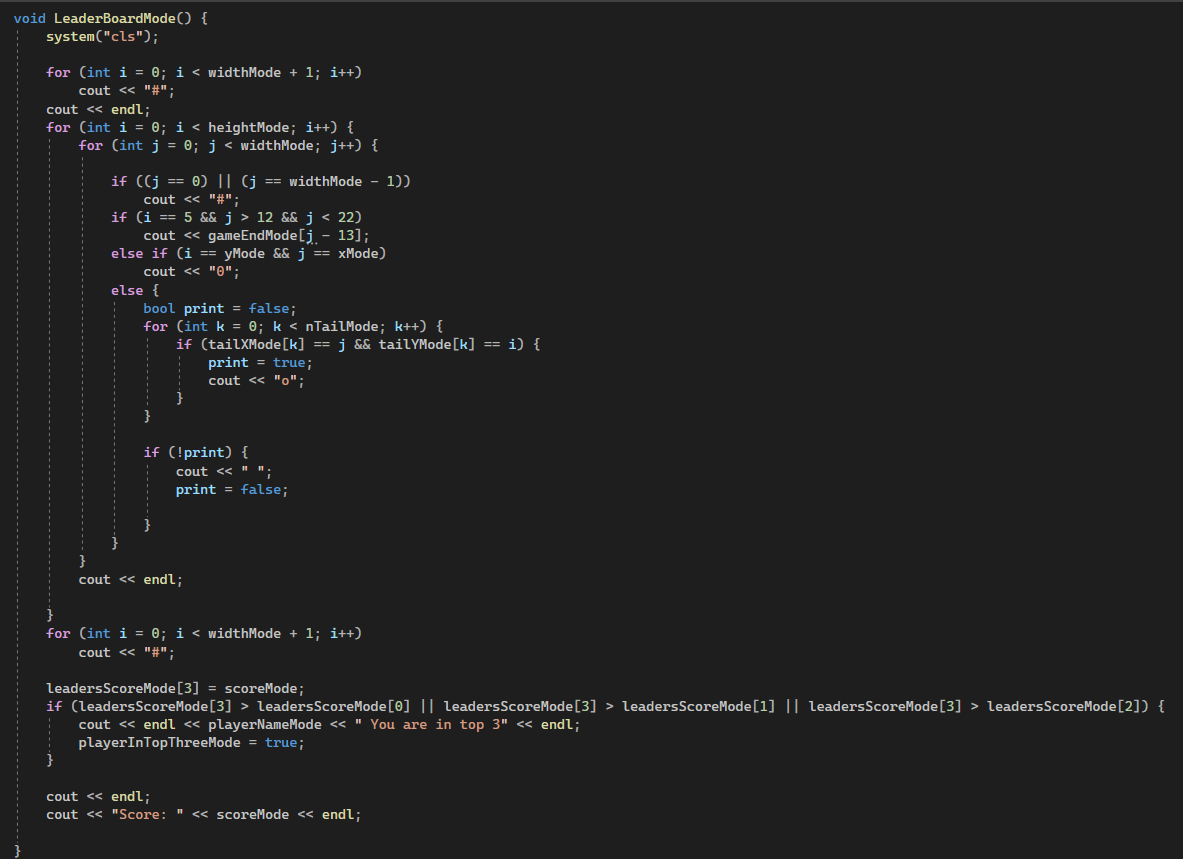
Функция DrawMode() используется для отображения игрового поля и текущего счета в режиме "Mode". Вот разбор того, как она работает:

Эта функция выполняет следующие действия:

1. Очищает консоль с помощью команды system("cls"), чтобы очистить предыдущий кадр перед отрисовкой нового игрового поля.
2. Если dirMode не равно 0, то выполняется следующее:
   * Отрисовывается верхняя граница игрового поля путем вывода символа # widthMode + 1 раз.
   * Отрисовывается игровое поле:
     + Для каждой строки и столбца игрового поля проверяется следующее:
       - Если текущая позиция соответствует позиции змейки (xMode и yMode), выводится символ 0.
       - Если текущая позиция соответствует позиции фрукта (fruitXMode и fruitYMode), выводится символ F.
       - Если текущая позиция соответствует позиции загадочного фрукта (mysteriosFruitXMode и mysteriosFruitYMode), выводится символ ?.
       - Если текущая позиция соответствует одной из позиций специальных фруктов (specialFruitXMode и specialFruitYMode), выводится символ S.
       - Если текущая позиция соответствует позиции хвоста змейки (tailXMode и tailYMode), выводится символ o.
       - В противном случае выводится пустой символ.
   * Завершается отрисовка игрового поля.
3. Если dirMode равно 0, то выполняется следующее:
   * Отрисовывается верхняя граница игрового поля путем вывода символа # widthMode + 1 раз.
   * Отрисовывается игровое поле:
     + Для каждой строки и столбца игрового поля проверяется следующее:
       - Если текущая позиция соответствует позиции змейки, выводится символ 0.
       - Если текущая позиция соответствует позиции хвоста змейки, выводится символ o.
       - Если текущая позиция соответствует определенной позиции (i == 5 и j > 10 и j < 25), выводится символ из массива startTheGameMode, который представляет сообщение "Start the Game".
       - В противном случае выводится пустой символ.
4. Завершается отрисовка игрового поля путем вывода нижней границы игрового поля с помощью символа # widthMode + 1 раз.
5. Выводится текущий счет (scoreMode).

Таким образом, функция DrawMode() отрисовывает игровое поле и текущий счет в режиме "Mode".

**Функция LeaderBoardMode()**

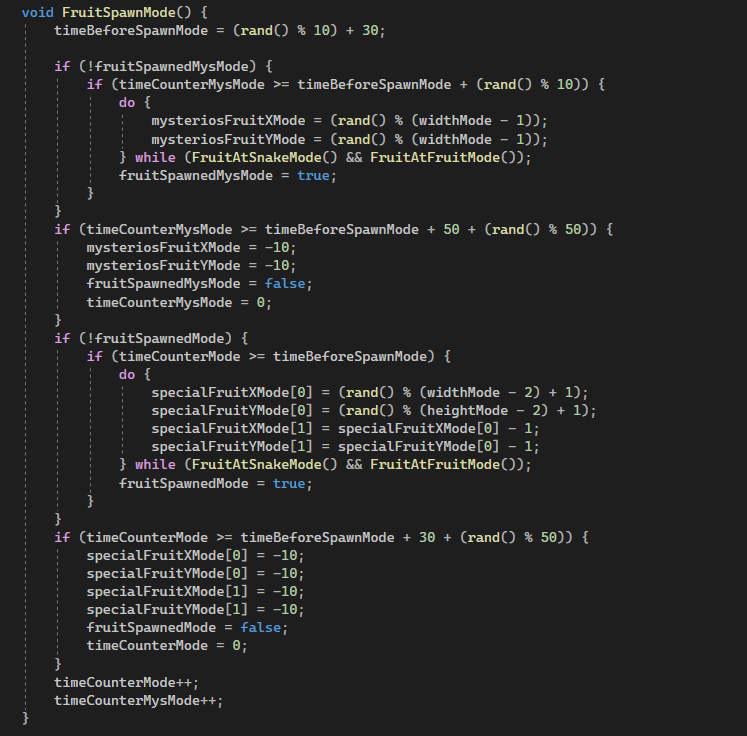
****

Функция LeaderBoardMode() используется для отображения доски лидеров и текущего счета в режиме "Mode". Вот разбор того, как она работает:

Эта функция выполняет следующие действия:

1. Очищает консоль с помощью команды system("cls"), чтобы очистить предыдущий кадр перед отрисовкой новой доски лидеров.
2. Отрисовывает верхнюю границу доски лидеров с помощью символа # widthMode + 1 раз.
3. Отрисовывает доску лидеров:
   * Для каждой строки и столбца доски лидеров проверяется следующее:
     + Если текущая позиция соответствует боковой границе доски лидеров, выводится символ #.
     + Если текущая позиция соответствует определенной позиции (i == 5 и j > 12 и j < 22), выводится символ из массива gameEndMode, который представляет сообщение "Game Over".
     + Если текущая позиция соответствует позиции змейки (xMode и yMode), выводится символ 0.
     + Если текущая позиция соответствует позиции хвоста змейки (tailXMode и tailYMode), выводится символ o.
     + В противном случае выводится пустой символ.
4. Завершается отрисовка доски лидеров путем вывода нижней границы доски леров с помощью символа # widthMode + 1 раз.
5. Обновляется массив leadersScoreMode, заменяя четвертое значение на текущий счет scoreMode.
6. Проверяется, является ли текущий счет лучшим из трех лучших результатов. Если да, выводится сообщение о том, что игрок находится в топ-3, и устанавливается флаг playerInTopThreeMode в значение true.
7. Выводится текущий счет scoreMode.

**Функция FruitSpawnMode()**

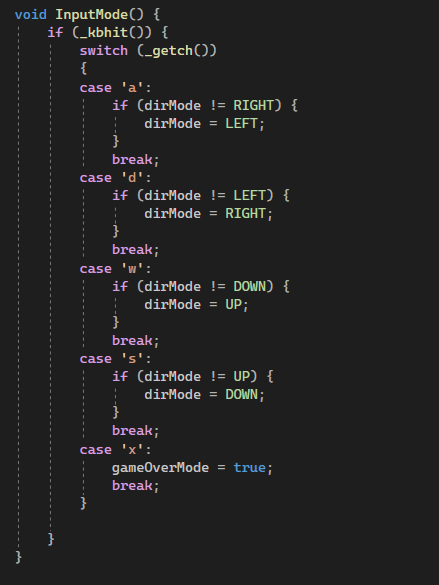


Функция FruitSpawnMode() используется для генерации специальных фруктов в режиме "Mode". Вот разбор того, как она работает:

В основном, функция генерирует случайные координаты для специальных фруктов (mysteriosFruitXMode и mysteriosFruitYMode) и двух частей специального фрукта (specialFruitXMode и specialFruitYMode). Она также отслеживает время до появления и исчезновения фруктов с помощью счетчиков времени (timeCounterMysMode и timeCounterMode).

Координаты фруктов и флаги fruitSpawnedMysMode и fruitSpawnedMode используются для определения, нужно ли отобразить фрукты на игровом поле.

**Функция InputMode()**



Функция InputMode() отвечает за обработку пользовательского ввода в режиме "Mode". Вот разбор её работы:

Функция использует две функции из стандартной библиотеки <conio.h>: \_kbhit() и \_getch(). \_kbhit() проверяет, доступны ли символы во входном потоке. Если функция возвращает ненулевое значение, значит, есть символ для чтения. Затем \_getch() считывает символ из входного потока без его отображения на экране.

