

# 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.
- **Constants** (constants.h) – константы проекта.
- **Point** (Point.h, Point.cpp) – точка игрового поля.
- **RandomPoint** (RandomPoint.h, RandomPoint.cpp) – случайная 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).

## Constants:

**const int GAME\_FIELD\_SIZE = 20;** - размеры игрового поля.

**namespace Directions {** - константы для направлений движения змейки.

**const int RIGHT = 0;**

**const int DOWN = 1;**

**const int LEFT = 2;**

**const int UP = 3;**

**}**

## Point:

**struct Point {**

**int x = 0;** - x-координата точки.

**int y = 0;** - y-координата точки.

**Point();** - конструктор по умолчанию (x = 0; y = 0).

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

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

**};**

## RandomPoint:

**struct RandomPoint : public Point {**

**RandomPoint();** - конструктор по умолчанию. Координаты точки - случайные числа в пределах размеров игрового поля.

**RandomPoint(const std::vector<Point>& acceptable\_points);** - конструктор с указанием недопустимых точек. Координаты точки – случайные числа в пределах размеров игрового поля за исключением пар, образующих недопустимые точки

**};**

## 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:**

**Snake();** - конструктор по умолчанию. Инициализирует змейку (init\_snake\_dots) с размером и направлением по умолчанию (DEFAULT\_SIZE, Directions::RIGHT).

**Point get\_point\_by\_index(int index);** - геттер для получения координат точки змейки по её индексу в массиве с координатами точек змейки.

**int get\_size();** - геттер для размера змейки.

**void move();** - сдвиг змейки на одну клетку в установленном направлении.

**void change\_direction(int new\_direction);** - смена направления движения змейки (используются константы Direction::)

**bool check\_collision\_with\_body();** - проверка на совпадение координат головы змейки и какой либо клетки её тела.

**void increase\_size();** - увеличение длины змейки на одну клетку.

**std::vector<Point> get\_points();** - геттер для координат точек змейки.

**private:**

**static const int DEFAULT\_SIZE = 3;** - размер змейки по умолчанию.

**std::vector<Point> dots;** - массив с координатами точек тела змейки.

**int size;** - размер змейки.

**int direction;** - текущее направление (используются константы Direction::).

**void init\_snake\_dots();** - инициализация змейки в начальных координатах (в один ряд, горизонтально, последняя точка находится в координатах (0; 0).

**bool is\_change\_of\_direction\_correct(int new\_direction);** - проверка корректности смены направления (используется в change\_direction).

};

### GameField:

**class GameField {**

**public:**

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

**void update();** - перерисовка яблока и змейки на поле.

**void move\_snake();** - движение змейки (делегирование объекту класса Snake).

**void check\_collision\_with\_apple();** - проверка на совпадение координат головы змейки и яблока. Если совпадение есть, длина змейки увеличивается и генерируется новое яблоко.

**bool is\_game\_over();** - проверка на проигрыш (столкновение змейки с самой собой или границами игрового поля).

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

**private:**

**static const char SNAKE\_SYMBOL = '#';** - символ для обозначения тела змейки на игровом поле.

**static const char VOID\_SYMBOL = '-';** - символ для обозначения пустой клетки на игровом поле.

**static const char APPLE\_SYMBOL = '\*';** - символ для обозначения яблока на игровом поле.

**const int size = GAME\_FIELD\_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 символами пустой клетки).

**bool check\_collision\_with\_snake\_body();** - проверка на столкновение змейки с самой собой (делегирование объекту класса Snake).

**bool check\_collision\_with\_borders();** - проверка на столкновение змейки с границами игрового поля.

**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).

**std::vector<Point> generate\_acceptable\_points\_for\_new\_apple();** - генерация списка клеток, в которых можно сгенерировать новое яблоко (все клетки игрового поля за исключением клеток, в которых находится змейка).

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

**};**