

TDAs.

Rodríguez Tabares Juan

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

Abstract

Este trabajo comprende a la elaboracion de la actividad numero 3 de la asignatura Estructura de datos II, de la carrera de Ingenieria en computación llevada acabo en el ciclo escolar 2020B.

1. Introducción

En este documento veremos el desarrollo del trabajo encargado el cual consiste en adaptar el programa previamente entregado por el profesor para que este pueda cargar datos de un archivo a una lista y poder trabajar con el programa con esos datos o añadir mas segun el usuario deseé.

2. Desarrollo

En este punto veremos como se relizan los cambios en cada uno de los archivos para poder tener tener el resultado deseado en el programa.

2.1. Lista.h

En esta parte del programa se añadió un nuevo metotodo llamado readFromDisk el cual es una funcion de tipo void sin parametros de entrada.

Declaracion de readFromDisk: Linea 47.

Definicion de readFromDisk: Linea 171-178.

Esta funcion contiene la adición de los datos a el programa en base a una funcion creada que se hablara de ella mas a delante (Leer). Una vez ejecutada la función se añaden todos los elementos del archivo.

2.2. Producto.h

En este archivo del programa se genero un nuevo metodo el cual es Leer(), la cual es una funcion del tipo Producto la cual no recibe parametro alguno.

Declaracion de Leer(): Linea 21.

Definicion de Leer(): Linea 77-106.

Esta funcion tiene el trabajo de leer el archivo para poder colocar los datos dentro de un objeto del tipo producto y asi pueda retornarse y poder añadirse a la lista de forma correcta.

2.3. main.cpp

Este archivo del trabajo se realizaron pocos cambios en comparación a los demas archivos. Los cambios fueron los siguientes:

Linea 24: Declaracion de la apertura del archivo bajo el nombre "file".

Linea 30-32: Comprobacion para saber si existe el archivo en memoria y poder leer los datos dentro de este. Caso contrario se ejecuta el programa y se genera el archivo al final del programa para su proximo uso.

2.4. Código en C++

Lista.h: Aquí es donde se declaran todos los metodos y atributos de la lista para poder añadir los objetos que se crean.

Recordatorio: Las líneas añadidas fueron:

Línea 47

Línea 171-178.

```
1 #ifndef LISTA_H_INCLUDED
2 #define LISTA_H_INCLUDED
3 using namespace std;
4 #include <iostream>
5 #include <iostream>
6 #include <sstream>
7 #include <fstream>
8 #include <CString>
9 template <class T>
10 class Lista;
11 template <class T>
12 class NodoLista
13 {
14 private:
15     NodoLista<T>* Liga;
16     T Info;
17 public:
18     NodoLista();
19     T RegresaInfo();
20     friend class Lista<T>;
21 };
22
23 template <class T>
24 NodoLista<T>::NodoLista()
25 {
26     this->Liga = NULL;
27 }
28
29 template <class T>
30 T NodoLista<T>::RegresaInfo()
31 {
32     return Info;
33 }
34
35 template <class T>
36 class Lista
37 {
38 private:
39     NodoLista<T>* Primero;
40 public:
41     Lista();
42     NodoLista<T>* RegresaPrimero();
43     void CreaInicio();
44     void CreaFinal();
45     void ImprimeIterativo();
46     void EscribeFile();
47     void readFromDisk(const string&);
48     void ImprimeRecursivo(NodoLista<T>*);
49     void ImprimeUnNodo(NodoLista<T>*);
50     void InsertaInicio(T);
51     void InsertaFinal(T);
52     void InsertaOrdenCrec(T);
53     int InsertaAntes(T, T);
54     int InsertaDespues(T, T);
55     int EliminaPrimero();
56     int EliminaUltimo();
57     int EliminaUnNodo(T);
58     int EliminaAnterior(T);
59     int EliminaDespues(T);
60     NodoLista<T>* BuscaDesordenada(T);
61     NodoLista<T>* BuscaOrdenada(T);
62     NodoLista<T>* BuscaRecursivo(T, NodoLista<T>*);
```

