

TDA Arbol.

Rodríguez Tabares Juan

Ingeniería en computación
Centro Universitario de Ciencias Exactas e Ingenierías
Universidad de Guadalajara

Abstract

Este trabajo comprende a la elaboración de la actividad numero 5 de la asignatura Estructura de datos II, de la carrera de Ingeniería en computación llevada acabo en el ciclo escolar 2020B.

1. Introducción

En este documento veremos como se implemento la practica propuesta la cual consiste en implementar 2 nuevas funciones al codigo que se nos fue entregado, dichas funciones deberian hacer lo siguiente:

La primer función debiera cargar todos los datos de un archivo.

La segunda función debiera cargar un dato de un archivo el dato debiera ser indicado por el usuario.

2. Desarrollo

En este punto veremos como se adapto el programa a los requerimientos pedidos paso a paso para llevar acabo su correcto funcionamiento.

NOTA: Se omitieron los archivos *Producto.h* y *WritePerson.cpp* ya que los cambios realizados en estos fueron nulos o irrelevantes para lo pedido en la practica.

2.1. ArbolBinBus.h

Las modificaciones de esta parte fueron casi nulas solo por una funcion que fue eliminada de nombre *WriteTreeFile*, ya que no fue requerida.

2.2. main.cpp

Este archivo de el trabajo fue la que mayores cambios tuvieron ya que aqui se añadieron las 2 funciones requeridas junto con algunas mejoras al programa mencionadas a continuación:

Linea 6-10: Fueron declaradas 2 nuevas variables una de estado estatico llamada *recorrido* y otra de tipo booleano llamada *band* declarada por defecto en falsa.

Linea 27 y 28: Fueron añadidas 2 nuevas funciones al menu.

Linea 103-110: Fue añadida una validacion la cual indica si la carga de datos del archivo ya fue realizada o no, si ya fue hecha esta opcion quedara inhabilitada por la variable bandera.

Linea 113-119: Fue añadida la funcion de añadir producto de el archivo por nombre.

2.3 Código en C++

Explicación de la función 1: Añadir todo lo escrito en el archivo

Línea 132-181: Iniciamos la declaración de un buffer que guardara todo lo contenido en el archivo, la variable *lectura* guardara el número de espacios que necesita el producto. Las siguientes tres variables guardaran los datos del producto. La variable *i* será la encargada de hacer el recorrido durante el buffer, esta se iguala a recorrido ya que esta será la que guarde el punto donde se quedó la anterior iteración. La siguiente variable comprobará el número de delimitadores recorridos y así saber qué dato se está leyendo. Las últimas 3 variables llevarán a cabo la cuenta de *id*, *nombre*, *precio*.

Línea 144-150: Abrimos el archivo, si no se abre lanza error, si se abre se leen los datos y se guardan en el buffer.

Línea 151-155: Este while se encargará de guardar en *lectura* los espacios que requiere el producto.

Línea 156: Se aumenta *i* en 1 para que este este colocado en lo que sería el primer dato (id del producto).

Línea 157-171: Bucle while el cual se encargará de recorrer el buffer para colocar los datos en sus debidos arreglos.

Línea 158: If que lleva la cuenta de delimitadores para llevar el control de datos.

Línea 159-169: If para saber que estamos leyendo un dato de nuestro interés.

Línea 160-162: Bloque de ifs el cual se encarga de meter los datos de el buffer en el arreglo que pertenecen.

Línea 163-167: Asignación a cada uno de los arreglos un fin y a su vez un fin al while.

Línea 170: Incremento de *i* para que se vaya recorriendo el buffer.

Línea 172: asignación de *i* a la variable global recorrido (remarcando que esta guarda el punto en el que se quedó *i* o sea en el inicio de el siguiente producto).

173-174: Transformación de char a su debido tipo de dato de los arreglos *id*, *precio*

Línea 175: Creación de un producto con los datos obtenidos del recorrido.

Línea 176: Inserción de el producto al árbol.

Línea 177-180: Bloque if para saber si hemos llegado al final de el buffer, si este no se cumple cerrará el archivo y abrirá paso a la lectura de los siguientes productos con la función recursiva.