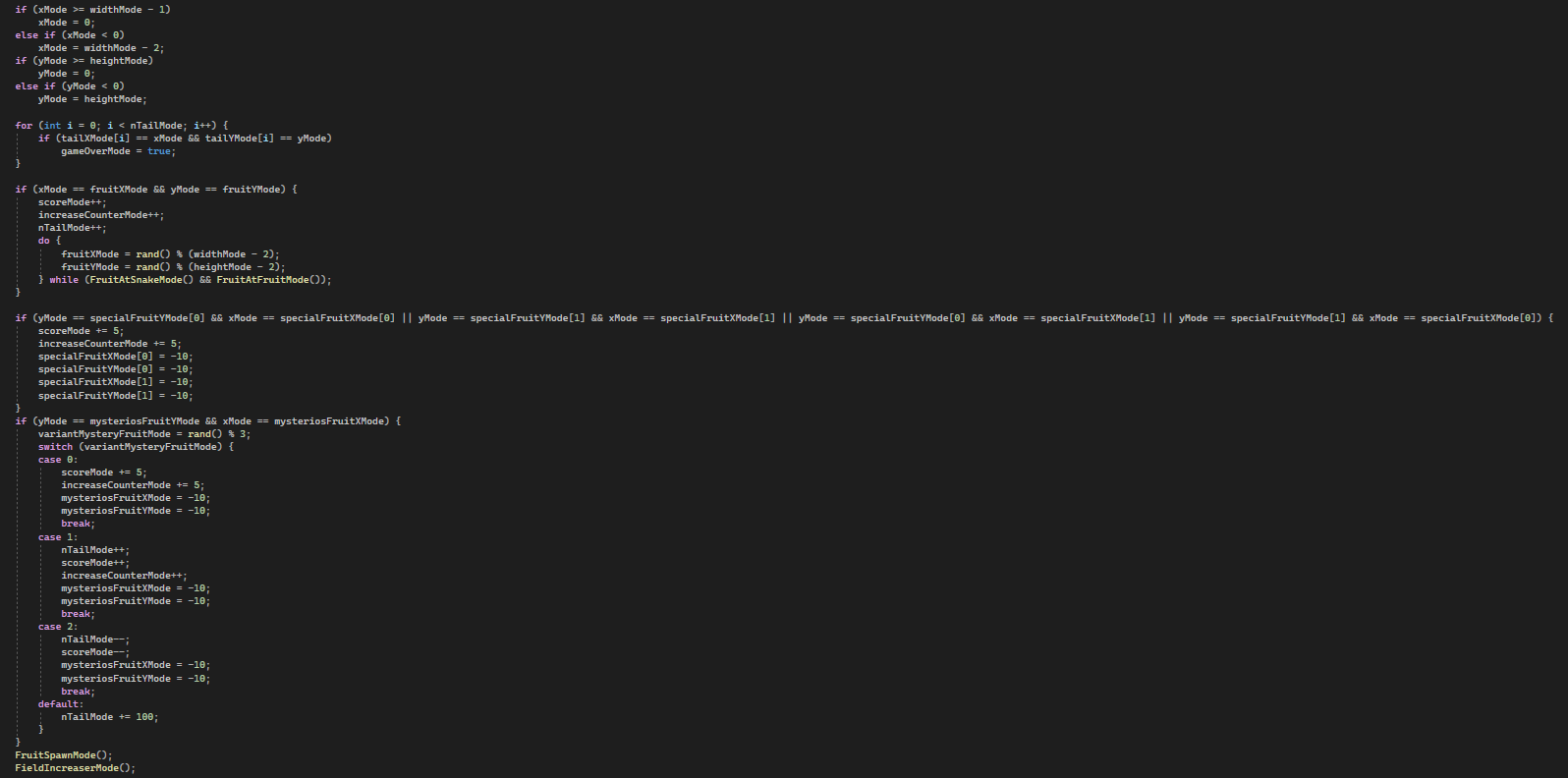
В функции InputMode() символы проверяются в switch-конструкции. Если символ соответствует одному из случаев ('a', 'd', 'w', 's', 'x'), то происходит соответствующая обработка:

* Если символ 'a' и текущее направление не право (RIGHT), то направление движения устанавливается на лево (LEFT).
* Если символ 'd' и текущее направление не лево (LEFT), то направление движения устанавливается на право (RIGHT).
* Если символ 'w' и текущее направление не вниз (DOWN), то направление движения устанавливается вверх (UP).
* Если символ 's' и текущее направление не вверх (UP), то направление движения устанавливается вниз (DOWN).
* Если символ 'x', то устанавливается флаг gameOverMode, сигнализирующий об окончании игры.

