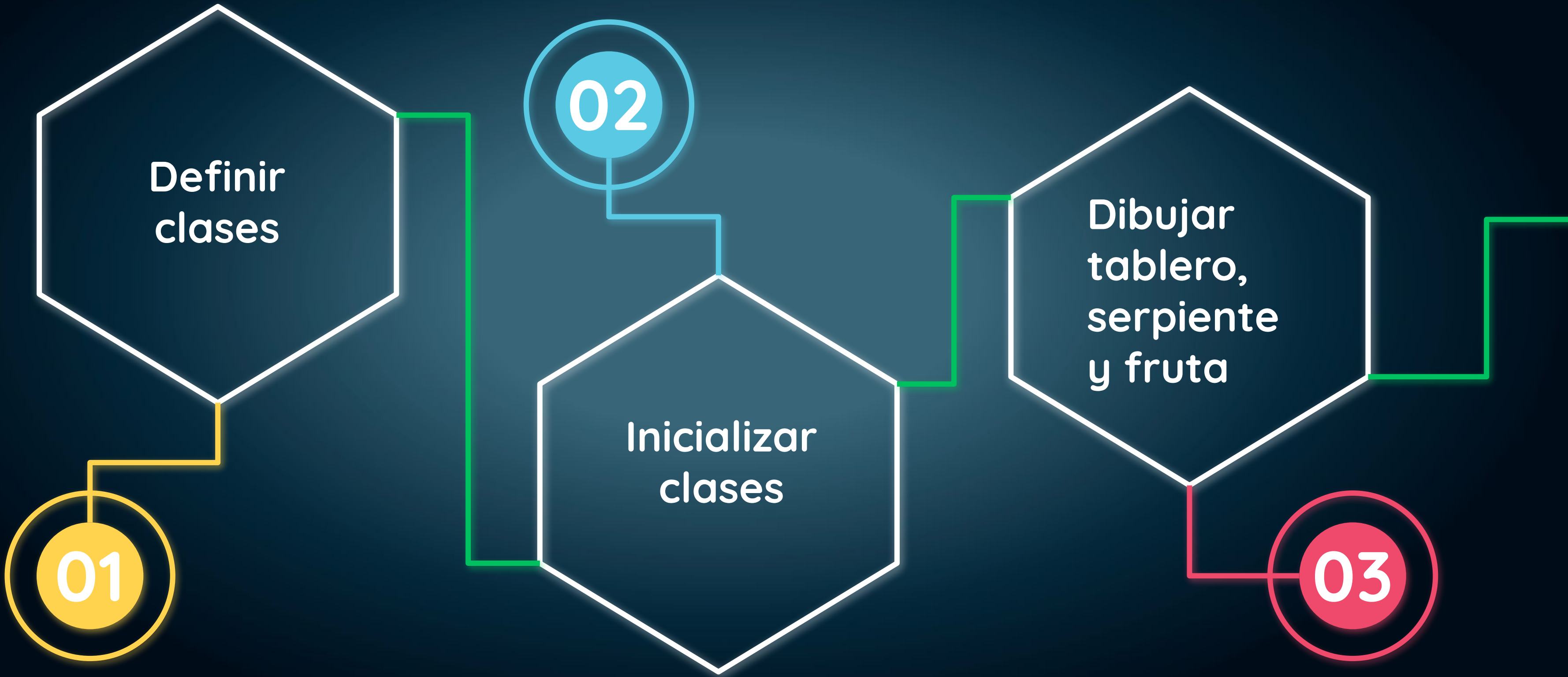
JUEGO SNAKE

- Josue Carpio
- Ariana Leon
- Anthony Rodriguez

Universidad Católica San Pablo

Docente: DSc. Manuel Eduardo Loaiza Fernández
Departamento de de ciencia de la computación
Universidad católica San Pablo
Semestre 2021-II
Arequipa-Perú

