AyP/CODIGOS GENERADOS POR IA.cpp

```cpp
/* PROMPT: escribe un código en c++ para encontrar todos los números primos
hasta un número ingresado*/
#include <iostream>

using namespace std;

bool isPrime(int number) {
    if (number <= 1) {
        return false;
    }
    if (number <= 3) {
        return true;
    }
    if (number % 2 == 0 || number % 3 == 0) {
        return false;
    }

    for (int i = 5; i * i <= number; i += 6) {
        if (number % i == 0 || number % (i + 2) == 0) {
            return false;
        }
    }

    return true;
}

int main() {
    int n;
    cout << "Enter a positive integer: ";
    cin >> n;

    cout << "Prime numbers up to " << n << " are:" << endl;

    for (int i = 2; i <= n; ++i) {
        if (isPrime(i)) {
            cout << i << " ";
        }
    }

    cout << endl;

    return 0;
}


/* PROMPT: escribe un código en c++ para encontrar todos los números primos
hasta un número ingresado*/
#include <iostream>
using namespace std;

bool esPrimo(int num) {
```

```
    if (num <= 1) {
        return false;
    }
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return false;
        }
    }
    return true;
}

int main() {
    int limite;

    cout << "Ingresa un número límite: ";
    cin >> limite;

    cout << "Números primos hasta " << limite << ": ";
    for (int i = 2; i <= limite; i++) {
        if (esPrimo(i)) {
            cout << i << " ";
        }
    }

    cout << endl;

    return 0;
}

/* BINGAI: write the entire code for pong in c++*/

class GameObject {
  protected:
    int x; // x coordinate
    int y; // y coordinate
    int w; // width
    int h; // height
    SDL_Texture* texture; // image
  public:
    GameObject(int x, int y, int w, int h, SDL_Texture* texture); // constructor
    virtual ~GameObject(); // destructor
    virtual void update() = 0; // update position and state
    virtual void render(SDL_Renderer* renderer) = 0; // draw on screen
};

class Ball : public GameObject {
  private:
    int dx; // horizontal speed
    int dy; // vertical speed
  public:
    Ball(int x, int y, int w, int h, SDL_Texture* texture); // constructor
    ~Ball(); // destructor
    void update() override; // bounce off walls and paddles
    void render(SDL_Renderer* renderer) override; // draw on screen
};
```

```cpp

class Paddle : public GameObject {
  private:
    int speed; // vertical speed
  public:
    Paddle(int x, int y, int w, int h, SDL_Texture* texture); // constructor
    ~Paddle(); // destructor
    void update() override; // move up or down according to input
    void render(SDL_Renderer* renderer) override; // draw on screen
};

class Score : public GameObject {
  private:
    int value; // score value
    TTF_Font* font; // font
    SDL_Color color; // color
  public:
    Score(int x, int y, int w, int h, TTF_Font* font, SDL_Color color); // constructor
    ~Score(); // destructor
    void update() override; // increase or reset value
    void render(SDL_Renderer* renderer) override; // draw on screen
};

class GameManager {
  private:
    SDL_Window* window; // window
    SDL_Renderer* renderer; // renderer
    SDL_Event event; // event handler
    bool running; // game state
    Ball* ball; // ball object
    Paddle* paddle1; // paddle1 object
    Paddle* paddle2; // paddle2 object
    Score* score1; // score1 object
    Score* score2; // score2 object
  public:
    GameManager(); // constructor
    ~GameManager(); // destructor
    void init(); // initialize game components
    void loadMedia(); // load images and fonts
    void handleEvents(); // handle user input and window events
    void update(); // update game logic and objects
    void render(); // render game objects on screen
    void clean(); // free memory and quit SDL subsystems
};

/* CHATGPT3.5: write the entire code for pong in c++*/

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

using namespace std;

bool quit = false;
```

```
