Programación 2

Examen de teoría (julio 2018)

9 de julio de 2018

Instrucciones

- **Duración: 3 horas**
- El fichero del primer problema debe llamarse `extBasesShips.cc`. Para el segundo problema es necesario entregar cuatro ficheros, llamados `Animal.cc, Animal.h, Shelter.cc, Shelter.h`. Pon tu DNI y tu nombre en un comentario al principio de los ficheros fuente.
- La entrega se realizará como en las prácticas, a través del servidor del DLSI (`http://pracdlsi.dlsi.ua.es`), en el enlace `Programación 2`. Puedes realizar varias entregas, aunque sólo se corregirá la última.
- En la página web de la asignatura `http://www.dlsi.ua.es/asignaturas/p2` tienes disponibles algunos ficheros que te pueden servir de ayuda para resolver los problemas del examen, y el apartado `Reference` de la web `www.cplusplus.com`.

Problemas

1. **(6 puntos)** Un joven informático inexperto de la Alianza Rebelde ha guardado los datos de bases y naves mezclados en un único fichero binario, con registros en los que ha almacenado indistintamente bases o naves. El registro tiene el siguiente formato:

```
struct RegBin {
   char name[15];
   int p;          // people o maxPeople
   int e;          // equipment o maxEquipment
   double x;       // solo bases
   double y;       // solo bases
}
```

   Además, ha mezclado datos reales con datos de prueba, por lo que su sustituto se ha tenido que revisar uno a uno los registros del fichero, y ha obtenido un fichero de texto con los datos reales, como el del siguiente ejemplo:

```
base:"B1" pos:0
  ship :"GR-75 12:3:7"   pos  : 4
 base:  "B2 main base" pos :7
base:"D'Qar"  pos: 15
ship : "Cr. Dantooine" pos:143
base: "Tierfon 2:3" pos: 253
```

   Lamentablemente, el sustituto no puede continuar con su trabajo[1] y debes diseñar un programa que lea el fichero de texto y genere dos ficheros de texto llamados "`bases.txt`" y "`ships.txt`" con los datos de las bases y naves (respectivamente) obtenidos del fichero binario usando como referencia las líneas que aparecen en el fichero de texto que contiene los datos reales.[2] Para ello, se debe leer cada línea del fichero de texto, y hacer lo siguiente:

   - Averiguar si se trata de una base o una nave mirando la palabra antes de "`:`"; si no es "`base`" ni "`ship`"[3] debe emitir un error que indique el contenido y número de línea incorrecto, siendo la primera línea la línea número 1:

[1] Enfadó a un *wookie*, y no quedó mucho de él.

[2] No se debe leer y almacenar en memoria ninguno de los dos ficheros, ni el binario ni el de texto, se debe leer la cantidad mínima de información necesaria cada vez: un registro, o una línea.

[3] Los blancos alrededor de la palabra se deben ignorar. Por otro lado, el sustituto pudo equivocarse en esa palabra, pero el resto de datos se sabe que son correctos.

```
Error line 7: shup:"B7" pos:887
```

- Extraer el nombre entre comillas, para compararlo posteriormente con el del registro binario.
- Obtener la posición en el fichero binario del registro de la base/nave, que es el valor que aparece después de "`pos:`"[4]
- Obtener el registro binario correspondiente a esa posición, teniendo en cuenta que:
  (a) El primer registro está en la posición 0, como se puede ver en el ejemplo.
  (b) No habrá posiciones incorrectas, es decir, en la posición indicada con `pos` siempre habrá un registro. Y siempre aparecerán ordenadas de menor a mayor.[5]

  Una vez se ha leído el registro, se debe comprobar que el nombre extraído del fichero de texto coincide con el nombre que aparece en el registro binario; si no coincide, se debe emitir un error como el del siguiente ejemplo:

  ```
  Error line 25: name "main base" does not match with binary file (name="B2 main base")
  ```

  donde "`main base`" es el nombre en el fichero de texto, y "`B2 main base`" el del fichero binario.
- Si hay algún error en una línea se emitirá el mensaje de error correspondiente y se ignorará dicha línea, y se seguirá leyendo la siguiente.
- Si no hay errores, se debe escribir la base o nave en el fichero correspondiente. El fichero de bases tendrá el mismo formato que el de la práctica 2:

  ```
  "B1",250:2600,(-0.126,1.23)
  ```

  donde "`B1`" es el nombre, `250` la cantidad de personas, `2500` la cantidad de equipamiento y `(-0.126,1.23)` la coordenada.
  El fichero de naves tendrá un formato muy similar al de la práctica 2:

  ```
  (350,5000) "Cr. Dantooine"
  ```

  donde `350` es la cantidad máxima de personas, `5000` la cantidad máxima de equipamiento y "`Cr. Dantooine`" el nombre.