Pasos


01
Definir clases
02
Inicializar clases
03
Dibujar tablero, serpiente y fruta

Pasos

04

Reconocer el teclado

05

Movimiento de la serpiente

06

Maneras de morir

Pasos

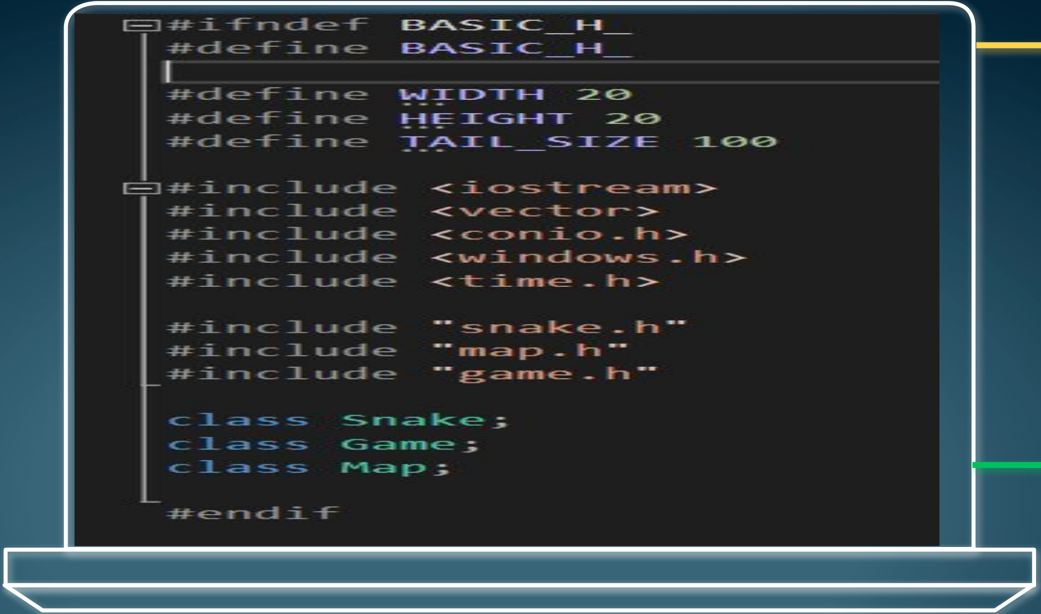
07

Comer las frutas

Correr main

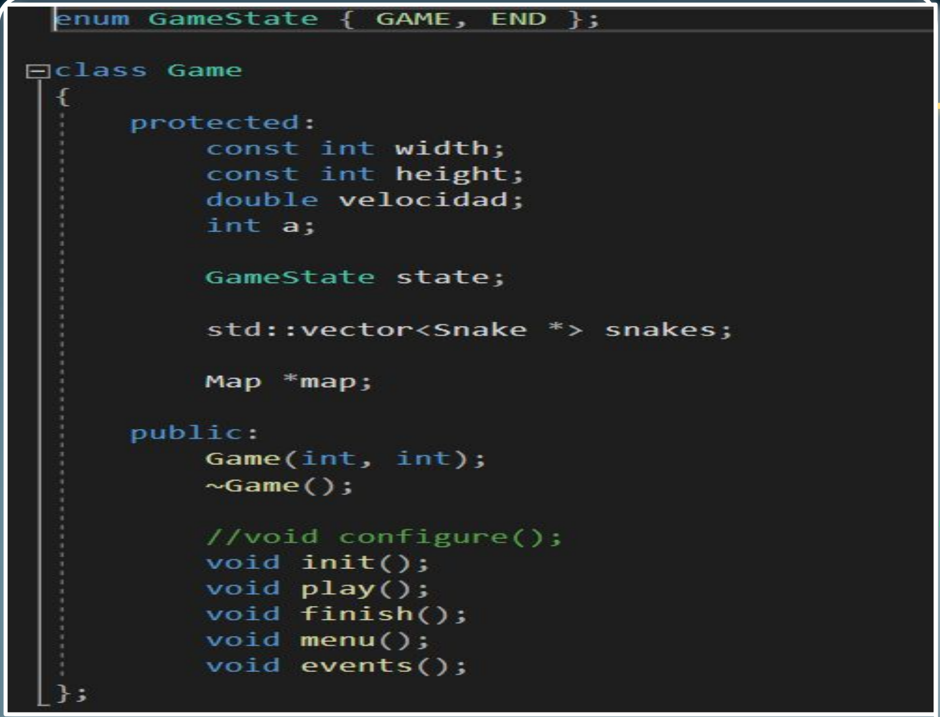
08

Incluir librerías



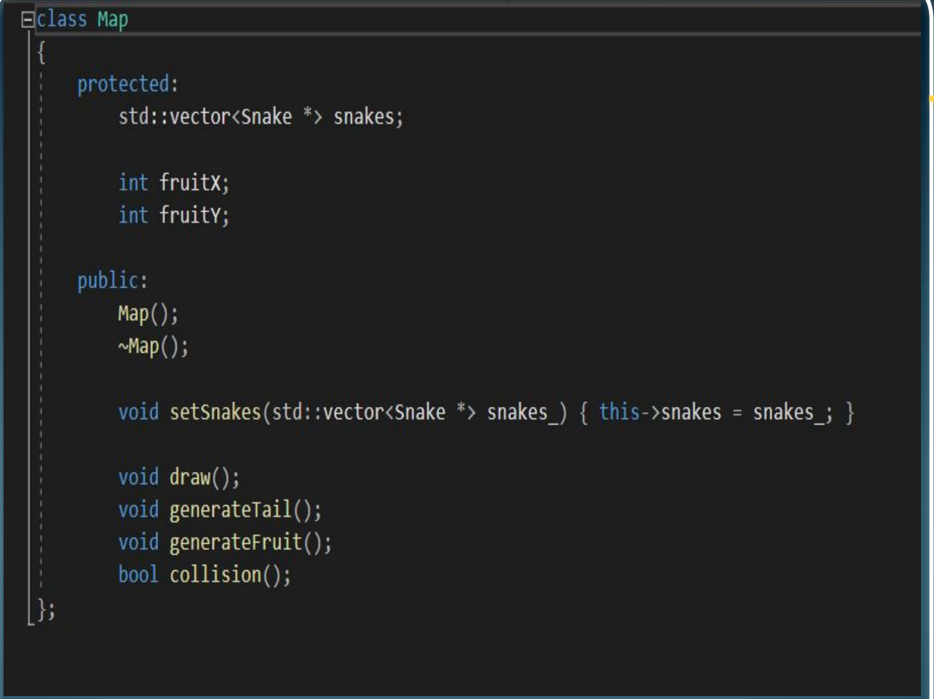
```
#ifndef BASIC_H_
#define BASIC_H_

#define WIDTH 20
#define HEIGHT 20
#define TAIL_SIZE 100

#include <iostream>
#include <vector>
#include <conio.h>
#include <windows.h>
#include <time.h>

#include "snake.h"
#include "map.h"
#include "game.h"

class Snake;
class Game;
class Map;

#endif
```