Таким образом, функция InputMode() обрабатывает нажатия клавиш пользователя и изменяет направление движения змейки соответственно. Также, при нажатии клавиши 'x' устанавливается флаг gameOverMode, что приводит к завершению игры.

**Функция LogicMode()**





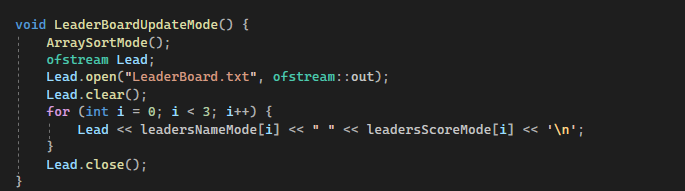
Функция LogicMode() отвечает за движение и поведение змейки в определенном режиме игры.

Давайте разберем код и поймем его функциональность:

1. srand(time(NULL));: Инициализирует генератор случайных чисел текущим временем, чтобы гарантировать разные случайные значения при каждом запуске программы.
2. int prevX = tailXMode[0]; и int prevY = tailYMode[0];: Хранят предыдущую позицию хвоста змейки, которая изначально устанавливается в текущую позицию головы змейки.
3. Обновление позиции хвоста: Код обновляет позиции сегментов хвоста змейки, сдвигая каждый сегмент на позицию предыдущего сегмента, начиная со второго сегмента хвоста.
4. Обновление позиции головы змейки: Позиция головы змейки обновляется в зависимости от текущего направления (dirMode). Голова змейки перемещается на один шаг в указанном направлении.
5. Обработка пересечения границ: Если голова змейки пересекает границы игровой области, она перемещается на противоположную сторону.

Также в коде присутствуют проверки на столкновение змейки с фруктами, особыми фруктами и загадочным фруктом. При столкновении с фруктом увеличивается счетчик очков и длина змейки, а также генерируется новое положение фрукта. При столкновении с особыми фруктами или загадочным фруктом также изменяются счетчик очков и длина змейки в зависимости от случайно выбранной вариации.

**Функция LeaderBoardUpdateMode()**

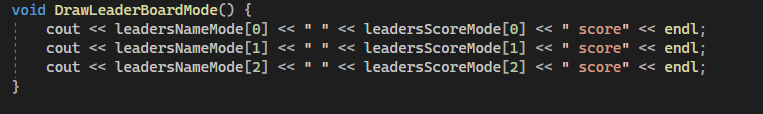


Функция LeaderBoardUpdateMode() отвечает за обновление таблицы лидеров в определенном режиме игры. Давайте разберем данный код:

1. ArraySortMode();: Вызывается функция ArraySortMode(), которая предположительно сортирует массивы leadersNameMode и leadersScoreMode по убыванию (от наибольшего к наименьшему). Это может быть реализовано с использованием популярных алгоритмов сортировки, таких как сортировка пузырьком, сортировка вставками или быстрая сортировка.
2. ofstream Lead;: Создается объект Lead класса ofstream, который используется для записи данных в файл.
3. Lead.open("LeaderBoard.txt", ofstream::out);: Открывается файл с именем "LeaderBoard.txt" в режиме записи (ofstream::out). Если файл не существует, он будет создан. Если файл уже существует, его содержимое будет заменено.
4. Lead.clear();: Очищается содержимое файла.
5. for (int i = 0; i < 3; i++) { ... }: Выполняется цикл три раза для трех лучших результатов в таблице лидеров.
6. Lead << leadersNameMode[i] << " " << leadersScoreMode[i] << '\n';: Записывается имя лидера и его счет в файл. Имя и счет разделяются пробелом, а каждая запись завершается символом новой строки.
7. Lead.close();: Файл закрывается после записи всех данных.

Таким образом, данная функция сортирует массивы с именами и счетами лидеров, очищает файл таблицы лидеров, затем записывает три лучших результата в файл "LeaderBoard.txt".

