18-3-2021

Pilas (Torres hanoi)

Practica: 3

Materia: Seminario de estructura de datos 1

Sección: D13.

Código: 216584703

Carrera: Ingeniería en computación.

Nombre alumno: Padilla Pérez Jorge Daray

Nombre profesor: Julio Esteban Valdes Lopez

Índice general

Introducción

Mi práctica consiste en la implementación De un juego llamado torres de hanoi utilizando pilas en lenguaje c++, en donde se puede apreciar la integración de la frase: divide y vencerás, además de introducir el tema de las pilas y el usar posiciones para agregar y quitar cosas.

Al realizar mi practica obtuve como resultado la incorrecta integración e implementación de las pilas dentro del código, en el cual es necesario asegurar que cada llamada a la pila debe estar definida sobre un problema menos complejo que el que dio lugar a la llamada, como el que yo diseñe, pero a pesar de todo mi esfuerzo no logre completar la practica a tiempo y tampoco funciona correctamente.

Pantallazo

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <string>
#define TAMMAX 500

using namespace std;

void menu();

//definicion de llamado a las pilas
typedef int tipo_dato;
typedef struct{
} pila;

pila pilaOrig, pilaDest, pilaAux ;

//Lista cosas
struct Lista{
tipo_dato datos[TAMMAX];
void inicializa();
bool vacia();
bool llena();
int tope;
void push(tipo_dato elem, pila(int datos[],int tope));
void pop(pila(int datos[],int tope));
int top(pila(int datos[],int tope));
void hanoi(struct pilaOrig,struct pilaDest,struct pilaAux,int numdiscos);
```

Aquí se intentó la implementación de una función tipo estructura para las pilas a la cual le agregue la pila original, la secundaria y la ultima.

Además de el inicio de las funciones consecutivas para su funcionamiento.

```
Lista(){
 inicializa();
 }
 };

void Lista::inicializa(){
 tope = -1;
 }

bool Lista::vacia(){
 if (tope == -1)
 {
     return true;
 }
 else{
     return false;
 }
 }

bool Lista::llena(){
if (tope == TAMMAX -1 ){
     return true;
 }
 else{
     return false;
 }
 }
```

Aquí se inicializa el constructor para la estructura lista y empezamos poniendo las funciones para inicializar, y para medir los errores que pueda generar al jugar el juego.

Aquí se mira donde se les dan los parámetros adecuados a las funciones para su correcto funcionamiento.

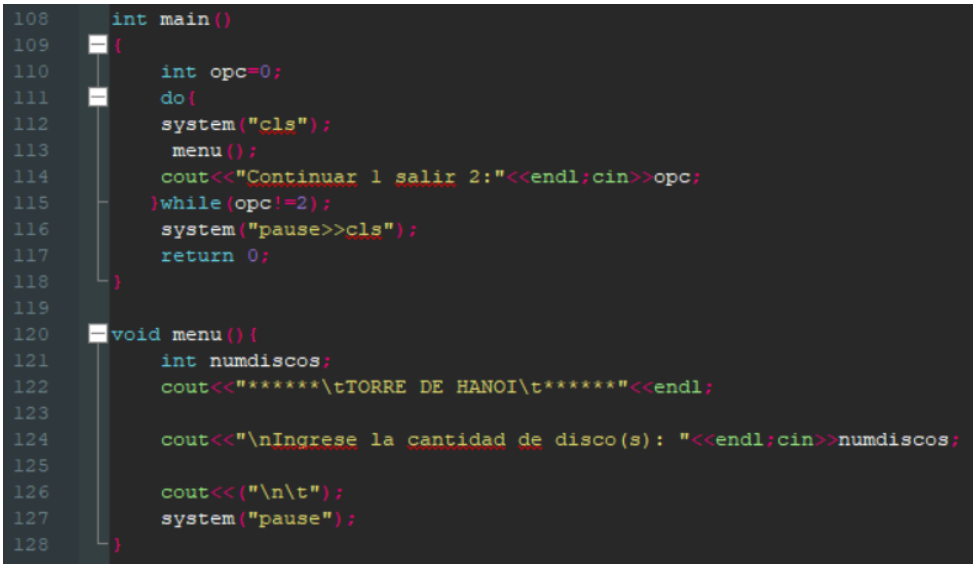
Pantallazos

```
57    //Ingresar los discos a la torre
58    void Lista::push(tipo_dato elem, pila(int datos[],int tope)){
59    if (llena()){
60        cout<<"desbordamiento de datos"<<endl;
61        system("pause");
62        return;
63    }
64    else{
65        tope = tope + 1;
66        datos[tope] = elem;
67    }
68    }
69
70    //Sacar los discos a la torre
71    void Lista::pop(pila(int datos[],int tope)){
72    if (vacia()){
73        cout<<"Insuficiencia de datos"<<endl;
74        system("pause");
75        return;
76    }
77    else{
78        tope = tope - 1;
79    }
80    }
```

Aquí se escribe la función push para ingresar los discos, y pop que es para sacar los discos como dice en texto comentado.

