

Prime Jack - hitos y prompts

Valentina Mesa

September 2024

1 Hito. Estructura inicial del código en Python y c++

Prompt Usado:

“Eres un desarrollador y te contratan para realizar un juego en Python y C++. El juego consiste en lo siguiente:

En lugar de cartas, se usan 2 dados de 6 lados. El objetivo es llegar a 23 o al número más cercano, priorizando números primos. Un número primo gana frente a uno mayor. Entre primos, gana el mayor.

Si un jugador saca dobles, puede dividir el lanzamiento y usar 3 dados adicionales para cada nueva tirada.

Si sale un 12, el jugador puede lanzar solo 1 dado más, con un máximo de 18.”

Resultado:

ChatGPT generó un código con términos que aún no comprendo bien, lo que lo hace algo complejo. Utiliza decoradores para penalizar automáticamente a los jugadores que exceden el valor permitido, además de listas por comprensión y funciones lambda para simplificar operaciones. La implementación con clases también mejora la estructura del código.

Listing 1: Código en Python

```
1 import random
2
3 def penalizar_si_excede(func):
4     return lambda *args, **kwargs:
5         max(func(*args, **kwargs) - random.randint(1, 5), 0)
6         if func(*args, **kwargs) > 23 else func(*args, **kwargs)
7
8 class Jugador:
9     def __init__(self, nombre):
```

```

10         self.nombre = nombre
11         self.puntaje = 0
12
13     @penalizar_si_excede
14     def jugar_turno(self):
15         dados = [random.randint(1, 6) for _ in range(2)]
16         self.puntaje = sum(dados)
17         if dados[0] == dados[1]:
18             self.puntaje = sum([random.randint(1, 6) for _ in range(3)])
19         if self.puntaje == 12:
20             self.puntaje += random.randint(1, 6)
21         return self.puntaje
22
23 def es_primo(num):
24     return num > 1 and all(num % i for i in range(2, int(num**0.5) + 1))
25
26 def determinar_ganador(j1, j2):
27     if es_primo(j1.puntaje) != es_primo(j2.puntaje):
28         return f"{j1.nombre} gana con un n mero primo."
29     if es_primo(j1.puntaje) else f"{j2.nombre} gana con un n mero primo."
30     return f"{j1.nombre} gana." if j1.puntaje > j2.puntaje else f"{j2.nombre} gana."
31
32 jugador1, jugador2 = Jugador("Jugador 1"), Jugador("Jugador 2")
33 print(determinar_ganador(jugador1, jugador2))

```

Listing 2: Código en C++

```

1  #include <iostream>
2  #include <vector>
3  #include <cstdlib> // Para rand() y srand()
4  #include <ctime>   // Para time()
5  #include <cmath>   // Para sqrt()
6
7  using namespace std;
8
9  // Funci n para penalizar si el puntaje excede 23
10 int penalizar_si_excede(int puntaje) {
11     if (puntaje > 23) {
12         int penalizacion = rand() % 5 + 1; // Genera un n mero aleatorio entre 1 y 5
13         return max(puntaje - penalizacion, 0);
14     }
15     return puntaje;
16 }
17
18 // Clase Jugador
19 class Jugador {
20 public:
21     string nombre;
22     int puntaje;
23
24     Jugador(string nombre) : nombre(nombre), puntaje(0) {}
25
26     int jugar_turno() {
27         vector<int> dados = {rand() % 6 + 1, rand() % 6 + 1}; // Dos dados de 1 a 6
28         puntaje = dados[0] + dados[1];
29
30         // Si los dados son iguales (dobles), tirar 3 dados adicionales