**Функция DrawLeaderBoardMode()**

****

Функция DrawLeaderBoardMode() отвечает за вывод таблицы лидеров в определенном режиме игры. Давайте разберем данный код:

1. cout << leadersNameMode[0] << " " << leadersScoreMode[0] << " score" << endl;: Выводится имя первого лидера и его счет в виде "Имя Счет score" на стандартный вывод (консоль). Здесь leadersNameMode[0] предполагается содержит имя первого лидера, а leadersScoreMode[0] - его счет.
2. cout << leadersNameMode[1] << " " << leadersScoreMode[1] << " score" << endl;: Аналогично выводится имя и счет второго лидера.
3. cout << leadersNameMode[2] << " " << leadersScoreMode[2] << " score" << endl;: Аналогично выводится имя и счет третьего лидера.

Таким образом, функция DrawLeaderBoardMode() просто выводит на консоль имена и счета трех лучших лидеров в таблице лидеров для определенного режима игры.

**Функция easyGameMode()**

****

Функция easyGameMode() представляет режим игры с настройками "легкий". Давайте разберем данный код:

1. GetLeaderBoardMode();: Вызывается функция GetLeaderBoardMode(), которая, предположительно, получает таблицу лидеров для данного режима игры.
2. while (restartTheGameMode) { ... }: Выполняется цикл, пока переменная restartTheGameMode истинна. Это позволяет игроку повторно играть после завершения игры.
3. system("cls");: Очищается экран консоли с помощью команды cls (для операционной системы Windows).
4. SetupMode();: Вызывается функция SetupMode(), которая, вероятно, устанавливает начальные настройки игры в данном режиме.
5. DrawMode();: Вызывается функция DrawMode(), которая отображает игровое поле и состояние игры.
6. gameOverMode = false;: Устанавливается значение переменной gameOverMode в ложь, чтобы начать новую игру.
7. while (!gameOverMode) { ... }: Выполняется цикл, пока переменная gameOverMode ложна. Это позволяет игроку играть в текущей игровой сессии до тех пор, пока игра не завершится.
8. DrawMode();: Вызывается функция DrawMode(), чтобы отобразить обновленное состояние игры после каждого шага.
9. InputMode();: Вызывается функция InputMode(), которая, предположительно, обрабатывает ввод игрока (например, управление змейкой).
10. LogicMode();: Вызывается функция LogicMode(), которая обрабатывает логику движения и поведения змейки.
11. Sleep(100 - 3 \* nTailMode);: Приостанавливается выполнение программы на указанное количество миллисекунд. Здесь используется формула для задержки, которая зависит от длины хвоста змейки (nTailMode), что может означать ускорение игры с увеличением длины змейки.
12. LeaderBoardMode();: Вызывается функция LeaderBoardMode(), которая отображает таблицу лидеров для данного режима игры.
13. if (playerInTopThreeMode) LeaderBoardUpdateMode();: Если игрок попадает в топ-3 лидеров, вызывается функция LeaderBoardUpdateMode(), которая обновляет таблицу лидеров.
14. break;: Происходит выход из цикла while (restartTheGameMode) после завершения одной игровой сессии.

Таким образом, функция easyGameMode() реализует основной цикл игры в режиме "легкий" с отображением игрового поля, обработкой ввода игрока, обновлением состояния игры, отображением таблицы лидеров и возможностью повторной игры.

**Заключение**

Проект "Игра Змейка" на C++ представляет собой реализацию обновления классической аркадной игры с использованием языка программирования C++. Он был успешно разработан с учетом основных механик игры, включая управление змейкой, генерацию пищи, обработку столкновений и отображение счета.

Проект "Игра Змейка" на C++ представляет потенциал для привлечения и развлечения широкой аудитории, особенно любителей аркадных игр. С учетом дальнейших улучшений и новых функций, игра может стать еще более привлекательной и популярной.



Автор фотографии:

Аманов Артур Русланович