Presentación:

Practica 7. Torres de Hanoi.

Nombre: Farfán de León José Osvaldo

Código: 214796622

Carrera: Ingeniería en computación

Materia: Estructura de Datos I

Profesor: Julio Esteban Valdés López

Sección: “D13”

Fecha de entrega: 19/11/2020

Farfán de león José Osvaldo

Introducción:

Objetivo de la practica: Construir un juego de las torres de hanoi.

Fundamentación teórica: Tres pilas en las cuales se debe mover todos los discos a la ultima pila, siempre y cuando los discos de la parte de arriba sean menores que los de la parte de debajo de cada pila.

Análisis del problema: Crear 3 pilas en las cuales se estarán moviendo los datos entre ellas.

Datos de entrada y precondiciones: Cantidad de discos ingresados por el usuario.

Datos o elementos de salida: Pila C que contiene todos los discos.

Resultados obtenidos (captura de pantalla):

Código fuente en C++:

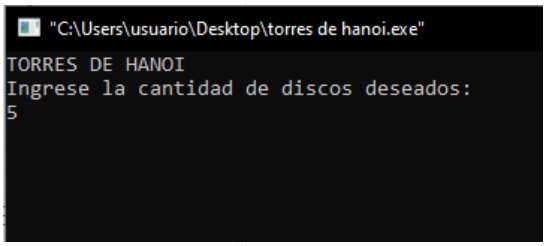
```
#include <iostream>
 #include <stdlib.h>
 #include <ctime>
 #include <windows.h>

 using namespace std;
 int salir=0;

  void gotoxy(int x,int y){
       HANDLE hcon;
       hcon = GetStdHandle(STD_OUTPUT_HANDLE);
       COORD dwPos;
       dwPos.X = x;
       dwPos.Y= y;
       SetConsoleCursorPosition(hcon,dwPos);
  }

  void TextColor(int color){
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);
 }

 struct Nodo{
   int datos;
   struct Nodo *siguiente;
 };


 void inicializa(struct Nodo *&pila);
 int vacia(struct Nodo *&pila);
 void push(int elemento, struct Nodo *&pila);
 void pop(struct Nodo *&pila);
 int top(struct Nodo *&pila);
 void llena(struct Nodo *&pila, int numDiscos);
 void imprime(struct Nodo *&pila, int clv);
 void imprime2(struct Nodo *&pila, int clv);
 void imprime3(struct Nodo *&pila, int clv);
 void menu(struct Nodo *&pila1, struct Nodo *&pila2, struct Nodo *&pila3);
 void ganador(struct Nodo *&pila1, struct Nodo *&pila2);

 void inicializa (struct Nodo *&pila){
   pila = NULL;
```

```
42  }
43
44  int vacia(struct Nodo *&pila){
45      if(pila == NULL){
46          return true;
47      }else{
48          return false;
49      }
50  }
51
52  void llena(struct Nodo *&pila, int numDiscos){
53      for(int i = numDiscos; i >= 1; i--){
54          push(i, pila);
55      }
56  }
57
58  void push(int elemento, struct Nodo *&pila){
59      struct Nodo *aux = new struct Nodo;
60      aux->datos = elemento;
61      aux->siguiente = pila;
62      pila = aux;
63  }
64
65  void pop(struct Nodo *&pila){
66      if(vacia(pila) == 1){
67          system("CLS");
68          TextColor(4);
69          gotoxy(12,10);cout<<"**********************"<<endl;
70          gotoxy(12,11);cout<<"*   Falta de datos   *"<<endl;
71          gotoxy(12,12);cout<<"**********************"<<endl;
72          TextColor(15);
73          system("pause");
74          system("cls");
75      }else{
76          struct Nodo *aux = new struct Nodo;
77          aux = pila;
78          pila = pila->siguiente;
79          delete aux;
80      }
81  }
82
83  int top(struct Nodo *&pila){
84      if (vacia(pila) == 1){
85          return -1;
86      }else{
87          return pila->datos;
88      }
89  }
90
91  void menu(struct Nodo *&pila1, struct Nodo *&pila2, struct Nodo *&pila3){
92      int VC = 0, clv=0;
93      do{
94          system("cls");
95          int opcion = 0;
96          TextColor(4);
97          imprime(pila1, 4);
98          TextColor(3);
99          imprime2(pila2, 3);
100         TextColor(6);
101         imprime3(pila3, 6);
102         TextColor(15);
103         gotoxy(40,2);cout<<"1.- Mueve de pila A a B";
104         gotoxy(40,3);cout<<"2.- Mueve de pila A a C";
105         gotoxy(40,4);cout<<"3.- Mueve de pila B a A";
106         gotoxy(40,5);cout<<"4.- Mueve de pila B a C";
107         gotoxy(40,6);cout<<"5.- Mueve de pila C a A";
108         gotoxy(40,7);cout<<"6.- Mueve de pila C a B";
109         gotoxy(40,8);cout<<"0. SALIR";
110         gotoxy(40,9);cout<<"OPCION: ";
111         gotoxy(40,10);cin>>opcion;
112         switch(opcion){
```

```
113             case 1:
114                 if(top(pila2) == -1 || top(pila1) < top(pila2)){
115                     push(top(pila1), pila2);
116                     pop(pila1);
117                 }else if(top(pila1) > top(pila2)){
118                     gotoxy(20,15); cout<<"Imposible"<<endl;
119                     system("pause");
120                 }
121             break;
122             case 2:
123                 if(top(pila3) == -1 || top(pila1) < top(pila3)){
124                     push(top(pila1), pila3);
125                     pop(pila1);
126                 }else if(top(pila1) > top(pila3)){