```
82    int Lista::top(pila(int datos[],int tope)){
83    if (vacia()){
84        cout<<"Insuficiencia de datos"<<endl;
85        system("pause");
86        return false;
87    }
88    else{
89        return datos[tope];
90    }
91    }
92
93    void Lista::hanoi(struct pilaOrig,struct pilaDest,struct pilaAux,int numdiscos){
94
95    if (numdiscos == 0){
96        return;
97    }
98    else{
99    hanoi (pilaOrig, pilaAux, pilaDest, numdiscos - 1);
100
101   push (top (pilaOrig), pilaDest);
102   pop (pilaOrig);
103
104   hanoi (pilaAux, pilaDest, pilaOrig, numdiscos - 1);
105   }
106   }
```

Aquí se muestra la función top que sirve para retornar el valor de los datos que estén en el tope de la pila, además de la función hanoi para la correcta implementación de este, mas a parte de aplicar la recursividad mandándose a llamar a si misma la función



```
int main()
{
    int opc=0;
    do{
    system("cls");
     menu();
    cout<<"Continuar 1 salir 2:"<<endl;cin>>opc;
   }while(opc!=2);
    system("pause>>cls");
    return 0;
}

void menu(){
    int numdiscos;
    cout<<"******\tTORRE DE HANOI\t******"<<endl;

    cout<<"\nIngrese la cantidad de disco(s): "<<endl;cin>>numdiscos;

    cout<<("\n\t");
    system("pause");
}
```

aquí se aprecia el menú en la cual se pregunta la cantidad de discos y se almacena en numdiscos para poder contarlos el problema es que me da errores a la hora de implementar estos mismos por lo que no se como solucionarlo pero sigo investigando como hacerle, lastimosamente tengo que entregarlo aunque no funcione para que se pueda evaluar de mejor forma.

Conclusión

Respecto a la realización del código concluyo que está muy difícil todavía para mi y que en este caso que no logro entregarlo completo tengo que seguir estudiando sobre el tema y seguir progresando, teniendo que echarle mas ganas aun de lo que ya hacia para poder concluir bien con el tema de pilas y su correcta implementación.

Código Fuente

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <string>
#define TAMMAX 500

using namespace std;

void menu();

//definicion de llamado a las pilas
typedef int tipo_dato;
typedef struct{
} pila;

pila pilaOrig, pilaDest, pilaAux ;

//Lista cosas
struct Lista{
tipo_dato datos[TAMMAX];
void inicializa();
bool vacia();
bool llena();
int tope;
void push(tipo_dato elem, pila(int datos[],int tope));
void pop(pila(int datos[],int tope));
int top(pila(int datos[],int tope));
```

```
29  void hanoi(struct pilaOrig,struct pilaDest,struct pilaAux,int numdiscos);
30  Lista(){
31  inicializa();
32  }
33  };
34
35  void Lista::inicializa(){
36  tope = -1;
37  }
38
39  bool Lista::vacia(){
40  if (tope == -1)
41  {
42     return true;
43  }
44  else{
45     return false;
46  }
47  }
48
49  bool Lista::llena(){
50  if (tope == TAMMAX -1 ){
51     return true;
52  }
53  else{
54     return false;
55  }
56  }
```

```
57
58  //Ingresar los discos a la torre
59  void Lista::push(tipo_dato elem, pila(int datos[],int tope)){
60  if (llena()){
61     cout<<"desbordamiento de datos"<<endl;
62     system("pause");
63     return;
64  }
65  else{
66     tope = tope + 1;
67     datos[tope] = elem;
68  }
69  }
70
71  //Sacar los discos a la torre
72  void Lista::pop(pila(int datos[],int tope)){
73  if (vacia()){
74     cout<<"Insuficiencia de datos"<<endl;
75     system("pause");
76     return;
77  }
78  else{
79     tope = tope - 1;
80  }
81  }
82
83  int Lista::top(pila(int datos[],int tope)){
84  if (vacia()){
85     cout<<"Insuficiencia de datos"<<endl;
```

```
86
87      system("pause");
88      return false;
89    }
90    else{
91      return datos[tope];
92    }
93    }
94
95    void Lista::hanoi(struct pilaOrig,struct pilaDest,struct pilaAux,int numdiscos){
96
97    if (numdiscos == 0){
98      return;
99    }
100   else{
101   hanoi (pilaOrig, pilaAux, pilaDest, numdiscos - 1);
102
103   push (top (pilaOrig), pilaDest);
104   pop (pilaOrig);
105
106   hanoi (pilaAux, pilaDest, pilaOrig, numdiscos - 1);
107   }
108   }
109
110   int main()
111   {
112           int opc=0;
113     do{
114           system("cls");
```

```
115
116             menu();
117     cout<<"Continuar 1 salir 2:"<<endl;cin>>opc;
118    }while(opc!=2);
119     system("pause>>cls");
120     return 0;
121  }
122
123  void menu(){
124     int numdiscos;
125     cout<<"******\tTORRE DE HANOI\t******"<<endl;
126
127          cout<<"\nIngrese la cantidad de disco(s): "<<endl;cin>>numdiscos;
128
129          cout<<("\n\t");
130          system("pause");
131  }
```