Definir clases



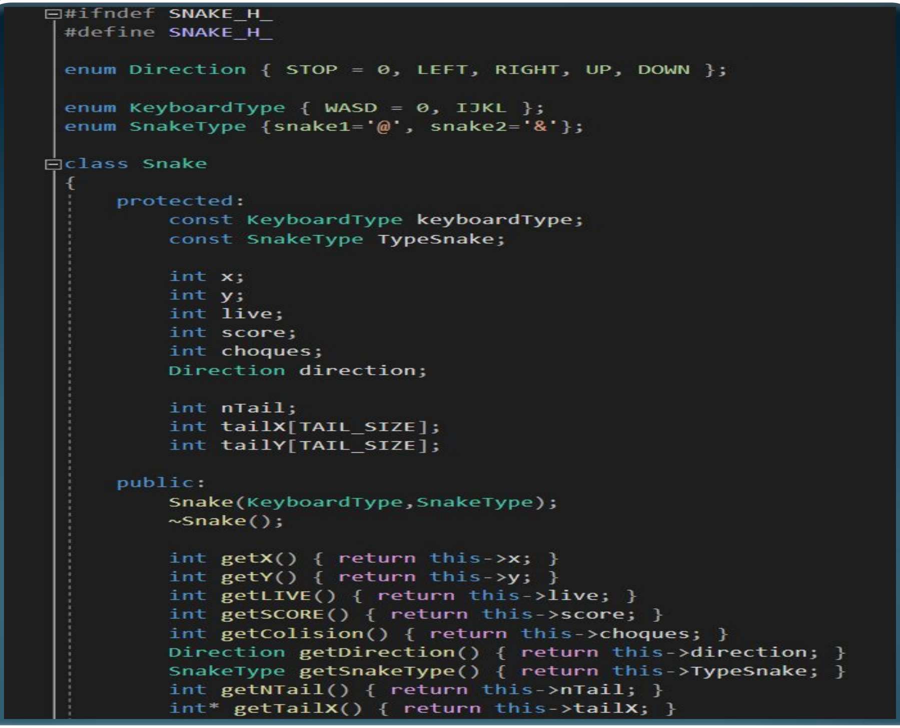
```
enum GameState { GAME, END };

class Game
{
	protected:
		const int width;
		const int height;
		double velocidad;
		int a;

		GameState state;

		std::vector<Snake *> snakes;

		Map *map;

	public:
		Game(int, int);
		~Game();

		//void configure();
		void init();
		void play();
		void finish();
		void menu();
		void events();
};
```

Definir clases



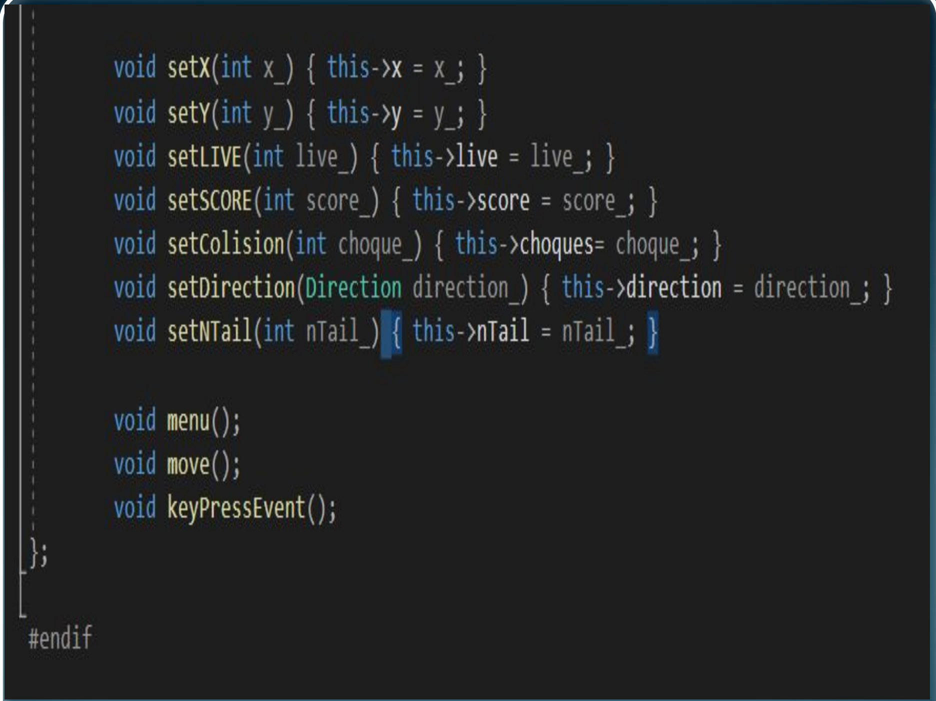
```
class Map
{
    protected:
        std::vector<Snake *> snakes;

        int fruitX;
        int fruitY;

    public:
        Map();
        ~Map();

        void setSnakes(std::vector<Snake *> snakes_) { this->snakes = snakes_; }

        void draw();
        void generateTail();
        void generateFruit();
        bool collision();
};
```

Definir clases



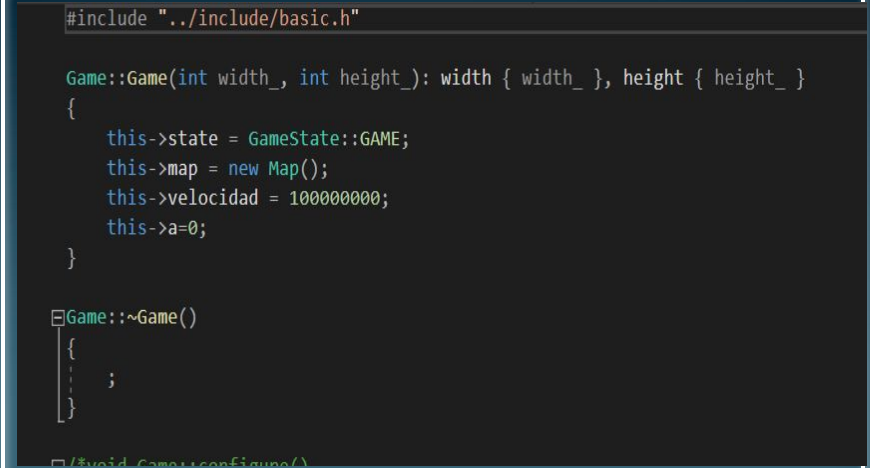
```cpp
#ifndef SNAKE_H_
#define SNAKE_H_

enum Direction { STOP = 0, LEFT, RIGHT, UP, DOWN };

enum KeyboardType { WASD = 0, IJKL };
enum SnakeType {snake1='@', snake2='&'};

class Snake
{
    protected:
        const KeyboardType keyboardType;
        const SnakeType TypeSnake;

        int x;
        int y;
        int live;
        int score;
        int choques;
        Direction direction;

        int nTail;
        int tailX[TAIL_SIZE];
        int tailY[TAIL_SIZE];

    public:
        Snake(KeyboardType,SnakeType);
        ~Snake();

        int getX() { return this->x; }
        int getY() { return this->y; }
        int getLIVE() { return this->live; }
        int getSCORE() { return this->score; }
        int getColision() { return this->choques; }
        Direction getDirection() { return this->direction; }
        SnakeType getSnakeType() { return this->TypeSnake; }
        int getNTail() { return this->nTail; }
        int* getTailX() { return this->tailX; }
```