Explicación de la función 2: Añadir cierto producto de el archivo

Línea 186-237: Para no alargar más de lo debido el archivo dire que es exactamente lo mismo que la primera función pero con la diferencia de que esta contiene una validación para saber si el producto se encontró o no.

Línea 227-236: Bloque if que comprueba si la cadena que el usuario pasó para buscar es igual al nombre del producto encontrada en el archivo caso contrario seguirá de forma recursiva el programa.

2.3. Código en C++

ArbolBinBus.h: Aquí se pueden observar los cambios realizados que se explicaron en la parte 2.1 del desarrollo.

```
1 #ifndef ARBOLBINBUS_H_INCLUDED
2 #define ARBOLBINBUS_H_INCLUDED
3 #include <iostream>
4 #include <fstream>
5 #include <string.h>
6 #include "Producto.h"
7 template <class T>
8 class ArbolBinBus;
9
```

2.3 Código en C++

```
10 template <class T>
11 class NodoArbol
12 {
13     private:
14         T Info;
15         NodoArbol<T> *HijoIzq;
16         NodoArbol<T> *HijoDer;
17     public:
18         NodoArbol();
19         T RegresaInfo();
20         void ActualizaInfo(T);
21         friend class ArbolBinBus<T>;
22 };
23
24 template <class T>
25 NodoArbol<T>::NodoArbol()
26 {
27     HijoIzq= NULL;
28     HijoDer= NULL;
29 }
30
31 template <class T>
32 T NodoArbol<T>::RegresaInfo()
33 {
34     return Info ;
35 }
36
37 template <class T>
38 void NodoArbol<T>::ActualizaInfo(T Dato)
39 {
40     Info= Dato ;
41 }
42
43 template <class T>
44 class ArbolBinBus
45 {
46     private:
47         NodoArbol<T> *Raiz;
48         char* SavePreorden ( NodoArbol<T> *, char *);
49     public:
50         ArbolBinBus ();
51         NodoArbol<T> *RegresaRaiz();
52         void Preorden (NodoArbol<T> *);
53         int WriteFile(std::ostream &);
54         void WritePreordenFile(std::ostream &, NodoArbol<T> * );
55         void Inorden (NodoArbol<T> *);
56         void Postorden (NodoArbol<T> *);
57         NodoArbol<T> * Busqueda (NodoArbol<T> *, T);
58         void InsertaNodoSinRep (NodoArbol<T> *, T);
59         void EliminaNodo (NodoArbol<T> *, T);
60 };
61
62 template <class T>
63 ArbolBinBus<T>::ArbolBinBus()
64 {
65     Raiz= NULL;
66 }
67
68 template <class T>
69 NodoArbol<T> *ArbolBinBus<T>::RegresaRaiz()
70 {
71     return Raiz;
72 }
73
74 template <class T>
75 void ArbolBinBus<T>::Preorden (NodoArbol<T> *Apunt)
76 {
77     if (Apunt)
78     {
79         std::cout<< Apunt->Info << " ";
80         Preorden (Apunt->HijoIzq);
```

