```cpp
#ifndef SNAKE_HPP
#define SNAKE_HPP

/// Includes
#include <vector>
#include "SFML/Graphics.hpp"

/// The snake representation
struct SnakeSegment
{
    SnakeSegment(int x, int y) : position(x, y) {};
    sf::Vector2i position;
};
using SnakeContainer = std::vector<SnakeSegment>;

/// Directions of that the snake might take
enum class Direction
{
    None,
    Up,
    Down,
    Left,
    Right
};

/// The snake class.
class Snake
{
    public:
        /** Default constructor */
        Snake(int l_blockSize);
        ~Snake();

        /// Helper methods
        Direction GetDirection();
        void SetDirection(Direction l_dir);
        int GetScore();
        int GetSpeed();
        sf::Vector2i GetPosition();
        int GetLives();
        void IncreaseScore();
        bool HasLost();

        void Lose();        /// Handle losing here.
        void ToggleLost();

        void Extend();      /// Grow the snake.
        void Reset();       /// Reset to starting position.

        void Move();        /// Movement method.
        void Tick();        /// Update method.
        void Cut(int l_segments);        ///Method for cutting snake.
        void Render(sf::RenderWindow& l_window);

        Direction GetPhysicalDirection();

    protected:

    private:
        void CheckCollision();      /// Checking for collision.
        SnakeContainer m_snakeBody;     /// Segment vector.
        int m_size;     /// Size of the graphics.
        Direction m_dir;        /// Current direction of the snake.
        int m_speed;        /// Speed.
        int m_lives;        /// Lives.
        int m_score;        /// Scores.
        bool m_lost;        /// Losing state.
        sf::RectangleShape m_bodyRect;      /// Shape used in rendering.
};

#endif // SNAKE_HPP
```