2.4 Código en C++

```
63
64
65 class Exception : public std::exception {
66 private:
67     std::string msg;
68
69 public:
70     explicit Exception(const char* message) : msg(message) {}
71
72     explicit Exception(const std::string& message) : msg(message) {}
73
74     virtual ~Exception() throw() { }
75
76     virtual const char* what() const throw() {
77         return msg.c_str();
78     }
79 };
80
81
82 template <class T>
83 Lista<T>::Lista()
84 {
85     Primero = NULL;
86 }
87
88 template <class T>
89 NodoLista<T>* Lista<T>::RegresaPrimero()
90 {
91     return Primero;
92 }
93
94 template <class T>
95 void Lista<T>::CreaInicio()
96 {
97     NodoLista<T>* P;
98     T Dato;
99     char Resp;
100     Primero = new NodoLista<T>();
101     cout << "Ingrese la informacin a almacenar: \n";
102     cin >> Dato;
103     Primero->Info = Dato;
104     cout << "\n Desea ingresar otro elemento (S/N)? ";
105     cin >> Resp;
106     while (Resp == 'S' || Resp == 's')
107     {
108         cout << "Ingrese la informacin: \n";
109         cin >> Dato;
110         P = new NodoLista<T>();
111         P->Info = Dato;
112         P->Liga = Primero;
113         Primero = P;
114         cout << "\n Desea ingresar otro elemento (S/N)? ";
115         cin >> Resp;
116     }
117 }
118
119 template <class T>
120 void Lista<T>::CreaFinal()
121 {
122     NodoLista<T>* P, * Ultimo;
123     T Dato;
124     char Resp;
125     Primero = new NodoLista<T>();
126     cout << "Ingrese la informacin a almacenar: \n";
127     cin >> Dato;
128     Primero->Info = Dato;
129     Ultimo = Primero;
130     cout << "\n Desea ingresar otro elemento (S/N)? ";
131     cin >> Resp;
132     while (Resp == 'S' || Resp == 's')
133     {
```

2.4 Código en C++

```
134     cout << "\nIngrese la informacin \n";
135     cin >> Dato;
136     P = new NodoLista<T>();
137     P->Info = Dato;
138     Ultimo->Liga = P;
139     Ultimo = P;
140     cout << "\n Desea ingresar otro elemento (S/N)? ";
141     cin >> Resp;
142 }
143 }
144 template <class T>
145 void Lista<T>::ImprimeIterativo()
146 {
147     NodoLista<T>* P;
148     P = Primero;
149     while (P)
150     {
151         cout << "\nInformacin: " << P->Info;
152         P = P->Liga;
153     }
154     cout << '\n';
155 }
156
157 template <class T>
158 void Lista<T>::EscribeFile()
159 {
160     NodoLista<T>* P;
161     P = Primero;
162     while (P)
163     {
164
165         P->Info.Capturar();
166         P = P->Liga;
167     }
168     cout << "\n Escrito en Disco";
169 }
170
171 template <class T>
172
173 void Lista<T>::readFromDisk(const string& fileName) {
174     T data,aux;
175     aux = data.Leer();
176     InsertaFinal(aux);
177
178     cout<<"\n Datos traídos del disco"<<"\n";
179 }
180
181 template <class T>
182 void Lista<T>::ImprimeRekursivo(NodoLista<T>* P)
183 {
184     if (P)
185     {
186         cout << "\nInformacin: " << P->Info;
187         ImprimeRekursivo(P->Liga);
188     }
189     cout << '\n';
190 }
191
192 template <class T>
193 void Lista<T>::ImprimeUnNodo(NodoLista<T>* P)
194 {
195     if (P)
196         cout << P->Info;
197 }
198
199 template <class T>
200 void Lista<T>::InsertaInicio(T Dato)
201 {
202     NodoLista<T>* P;
203     P = new NodoLista<T>();
204     P->Info = Dato;
```