2.3 Código en C++

```
81     Preorden(Apunt->HijoDer);
82 }
83 }
84
85 template <class T>
86 void ArbolBinBus<T>::WritePreordenFile(std::ostream &stream, NodoArbol<T> *Apunt )
87 {
88     if (Apunt)
89     {
90         Apunt->Info.WriteProductFile(stream);
91         WritePreordenFile(stream, Apunt->HijoIzq);
92         WritePreordenFile(stream, Apunt->HijoDer);
93     }
94 }
95
96 template <class T>
97 char* ArbolBinBus<T>::SavePreorden(NodoArbol<T> *Apunt, char * buffer)
98 {
99     if (Apunt)
100     {
101         std::stringstream s;
102         char clenght[100]="", ProducInf[150]="";
103         s<<strlen(Apunt->Info.getArgumString(ProducInf)); s>>clenght;
104         strcat(buffer, clenght);
105         strcat(buffer, "#");
106         strcat(buffer, Apunt->Info.getArgumString(ProducInf));
107         SavePreorden(Apunt->HijoIzq, buffer);
108         SavePreorden(Apunt->HijoDer, buffer);
109     }
110     return buffer;
111 }
112
113 template <class T>
114 int ArbolBinBus<T>::WriteFile(std::ostream &stream)
115 {
116     char buffer[2000]="";
117     strcpy(buffer, SavePreorden(Raiz, buffer));
118     stream.write(buffer, strlen(buffer));
119     return 0;
120 }
121
122 template <class T>
123 void ArbolBinBus<T>::Inorden (NodoArbol<T> *Apunt)
124 {
125     if (Apunt)
126     {
127         Inorden(Apunt->HijoIzq);
128         std::cout<< Apunt->Info << " ";
129         Inorden(Apunt->HijoDer);
130     }
131 }
132
133 template <class T>
134 void ArbolBinBus<T>::Postorden (NodoArbol<T> *Apunt)
135 {
136     if (Apunt)
137     {
138         Postorden(Apunt->HijoIzq);
139         Postorden(Apunt->HijoDer);
140         std::cout<< Apunt->Info << " ";
141     }
142 }
143
144 template <class T>
145 NodoArbol<T> * ArbolBinBus<T>::Busqueda (NodoArbol<T> *Apunt, T Dato)
146 {
147     if (Apunt)
148     {
149         if (Dato < Apunt->Info)
150             return Busqueda(Apunt->HijoIzq, Dato);
151         else
152             if (Dato > Apunt->Info)
```

2.3 Código en C++

```
152         return Busqueda(Apunt->HijoDer, Dato);
153     else
154         return Apunt;
155 else
156     return NULL;
157 }
158
159
160 template <class T>
161 void ArbolBinBus<T>::InsertaNodoSinRep(NodoArbol<T> *Apunt, T Dato)
162 {
163     NodoArbol<T> *ApAux;
164     if (Apunt)
165     {
166         if (Dato < Apunt->Info)
167         {
168             InsertaNodoSinRep(Apunt->HijoIzq, Dato);
169             Apunt->HijoIzq= Raiz;
170         }
171         else
172         if (Dato > Apunt->Info)
173         {
174             InsertaNodoSinRep(Apunt->HijoDer, Dato);
175             Apunt->HijoDer= Raiz;
176         }
177         Raiz= Apunt;
178     }
179     else
180     {
181         ApAux= new NodoArbol<T>();
182         ApAux->Info= Dato;
183         Raiz= ApAux;
184     }
185 }
186
187 template <class T>
188 void ArbolBinBus<T>::EliminaNodo(NodoArbol<T> *Apunt, T Dato)
189 {
190     if (Apunt)
191     {
192         if (Dato < Apunt->Info)
193         {
194             EliminaNodo(Apunt->HijoIzq, Dato);
195             Apunt->HijoIzq= Raiz;
196         }
197         else
198         if (Dato > Apunt->Info)
199         {
200             EliminaNodo(Apunt->HijoDer, Dato);
201             Apunt->HijoDer= Raiz;
202         }
203         else
204         {
205             NodoArbol<T> *ApAux1,*ApAux2,*ApAux3;
206             ApAux3= Apunt;
207             if (!ApAux3->HijoDer)
208                 if (!ApAux3->HijoIzq)
209                     Apunt= NULL;
210                 else
211                     Apunt= ApAux3->HijoIzq;
212             else
213                 if (!ApAux3->HijoIzq)
214                     Apunt= ApAux3->HijoDer;
215                 else
216                 {
217                     ApAux1= ApAux3->HijoIzq;
218                     ApAux2= ApAux3;
219                     while (ApAux1->HijoDer)
220                     {
221                         ApAux2= ApAux1;
222                         ApAux1= ApAux1->HijoDer;
```

2.3 Código en C++

```

223     }
224     ApAux3->Info= ApAux1->Info;
225     if (ApAux3 == ApAux2)
226         ApAux3->HijoIzq= NULL;
227     else
228         if (!ApAux1->HijoIzq)
229             ApAux2->HijoDer= NULL;
230         else
231             ApAux2->HijoDer= ApAux1->HijoIzq;
232     ApAux3= ApAux1;
233     }
234     delete(ApAux3);
235 }
236 Raiz= Apunt;
237 }
238 }
239
240 #endif // ARBOLBINBUS_H_INCLUDED

