GRUPO 5

SERGIO REVUELTAS ESTRADA, MIGUEL VICENTE LINARES Y ANTONIO VICENTE MARTÍN

PRACTICA 1 - OPERACIONES BÁSICAS DE E/S EN UN FICHERO DE REGISTROS – ZOO DATABASE

• Diseño y tipos de datos implementados. Justificación del diseño elegido.

PUBLIC	PRIVATE

class Animal: Clase que almacena todas las propieades de un animal.

class Animal

```
char name[50];
bool hair;
bool feathers;
bool eggs;
bool milk;
bool airborne;
bool aquatic;
bool predator;
bool toothed;
bool backbone;
bool breathes;
bool venomous;
bool fins;
bool tail;
bool domestic;
bool catsize;
int legs;
int type;
```

```
<Constructor-Destructor>
<Getters and Setters>
bool operator==(Animal&);
friend ostream& operator<<(ostream&, Animal);
```

class Registro: Con esta clase gestionamos la E/S de animales, así como la lista enlazada de registros eliminados.

class Registro

```
bool valido;
long direccion;
Animal animal;
```

```
<Constructor-Destructor>
<Getters and Setters>
```

class Cabecera: Clase con los datos correspondientes a la cabecera del fichero binario

class Cabecera
long nRegistros; long nEliminados; long tamRegistro; long primerHueco;
<Constructor-Destructor> <Getters and Setters>

class EntradaSalida: Gestiona las operaciones de E/S

class EntradaSalida
list<Animal*> animals;

void leerTexto();
void leerBinario();
void leerRegistro(int);
void escribir();
void mostrar();
void insertar(Animal*);
bool eliminar(long);
void vaciar();
long buscar(string);

Se ha elegido este tipo de diseño, para conseguir una mayor comodidad delegando la gestion de los registros, y sus metadatos a las clases descritas anteriormente.

CrearRLF: main

menu(EntradaSalida&) Funciona de interfaz de usuario. Contiene un switch que ejecuta una determinada función, dependiendo del carácter de entrada.

restaurarArchivo(EntradaSalida&) En caso de necesitar recuperar el archivo binario por defecto, esta función se encarga de sobreescribir los datos leyendo el archivo de texto.

• Descripción de los métodos más importantes, especialmente los que implementen las operaciones básicas de inserción, eliminación, etc.

leerTexto() Lee el archivo de texto zoo-data.txt y lo almacena en memoria en una lista de punteros a Animales

leerBinario() Lee los RLFs del archivo binario zoo-data.dat y los almacena en memoria en una list<Animal*>

leerRegistro(**int)** Lee un único registro del archivo. El primer registro es el 0.

escribir() Escribe la lista de animales en RLFs en el archivo binario zoo-data.dat y vacía la memoria.

mostrar() Muestra la lista de animales en memoria.

insertar(Animal*) Inserta un animal en el primer hueco libre del archivo.

NOTA: fstream archivoSalida("zoo-data.dat", **fstream::*out*|fstream::*in*** | fstream::*binary*);

bool eliminar(**long)** Elimina un RLF dado su número de registro, y devuelve true si se ha eliminado con éxito.

vaciar() Vacia la lista de punteros a animales, eliminando la memoria reservada.

long buscar(string) Dado el nombre(clave) de un animal, devuelve el numero de registro donde se encuentra dicho animal, o -1 sino se encuentra en disco.

• Cuadro con la planificación y desarrollo de tareas que ha realizado el grupo.

-Durante la 1ª semana de trabajo, se hizo un boceto inicial en grupo decidiendo:

- El numero y diseño de clases a utilizar (Animal, Registro, Cabecera…)

- La estructura de datos a usar en EntradaSalida (list), con objetos dinámicos o estáticos…

-Numero de campos con metadata en cabecera y RLFs (validez, nEliminados, …)

-En las semanas posteriores, mientras uno se encargaba de la implementación básica de las clases, otro se encargaba de la algoritmia de lectura-escritura, y un tercero de depurar el código y comentar los métodos (todo ello vía svn).

-Finalmente, se implementaron los métodos de lectura secuencial, inserción, eliminación y búsqueda de manera conjunta, además de un pequeño menú en la clase principal para trabajar con la aplicación.

• Principales dificultades que se han encontrado: falta de base teórica, problemas de programación, etc.

-Para efectuar una operación de escritura en el fichero binario no vacío, en una determinada posición, hay que establecer un flujo de datos, de entrada y salida a la vez.

-El tamaño de una clase (sizeof), no coincide con la suma del tamaño de sus propiedades.

-Para serializar una clase, sus propiedades deben ser estáticas, en caso contrario, se serializará el valor del puntero. Por esta razón, hemos prescindido del uso de la clase string como propiedad, aunque sí la hemos usado como variable temporal en la lectura de fichero de texto.

-Como nuestro conjunto de datos, carece de una clave explícita, al hacer una búsqueda por nombre, devolverá la primera ocurrencia de dicha búsqueda.

• Material adicional utilizado en el desarrollo de la práctica.

www.cplusplus.com

www.cppreference.com

SubVersion,

Google Code