2.4 Código en C++

```
205     P->Liga = Primero;
206     Primero = P;
207 }
208
209 template <class T>
210 void Lista<T>::InsertaFinal(T Dato)
211 {
212     NodoLista<T>* P, * Ultimo;
213     P = new NodoLista<T>();
214     P->Info = Dato;
215     if (Primero)
216     {
217         Ultimo = Primero;
218         while (Ultimo->Liga)
219             Ultimo = Ultimo->Liga;
220         Ultimo->Liga = P;
221     }
222     else
223         Primero = P;
224 }
225
226 template <class T>
227 void Lista<T>::InsertaOrdenCrec(T Dato)
228 {
229     NodoLista<T>* P, * Q, * Ant;
230     if (!Primero || Primero->Info > Dato)
231         InsertaInicio(Dato);
232     else
233     {
234         Q = Primero;
235         while (Q && Q->Info < Dato)
236         {
237             Ant = Q;
238             Q = Q->Liga;
239         }
240         P = new NodoLista<T>();
241         P->Info = Dato;
242         Ant->Liga = P;
243         P->Liga = Q;
244     }
245 }
246
247 template <class T>
248 int Lista<T>::InsertaAntes(T Dato, T Ref)
249 {
250     NodoLista<T>* P, * Ant, * Q;
251     int Resp = 1;
252     if (Primero)
253     {
254         Q = Primero;
255         while ((Q != NULL) && (Q->Info != Ref))
256         {
257             Ant = Q;
258             Q = Q->Liga;
259         }
260         if (Q != NULL)
261         {
262             P = new NodoLista<T>();
263             P->Info = Dato;
264             if (Primero == Q)
265             {
266                 P->Liga = Primero;
267                 Primero = P;
268             }
269             else
270             {
271                 Ant->Liga = P;
272                 P->Liga = Q;
273             }
274         }
275         else
```

2.4 Código en C++

```
276         Resp = 0;
277     }
278     else
279         Resp = -1;
280     return Resp;
281 }
282
283 template <class T>
284 int Lista<T>::InsertaDespues(T Dato, T Ref)
285 {
286     NodoLista<T>* Q, * P;
287     int Resp = 1;
288     if (Primero)
289     {
290         Q = Primero;
291         while ((Q != NULL) && (Q->Info != Ref))
292             Q = Q->Liga;
293         if (Q != NULL)
294         {
295             P = new NodoLista<T>();
296             P->Info = Dato;
297             P->Liga = Q->Liga;
298             Q->Liga = P;
299         }
300         else
301             Resp = 0;
302     }
303     else
304         Resp = -1;
305     return Resp;
306 }
307
308 template <class T>
309 int Lista<T>::EliminaPrimero()
310 {
311     NodoLista<T>* P;
312     int Resp = 1;
313     if (Primero)
314     {
315         P = Primero;
316         Primero = P->Liga;
317         delete (P);
318     }
319     else
320         Resp = 0;
321     return Resp;
322 }
323
324 template <class T>
325 int Lista<T>::EliminaUltimo()
326 {
327     NodoLista<T>* Ant, * P;
328     int Resp = 1;
329     if (Primero)
330     {
331         if (!Primero->Liga)
332         {
333             delete (Primero);
334             Primero = NULL;
335         }
336         else
337         {
338             P = Primero;
339             while (P->Liga)
340             {
341                 Ant = P;
342                 P = P->Liga;
343             }
344             Ant->Liga = NULL;
345             delete (P);
346         }
347     }
```