161 const int width = 20;
162 const int height = 10;
163 int ballX, ballY, playerX, playerY, enemyX, enemyY, score;
164
165 enum Direction { STOP = 0, LEFT, RIGHT, UP, DOWN };
166 Direction ballDir;
167
168 void Setup() {
169     quit = false;
170     ballX = width / 2;
171     ballY = height / 2;
172     playerX = width / 2;
173     playerY = height - 1;
174     enemyX = width / 2;
175     enemyY = 0;
176     score = 0;
177     ballDir = STOP;
178 }
179
180 void Draw() {
181     system("cls");
182
183     for (int i = 0; i < width + 2; i++)
184         cout << "#";
185     cout << endl;
186
187     for (int i = 0; i < height; i++) {
188         for (int j = 0; j < width; j++) {
189             if (j == 0)
190                 cout << "#";
191
192             if (i == ballY && j == ballX)
193                 cout << "O";
194             else if (i == playerY && j == playerX)
195                 cout << "P";
196             else if (i == enemyY && j == enemyX)
197                 cout << "E";
198             else
199                 cout << " ";
200
201             if (j == width - 1)
202                 cout << "#";
203         }
204         cout << endl;
205     }
206
207     for (int i = 0; i < width + 2; i++)
208         cout << "#";
209     cout << endl;
210
211     cout << "Score:" << score << endl;
212 }
213
214 void Input() {
215     if (_kbhit()) {
```

```
        switch (_getch()) {
        case 'a':
            ballDir = LEFT;
            break;
        case 'd':
            ballDir = RIGHT;
            break;
        case 'w':
            ballDir = UP;
            break;
        case 's':
            ballDir = DOWN;
            break;
        case 'x':
            quit = true;
            break;
        }
    }
}

void Logic() {
    switch (ballDir) {
    case LEFT:
        ballX--;
        break;
    case RIGHT:
        ballX++;
        break;
    case UP:
        ballY--;
        break;
    case DOWN:
        ballY++;
        break;
    }

    if (ballX < 0 || ballX >= width || ballY < 0 || ballY >= height) {
        quit = true;
    }

    if (ballX == playerX && ballY == playerY) {
        ballDir = UP;
        score++;
    }

    if (ballX == enemyX && ballY == enemyY) {
        ballDir = DOWN;
    }
}

int main() {
    Setup();

    while (!quit) {
        Draw();
```

```
        Input();
        Logic();
        Sleep(100);  // Sleep for a short duration to control the game speed
    }

    return 0;
}

/* BARD: write the entire code for pong in c++*/

#include <iostream>
#include <vector>

using namespace std;

class Ball {
public:
  int x, y;
  int dir_x, dir_y;
  int radius;

  Ball() {
    x = 0;
    y = 0;
    dir_x = -1;
    dir_y = 0;
    radius = 10;
  }

  void move() {
    x += dir_x;
    y += dir_y;
  }

  bool collidesWith(Ball other) {
    return (x - other.x)*(x - other.x) + (y - other.y)*(y - other.y) <=
radius*radius;
  }
};

class Paddle {
public:
  int x, y;
  int height;
  int width;

  Paddle(int x, int y, int height, int width) {
    this->x = x;
    this->y = y;
    this->height = height;
    this->width = width;
  }

  void moveUp() {
    y -= 10;
```

```
  }

  void moveDown() {
    y += 10;
  }
};

int main() {
  int width = 600;
  int height = 400;

  Ball ball;
  Paddle paddle1(100, height / 2, 20, 100);
  Paddle paddle2(width - 100, height / 2, 20, 100);

  while (true) {
    // Clear the screen
    cout << "\033[2J\033[0;0H";

    // Draw the ball
    cout << ball;

    // Draw the paddles
    cout << paddle1;
    cout << paddle2;

    // Move the ball
    ball.move();

    // Check for collisions
    if (ball.collidesWith(paddle1)) {
      ball.dir_x = 1;
    } else if (ball.collidesWith(paddle2)) {
      ball.dir_x = -1;
    }

    // Check if the ball went out of bounds
    if (ball.x < 0 || ball.x > width) {
      // Game over!
      break;
    }

    // Wait for a few milliseconds before the next frame
    Sleep(10);
  }

  return 0;
}
```