Definir clases



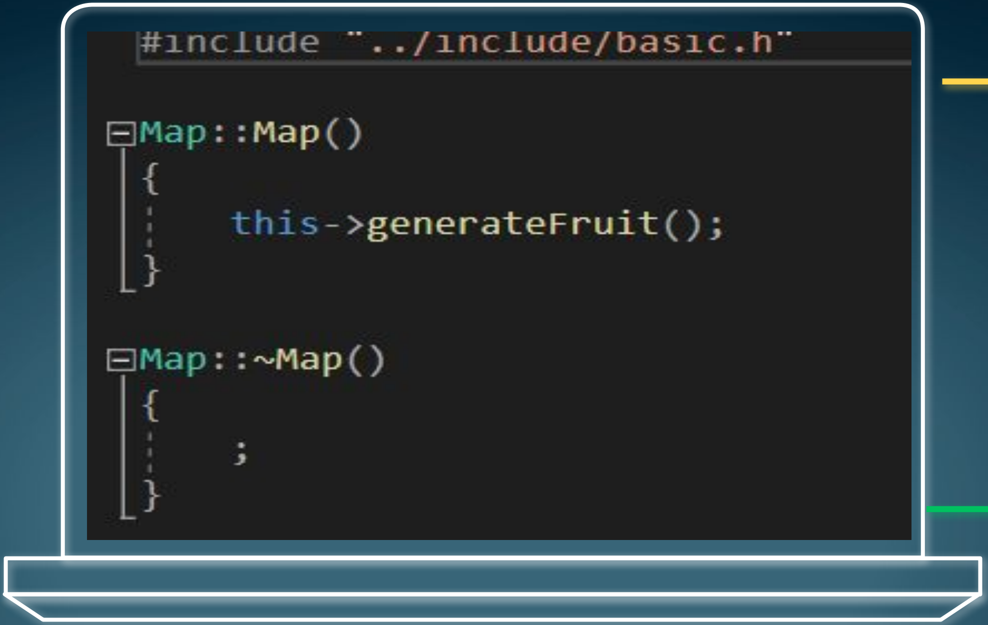
```
        void setX(int x_) { this->x = x_; }
        void setY(int y_) { this->y = y_; }
        void setLIVE(int live_) { this->live = live_; }
        void setSCORE(int score_) { this->score = score_; }
        void setColision(int choque_) { this->choques= choque_; }
        void setDirection(Direction direction_) { this->direction = direction_; }
        void setNTail(int nTail_) { this->nTail = nTail_; }

        void menu();
        void move();
        void keyPressEvent();
};

#endif
```

Inicializar variables



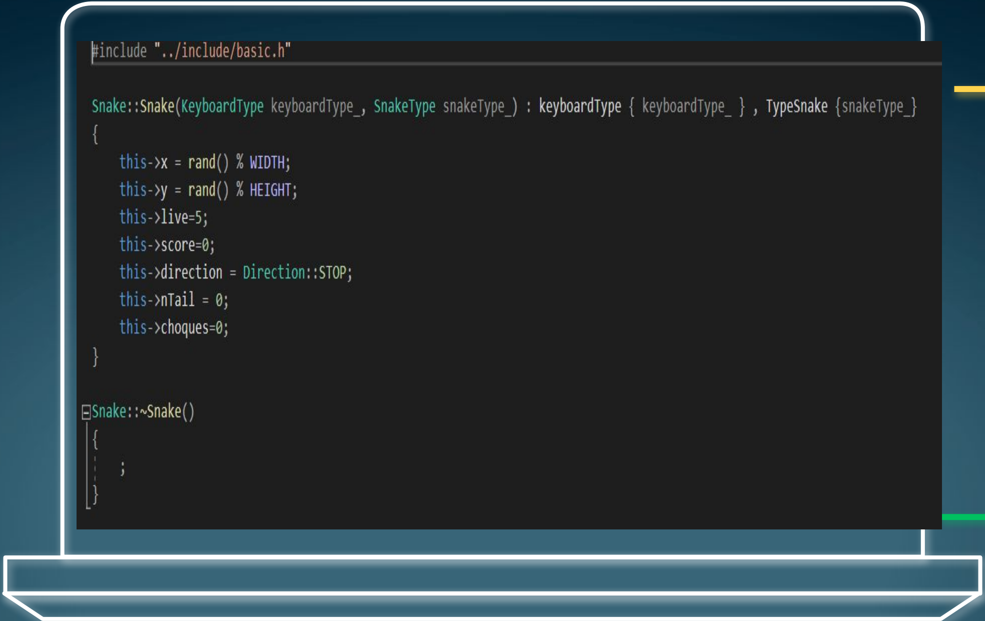
```
#include "../include/basic.h"

Game::Game(int width_, int height_): width { width_ }, height { height_ }
{
    this->state = GameState::GAME;
    this->map = new Map();
    this->velocidad = 100000000;
    this->a=0;
}

Game::~Game()
{
    ;
}

/*void Game::configure()
```