2.4 Código en C++

```
347     }
348     else
349         Resp = 0;
350     return Resp;
351 }
352
353 template <class T>
354 int Lista<T>::EliminaUnNodo(T Ref)
355 {
356     NodoLista<T>* P, * Ant = NULL;
357     int Resp = 1;
358     if (Primero)
359     {
360         P = Primero;
361         while ((P->Liga) && (P->Info != Ref))
362         {
363             Ant = P;
364             P = P->Liga;
365         }
366         if (P->Info != Ref)
367             Resp = 0;
368         else
369         {
370             if (Primero == P)
371                 Primero = P->Liga;
372             else
373                 Ant->Liga = P->Liga;
374             delete (P);
375         }
376     }
377     else
378         Resp = -1;
379     return Resp;
380 }
381
382 template <class T>
383 int Lista<T>::EliminaAnterior(T Ref)
384 {
385     NodoLista<T>* Q, * Ant, * P;
386     int Resp = 1;
387     if (Primero)
388     {
389         if (Primero->Info == Ref)
390             Resp = 2;
391         else
392         {
393             Q = Primero;
394             Ant = Primero;
395             while ((Q->Info != Ref) && (Q->Liga))
396             {
397                 P = Ant;
398                 Ant = Q;
399                 Q = Q->Liga;
400             }
401             if (Q->Info != Ref)
402                 Resp = 3;
403             else
404                 if (Primero->Liga == Q)
405                 {
406                     delete (Primero);
407                     Primero = Q;
408                 }
409                 else
410                 {
411                     P->Liga = Q;
412                     delete (Ant);
413                 }
414         }
415     }
416     else
417
```

2.4 Código en C++

```
418     Resp = 4;
419     return Resp;
420 }
421
422 template <class T>
423 NodoLista<T>* Lista<T>::BuscaDesordenada(T Ref)
424 {
425     NodoLista<T>* Q, * Resp = NULL;
426     if (Primero)
427     {
428         Q = Primero;
429         while ((Q->Info != Ref) && (Q->Liga))
430             Q = Q->Liga;
431         if (Q->Info == Ref)
432             Resp = Q;
433     }
434     return Resp;
435 }
436
437 template <class T>
438 NodoLista<T>* Lista<T>::BuscaOrdenada(T Ref)
439 {
440     NodoLista<T>* Q, * Resp = NULL;
441     if (Primero)
442     {
443         Q = Primero;
444         while ((Q->Info < Ref) && (Q->Liga))
445             Q = Q->Liga;
446         if (Q->Info == Ref)
447             Resp = Q;
448     }
449     return Resp;
450 }
451
452 template <class T>
453 NodoLista<T>* Lista<T>::BuscaRecursivo(T Dato, NodoLista<T>* Q)
454 {
455     if (Q)
456         if (Q->Info == Dato)
457             return Q;
458         else
459             return BuscaRecursivo(Dato, Q->Liga);
460     else
461         return NULL;
462 }
463
464 #endif // LISTA_H_INCLUDED
```

Producto.h.cpp: Aquí es donde se declaran los métodos y atributos de el objeto producto.

Recordatorio: Las líneas añadidas fueron:

Línea 21

Línea 77-106.

```
1  #ifndef PRODUCTO_H_INCLUDED
2  #define PRODUCTO_H_INCLUDED
3  #pragma warning(disable : 4996)
4  #include <iostream>
5  #include <sstream>
6  #include <fstream>
7  #include <Cstring>
8  #include "Lista.h"
9  using namespace std;
10 class Producto
11 {
12 private:
13     int Clave;
14     char NomProd[64];
15     double Precio;
16 public:
17     Producto();
18     Producto(int, const char[], double);
```


