24-2-2022

Practica 2 Torres de Hanói

Materia: Seminario de estructura de datos 1

Sección: D13.

Código: 216584703

Carrera: Ingeniería en computación.

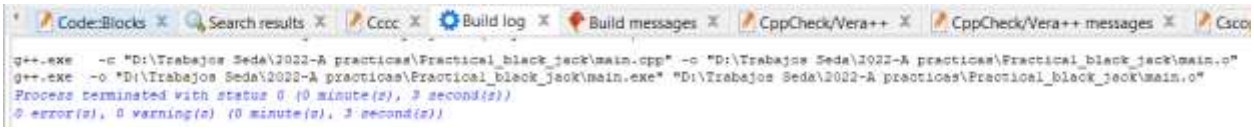
Nombre alumno: Padilla Pérez Jorge Daray

Nombre profesor: Julio Esteban Valdes Lopez

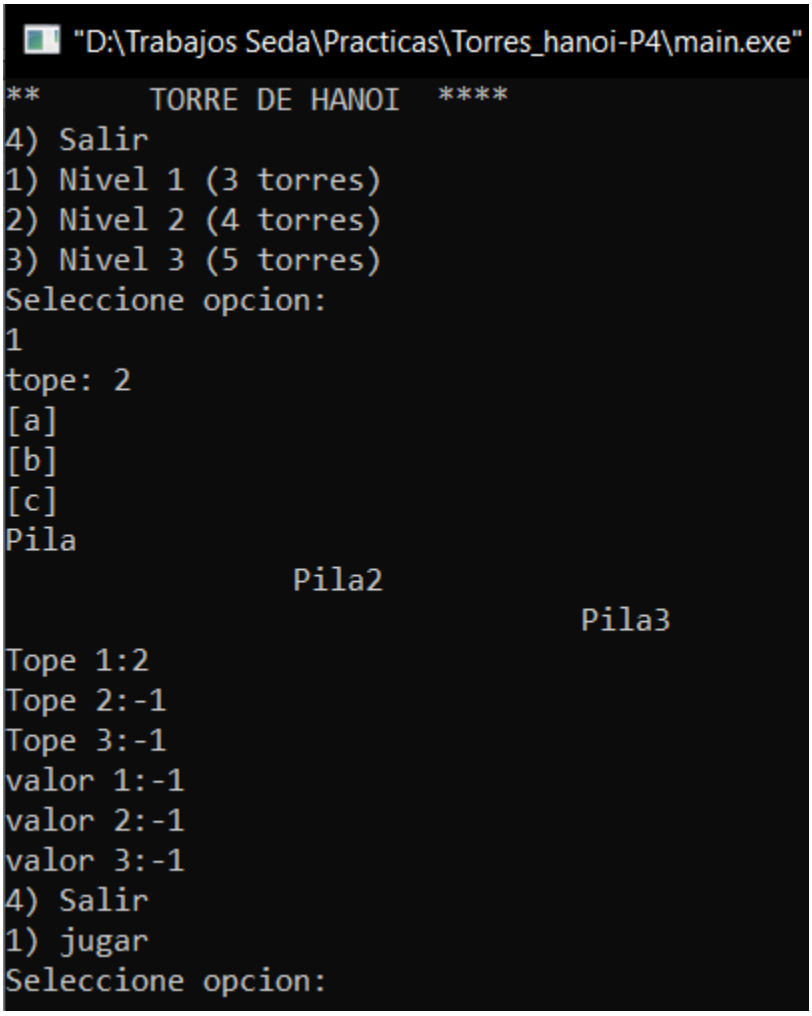
Introducción

En esta practica se realizó el juego Torres de hanoi el cual se realizo de una manera intermedia ya que, si funciona el juego, pero se puede hacer trampas para ganar lo cual no está permitido, por ende esta a medias, se espera acabarlo con mas tiempo y que se pueda revisar de una mejor forma.