Inicializar variables



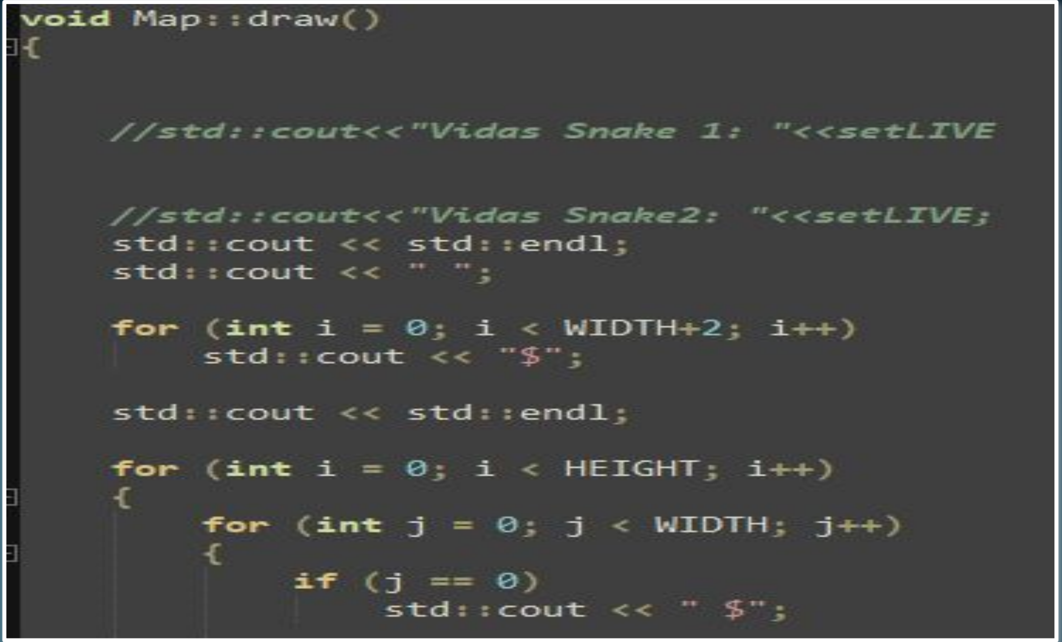
```
#include "../include/basic.h"

Map::Map()
{
    this->generateFruit();
}

Map::~Map()
{
    ;
}
```

Inicializar variables

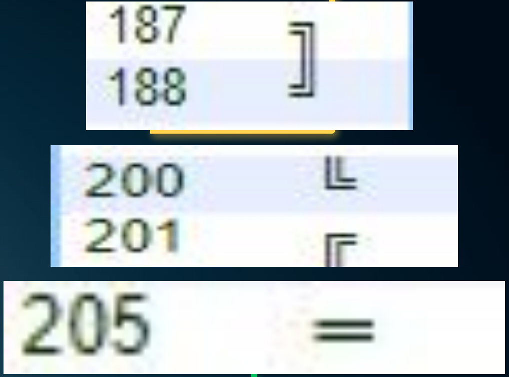


```
#include "../include/basic.h"

Snake::Snake(KeyboardType keyboardType_, SnakeType snakeType_) : keyboardType { keyboardType_ } , TypeSnake {snakeType_}
{
    this->x = rand() % WIDTH;
    this->y = rand() % HEIGHT;
    this->live=5;
    this->score=0;
    this->direction = Direction::STOP;
    this->nTail = 0;
    this->choques=0;
}

Snake::~Snake()
{
    ;
}
```

Función dibujar

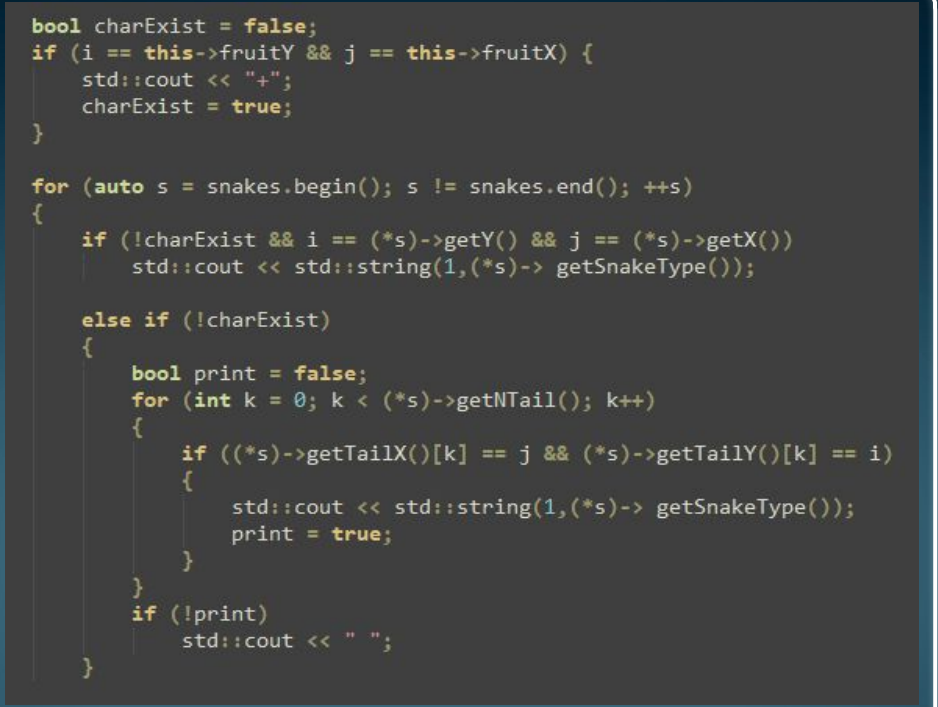


```
void Map::draw()
{


    //std::cout<<"Vidas Snake 1: "<<setLIVE


    //std::cout<<"Vidas Snake2: "<<setLIVE;
    std::cout << std::endl;
    std::cout << " ";

    for (int i = 0; i < WIDTH+2; i++)
        std::cout << "$";

    std::cout << std::endl;

    for (int i = 0; i < HEIGHT; i++)
    {
        for (int j = 0; j < WIDTH; j++)
        {
            if (j == 0)
                std::cout << " $";
```



