TDA ABSTRACTO

1

Por Antonio Miguel Morillo Chica
williams@correo.ugr.es

Índice general

Capítulo 1

Índice de clases

1.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 2

Indice de archivos

2.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

Capítulo 3

Documentación de las clases

3.1. Referencia de la Clase Cronologia

Métodos públicos

- Cronologia ()

  Constructor sin parámetros. Constructor sin parámetros. Crea una creonología vacía.
- Cronologia (const int filas)

  Constructor con parámetros.
- ∼Cronologia ()

  Destructor de la clase. Destructor de la clase. Elimina los objetos de tipo cronología.
- void crear (int filas)

  Crea una creonología de un tamño determinado.
- void destruir ()

  Destruye una cronología Destruye una cronología. Se usa dentro de la clase no sobre un objeto implicito.
- vector< string > GetEventos (int n)

  Consulta una lista de eventos de un año determinado.
- int getFilas ()

  Consultor de filas.

Amigas

- istream & **operator**>> (istream &entrada, Cronologia &crono)

3.1.1. Descripción detallada

Definición en la línea 18 del archivo cronologia.h.

3.1.2. Documentación del constructor y destructor

3.1.2.1. Cronologia()

```
Cronologia::Cronologia (
            const int filas )
```

Constructor con parámetros.

Parámetros

filas	cantidas de años distintos Constructor con parámetros. Crea una Cronologia con el tamaño de las filas

Definición en la línea 14 del archivo cronologia.cpp.

```
14                                      {
15   this->filas = filas;
16   crear(filas);
17 }
```

3.1.3. Documentación de las funciones miembro

3.1.3.1. crear()

```
void Cronologia::crear (
            int filas )
```

Crea una creonología de un tamño determinado.

Parámetros

filas	numero de años distintos Crea una creonología de un tamño determinado. Para ser usado dentro de la clase.

Definición en la línea 27 del archivo cronologia.cpp.

```
27                                {
28   destruir();
29   this->filas = filas;
30   this->lista_eventos = new ListaEventos[filas];
31 }
```

3.1.3.2. GetEventos()

```
vector< string > Cronologia::GetEventos (
            int n )
```

Consulta una lista de eventos de un año determinado.

Parámetros

n	año que se quiere consultar.

Devuelve

Devuelve un vector con cada uno de los eventos de ese año.

Definición en la línea 60 del archivo cronologia.cpp.

```
60                                          {
61   std::vector<string> v;
62   ListaEventos anioLista;
63
64   for (int i = 0; i < this->filas; i++) {
65     if (this->lista_eventos[i].getFecha() == n) {
66       v = this->lista_eventos[i].getEventos();
67     }
68   }
69   return v;
70 }
```

3.1.3.3. getFilas()

```
int Cronologia::getFilas ( )
```

Consultor de filas.

Devuelve

Devuelve la cantidad de años distintos dentro de una cronología determinada.

Definición en la línea 41 del archivo cronologia.cpp.

```
41                          {
42   return this->filas;
43 }
```

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- include/cronologia.h
- src/cronologia.cpp

3.2. Referencia de la Estructura EventoHistorico

Atributos públicos

- EventoHistorico * siguiente
- string evento

3.2.1. Descripción detallada

Definición en la línea 18 del archivo listaeventos.h.

3.2.2. Documentación de los datos miembro

3.2.2.1. evento

```
string EventoHistorico::evento
```

Nombre del evento

Definición en la línea 20 del archivo listaeventos.h.

3.2.2.2. siguiente

```
EventoHistorico* EventoHistorico::siguiente
```

Puntero al siguiente evento.