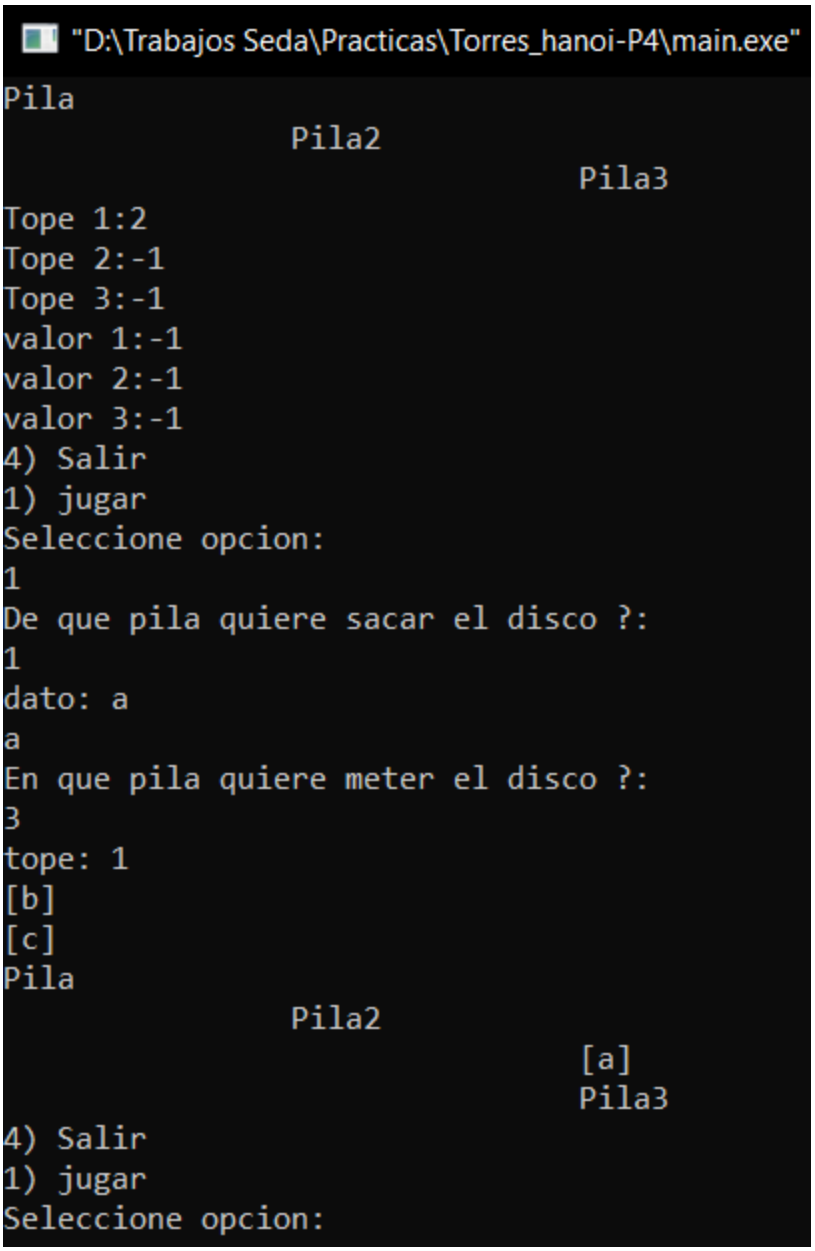
También se desarrolló la implementación de 3 niveles de dificultad desde 3 torres hasta 5, como la opción de retirarse si no se quiere continuar y volver a iniciar el juego una vez se gane y se complete el nivel que se tomo (Tener todos los discos en la última torre).



