Fundamentos de Programación

P2 Práctica 2 (Parte 1)

Grado en Ingeniería Informática
Grado en Ingeniería del Software
Grado en Ingeniería de Computadores

Raquel Hervás Ballesteros
Facultad de Informática
Universidad Complutense

Práctica 2

Trucos de magia con cartas

Práctica 2

El objetivo es realizar un simulador de trucos de cartas. Trabajaremos con la baraja francesa, donde los palos son picas, tréboles, diamantes y corazones, y para cada palo tenemos los "números" A, 2, 3,…, 9, 10, J, Q, K.

Se podrán realizar diferentes operaciones sobre mazos de cartas (barajar, cargar de fichero, cortar, repartir,…), y luego realizar varios trucos de magia utilizando las operaciones anteriores.

Práctica 2

Desarrollo incremental

- ✓ Versión 1: manipulación de mazos
  - ✓ Cargar, barajar, añadir mazo, cortar mazo, guardar
- ✓ Versión 2: Repartos de mazos
  - ✓ Separar en negras y rojas
  - ✓ Separar en bajas y altas
  - ✓ Repartir en tres montones
- ✓ Versión 3: Truco de los tres montones
- ✓ Versión 4: Truco de la posada
- ✓ Versión 5: Truco del jugador desconfiado (opcional)

Práctica 2

Entrega:

- ✓ 1 de febrero

Planificación recomendada:

- ✓ 8 de enero: versión 1
- ✓ 16 de enero: versión 2
- ✓ 23 de enero: versión 3
- ✓ 30 de enero: versión 4

¿Dudas de funcionamiento?

- ✓ Demo ejecutable en el campus virtual

Práctica 2

Representación de los datos

Representación de los datos

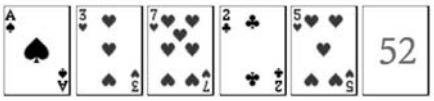
- ✓ Para representar las cartas:
  - ✓ Tipos enumerados para los palos (`tPalo`) y los números (`tNumero`)
    - — Para los números poned nombres representativos:

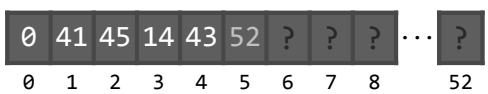
      `as, dos, tres, ..., jack, queen, king`
  - ✓ Tipo para la carta como sinónimo de `int`

    ```
    typedef int tCarta;
    ```
  - ✓ `CARTASPORPALO = 13`. Constante para las cartas por cada palo.

Representación de los datos

- ✓ Los mazos no siempre estarán completos
  - ✓ Necesitamos entonces marcar dónde termina el mazo
  - ✓ Usamos un centinela, un valor especial dentro del mazo que marca el final
- ✓ Mazo de cartas:
  - ✓ `MAXCARTAS = 53`: Tamaño del array (52 cartas + centinela)
  - ✓ Array de `tCarta` de `MAXCARTAS` posiciones
  - ✓ `CENTINELA = 52`. Para señalar el final del mazo





Recorridos y búsquedas en los mazos

- ✓ Recorridos
  - ✓ Como no sabemos cuántas cartas hay en el mazo, no podemos usar un `for`

```
int i = 0;
while (mazo[i] != CENTINELA) {
   ...
   i++
}
```

No vale contar cuántas cartas hay en el mazo y luego usar un `for`. ¡Es muy ineficiente!

Recorridos y búsquedas en los mazos

- ✓ Búsquedas
  - ✓ Esquema típico de búsqueda, pero el final lo marca el centinela, no el tamaño

```
int i = 0;
while (mazo[i] != CENTINELA && !encontrado) {
   ...
   i++
}
```

Práctica 2

Implementación

Subprogramas para dar el número y el palo

- ✓ `tPalo darPalo(tCarta carta)`
  - ✓ El valor del enumerado será el que esté en la posición
    `carta / CARTASPORPALO`
  - ✓ Con un molde se hace la transformación
    `tPalo(carta / CARTASPORPALO)`

- ✓ `tNumero darNumero(tCarta carta)`
  - ✓ El valor del enumerado será el que esté en la posición
    `carta % CARTASPORPALO`
  - ✓ Con un molde se hace la transformación
    `tNumero(carta % CARTASPORPALO)`