```

```

31         if (dados[0] == dados[1]) {
32             puntaje = 0;
33             for (int i = 0; i < 3; ++i) {
34                 puntaje += rand() % 6 + 1;
35             }
36         }
37
38         // Si el puntaje es 12, tirar 1 dado m s
39         if (puntaje == 12) {
40             puntaje += rand() % 6 + 1;
41         }
42
43         // Aplicar penalizaci n si excede 23
44         puntaje = penalizar_si_excede(puntaje);
45
46         return puntaje;
47     }
48 };
49
50 // Funci n para determinar si un n mero es primo
51 bool es_primo(int num) {
52     if (num <= 1) return false;
53     for (int i = 2; i <= sqrt(num); ++i) {
54         if (num % i == 0) return false;
55     }
56     return true;
57 }
58
59 // Funci n para determinar el ganador
60 string determinar_ganador(Jugador& j1, Jugador& j2) {
61     if (es_primo(j1.puntaje) != es_primo(j2.puntaje)) {
62         if (es_primo(j1.puntaje)) {
63             return j1.nombre + " gana con un n mero primo.";
64         } else {
65             return j2.nombre + " gana con un n mero primo.";
66         }
67     }
68
69     return (j1.puntaje > j2.puntaje) ? j1.nombre + " gana." : j2.nombre + " gana.";
70 }
71
72 int main() {
73     // Semilla para generar n meros aleatorios
74     srand(time(0));
75
76     // Crear los jugadores
77     Jugador jugador1("Jugador 1");
78     Jugador jugador2("Jugador 2");
79
80     // Cada jugador juega su turno
81     jugador1.jugar_turno();
82     jugador2.jugar_turno();
83
84     // Determinar y mostrar el ganador
85     cout << determinar_ganador(jugador1, jugador2) << endl;
86
87     return 0;

```

```
88 }
89
90 }
```

2 Hito. Simplificación del código en Python y c++

Prompt Usado:

“Solo puedes usar primitivas básicas. Condicionales, Ciclos,etc.”

Resultado:

Con este Prompt, chat gpt mejoró el código usando funciones simples como `es_primo` para verificar números primos, `lanzar_dados` para lanzar dados, y `jugar_ronda` para gestionar el turno de juego. Este enfoque es más directo y fácil de entender porque cada función realiza una tarea específica sin la complejidad de clases o decoradores. Sin embargo, el juego no corría como yo esperaba.

Listing 3: Código en Python

```
1 import random
2
3 # Funci n para verificar si un n mero es primo
4 def es_primo(n):
5     if n < 2:
6         return False
7     for i in range(2, int(n**0.5) + 1):
8         if n % i == 0:
9             return False
10    return True
11
12 # Funci n para lanzar dados
13 def lanzar_dados(cantidad=2):
14     return [random.randint(1, 6) for _ in range(cantidad)]
15
16 # Funci n para jugar una ronda
17 def jugar_ronda():
18     total = 0
19     while total < 23:
20         dados = lanzar_dados()
21         print(f"Lanzaste: {dados}")
22
23     # Si el jugador saca dobles
24     if dados[0] == dados[1]:
25         print(f" Sacaste dobles de {dados[0]}!")
26         # Permite dividir el lanzamiento
27         for _ in range(2): # Dos lanzamientos adicionales
28             nuevos_dados = lanzar_dados(3) # Se lanzan 3 dados
29             print(f"Nuevos dados tras dividir: {nuevos_dados}")
30             total += sum(nuevos_dados)
```

```

31         if total >= 23:
32             break
33         continue
34
35     # Si el jugador saca 12
36     if sum(dados) == 12:
37         print(" Sacaste un 12! Puedes lanzar un solo dado.")
38         dado_extra = lanzar_dados(1)[0]
39         print(f"Lanzaste el dado extra: {dado_extra}")
40         total += dado_extra
41         if total >= 23:
42             break
43         continue
44
45     total += sum(dados)
46
47     # Opci n de detener el juego si se llega a 23 o m s
48     if total >= 23:
49         break
50
51     print(f"Total acumulado: {total}")
52
53     return total
54
55 # Funci n para determinar el ganador
56 def determinar_ganador(jugador1, jugador2):
57     # Revisar si alguno tiene un n mero primo
58     if es_primo(jugador1) and es_primo(jugador2):
59         return 1 if jugador1 > jugador2 else 2
60     if es_primo(jugador1):
61         return 1
62     if es_primo(jugador2):
63         return 2
64
65     # Si ninguno tiene primos, gana el que est m s cerca a 23
66     return 1 if jugador1 <= 23 and (jugador1 > jugador2 or jugador2 > 23) else 2
67
68 # Juego principal
69 def juego_prime_jack():
70     print("Turno del Jugador 1")
71     jugador1 = jugar_ronda()
72     print(f"Puntuaci n final del Jugador 1: {jugador1}\n")
73
74     print("Turno del Jugador 2")
75     jugador2 = jugar_ronda()
76     print(f"Puntuaci n final del Jugador 2: {jugador2}\n")
77
78     ganador = determinar_ganador(jugador1, jugador2)
79     print(f"El ganador es el Jugador {ganador}")
80
81 # Ejecutar el juego
82 juego_prime_jack()