```

main.cpp: Es donde se mostrara la funcion principal del programa recuerde que la explicacion de las modificaciones esta en la parte 2.2 del desarrollo.

```

1  #include <iostream>
2  #include "ArbolBinBus.h"
3  #include "Producto.h"
4  // #include "WriteTreeFile.cpp"
5  using namespace std;
6  int static recorrido = 0;
7  bool band = false;
8
9  void getString(int);
10 void getStringName(int, char*);
11
12 ArbolBinBus<Producto> Inventario;
13 NodoArbol<Producto> *Ap1, *Ap2;
14
15 int Menu()
16 {
17     int Opcion;
18     system("cls");
19     do {
20         cout<<"\n\n\tOpciones de trabajo:\n";
21         cout<<"\t1.Ingresar nuevo producto.\n";
22         cout<<"\t2.Dar de baja un producto.\n";
23         cout<<"\t3.Reporte de todos los productos ordenados por clave.\n";
24         cout<<"\t4.Buscar un producto por clave.\n";
25         cout<<"\t5.Escribir en disco forma 1.\n";
26         cout<<"\t6.Escribir en disco forma 2.\n";
27         cout<<"\t7.Traer todos los datos del archivo.\n";
28         cout<<"\t8.Traer un dato especifico del archivo.\n";
29         cout<<"\t9.Terminar el proceso" << "\n\n";
30         cout<<"\tIngrese opción seleccionada: ";
31         cin>>Opcion;
32     } while (Opcion <1 || Opcion > 9);
33     return Opcion;
34 }
35
36 int main()
37 {
38     Producto Prod, auxiliar;
39     int Opc, Cla;
40     char aux[10] = "";
41     do {
42         Opc= Menu();
43         switch (Opc)
44         {
45             case 1:{
46                 cin>>Prod;
47                 Ap1= Inventario.RegresaRaiz();
48                 Inventario.InsertaNodoSinRep(Ap1, Prod);
49                 break;
50             }

```

2.3 Código en C++

```
51         case 2:{
52             cout<<"\n\nIngrese la clave del producto a eliminar:";
53             cin>>Cla;
54             Producto Prod(Cla,aux, 0);
55             Apl= Inventario.RegresaRaiz();
56             Inventario.EliminaNodo(Apl, Prod);
57             break;
58         }
59         case 3:{
60             Apl= Inventario.RegresaRaiz();
61             cout<<"\n\n\n-----\n\n";
62             cout<<"PRODUCTOS EN INVENTARIO\n\n";
63             cout<<"-----\n\n";
64             Inventario.Inorden(Apl);
65             break;
66         }
67         case 4: {
68             cout<<"\n\nIngrese la clave del producto a buscar:";
69             cin>>Cla;
70             Producto Prod(Cla, aux, 0);
71             Apl= Inventario.RegresaRaiz();
72             Ap2= Inventario.Busqueda(Apl, Prod);
73             if (Ap2)
74             {
75                 cout<<"\n\n\nExiste un producto registrado con esa clave.\n";
76                 auxiliar=Ap2->RegresaInfo();
77                 cout<<auxiliar;
78             }
79             else
80                 cout<<"\n\nNo se ha registrado ningn producto con esa clave. \n";
81             break;
82         }
83         case 5:{
84             cout<<"\n\n\n-----\n\n";
85             cout<<"Escritura en archivo preorden\n\n";
86             cout<<"-----\n\n";
87             ofstream myfile("filename.txt", ios::out);
88             Inventario.WriteFile(myfile);
89             myfile.close();
90             break;
91         }
92         case 6:{
93             cout<<"\n\n\n-----\n\n";
94             cout<<"Escritura en archivo preorden\n\n";
95             cout<<"-----\n\n";
96             Apl= Inventario.RegresaRaiz();
97             ofstream myfile("filename.txt", ios::out);
98             Inventario.WritePreordenFile(myfile,Apl );
99             myfile.close();
100             break;
101         }
102         case 7: {
103             if(!band){
104                 getString(recorrido);
105                 std::cout << "Datos traídos del disco" << '\n';
106                 band = true;
107             }
108             else{
109                 std::cout << "Los datos ya fueron insertados." << '\n';
110             }
111             break;
112         }
113         case 8:{
114             fflush(stdin);
115             std::cout << "Ingrese nombre a buscar" << '\n';
116             char name[10] = "";
117             std::cin.getline (name,10);
118             getStringName(recorrido, name);
119             std::cout << "Dato traído del disco" << '\n';
120             break;
121         }
```