El programa debe llamarse "`extBasesShips.cc`", y debe invocarse con dos argumentos:[6]

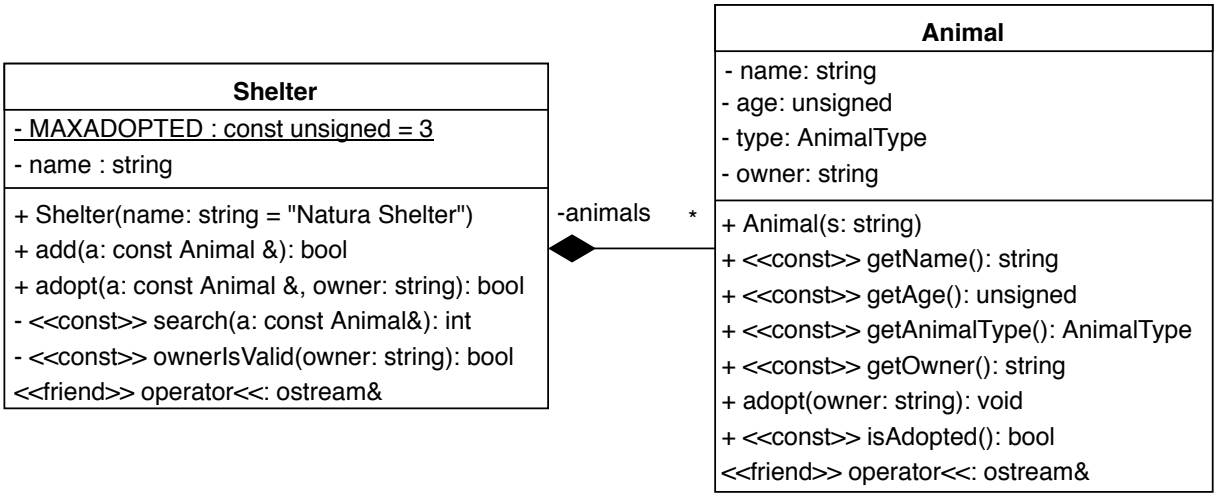
```
./extBasesShips binaryFile.bin correctData.txt
```

donde el primer argumento es el nombre del fichero binario y el segundo el del fichero de texto que hay que leer. Si no hay exactamente dos argumentos se debe emitir un mensaje de error adecuado, y también si no es posible abrir alguno de los ficheros para lectura o escritura.

[4] Para convertir una cadena a número entero puedes utilizar `atoi` de la librería `cstdlib`.
[5] Aunque evidentemente no se debería leer todo el fichero binario, solamente aquellos registros que se indiquen en el fichero de texto.
[6] En la web de la asignatura tienes unos ficheros de ejemplo.

2. **(4 puntos)**

Queremos hacer un programa para gestionar la adopción de animales en una protectora (en inglés, *Shelter*).



Un animal tendrá un nombre, una edad, es de un tipo `AnimalType` y puede tener opcionalmente un adoptante (`owner`). En el fichero `Animal.h` debemos declarar los tipos de animales: `enum AnimalType {Cat, Dog, Jerbo};`

El constructor de la clase `Animal` recibirá un `string` con el nombre, el tipo y la edad del animal separados por comas, por ejemplo: `"Bobby,Dog,3"`. En caso de que el tipo no esté entre los permitidos, este constructor deberá lanzar una excepción devolviendo la cadena con el tipo incorrecto. El método `adopt` debe asignar el adoptante (`owner`) recibido al atributo privado. El método `isAdopted` devolverá `true` si `owner` no es una cadena vacía, y `false` en caso contrario. El operador salida debe imprimir el nombre, la edad y su adoptante (sólo si lo tiene). Ejemplo: `"Bobby, age=3, owner=Juan"`.

La clase `Shelter` tiene un constructor que recibe opcionalmente el nombre de la protectora. El método `add` debe añadir un animal a la protectora, pero sólo si este no estaba ya añadido. Si el animal se puede añadir el método debe devolver `true`, y si no `false`. Para buscarlo, debe usarse el método `search`, que devuelve la posición de un animal en el vector si este existe (es decir, si ya hay alguno con el mismo nombre y tipo), o `-1` si no lo encuentra.

El metodo `adopt` de `Shelter` busca el animal recibido por parámetro en el vector de animales. Si se encuentra y se puede adoptar, el método asigna un propietario al animal del vector, devolviendo `true`. Si el animal no está o no se puede adoptar, el método devuelve `false`. Un animal se puede adoptar sólo si no está ya adoptado y su propietario puede adoptarlo (para saber esto debe usarse el método `ownerIsValid`). Si el animal no puede adoptarse, el método mostrará el mensaje `<nombre> cannot be adopted`, como puede verse en el ejemplo de ejecución.

El método `ownerIsValid` debe comprobar si el adoptante puede adoptar, devolviendo `true` si es posible o `false` si ya tiene `MAXADOPTED` animales.

Finalmente, el operador salida mostrará el mensaje `--- Adopted ---` con el listado de todos los animales adoptados, y después `--- Not adopted ---` con los no adoptados, como puede verse en el ejemplo de ejecución.

Dado el siguiente `main.cc`,[7] que puedes compilar con `g++ Animal.cc Shelter.cc main.cc -o ej2`:

```
#include "Shelter.h"

int main()
{
  try {
    Animal a1("Bobby,Dog,3");
    Animal a2("Jerry Harry,Jerbo,2");
    Animal a3("Thanos,Dog,5");

    Shelter s;
    s.add(a1);
    s.add(a2);
    s.add(a3);

    s.adopt(a1, "Juan");
    s.adopt(a1, "Juan"); // No se permite (ya adoptado)
    s.adopt(a2, "Pepe");

    cout << s;
    Animal a5("Jerry,Mouse,10");
  }
  catch(string message) {
    cout << "Animal type " << message
         << " not supported" << endl;
  }
}
```

Con este `main`, el programa debe mostrar:

```
Bobby cannot be adopted
---- Adopted -----
Bobby, age=3, owner=Juan
Jerry Harry, age=2, owner=Pepe
---- Not adopted -----
Thanos, age=5
Animal type Mouse not supported
```

[7] Lo tienes en la web de la asignatura.