2.4 Código en C++

```
19     double RegresaPrecio();
20     void Capturar();
21     Producto Leer();
22     int operator == (Producto);
23     int operator != (Producto);
24     int operator > (Producto);
25     int operator < (Producto);
26     friend istream& operator>> (istream&, Producto&);
27     friend ostream& operator<< (ostream&, Producto&);
28 };
29
30 Producto::Producto()
31 { }
32
33 Producto::Producto(int Cla, const char NomP[], double Pre)
34 {
35     Clave = Cla;
36     strcpy(NomProd, NomP);
37     Precio = Pre;
38 }
39
40 double Producto::RegresaPrecio()
41 {
42     return Precio;
43 }
44
45 void Producto::Capturar()
46 {
47     int a=0, b=0, c=0;
48     char converc[20];
49     stringstream ss;
50
51     ofstream o("archivo.bin", ios::app);
52     if (!o.good()) {
53         cout << "error al abrir archivo" << endl;
54     }
55     else {
56
57
58
59         ss << strlen((char*)&Clave);
60         ss >> converc;
61         o.write((char*)&converc, sizeof(int));
62         o.write((char*)&Clave, c);
63
64         a = strlen(NomProd);
65         o.write((char*)&a, sizeof(int));
66         o.write((char*)&NomProd, a);
67
68         ss << strlen((char*)&converc);
69         ss >> converc;
70         o.write((char*)&b, sizeof(int));
71         o.write((char*)&converc, b);
72
73     }
74     o.close();
75 }
76
77 Producto Producto::Leer()
78 {
79     int a=0, b=0, c=0;
80     Producto n;
81     string converc;
82     stringstream ss;
83     ifstream o("archivo.bin");
84     if (!o.good()) {
85         cout << "error al abrir archivo" << endl;
86     }
87     else{
88         ss << strlen((char*)&Clave);
89         ss >> converc;
```

2.4 Código en C++

```
90     o.read((char*)&c, sizeof(int));
91     o.read((char*)&Clave, c);
92
93     a = strlen(NomProd);
94     o.read((char*)&a, sizeof(int));
95     o.read((char*)&NomProd, a);
96
97
98     o.read((char*)&b, sizeof(int));
99     o.read((char*)&converc, b);
100     double d = stod(converc.c_str());
101     Producto pro(Clave, NomProd, d);
102     return pro;
103 }
104 o.close();
105 return n;
106 }
107
108
109 int Producto::operator == (Producto Prod)
110 {
111     int Resp = 0;
112     if (Clave == Prod.Clave)
113         Resp = 1;
114     return Resp;
115 }
116
117 int Producto::operator != (Producto Prod)
118 {
119     int Resp = 0;
120     if (Clave != Prod.Clave)
121         Resp = 1;
122     return Resp;
123 }
124
125 int Producto::operator > (Producto Prod)
126 {
127     int Resp = 0;
128     if (Clave > Prod.Clave)
129         Resp = 1;
130     return Resp;
131 }
132
133 int Producto::operator < (Producto Prod)
134 {
135     int Resp = 0;
136     if (Clave < Prod.Clave)
137         Resp = 1;
138     return Resp;
139 }
140 istream& operator>> (istream& Lee, Producto& ObjProd)
141 {
142     cout << "\n\nIngrese clave del producto: ";
143     Lee >> ObjProd.Clave;
144     cout << "\n\nIngrese nombre del producto: ";
145     Lee >> ObjProd.NomProd;
146     cout << "\n\nIngrese precio: ";
147     Lee >> ObjProd.Precio;
148     return Lee;
149 }
150
151 ostream& operator<< (ostream& Escribe, Producto& ObjProd)
152 {
153     Escribe << "\n\nDatos del producto\n";
154     Escribe << "\nClave: " << ObjProd.Clave;
155     Escribe << "\nNombre: " << ObjProd.NomProd;
156     Escribe << "\nPrecio: " << ObjProd.Precio << "\n";
157     return Escribe;
158 }
159
160
```