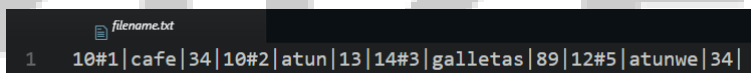
2.3 Código en C++

```
122         case 9:{
123             std::cout << "SALIENDO.." << '\n';
124             break;
125         }
126     }
127     system("pause");
128 } while (Opc >=1 && Opc< 9);
129 return 0;
130 }
131
132 void getString(int recorrido)
133 {
134     char buffer[2000] = "";
135     char lectura[2] = "";
136     char id[2] = "";
137     char nombre[10] = "";
138     char precio[10] = "";
139
140     int i = recorrido;
141     int delimitadores = 0;
142     int recor = 0, recor2 = 0, recor3 = 0;
143
144     ifstream o("filename.txt");
145     if(!o.good()){
146         std::cout << "error al abrir el archivo" << '\n';
147     }
148     else{
149         o.getline(buffer, 2000);
150     }
151     while (true) {
152         if(buffer[i] == '#') { lectura[i] = '\0'; break; }
153         lectura[i] = buffer[i];
154         i++;
155     }
156     i++;
157     while (true) {
158         if(buffer[i] == '|'){ delimitadores++; }
159         if(buffer[i] != '#' && buffer[i] != '|'){
160             if(delimitadores == 0){ id[recor] = buffer[i]; recor++;}
161             if(delimitadores == 1){ nombre[recor2] = buffer[i]; recor2++;}
162             if(delimitadores == 2){ precio[recor3] = buffer[i]; recor3++;}
163             if(delimitadores == 3){
164                 id[recor+1] = '\0';
165                 nombre[recor2+1] = '\0';
166                 precio[recor3+1] = '\0';
167                 break;
168             }
169         }
170         i++;
171     }
172     recorrido = i;
173     int pro_id = atoi(id);
174     double pro_precio = atof(precio);
175     Producto pro(pro_id, nombre, pro_precio);
176     Inventario.InsertaNodoSinRep(Inventario.RegresaRaiz(), pro);
177     if(buffer[recorrido] != '\0'){
178         o.close();
179         getString(recorrido);
180     }
181 }
182
183
184
185
186 void getStringName(int recorrido, char* producto)
187 {
188     char buffer[2000] = "";
189     char lectura[2] = "";
190     char id[2] = "";
191     char nombre[10] = "";
192     char precio[10] = "";
```