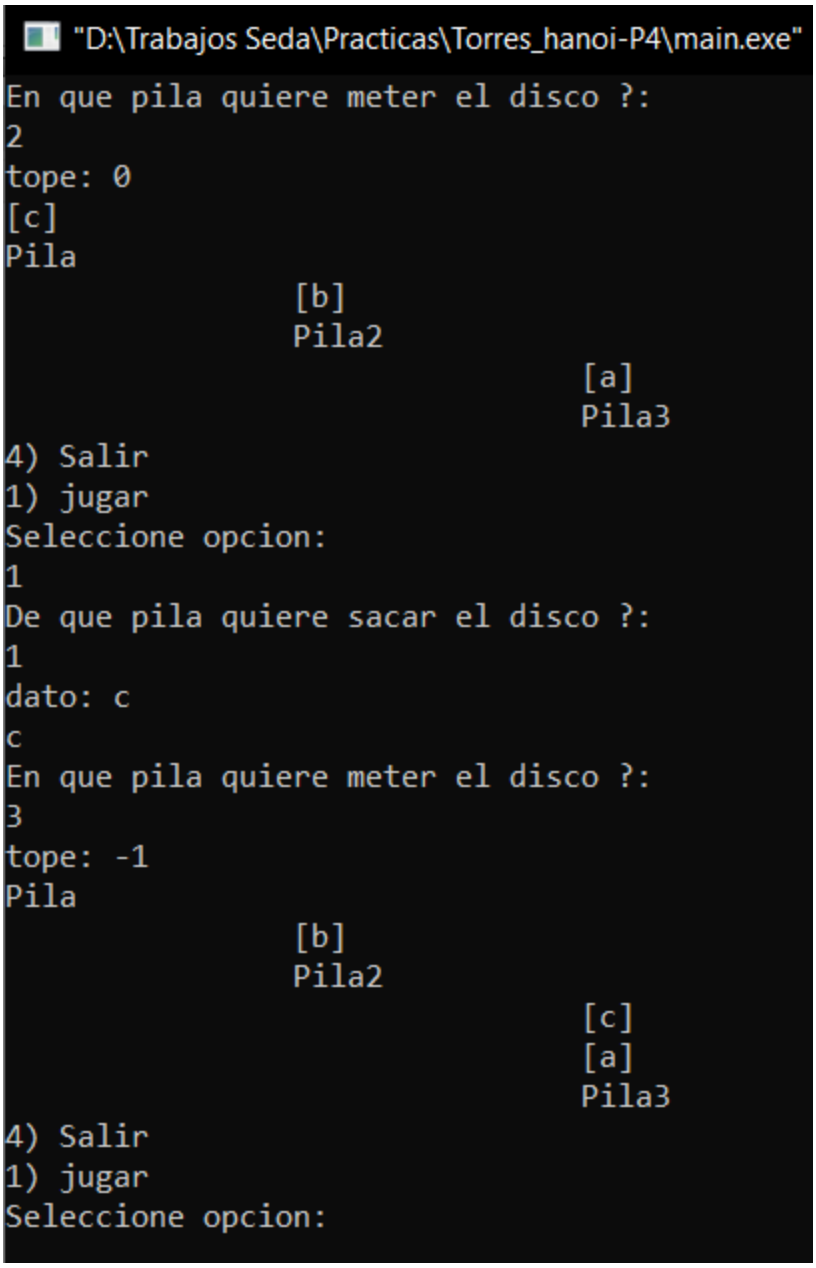
En primera no marca errores el programa.



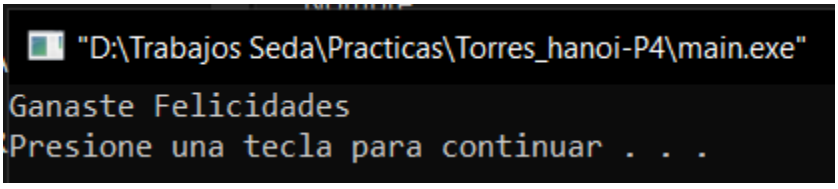
Una vez ejecutado se aprecia el menú, todo funciona solo que no cumple con las reglas.



Aquí se aprecia que saque el dato a de la primer torre y lo puse en la pila 3, este procedimiento se repite cuantas veces quieras, lo único es que se puede ganar haciendo trampas lo cual se debe modificar.



Como ven aquí se precia que se puede hacer trampa en el juego.



Una vez ganas te felicita el programa y se vuelve a iniciar para que escojas el nivel nuevamente.

Conclusión

Me falta todavía solucionar los problemas que se presentan a la hora de seguir las reglas del juego, mas que nada por falta de tiempo pero espero poder resolverlo lo antes posible para no atrasarme con el siguiente programa si es que se puede.

También me falto desarrollar los otros niveles pero ya nomas es copiar y pegar el primero pero con mas discos iniciales, no lo hice porque primero quiero que quede bien el primero para poder ahora si desarrollar los demás niveles para poder jugarlo de una mejor manera.