Definición en la línea 19 del archivo listaeventos.h.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- include/listaeventos.h

3.3. Referencia de la Clase ListaEventos

Métodos públicos

- ListaEventos ()
  Constructor por defecto. Contructor por defecto. Crea una Lista de eventos vacía.
- ListaEventos (string cadena)
  Constructor por parametros.
- ListaEventos (const ListaEventos &otra)
  Constructor de copia.
- ∼ListaEventos ()
  Destructor Destructor: destruye automaticamente la basura.
- void destruir ()
  Destruye el objeto Destruye el objeto implicito.
- ListaEventos & copiarLista (const ListaEventos &otra)
  Copia todos los eventos.
- ListaEventos & operator= (const ListaEventos &otra)
  Sobrecarga del operator=.
- ListaEventos & operator+ (const string nueva_lista)
  Sobrecarga del operador+.
- void insertar (string evento)
  Inserta un nuevo año a la cronología.
- void insertarLista (string list)
  Inserta una lista completa de eventos.
- string getEvento (int i)
  Muestra el titulo de un evento.
- vector< string > getEventos ()
  Consulta los eventos de toda una fecha.
- int getColumnas ()
  Cantidad de años dentro de la cronología.
- const int getFecha ()
  Consulta la fecha de la lista de eventos.

3.3.1. Descripción detallada

Definición en la línea 24 del archivo listaeventos.h.

3.3.2. Documentación del constructor y destructor

3.3.2.1. ListaEventos() [1/2]

```
ListaEventos::ListaEventos (
            string cadena )
```

Constructor por parametros.

Parámetros

cadena	sucesión de eventoshistoricos separados por # Constructor por parametros. Construye una lista de eventos de tamaño indefinido y escribe en la primera fila los eventos historicos que tenga el string cadena.

Definición en la línea 13 del archivo listaeventos.cpp.

```
13                                      {
14     insertar(cadena);
15 }
```

3.3.2.2. ListaEventos() [2/2]

```
ListaEventos::ListaEventos (
              const ListaEventos & otra )
```

Constructor de copia.

Parámetros

otra	lista que queremos copiar

Definición en la línea 19 del archivo listaeventos.cpp.

```
19                                                  {
20     copiarLista(otra);
21 }
```

3.3.3. Documentación de las funciones miembro

3.3.3.1. copiarLista()

```
ListaEventos & ListaEventos::copiarLista (
              const ListaEventos & otra )
```

Copia todos los eventos.

Parámetros

otra	Lista de eventos por la que queremos sustituirla Copia todos los eventos de una lista al objeto implicito.

Definición en la línea 45 del archivo listaeventos.cpp.

```
45                                                          {
46   if(otra.cabecera != 0){
47     destruir();
48     EventoHistorico *aux = otra.cabecera;
49
50     this->fecha = otra.fecha;
```

```
51     this->columnas = otra.columnas;
52
53     while(aux->siguiente != 0){
54       insertar(aux->evento);
55       aux = aux -> siguiente;
56     }
57
58     delete aux;
59   }
60
61   return *this;
62 }
```

3.3.3.2. getColumnas()

```
int ListaEventos::getColumnas ( )
```

Cantidad de años dentro de la cronología.

Devuelve

Devuelve el tamaño de la lista enlazada principal (columnas).

Definición en la línea 151 del archivo listaeventos.cpp.

```
151                             {
152   return this->columnas;
153 }
```

3.3.3.3. getEvento()

```
string ListaEventos::getEvento (
              int i )
```

Muestra el titulo de un evento.

Parámetros

i	celda que contiene el evento

Devuelve

Devuelve el string que contiene.

Definición en la línea 123 del archivo listaeventos.cpp.

```
123                                 {
124   string cadena = "";
125   EventoHistorico *ptr = cabecera;
126   int cnt = 0;
127   if(ptr != 0){
128     while (cnt != i) {
129       ptr = ptr -> siguiente;
130       cnt++;
131     }
132     cadena = ptr-> evento;
133   }
134   return cadena;
135 }
```

3.3.3.4. getEventos()

```
vector< string > ListaEventos::getEventos ( )
```

Consulta los eventos de toda una fecha.

Devuelve

devuelve la lista de eventos en un vector de strings.

Definición en la línea 139 del archivo listaeventos.cpp.

```
139                                     {
140   std::vector<string> v;
141   string evento;
142   for (int i = 0; i < this->columnas; i++) {
143     evento = getEvento(i);
144     v.push_back(evento);
145   }
146   return v;
147 }
```

3.3.3.5. getFecha()

```
const int ListaEventos::getFecha ( )
```

Consulta la fecha de la lista de eventos.