2.4 Código en C++

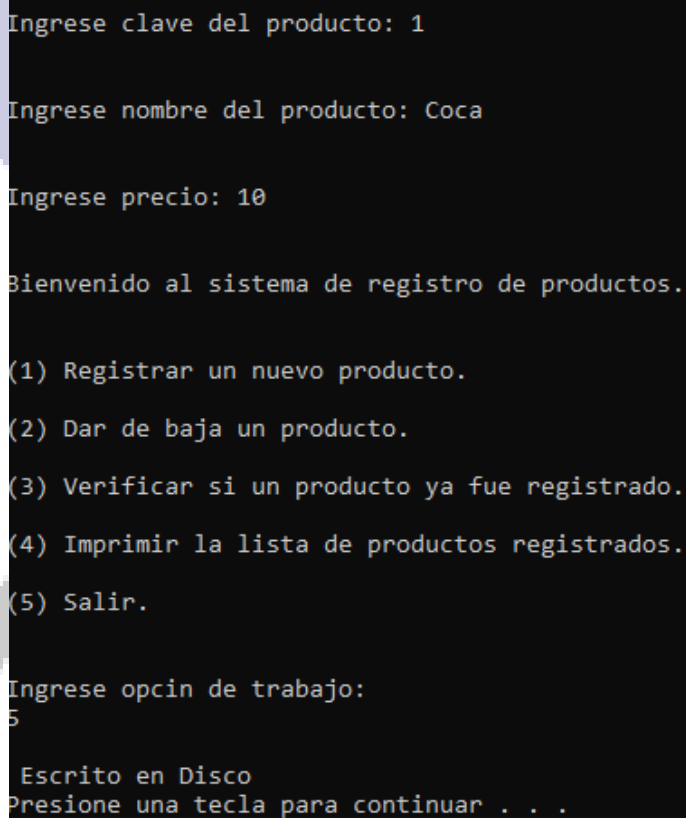
161 #endif // PRODUCTO_H_INCLUDED

main.cpp: Es donde se mostrara la funcion principal del programa. **Recordatorio:** Las lineas añadidas fueron:
Linea 24
Linea 30-32.

```
1 #include <Windows.h>
2 #include <iostream>
3 #include "Producto.h"
4 #include "Lista.h"
5 using namespace std;
6
7 int Menu()
8 {
9     int Opc;
10    cout << "\n\nBienvenido al sistema de registro de productos.\n\n";
11    cout << "\n(1) Registrar un nuevo producto.\n";
12    cout << "\n(2) Dar de baja un producto.\n";
13    cout << "\n(3) Verificar si un producto ya fue registrado.\n";
14    cout << "\n(4) Imprimir la lista de productos registrados.\n";
15    cout << "\n(5) Salir.\n";
16    cout << "\n\nIngrese opción de trabajo:\n";
17    cin >> Opc;
18    return Opc;
19 }
20
21 int main()
22 {
23
24    ifstream file("archivo.bin");
25    Lista<Producto> ListaProds;
26    Producto ObjProd;
27    NodoLista<Producto> * Apunt;
28    int Opc, Res, Clave;
29
30    if (file) {
31        ListaProds.readFromDisk("archivo.bin");
32    }
33
34
35    Opc = Menu();
36    while (Opc >= 1 && Opc <= 4)
37    {
38        switch (Opc)
39        {
40            case 1: {
41                cout << "\n\nIngrese datos del producto a registrar:\n";
42                cin >> ObjProd;
43                ListaProds.InsertaOrdenCrec(ObjProd);
44                break;
45            }
46            case 2: {
47                cout << "\n\nIngrese la clave del producto a eliminar:\n";
48                cin >> Clave;
49                Producto Produc(Clave, "", 0);
50                Res = ListaProds.EliminaUnNodo(Produc);
51                switch (Res)
52                {
53                    case 1: cout << "\n\nEl producto ya fue eliminado.\n";
54                        break;
55                    case 0: cout << "\n\nEse producto no se encuentra registrado.\n";
56                        break;
57                    case -1: cout << "\n\nNo hay productos registrados.\n";
58                        break;
59                }
60                break;
61            }
62            case 3:
63            {
64                cout << "\n\nIngrese la clave del producto a buscar:\n";
```

```
65     cin >> Clave;
66     Producto Produc(Clave, "", 0);
67     Apunt = ListaProds.BuscaOrdenada(Produc);
68     if (!Apunt)
69         cout << "\n\nEse producto no e s t registrado.\n\n";
70     else
71     {
72         cout << "\n\nEse producto e s t registrado.\n";
73         ListaProds.ImprimeUnNodo(Apunt);
74     }
75     break;
76 }
77 case 4:ListaProds.ImprimeRekursivo(ListaProds.RegresaPrimero());
78     break;
79 }
80 Opc = Menu();
81 }
82 ListaProds.EscribeFile();
83 return 0;
84 }
```

3. Resultados



```
Ingrese clave del producto: 1

Ingrese nombre del producto: Coca

Ingrese precio: 10

Bienvenido al sistema de registro de productos.

(1) Registrar un nuevo producto.
(2) Dar de baja un producto.
(3) Verificar si un producto ya fue registrado.
(4) Imprimir la lista de productos registrados.
(5) Salir.

Ingrese opcin de trabajo:
5

Escrito en Disco
Presione una tecla para continuar . . .
```

Figura 1: Primera parte de la ejecución del programa la cual consiste en el registro de productos.