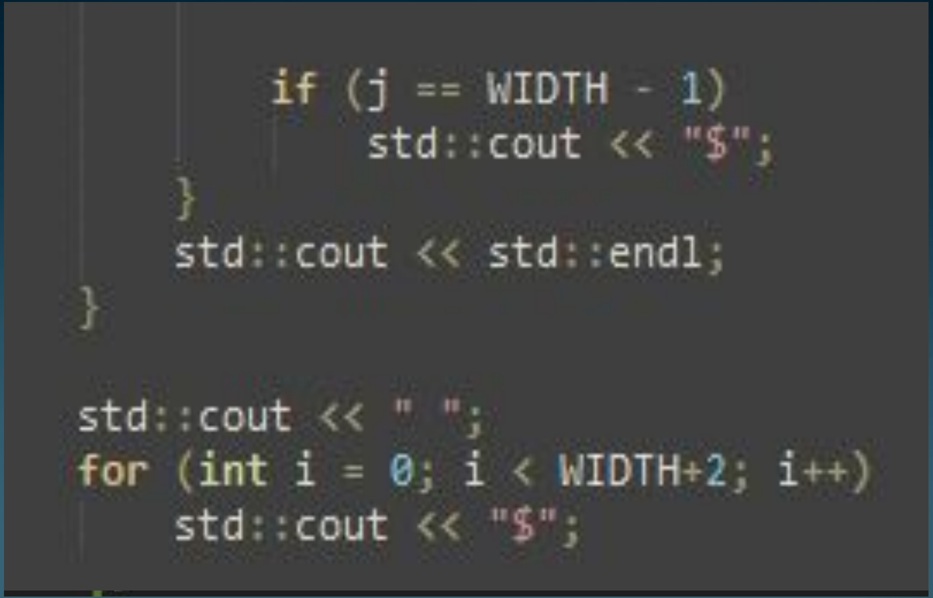
```
187    ╗
188    ╝
200    ╚
201    ╔
205    ═
```

Función dibujar



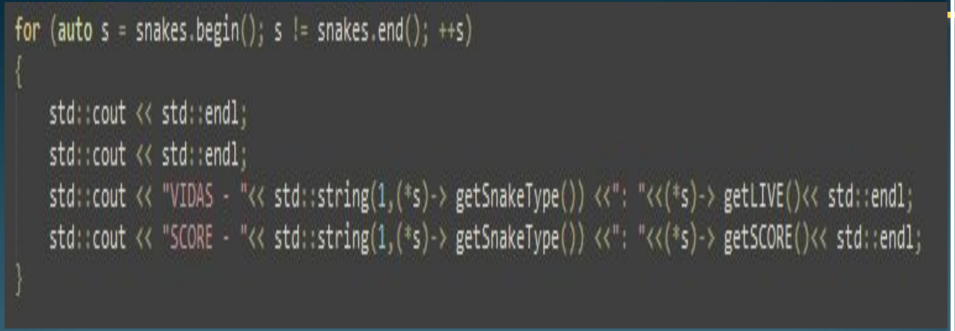
```
bool charExist = false;
if (i == this->fruitY && j == this->fruitX) {
    std::cout << "+";
    charExist = true;
}

for (auto s = snakes.begin(); s != snakes.end(); ++s)
{
    if (!charExist && i == (*s)->getY() && j == (*s)->getX())
        std::cout << std::string(1,(*s)-> getSnakeType());

    else if (!charExist)
    {
        bool print = false;
        for (int k = 0; k < (*s)->getNTail(); k++)
        {
            if ((*s)->getTailX()[k] == j && (*s)->getTailY()[k] == i)
            {
                std::cout << std::string(1,(*s)-> getSnakeType());
                print = true;
            }
        }
        if (!print)
            std::cout << " ";
    }
```