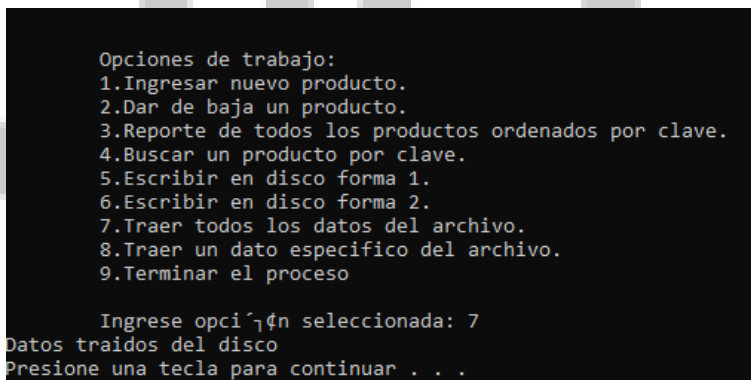
```
193
194 int i = recorrido;
195 int delimitadores = 0;
196 int recor = 0, recor2 = 0, recor3 = 0;
197
198 ifstream o("filename.txt");
199 if(!o.good()){
200     std::cout << "error al abrir el archivo" << '\n';
201 }
202 else{
203     o.getline(buffer, 2000);
204 }
205 while (true) {
206     if(buffer[i] == '#') { lectura[i] = '\0'; break; }
207     lectura[i] = buffer[i];
208     i++;
209 }
210 i++;
211 while (true) {
212     if(buffer[i] == '|'){ delimitadores++; }
213     if(buffer[i] != '#' && buffer[i] != '|'){
214         if(delimitadores == 0){ id[recor] = buffer[i]; recor++;}
215         if(delimitadores == 1){ nombre[recor2] = buffer[i]; recor2++;}
216         if(delimitadores == 2){ precio[recor3] = buffer[i]; recor3++;}
217         if(delimitadores == 3){
218             id[recor+1] = '\0';
219             nombre[recor2+1] = '\0';
220             precio[recor3+1] = '\0';
221             break;
222         }
223     }
224     i++;
225 }
226 recorrido = i;
227 if(strcmp(producto, nombre) == 0){
228     o.close();
229     int pro_id = atoi(id);
230     double pro_precio = atof(precio);
231     Producto pro(pro_id, nombre, pro_precio);
232     Inventario.InsertaNodoSinRep(Inventario.RegresaRaiz(), pro);
233 }
234 else{
235     getStringName(recorrido, producto);
236 }
237 }
```

3. Resultados



ID	Nombre	Precio
10#1	cafe	34
10#2	atun	13
14#3	galletas	89
12#5	atunwe	34

Figura 1: Datos dentro del archivo.



```
Opciones de trabajo:
1.Ingresar nuevo producto.
2.Dar de baja un producto.
3.Reporte de todos los productos ordenados por clave.
4.Buscar un producto por clave.
5.Escribir en disco forma 1.
6.Escribir en disco forma 2.
7.Traer todos los datos del archivo.
8.Traer un dato especifico del archivo.
9.Terminar el proceso

Ingrese opci^n seleccionada: 7
Datos traídos del disco
Presione una tecla para continuar . . .
```

Figura 2: Opcion 7 que es la que trae los datos del archivo.

PRODUCTOS EN INVENTARIO

Datos del producto

Clave: 1
Nombre: cafe
Precio: 34

Datos del producto

Clave: 2
Nombre: atun
Precio: 13

Datos del producto

Clave: 3
Nombre: galletas
Precio: 89

Datos del producto

Clave: 5
Nombre: atunwe
Precio: 34
Presione una tecla para continuar . . .

Figura 3: Carga de datos del archivo con éxito!.

Opciones de trabajo:
1.Ingresar nuevo producto.
2.Dar de baja un producto.
3.Reporte de todos los productos ordenados por clave.
4.Buscar un producto por clave.
5.Escribir en disco forma 1.
6.Escribir en disco forma 2.
7.Traer todos los datos del archivo.
8.Traer un dato especifico del archivo.
9.Terminar el proceso

Ingrese opción seleccionada: 8
Ingrese nombre a buscar
galletas
Dato traído del disco
Presione una tecla para continuar . . .

Figura 4: Opcion 8 que es la que trae un producto en especifico del archivo.

```
Ingrese opción seleccionada: 3

-----

PRODUCTOS EN INVENTARIO

-----

Datos del producto

Clave: 3
Nombre: galletas
Precio: 89
Presione una tecla para continuar . . .
```

Figura 5: Carga de el dato del archivo con éxito!.

```
Opciones de trabajo:
1.Ingresar nuevo producto.
2.Dar de baja un producto.
3.Reporte de todos los productos ordenados por clave.
4.Buscar un producto por clave.
5.Escribir en disco forma 1.
6.Escribir en disco forma 2.
7.Traer todos los datos del archivo.
8.Traer un dato específico del archivo.
9.Terminar el proceso

Ingrese opción seleccionada: 7
Los datos ya fueron insertados.
Presione una tecla para continuar . . .
```

Figura Extra: Comprobacion de validación.

4. Conclusiones

La realización de esta actividad fue bastante mas facil que las anteriores ya que me di la libertad de hacerla un poco fuera de lo esperado, viendo que el resultado es satisfactorio y no se encuentran problemas creo que podemos pasar por alto las expectativas.