```

Listing 4: C3digo en C++

```

1 #include <iostream>
2 #include <vector>

```

```

3 #include <cstdlib> // Para rand() y srand()
4 #include <ctime> // Para time()
5 #include <cmath> // Para sqrt()
6
7 using namespace std;
8
9 // Funci n para penalizar si el puntaje excede 23
10 int penalizar_si_excede(int puntaje) {
11     if (puntaje > 23) {
12         int penalizacion = rand() % 5 + 1; // Genera un n mero aleatorio entre 1 y 5
13         return max(puntaje - penalizacion, 0);
14     }
15     return puntaje;
16 }
17
18 // Clase Jugador
19 class Jugador {
20 public:
21     string nombre;
22     int puntaje;
23
24     Jugador(string nombre) : nombre(nombre), puntaje(0) {}
25
26     int jugar_turno() {
27         vector<int> dados = {rand() % 6 + 1, rand() % 6 + 1}; // Dos dados de 1 a 6
28         puntaje = dados[0] + dados[1];
29
30         // Si los dados son iguales (dobles), tirar 3 dados adicionales
31         if (dados[0] == dados[1]) {
32             puntaje = 0;
33             for (int i = 0; i < 3; ++i) {
34                 puntaje += rand() % 6 + 1;
35             }
36         }
37
38         // Si el puntaje es 12, tirar 1 dado m s
39         if (puntaje == 12) {
40             puntaje += rand() % 6 + 1;
41         }
42
43         // Aplicar penalizaci n si excede 23
44         puntaje = penalizar_si_excede(puntaje);
45
46         return puntaje;
47     }
48 };
49
50 // Funci n para determinar si un n mero es primo
51 bool es_primo(int num) {
52     if (num <= 1) return false;
53     for (int i = 2; i <= sqrt(num); ++i) {
54         if (num % i == 0) return false;
55     }
56     return true;
57 }
58
59 // Funci n para determinar el ganador

```

```

60 string determinar_ganador(Jugador& j1, Jugador& j2) {
61     if (es_primo(j1.puntaje) != es_primo(j2.puntaje)) {
62         if (es_primo(j1.puntaje)) {
63             return j1.nombre + " gana con un n mero primo.";
64         } else {
65             return j2.nombre + " gana con un n mero primo.";
66         }
67     }
68
69     return (j1.puntaje > j2.puntaje) ? j1.nombre + " gana." : j2.nombre + " gana.";
70 }
71
72 int main() {
73     // Semilla para generar n meros aleatorios
74     srand(time(0));
75
76     // Crear los jugadores
77     Jugador jugador1("Jugador 1");
78     Jugador jugador2("Jugador 2");
79
80     // Cada jugador juega su turno
81     jugador1.jugar_turno();
82     jugador2.jugar_turno();
83
84     // Determinar y mostrar el ganador
85     cout << determinar_ganador(jugador1, jugador2) << endl;
86
87     return 0;
88 }
89
90 }

```

3 Hito. Correcciones del código en Python y c++

Prompt Usado:

Como el juego con el anterior código creaba la partida completa con solo ejecutar el código, le pedi a chat gpt lo siguiente: “El jugador debe escoger si desea lanzar otra vez o quedarse con el puntaje. Corrige eso.”