Función dibujar



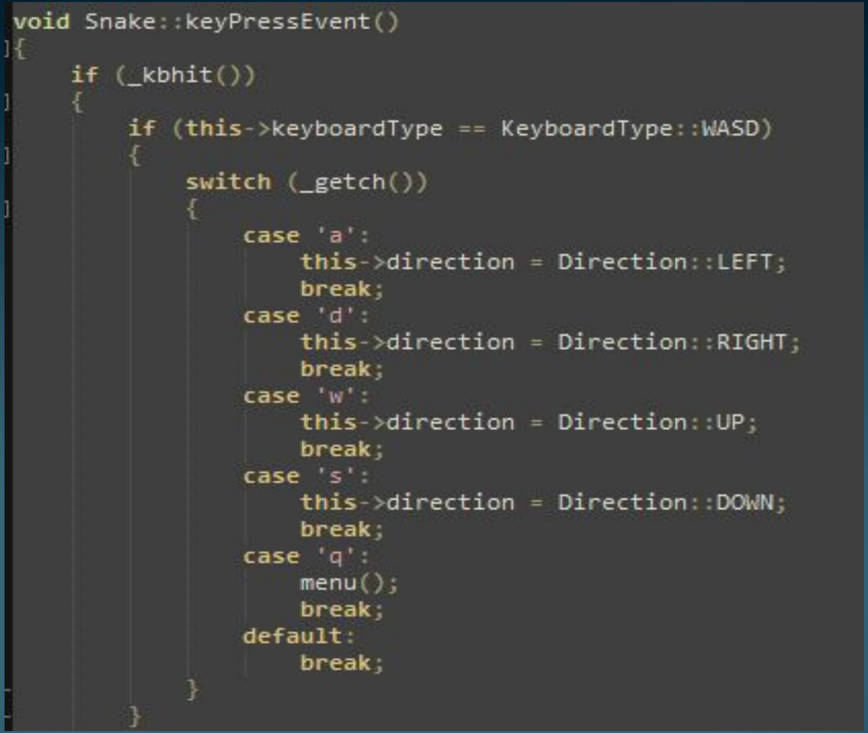
```
            if (j == WIDTH - 1)
                std::cout << "$";
        }
        std::cout << std::endl;
    }

    std::cout << " ";
    for (int i = 0; i < WIDTH+2; i++)
        std::cout << "$";
```

Función dibujar



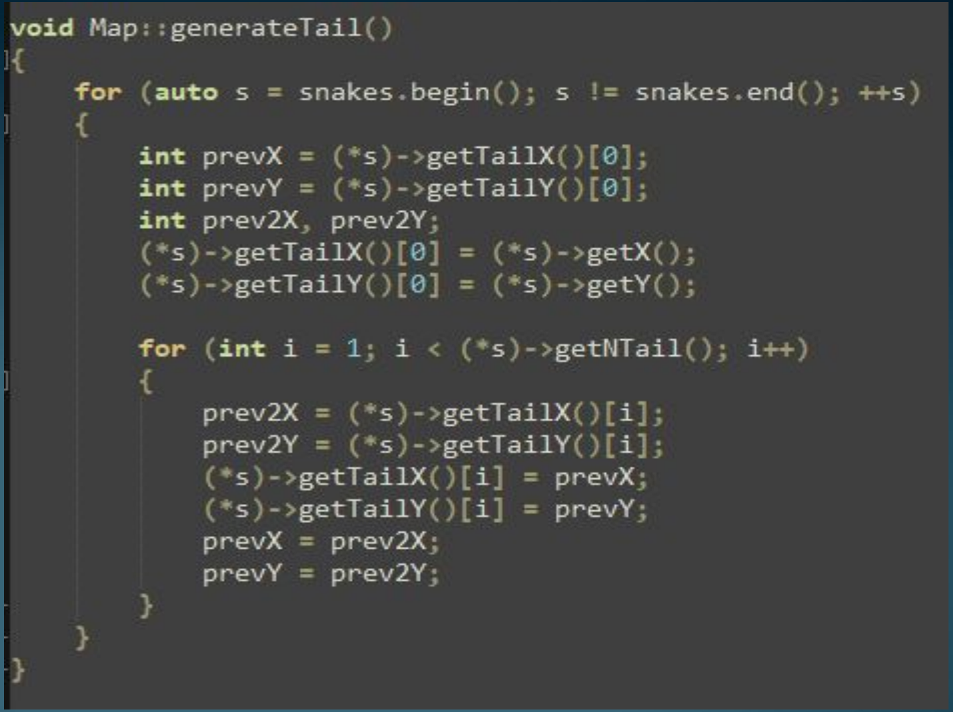
```cpp
for (auto s = snakes.begin(); s != snakes.end(); ++s)
{
    std::cout << std::endl;
    std::cout << std::endl;
    std::cout << "VIDAS - "<< std::string(1,(*s)-> getSnakeType()) <<": "<<(*s)-> getLIVE()<< std::endl;
    std::cout << "SCORE - "<< std::string(1,(*s)-> getSnakeType()) <<": "<<(*s)-> getSCORE()<< std::endl;
}
```

Función teclado



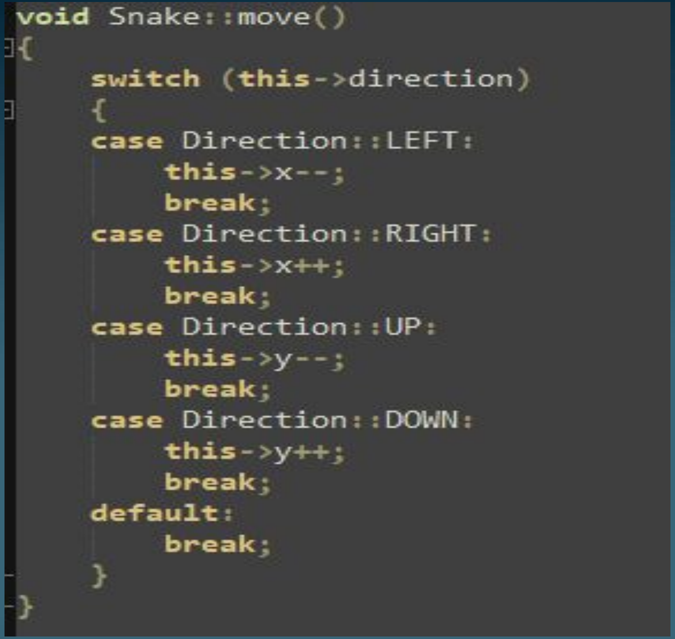
```
void Snake::keyPressEvent()
{
    if (_kbhit())
    {
        if (this->keyboardType == KeyboardType::WASD)
        {
            switch (_getch())
            {
                case 'a':
                    this->direction = Direction::LEFT;
                    break;
                case 'd':
                    this->direction = Direction::RIGHT;
                    break;
                case 'w':
                    this->direction = Direction::UP;
                    break;
                case 's':
                    this->direction = Direction::DOWN;
                    break;
                case 'q':
                    menu();
                    break;
                default:
                    break;
            }
        }
```

