

## AyP/CODIGOS GENERADOS POR IA.cpp

```
1  /* PROMPT: escribe un código en c++ para encontrar todos los números primos
2  hasta un número ingresado*/
3  #include <iostream>
4
5  using namespace std;
6
7  bool isPrime(int number) {
8      if (number <= 1) {
9          return false;
10     }
11     if (number <= 3) {
12         return true;
13     }
14     if (number % 2 == 0 || number % 3 == 0) {
15         return false;
16     }
17
18     for (int i = 5; i * i <= number; i += 6) {
19         if (number % i == 0 || number % (i + 2) == 0) {
20             return false;
21         }
22     }
23
24     return true;
25 }
26
27 int main() {
28     int n;
29     cout << "Enter a positive integer: ";
30     cin >> n;
31
32     cout << "Prime numbers up to " << n << " are:" << endl;
33
34     for (int i = 2; i <= n; ++i) {
35         if (isPrime(i)) {
36             cout << i << " ";
37         }
38     }
39
40     cout << endl;
41
42     return 0;
43 }
44
45
46 /* PROMPT: escribe un código en c++ para encontrar todos los números primos
47 hasta un número ingresado*/
48 #include <iostream>
49 using namespace std;
50
51 bool esPrimo(int num) {
```

```
52     if (num <= 1) {
53         return false;
54     }
55     for (int i = 2; i * i <= num; i++) {
56         if (num % i == 0) {
57             return false;
58         }
59     }
60     return true;
61 }
62
63 int main() {
64     int limite;
65
66     cout << "Ingresa un número límite: ";
67     cin >> limite;
68
69     cout << "Números primos hasta " << limite << ": ";
70     for (int i = 2; i <= limite; i++) {
71         if (esPrimo(i)) {
72             cout << i << " ";
73         }
74     }
75
76     cout << endl;
77
78     return 0;
79 }
80
81 /* BINGAI: write the entire code for pong in c++*/
82
83 class GameObject {
84     protected:
85         int x; // x coordinate
86         int y; // y coordinate
87         int w; // width
88         int h; // height
89         SDL_Texture* texture; // image
90     public:
91         GameObject(int x, int y, int w, int h, SDL_Texture* texture); // constructor
92         virtual ~GameObject(); // destructor
93         virtual void update() = 0; // update position and state
94         virtual void render(SDL_Renderer* renderer) = 0; // draw on screen
95 };
96
97 class Ball : public GameObject {
98     private:
99         int dx; // horizontal speed
100        int dy; // vertical speed
101    public:
102        Ball(int x, int y, int w, int h, SDL_Texture* texture); // constructor
103        ~Ball(); // destructor
104        void update() override; // bounce off walls and paddles
105        void render(SDL_Renderer* renderer) override; // draw on screen
106 };
```

```
107
108 class Paddle : public GameObject {
109     private:
110         int speed; // vertical speed
111     public:
112         Paddle(int x, int y, int w, int h, SDL_Texture* texture); // constructor
113         ~Paddle(); // destructor
114         void update() override; // move up or down according to input
115         void render(SDL_Renderer* renderer) override; // draw on screen
116 };
117
118 class Score : public GameObject {
119     private:
120         int value; // score value
121         TTF_Font* font; // font
122         SDL_Color color; // color
123     public:
124         Score(int x, int y, int w, int h, TTF_Font* font, SDL_Color color); //
constructor
125         ~Score(); // destructor
126         void update() override; // increase or reset value
127         void render(SDL_Renderer* renderer) override; // draw on screen
128 };
129
130 class GameManager {
131     private:
132         SDL_Window* window; // window
133         SDL_Renderer* renderer; // renderer
134         SDL_Event event; // event handler
135         bool running; // game state
136         Ball* ball; // ball object
137         Paddle* paddle1; // paddle1 object
138         Paddle* paddle2; // paddle2 object
139         Score* score1; // score1 object
140         Score* score2; // score2 object