Subprograma para mostrar una carta

- ✓ `void escribirCarta(tCarta carta)`

1. Se obtiene el palo de la carta con `darPalo(…)`
   - ✓ Para traducir a un `string` se usa un `switch`
2. Se obtiene el número de la carta con `darNumero(…)`
   - ✓ Para traducir a un número se usa un `switch`
     - — Para el as, jota, rey y reina se usa un texto
     - — Para los demás se puede calcular el número automáticamente

       `int(numero) + 1`
     - — Si por ejemplo la carta tiene como número el valor `dos` de `tNumero`:

       `int(2) = 1 + 1 ⟶ 2`

Subprograma para cargar un mazo de fichero

- ✓ `bool cargarMazo(tMazo mazo)`
- ✓ Cada línea del fichero contiene:
  - ✓ Una letra indicando el palo
  - ✓ Un número indicando el número

```
p 1
d 4
p 12
d 13
t 2
t 4
c 12
x
```

- ✓ Como las cartas se representan como enteros del 0 al 51 hay que hacer una transformación según el palo
  - ✓ Si `palo=='p'`, entonces `tCarta(num-1)`
  - ✓ Si `palo=='t'`, entonces `tCarta(CARTASPORPALO + num-1)`
  - ✓ Si `palo=='d'`, entonces `tCarta(2*CARTASPORPALO + num-1)`
  - ✓ Si `palo=='c'`, entonces `tCarta(3*CARTASPORPALO + num-1)`
- ✓ Al final de la lectura, hay que poner el centinela del array en la siguiente posición libre

Subprograma para barajar un mazo de cartas

- ✓ `void barajarMazo(tMazo mazo)`

1. Se calcula cuántas cartas hay en el mazo
   - ✓ Usando la función `int cuantasEnMazo(const tMazo mazo)`
2. Se calcula el número de intercambios (3 * cartas en el mazo)
3. Mientras no hayamos hecho todos los intercambios
   - ✓ Se eligen dos posiciones aleatorias válidas
   - ✓ Se intercambian

Subprograma para unir dos mazos

- ✓ `bool unirMazos(tMazo mazoOriginal, const tMazo nuevoMazo)`

1. Se calcula cuántas cartas hay en cada mazo
   - ✓ Usando la función `int cuantasEnMazo(const tMazo mazo)`
2. Si la suma es mayor que 52, no caben las cartas en el mazo
   - ✓ Se devolverá `false` al final de la función
3. Si sí caben, a partir de la última posición ocupada, hay que introducir las cartas del mazo nuevo
   - ✓ Se añaden cartas hasta que se llegue al centinela del mazo nuevo
   - ✓ Se coloca el centinela al final del mazo original con las cartas nuevas

Subprograma para partir un mazo en dos

- ✓ `bool partirMazo(tMazo mazoOrigen, int cuantasCoger, tMazo mazoDestino)`

1. Se comprueba si hay suficientes cartas para coger
   - ✓ Usando la función `int cuantasEnMazo(const tMazo mazo)`
2. Si no hay suficientes cartas
   - ✓ Se devolverá `false` al final de la función
3. Si sí hay suficientes
   - ✓ Pasamos al `mazoDestino` las cartas que nos indiquen en `cuantasCorger`
   - ✓ Ponemos el centinela en el `mazoDestino`
   - ✓ Eliminamos del mazo origen las cartas que hemos pasado al `mazoDestino`
     - — Para ello desplazamos hacia la izquierda todas las cartas que quedan tantas posiciones como cartas hayamos llevado al `mazoDestino`

Subprograma para cortar un mazo

- ✓ `void cortarMazo(tMazo mazo, int cuantas)`

1. Se parte el mazo usando la función `partirMazo(...)`
2. Si todo ha salido bien, se unen los mazos con `unirMazos(...)`

Otros subprogramas

- ✓ `void guardarMazo(const tMazo mazo)`
  - ✓ La traducción para escribir en fichero se hace similar a la de `escribirCarta(…)`
- ✓ `void crearMazoVacio(tMazo mazo)`
  - ✓ Se pone el centinela en la posición 0 del array
- ✓ `int cuantasEnMazo(const tMazo mazo)`
  - ✓ Recorrido para contar las posiciones ocupadas del array
- ✓ `void escribirMazo(const tMazo mazo)`
  - ✓ Recorrido. Para cada carta del array se llama a `escribirCarta(…)`

Fundamentos de Programación

Práctica 2
(Parte 1)

Grado en Ingeniería Informática
Grado en Ingeniería del Software
Grado en Ingeniería de Computadores

Raquel Hervás Ballesteros
Facultad de Informática
Universidad Complutense