Devuelve

Devuelve la fecha de los eventos de la lista

Definición en la línea 157 del archivo listaeventos.cpp.

```
157                                {
158   return this->fecha;
159 }
```

3.3.3.6. insertar()

```
void ListaEventos::insertar (
             string evento )
```

Inserta un nuevo año a la cronología.

Parámetros

evento	que se quiere insertar

Devuelve

Añade al objeto implícito un año nuevo en la posición correspondiente.

Precondición

el año ya ha sido definido.

Definición en la línea 77 del archivo listaeventos.cpp.

```
77                                      {
78
79   EventoHistorico *evento = new EventoHistorico();
80
81   evento -> evento = event;
82   evento -> siguiente = 0;
83
84   if(cabecera== 0)
85     cabecera = evento;
86   else{
87     EventoHistorico *ptr;
88     ptr = cabecera;
89     while (ptr->siguiente != 0) ptr = ptr->siguiente;
90     ptr->siguiente = evento;
91   }
92
93   this->columnas++;
94 }
```

3.3.3.7. insertarLista()

```
void ListaEventos::insertarLista (
              string list )
```

Inserta una lista completa de eventos.

Parámetros

list	lista de eventos que se quieren insertar

Devuelve

Añade al objeto implícito una los eventos correspondientes. el año ha de ser fijado

Definición en la línea 98 del archivo listaeventos.cpp.

```
98                                          {
99
100   string evento;
101   evento.resize(list.size());
102   int anio = stoi(list.substr(0,4));
103   this->fecha = anio;
104
105   int cnt = 0;
106   for (int i = 5; i < (int)list.size(); i++) {
107     if(list[i] != '#'){
108       evento.push_back(list[i]);
109       cnt++;
110     }else{
111       insertar(evento);
112       evento.clear();
113       cnt = 0;
```

```
114     }
115   }
116
117   // En caso de no insertar el último lo inserto
118   insertar(evento);
119 }
```

3.3.3.8. operator+()

```
ListaEventos& ListaEventos::operator+ (
              const string nueva_lista )
```

Sobrecarga del operador+.

Parámetros

nueva_lsita	sucesión de eventoshistoricos separados por # Sobrecarga del operador+ suma dos listas en una sola construyendo una sola lista donde la segunda es una concatenación de la siguiente.

3.3.3.9. operator=()

```
ListaEventos & ListaEventos::operator= (
              const ListaEventos & otra )
```

Sobrecarga del operator=.

Parámetros

otra	Lista de eventos por la que queremos sustituirlo Sobrecarga del operator= iguala la primera lista a la segunda. Esta sí se puede igualar inmediatamente a otra.

Definición en la línea 66 del archivo listaeventos.cpp.

```
66                                                    {
67   if(otra.cabecera != 0){
68     destruir();
69     copiarLista(otra);
70   }
71
72   return *this;
73 }
```

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- include/listaeventos.h
- src/listaeventos.cpp

Capítulo 4

Documentación de archivos

4.1. Referencia del Archivo include/cronologia.h

Clase para la estructura de datos de un vector dinámico de listas enlazadas. Permite el manejo de listas enlazadas dentro de un vector dinámico.

```
#include "listaeventos.h"
#include <istream>
#include <fstream>
```

Clases

- class Cronologia

4.1.1. Descripción detallada

Clase para la estructura de datos de un vector dinámico de listas enlazadas. Permite el manejo de listas enlazadas dentro de un vector dinámico.

Autor

Antonio Miguel Morillo Chica

Fecha

10 Octubre de 2016

4.2. Referencia del Archivo include/listaeventos.h

Clase para la estructura de datos de lista de strings. Permite el manejo de cadenas (strings) en una lista enlazada.

```
#include <string>
#include <vector>
#include <iostream>
```

Clases

- struct EventoHistorico
- class ListaEventos

4.2.1. Descripción detallada

Clase para la estructura de datos de lista de strings. Permite el manejo de cadenas (strings) en una lista enlazada.

Autor

Antonio Miguel Morillo Chica

Fecha

8 Junio de 2016

Índice alfabético