141     public:
142         GameManager(); // constructor
143         ~GameManager(); // destructor
144         void init(); // initialize game components
145         void loadMedia(); // load images and fonts
146         void handleEvents(); // handle user input and window events
147         void update(); // update game logic and objects
148         void render(); // render game objects on screen
149         void clean(); // free memory and quit SDL subsystems
150 };
151
152 /* CHATGPT3.5: write the entire code for pong in c++*/
153
154 # include <iostream>
155 # include <conio.h>
156 # include <windows.h>
157
158 using namespace std;
159
160 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()) {
```

```
216     switch (_getch()) {
217     case 'a':
218         ballDir = LEFT;
219         break;
220     case 'd':
221         ballDir = RIGHT;
222         break;
223     case 'w':
224         ballDir = UP;
225         break;
226     case 's':
227         ballDir = DOWN;
228         break;
229     case 'x':
230         quit = true;
231         break;
232     }
233 }
234 }
235
236 void Logic() {
237     switch (ballDir) {
238     case LEFT:
239         ballX--;
240         break;
241     case RIGHT:
242         ballX++;
243         break;
244     case UP:
245         ballY--;
246         break;
247     case DOWN:
248         ballY++;
249         break;
250     }
251
252     if (ballX < 0 || ballX >= width || ballY < 0 || ballY >= height) {
253         quit = true;
254     }
255
256     if (ballX == playerX && ballY == playerY) {
257         ballDir = UP;
258         score++;
259     }
260
261     if (ballX == enemyX && ballY == enemyY) {
262         ballDir = DOWN;
263     }
264 }
265
266 int main() {
267     Setup();
268
269     while (!quit) {
270         Draw();
```

```
271     Input();
272     Logic();
273     Sleep(100); // Sleep for a short duration to control the game speed
274 }
275
276     return 0;
277 }
278
279 /* BARD: write the entire code for pong in c++*/
280
281 #include <iostream>
282 #include <vector>
283
284 using namespace std;
285
286 class Ball {
287 public:
288     int x, y;
289     int dir_x, dir_y;
290     int radius;
291
292     Ball() {
293         x = 0;
294         y = 0;
295         dir_x = -1;
296         dir_y = 0;
297         radius = 10;
298     }
299
300     void move() {
301         x += dir_x;
302         y += dir_y;
303     }
304
305     bool collidesWith(Ball other) {
306         return (x - other.x)*(x - other.x) + (y - other.y)*(y - other.y) <=
radius*radius;
307     }
308 };
309
310 class Paddle {
311 public:
312     int x, y;
313     int height;
314     int width;
315
316     Paddle(int x, int y, int height, int width) {
317         this->x = x;
318         this->y = y;
319         this->height = height;
320         this->width = width;
321     }
322
323     void moveUp() {
324         y -= 10;
```

```
325     }
326
327     void moveDown() {
328         y += 10;
329     }
330 };
331
332 int main() {
333     int width = 600;
334     int height = 400;
335
336     Ball ball;
337     Paddle paddle1(100, height / 2, 20, 100);
338     Paddle paddle2(width - 100, height / 2, 20, 100);
339
340     while (true) {
341         // Clear the screen
342         cout << "\033[2J\033[0;0H";
343
344         // Draw the ball
345         cout << ball;
346
347         // Draw the paddles
348         cout << paddle1;
349         cout << paddle2;
350
351         // Move the ball
352         ball.move();
353
354         // Check for collisions
355         if (ball.collidesWith(paddle1)) {
356             ball.dir_x = 1;
357         } else if (ball.collidesWith(paddle2)) {
358             ball.dir_x = -1;
359         }
360
361         // Check if the ball went out of bounds
362         if (ball.x < 0 || ball.x > width) {
363             // Game over!
364             break;
365         }
366
367         // Wait for a few milliseconds before the next frame
368         Sleep(10);
369     }
370
371     return 0;
372 }
373
374
```