Resultado:

Chat gpt, creó la variable `seguir_jugando` que controla si el jugador desea continuar lanzando dados o detenerse. Después de cada lanzamiento, se pregunta al jugador si quiere seguir o quedarse con el total actual; si elige no continuar o si el total alcanza o supera 23, el turno termina. Estos ajustes permiten a los jugadores decidir si quieren seguir jugando para aumentar su puntuación o detenerse para protegerla.

Listing 5: Código en Python

```

1 import random
2
3 # Funci n para verificar si un n mero es primo
4 def es_primo(n):
5     if n < 2:
6         return False
7     for i in range(2, int(n**0.5) + 1):
8         if n % i == 0:
9             return False
10    return True
11
12 # Funci n para lanzar dados
13 def lanzar_dados(cantidad=2):
14     return [random.randint(1, 6) for _ in range(cantidad)]
15
16 # Funci n para jugar una ronda
17 def jugar_ronda():
18     total = 0
19     seguir_jugando = True
20
21     while seguir_jugando and total < 23:
22         dados = lanzar_dados()
23         print(f"Lanzaste: {dados}")
24
25         # Si el jugador saca dobles
26         if dados[0] == dados[1]:
27             print(f" Sacaste dobles de {dados[0]}!")
28             # Permite dividir el lanzamiento
29             for _ in range(2): # Dos lanzamientos adicionales
30                 nuevos_dados = lanzar_dados(3) # Se lanzan 3 dados
31                 print(f"Nuevos dados tras dividir: {nuevos_dados}")
32                 total += sum(nuevos_dados)
33                 if total >= 23:
34                     break
35             continue
36
37         # Si el jugador saca 12
38         if sum(dados) == 12:
39             print(" Sacaste un 12! Puedes lanzar un solo dado.")
40             dado_extra = lanzar_dados(1)[0]
41             print(f"Lanzaste el dado extra: {dado_extra}")
42             total += dado_extra
43             if total >= 23:
44                 break
45             continue
46
47         total += sum(dados)
48
49     print(f"Total acumulado: {total}")
50
51 # Preguntar al jugador si desea seguir lanzando o quedarse con su total actual
52 if total < 23:
53     respuesta = input(" Quieres lanzar de nuevo? (s/n): ").strip().lower()
54     if respuesta != 's':
55         seguir_jugando = False
56

```



```

57     return total
58
59 # Funci n para determinar el ganador
60 def determinar_ganador(jugador1, jugador2):
61     # Revisar si alguno tiene un n mero primo
62     if es_primo(jugador1) and es_primo(jugador2):
63         return 1 if jugador1 > jugador2 else 2
64     if es_primo(jugador1):
65         return 1
66     if es_primo(jugador2):
67         return 2
68
69     # Si ninguno tiene primos, gana el que est m s cerca a 23
70     return 1 if jugador1 <= 23 and (jugador1 > jugador2 or jugador2 > 23) else 2
71
72 # Juego principal
73 def juego_prime_jack():
74     print("Turno del Jugador 1")
75     jugador1 = jugar_ronda()
76     print(f"Puntuaci n final del Jugador 1: {jugador1}\n")
77
78     print("Turno del Jugador 2")
79     jugador2 = jugar_ronda()
80     print(f"Puntuaci n final del Jugador 2: {jugador2}\n")
81
82     ganador = determinar_ganador(jugador1, jugador2)
83     print(f"El ganador es el Jugador {ganador}")
84
85 # Ejecutar el juego
86 juego_prime_jack()

```

```

1     #include <iostream>
2     #include <cstdlib> // Para rand() y srand()
3     #include <ctime>   // Para time()
4
5     using namespace std;
6
7     // Funci n para generar un n mero aleatorio en un rango
8     int generar_numero_aleatorio(int min, int max) {
9         return min + rand() % (max - min + 1);
10    }
11
12    // Funci n para verificar si un n mero es primo
13    bool es_primo(int num) {
14        if (num <= 1) return false;
15        for (int i = 2; i * i <= num; ++i) {
16            if (num % i == 0) return false;
17        }
18        return true;
19    }
20
21    // Funci n para jugar una ronda
22    int jugar_ronda() {
23        int total = 0;
24        while (total < 23) {
25            int dado1 = generar_numero_aleatorio(1, 6);
26            int dado2 = generar_numero_aleatorio(1, 6);

