

# Техническая документация к проекту snake-game

Автор: [Yu-Leo](#)

GitHub: [snake-game](#)

Язык: C++ (стандарт C++17)

## Модули:

- **Main** (main.cpp) – главный запускаемый файл проекта.
- **Point** (Point.h, Point.cpp) – точка игрового поля.
- **Apple** (Apple.h, Apple.cpp) – яблоко, располагающееся на игровом поле.
- **Snake** (Snake.h, Snake.cpp) – игрок (змейка).
- **GameField** (GameField.h, GameField.cpp) – игровое поле.

## Main:

Управление игровым процессом. В цикле происходит ожидание ввода команды для управления змейкой (wasd), в случае ввода любой другой команды происходит окончание игры. Для ввода команды необходимо нажать Enter. После чего происходит обработка всех событий, перерисовка игрового поля и его вывод в консоль (терминал).

## Point:

**struct Point {**

**int x = 0;** - x-координата точки.

**int y = 0;** - y-координата точки.

**Point(int x, int y);** - конструктор с произвольным заданием координат

**Point() : Point(0, 0) {};** - конструктор по умолчанию.

**bool operator==(const Point& other);** - перегрузка оператора сравнения.  
Точки равны, если равны их координаты.

**};**

## Apple:

```
class Apple {
```

```
public:
```

**Apple();** - конструктор по умолчанию. Использует конструктор по умолчанию структуры Point.

**Apple(const Point& point);** - конструктор с указанием конкретной точки, в которой необходимо создать яблоко.

**Point get\_coordinates();** - геттер для поля coordinates.

```
private:
```

**Point coordinates;** - точка, в которой находится яблоко.

```
};
```

Snake:

```
class Snake {
```

```
public:
```

**enum Directions {** - константы направлений движения змейки.

**RIGHT,**

**DOWN,**

**LEFT,**

**UP**

```
};
```

**Snake();** - конструктор по умолчанию (пустой).

**void move\_head();** - изменение координаты головы в зависимости от заданного направления.

**void set\_field\_size(int field\_size);** - инициализация размеров поля.

**Point get\_head\_pos() const;** - геттер позиции головы.

**int get\_length() const;** - геттер длины.

**int get\_direction() const;** - геттер направления.

**void increase\_length();** - увеличение длины на одну ячейку.

**void change\_direction(int new\_direction);** - смена направления.

**private:**

**int length = 3;** - длина.

**Point head\_position = Point(this->length - 1, 0);** - позиция головы.

**int direction = Directions::RIGHT;** - направление.

**int field\_size = 0;** - размер поля, на котором используется объект класса.  
Необходимо инициализировать фактическим значением!

**};**

**GameField:**

**class GameField {**

**public:**

**GameField();** - конструктор по умолчанию. Инициализация поля, генерация яблока, отрисовка яблока и змейки (render\_apple, render\_apple).

**void move\_snake();** - движение змейки: изменение координаты головы, проверка столкновений, изменение значений в клетках тела змейки.

**void turn\_snake(int direction);** - смена направления движения змейки (делегирование объекту класса Snake).

**int get\_snake\_direction() const;** - геттер направления движения змейки.

**private:**

**enum Symbols {** - символы для отображения игрового поля в консоли (терминале).

**SNAKE = '#',**

**NONE = '-',**

**APPLE = '\*'**

**};**

**static const int FIELD\_CELL\_TYPE\_NONE = 0;** - значения в пустых ячейках.

**static const int FIELD\_CELL\_TYPE\_APPLE = -1;** - значение в ячейке с яблоком.

**static const int DEFAULT\_SIZE = 20;** - размер игрового поля по умолчанию.

**const int size = this->DEFAULT\_SIZE;** - размер игрового поля.

**std::vector<std::vector<char>> field;** - матрица игрового поля.

**Snake snake;** - сама змейка (объект класса Snake).

**Apple apple;** - яблоко.

**void resize\_matrix();** - установка размеров матрицы игрового поля.

**void init\_field();** - инициализация игрового поля (заполнение значениями пустой ячейки).

**void check\_collisions();** - проверка на столкновение головы змейки с другими клетками.

**void grow\_snake();** - увеличение значений в клетках змейки.

**void decrease\_snake\_cells();** - уменьшение значений в клетках змейки.

**void render\_snake();** - отрисовка змейки на игровом поле (заполнение клеток, в которых находится змейка символами для обозначения тела змейки).

**void render\_apple();** - отрисовка яблока на игровом поле (заполнение клетки, в которой находится яблоко символом для обозначения яблока).

**bool is\_cell\_empty(const Point& cell);** - проверка клетки на пустоту.

**int count\_empty\_cells();** - получение количества пустых клеток.

**Point get\_random\_empty\_cell();** - получение случайной пустой ячейки.

**friend void print\_cell(std::ostream& out, const GameField& game\_field, const Point& cell);** - дружественная функция для вывода одной клетки игрового поля в консоль (терминал).

**friend std::ostream& operator<< (std::ostream& out, const GameField& game\_field);** - дружественная функция для вывода матрицы игрового поля в консоль (терминал).

**};**