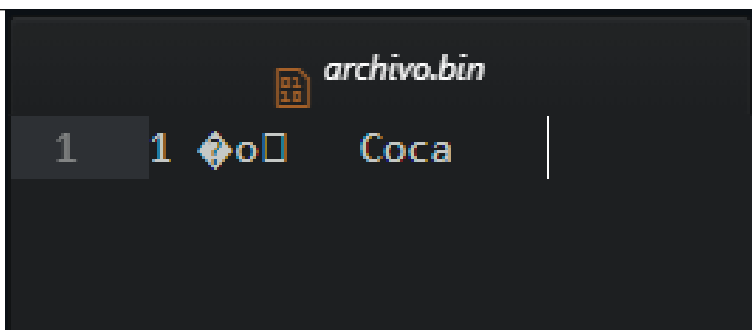


Figura 2: Vista de la generacion del archivo con el contenido que se le fue añadido en el programa.

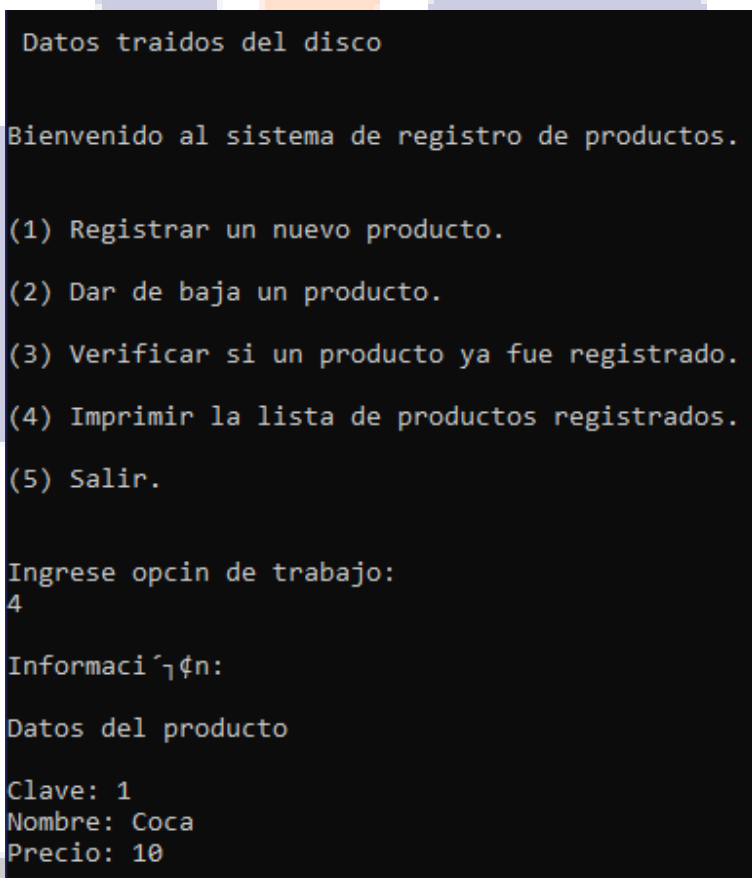


Figura 3: Carga de datos del archivo con éxito!.

4. Conclusiones

La realización de este trabajo no fue tan fácil como se veía, al momento de tener que recuperar el double fue la mayor parte del trabajo ya que no se recuperaba de forma completa o al menos ese fue el problema que sufrí, fuera de lo anteriormente mencionado fue relativamente fácil la aplicación de la lectura del archivo.