Solitario de las dos cartas

Tipo `tCarta`

Una carta de la baraja española es una pareja de valores formada por una figura (`AS`, `DOS`, `TRES`, `CUATRO`, `CINCO`, `SEIS`, `SIETE`, `SOTA`, `CABALLO`, `REY`) y un palo (`OROS`, `COPAS`, `ESPADAS`, `BASTOS`).

Solitario de las dos cartas (carta.h)

```
1  #ifndef TCARTA_H
2  #define TCARTA_H

4  enum tpalo {OROS, COPAS, ESPADAS, BASTOS};
5  enum tfig {AS, DOS, TRES, CUATRO, CINCO,
6             SEIS, SIETE, SOTA, CABALLO, REY};

8  class tCarta {
9  public:
10    tCarta(tpalo p = OROS, tfig f = AS) : palo_(p), fig_(f) {};
11    tpalo palo() const { return palo_; };
12    tfig fig() const { return fig_; };
13    friend std::ostream& operator <<(std::ostream& os,
14                                     const tCarta& c);
15 private:
16    tpalo palo_;
17    tfig fig_;
18    static const char* nom_palo[], * nom_fig[];
19 };

21 #endif // TCARTA_H
```

Solitario de las dos cartas (carta.cpp)

```
1  #include <iostream>
2  #include "carta.h"

4  const char* tCarta::nom_palo[] =
5                {"OROS␣␣␣", "COPAS␣␣", "ESPADAS", "BASTOS␣"};
6  const char* tCarta::nom_fig[] =
7                {"AS␣␣␣␣␣", "DOS␣␣␣␣", "TRES␣␣␣", "CUATRO␣",
8                 "CINCO␣␣", "SEIS␣␣␣", "SIETE␣␣",
9                 "SOTA␣␣␣", "CABALLO", "REY␣␣␣␣"};

11 std::ostream& operator <<(std::ostream& os, const tCarta& c)
12 {
13    os << "(" << (c.fig() < 0 ? "-------"
14                              : tCarta::nom_fig[c.fig()])
15       << ",␣" << tCarta::nom_palo[c.palo()] << ")";
16    return os;
17 }
```

Solitario de las dos cartas. Elementos del juego

Baraja

Un vector de 40 cartas.

Mazo y montón de descartes

Ambos elementos del juego los representamos como pilas de cartas, ya que se trata de secuencias en las que las cartas se añaden y quitan por el mismo extremo.

Bases

Tenemos cuatro bases, una por cada palo de la baraja. Según el desarrollo del juego, sólo necesitamos conocer la última figura colocada en cada base. Por tanto, representamos las bases mediante un vector de cuatro figuras. Una base vacía tiene el valor -1.

Solitario de las dos cartas

```
1  #include "carta.h"
2  #include "pilavec.h"

4  using namespace std;

6  bool solitario(tCarta* baraja, tfig* base)
7  {
8     // Elementos del juego
9     Pila<tCarta> Mazo(40),
10                 Monton(40);

12    // Preparar el mazo...
13    for (int i = 0; i < 40; i++) {
14       Mazo.push(baraja[i]);
15    }
16    // ... y las bases vacías.
17    for (int i = 0; i < 4; i++)
18       base[i] = tfig(-1);
```

Solitario de las dos cartas

```
20      // Jugar
21      int colocadas;
22      do {
23          colocadas = 0;
24          while (!Mazo.vacia()) {
25              // Pasar dos cartas (si hay) del mazo al montón de descartes.
26              // ...

28              // Pasar cartas del montón de descartes a sus bases mientras
29              // sea posible.
30              while (!Monton.vacia() && Monton.tope().fig() ==
31                                        base[Monton.tope().palo()] + 1) {
32                  base[Monton.tope().palo()] = Monton.tope().fig();
33                  Monton.pop();
34                  colocadas++;
35              }
36          } // Mazo vacío.
```

Solitario de las dos cartas

```
37          // Reponer el mazo con las cartas del montón de descartes.
38          // ...

40          // Parar si no quedan cartas en el mazo o bien en el último
41          // ciclo no se ha colocado ninguna en su base.
42      } while (!Mazo.vacia() && colocadas > 0);

44      return (Mazo.vacia()); // TRUE (éxito) si el mazo está vacío, o lo
45                             // que es lo mismo, todas las cartas están
46                             // en sus bases.
47  }
```

Solitario de las dos cartas (programa de prueba)

```
1  #include <iostream>
2  #include <algorithm>
3  #include "carta.h"

5  using namespace std;

7  int main()
8  {
9     tCarta baraja[40];
10    tfig base[4];

12    // Generar las 40 cartas de la baraja española.
13    for (int i = 0, j = OROS; j <= BASTOS; j++)
14       for (int k = AS; k <= REY; k++, i++)
15          baraja[i] = tCarta(tpalo(j), tfig(k));
```

Solitario de las dos cartas (programa de prueba)

```
17      srand(time(0));
18      bool victoria = false;
19      int partidas = 0;
20      do {
21         random_shuffle(baraja, baraja+40); // Barajar las cartas
22         victoria = solitario(baraja, base); // Jugar
23         // Mostrar resultado
24         cout << "Partidas␣jugadas␣=␣" << ++partidas << endl;
25         cout << "Últimas␣cartas␣colocadas:\n"
26              << "\t" << tCarta(OROS, base[OROS]) << endl
27              << "\t" << tCarta(COPAS, base[COPAS]) << endl
28              << "\t" << tCarta(ESPADAS, base[ESPADAS]) << endl
29              << "\t" << tCarta(BASTOS, base[BASTOS]) << endl
30              << endl;
31      } while (!victoria); // Parar al ganar una partida.
32      cout << "¡Conseguido!␣Bien␣hecho.\n" << endl;
33  }
```