```

```

27     cout << "Lanzaste: " << dado1 << " y " << dado2 << endl;
28
29     if (dado1 == dado2) {
30         cout << " Sacaste dobles de " << dado1 << "!" << endl;
31         for (int i = 0; i < 3; ++i) {
32             int dado_extra = generar_numero_aleatorio(1, 6);
33             cout << "Nuevo dado: " << dado_extra << endl;
34             total += dado_extra;
35         }
36     } else if (dado1 + dado2 == 12) {
37         cout << " Sacaste un 12! Puedes lanzar un solo dado." << endl;
38         int dado_extra = generar_numero_aleatorio(1, 6);
39         cout << "Dado extra: " << dado_extra << endl;
40         total += dado_extra;
41     } else {
42         total += dado1 + dado2;
43     }
44
45     if (total >= 23) break;
46     cout << "Total acumulado: " << total << endl;
47 }
48
49 return total;
50 }
51
52 // Funci n para determinar el ganador
53 int determinar_ganador(int puntaje1, int puntaje2) {
54     bool primo1 = es_primo(puntaje1);
55     bool primo2 = es_primo(puntaje2);
56
57     if (primo1 && !primo2) return 1;
58     if (!primo1 && primo2) return 2;
59     if (primo1 && primo2) return (puntaje1 > puntaje2) ? 1 : 2;
60     return (puntaje1 > puntaje2) ? 1 : 2;
61 }
62
63 int main() {
64     srand(static_cast<unsigned int>(time(0)));
65
66     cout << "Turno del Jugador 1" << endl;
67     int puntaje1 = jugar_ronda();
68     cout << "Puntuaci n final del Jugador 1: " << puntaje1 << endl;
69
70     cout << "Turno del Jugador 2" << endl;
71     int puntaje2 = jugar_ronda();
72     cout << "Puntuaci n final del Jugador 2: " << puntaje2 << endl;
73
74     int ganador = determinar_ganador(puntaje1, puntaje2);
75     cout << "El ganador es el Jugador " << ganador << endl;
76
77     return 0;
78 }

```

4 Hito. Correcciones en c++

Prompt Usado:

”Elimina algunas bibliotecas y usa solo las funcionalidades básicas necesarias en c++”

Resultado:

Las bibliotecas se redujeron a solo `<iostream>`, `<cstdlib>`, y `<ctime>`, y la generación de números aleatorios se simplifica con la función `generar_numero_aleatorio`. Se eliminaron `vector` y `cmath`, utilizando solo operaciones básicas de C++. El código resultante es más lineal y directo, con menos funciones y más centrado en las operaciones fundamentales del juego.

```
1 #include <iostream>
2 #include <cstdlib> // Para rand() y srand()
3 #include <ctime> // Para time()
4
5 using namespace std;
6
7 // Funci n para generar un n mero aleatorio en un rango
8 int generar_numero_aleatorio(int min, int max) {
9     return min + rand() % (max - min + 1);
10 }
11
12 // Funci n para verificar si un n mero es primo
13 bool es_primo(int num) {
14     if (num <= 1) return false;
15     for (int i = 2; i * i <= num; ++i) {
16         if (num % i == 0) return false;
17     }
18     return true;
19 }
20
21 // Funci n para jugar una ronda
22 int jugar_ronda() {
23     int total = 0;
24     while (total < 23) {
25         int dado1 = generar_numero_aleatorio(1, 6);
26         int dado2 = generar_numero_aleatorio(1, 6);
27         cout << "Lanzaste: " << dado1 << " y " << dado2 << endl;
28
29         if (dado1 == dado2) {
30             cout << " Sacaste dobles de " << dado1 << "!" << endl;
31             for (int i = 0; i < 3; ++i) {
32                 int dado_extra = generar_numero_aleatorio(1, 6);
33                 cout << "Nuevo dado: " << dado_extra << endl;
34                 total += dado_extra;
35             }
36         } else if (dado1 + dado2 == 12) {
37             cout << " Sacaste un 12! Puedes lanzar un solo dado." << endl;
38             int dado_extra = generar_numero_aleatorio(1, 6);
39             cout << "Dado extra: " << dado_extra << endl;
```

```

40         total += dado_extra;
41     } else {
42         total += dado1 + dado2;
43     }
44
45     if (total >= 23) break;
46     cout << "Total acumulado: " << total << endl;
47 }
48
49     return total;
50 }
51
52 // Funci n para determinar el ganador
53 int determinar_ganador(int puntaje1, int puntaje2) {
54     bool primo1 = es_primo(puntaje1);
55     bool primo2 = es_primo(puntaje2);
56
57     if (primo1 && !primo2) return 1;
58     if (!primo1 && primo2) return 2;
59     if (primo1 && primo2) return (puntaje1 > puntaje2) ? 1 : 2;
60     return (puntaje1 > puntaje2) ? 1 : 2;
61 }
62
63 int main() {
64     srand(static_cast<unsigned int>(time(0)));
65
66     cout << "Turno del Jugador 1" << endl;
67     int puntaje1 = jugar_ronda();
68     cout << "Puntuaci n final del Jugador 1: " << puntaje1 << endl;
69
70     cout << "Turno del Jugador 2" << endl;
71     int puntaje2 = jugar_ronda();
72     cout << "Puntuaci n final del Jugador 2: " << puntaje2 << endl;
73
74     int ganador = determinar_ganador(puntaje1, puntaje2);
75     cout << "El ganador es el Jugador " << ganador << endl;
76
77     return 0;
78 }

```

Fortalezas y debilidades.

El uso de la IA fue muy útil para generar y simplificar el código, proporcionándome soluciones y explicaciones rápidas. Sin embargo, al principio, el código generado era complejo debido al uso de técnicas avanzadas como decoradores y listas por comprensión, lo cual complicó mi entendimiento.

Pros

La simplificación del código mediante el uso de funciones básicas y condicionales fue fundamental. Esto facilitó la comprensión y la implementación del juego.

Contras

El código inicial en Python y C++ contenía elementos avanzados que no comprendía completamente. Esto hizo que la implementación y depuración fueran más difíciles hasta que con la ayuda de la misma IA logré simplificarlo.

5 Problemas enfrentados y soluciones

El código inicial era complejo por usar decoradores y listas por comprensión, lo que simplifiqué eliminando estos elementos y utilizando solo funciones básicas. Además, modifiqué la implementación para permitir a los jugadores decidir si continuar o detenerse mediante la variable `seguir_jugando`, y reduje las bibliotecas en C++ a las esenciales `<iostream>`, `<cstdlib>`, y `<ctime>`, simplificando así la generación de números aleatorios.

6 Cómo funciona mi código?

En Python, el código usa varias funciones clave como: `es_primo(n)` para verificar números primos, `lanzar_dados(cantidad=2)` para lanzar dados y `jugar_ronda()` para gestionar una ronda del juego. La función `determinar_ganador(jugador1, jugador2)` decide el ganador basándose en números primos o el puntaje más cercano a 23. Finalmente, `juego_prime_jack()` coordina el juego entre dos jugadores y determina el ganador.

Así mismo, En C++, el código usa `generar_numero_aleatorio(min, max)` para generar números aleatorios, `es_primo(num)` para verificar números primos y `jugar_ronda()` para manejar el juego, incluyendo lanzamientos de dados y casos especiales. La función `determinar_ganador(puntaje1, puntaje2)` usa booleanos para comparar si las puntuaciones son primas y decide el ganador. Finalmente, `main()` ejecuta el juego, gestionando turnos y mostrando el resultado.

Link juego c++ [github](#).

Link juego python [github](#).