Movimiento de la cola



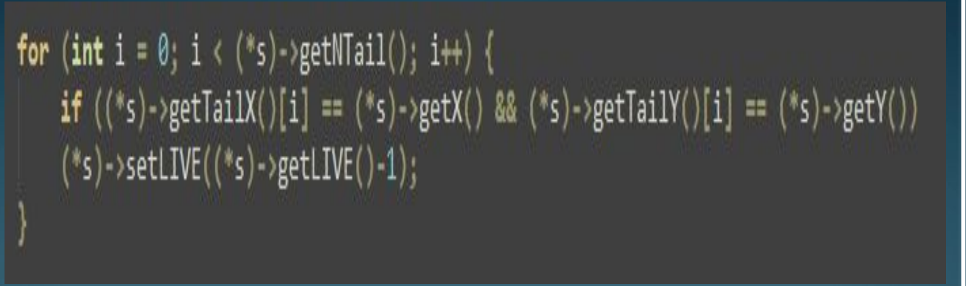
```cpp
void Map::generateTail()
{
    for (auto s = snakes.begin(); s != snakes.end(); ++s)
    {
        int prevX = (*s)->getTailX()[0];
        int prevY = (*s)->getTailY()[0];
        int prev2X, prev2Y;
        (*s)->getTailX()[0] = (*s)->getX();
        (*s)->getTailY()[0] = (*s)->getY();

        for (int i = 1; i < (*s)->getNTail(); i++)
        {
            prev2X = (*s)->getTailX()[i];
            prev2Y = (*s)->getTailY()[i];
            (*s)->getTailX()[i] = prevX;
            (*s)->getTailY()[i] = prevY;
            prevX = prev2X;
            prevY = prev2Y;
        }
    }
}
```

Movimiento Snake



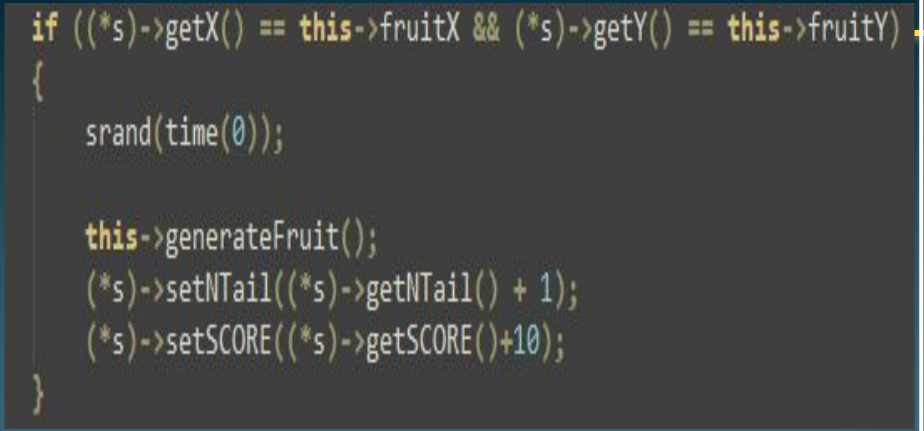
```
void Snake::move()
{
    switch (this->direction)
    {
    case Direction::LEFT:
        this->x--;
        break;
    case Direction::RIGHT:
        this->x++;
        break;
    case Direction::UP:
        this->y--;
        break;
    case Direction::DOWN:
        this->y++;
        break;
    default:
        break;
    }
}
```

Maneras de morir



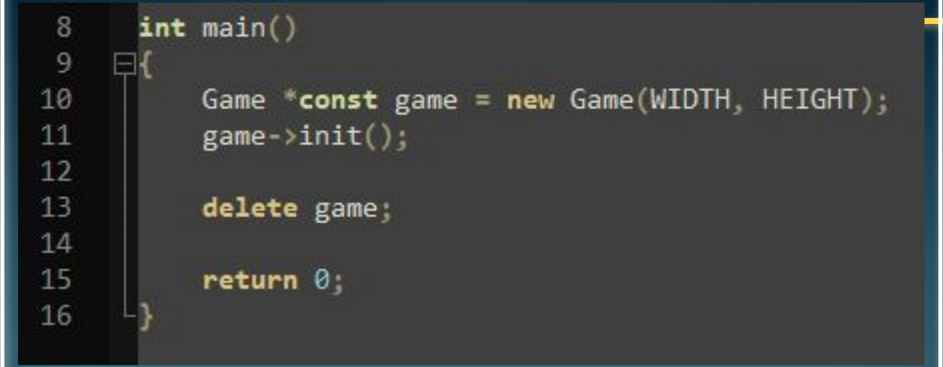
```
for (int i = 0; i < (*s)->getNTail(); i++) {
    if ((*s)->getTailX()[i] == (*s)->getX() && (*s)->getTailY()[i] == (*s)->getY())
    (*s)->setLIVE((*s)->getLIVE()-1);
}
```

Comer frutas



```
if ((*s)->getX() == this->fruitX && (*s)->getY() == this->fruitY)
{
    srand(time(0));

    this->generateFruit();
    (*s)->setNTail((*s)->getNTail() + 1);
    (*s)->setSCORE((*s)->getSCORE()+10);
}
```

Llamar funciones



```
int main()
{
    Game *const game = new Game(WIDTH, HEIGHT);
    game->init();

    delete game;

    return 0;
}
```

https://github.com/Anthony-Rodriguez18/SNAKE_CON_CLASES

GRACIAS