127                     gotoxy(20,15); cout<<"Imposible"<<endl;
128                     system("pause");
129                 }
130             break;
131             case 3:
132                 if(top(pila1) == -1 || top(pila2) < top(pila1)){
133                     push(top(pila2), pila1);
134                     pop(pila2);
135                 }else if(top(pila2) > top(pila1)){
136                     gotoxy(20,15); cout<<"Imposible"<<endl;
137                     system("pause");
138                 }
139             break;
140             case 4:
141                 if(top(pila3) == -1 || top(pila2) < top(pila3)){
142                     push(top(pila2), pila3);
143                     pop(pila2);
144                 }else if(top(pila2) > top(pila3)){
145                     gotoxy(20,15); cout<<"Imposible"<<endl;
146                     system("pause");
147                 }
148             break;
149             case 5:
150                 if(top(pila1) == -1 || top(pila3) < top(pila1)){
151                     push(top(pila3), pila1);
152                     pop(pila3);
153                 }else if(top(pila3) > top(pila1)){
154                     gotoxy(20,15); cout<<"Imposible"<<endl;
155                     system("pause");
156                 }
157             break;
158             case 6:
159                 if(top(pila2) == -1 || top(pila3) < top(pila2)){
160                     push(top(pila3), pila2);
161                     pop(pila3);
162                 }else if(top(pila3) > top(pila2)){
163                     gotoxy(20,15); cout<<"Imposible"<<endl;
164                     system("pause");
165                 }
166             break;
167             case 0:{
168                 salir = 1;
169                 system("cls");
170                 cout<<"FIN DEL PROGRAMA"<<endl;
171                 break;
172             }
173             default:
174                 cout<<"Error... Intentelo de nuevo";
175                 menu(pila1, pila2, pila3);
176             break;
177         }
178         ganador(pila1, pila2);
179     }while(salir!=1);
180 }
181
182 void imprime(struct Nodo *&pila, int clv){
183     int cont=2;
```

```
184      TextColor(clv);
185      struct Nodo *aux = new struct Nodo;
186      aux = pila;
187      while(aux != NULL){
188          gotoxy(2,cont);cout<<aux->datos<<""<<endl;
189          aux = aux->siguiente;
190          cont++;
191      }
192      gotoxy(2,10);cout<<"A"<<endl;
193  }
194  void imprime2(struct Nodo *&pila, int clv){
195      int cont=2;
196      TextColor(clv);
197      struct Nodo *aux = new struct Nodo;
198      aux = pila;
199      while(aux != NULL){
200          gotoxy(10,cont);cout<<aux->datos<<""<<endl;
201          aux = aux->siguiente;
202          cont++;
203      }
204      gotoxy(10,10);cout<<"B"<<endl;
205  }
206  void imprime3(struct Nodo *&pila, int clv){
207      int cont=2;
208      TextColor(clv);
209      struct Nodo *aux = new struct Nodo;
210      aux = pila;
211      while(aux != NULL){
212          gotoxy(20,cont);cout<<aux->datos<<""<<endl;
213          aux = aux->siguiente;
214          cont++;
215      }
216      gotoxy(20,10);cout<<"C"<<endl;
217  }
218
219
220  void ganador(struct Nodo *&pila1, struct Nodo *&pila2){
221      if(vacia(pila1) == 1 && vacia(pila2)==1){
222          salir=1;
223          system("cls");
224          TextColor(48);
225          cout<<" ------------------------------------------- "<<endl;
226          cout<<"|                                           |"<<endl;
227          cout<<"| F E L I C I D A D E S  H A S  G A N A D O |"<<endl;
228          cout<<"|                                           |"<<endl;
229          cout<<" ------------------------------------------- "<<endl;
230          TextColor(15);
231      }else{
232          return;
233      }
234  }
235
236  int main(){
237      int num;
238      struct Nodo *pila1;
239      struct Nodo *pila2;
240      struct Nodo *pila3;
241      inicializa(pila1);
242      inicializa(pila2);
243      inicializa(pila3);
244      cout<<"TORRES DE HANOI"<<endl;
245      cout<<"Ingrese la cantidad de discos deseados: "<<endl;
246      cin>>num;
247      llena(pila1, num);
248      system("cls");
249      menu(pila1, pila2, pila3);
250  }
```

Pantalla de ejecución:



Farfán de león José Osvaldo

Conclucion:

Pues me enseñe a utilizar mejor como es que se maneja las pilas estructuradas y como es su funcionamiento aun más, entender bien como es que se guardan los datos en ellas y como es que nos ayudan a mantener la memoria que solo es ocupada y no memoria de más.

Solo que debemos tener mucho cuidado cuando la pila ya no tenga datos.

19 de noviembre de 2020