Codigo fuente

```
#include <iostream>
#define TAMMAX 500

using namespace std;
typedef char tipo_dato;
void menu();

class Pila{
    public:
        tipo_dato datos[TAMMAX];
        int tope, valor;
        Pila();
        bool vacia();
        bool llena();
        void push(tipo_dato elem);
        tipo_dato pop();
        tipo_dato top();
        void imprimir_pila();
        void imprimir_pila2();
        void imprimir_pila3();
        void ganador();
        };

Pila::Pila(){
    tope = -1;
    valor = -1;
}

bool Pila::vacia()
{
    return tope==-1;
```

```
}

bool Pila::llena()
{
    return tope == TAMMAX-1;
}

void Pila::push(tipo_dato e)
{
    if (llena()){
        cout<<"desbordamiento de datos"<<endl;
        system("pause");
        return;
    }
    else{
        tope++;
        datos[tope] = e;
    }
}

tipo_dato Pila::pop(){
    if(vacia()){
        cout<<"Insuficiencia de datos"<<endl;
        system("pause");
        return false;
    } else {
        tope--;
        return datos[tope+1];
    }
}

tipo_dato Pila::top(){
```

```
    if(vacia()){
        cout<<"Insuficiencia de datos"<<endl;
        system("pause");
        return false;
    } else {
        return datos[tope+1];
    }
}

void Pila::imprimir_pila(){
    cout << "tope: "<<tope<<endl;
    for (int t = tope; t > -1; t--){
        std::cout <<"["<<datos[t]<<"]"<< std::endl;
    }
            std::cout <<"Pila"<< std::endl;
}

void Pila::imprimir_pila2(){
            for (int t = tope; t > -1; t--){
            std::cout <<"\t\t["<<datos[t]<<"]"<< std::endl;
            }
            std::cout <<"\t\tPila2"<< std::endl;
}

void Pila::imprimir_pila3(){
             for (int t = tope; t > -1; t--){
            std::cout <<"\t\t\t\t["<<datos[t]<<"]"<< std::endl;
            }
            std::cout <<"\t\t\t\tPila3"<< std::endl;
}

using namespace std;
```

```
int main()
{
    int opc=0;
    do{
      system("cls");
       menu();
    cout<<"Continuar 1 salir 4:"<<endl;cin>>opc;
   }while(opc!=4);
    system("pause>>cls");
    return 0;
}

void menu(){
int opc=0;
      cout<<"**\tTORRE DE HANOI\t****"<<endl;
      cout<<"4) Salir"<<endl;
      cout<<"1) Nivel 1 (3 torres)"<<endl;
      cout<<"2) Nivel 2 (4 torres)"<<endl;
      cout<<"3) Nivel 3 (5 torres)"<<endl;
    cout<<"Seleccione opcion:"<<endl;
    cin>>opc;
      switch(opc){

            case 1: {
                Pila miPila,miPila2,miPila3;
                miPila.push('c');
            miPila.push('b');
            miPila.push('a');
            miPila.imprimir_pila();
            miPila2.imprimir_pila2();
            miPila3.imprimir_pila3();
```

```
            cout <<"Tope 1:"<<miPila.tope<<endl;
            cout <<"Tope 2:"<<miPila2.tope<<endl;
            cout <<"Tope 3:"<<miPila3.tope<<endl;
            cout <<"valor 1:"<<miPila.valor<<endl;
            cout <<"valor 2:"<<miPila.valor<<endl;
            cout <<"valor 3:"<<miPila.valor<<endl;
            do{
            int opc1=0;
                cout<<"4) Salir"<<endl;
                cout<<"1) jugar"<<endl;
                cout<<"Seleccione opcion:"<<endl;
                cin>>opc1;
                switch(opc1){

                case 1: {
                    int torre;
                    cout <<"De que pila quiere sacar el disco
?:"<<endl;cin>>torre;
                    if (torre == 1){
                        if (miPila.datos[miPila.tope] == 'a'){
                            miPila.valor = 2;
                        }
                        else if(miPila.datos[miPila.tope] == 'b'){
                            miPila.valor = 1;
                        }
                        else if(miPila.datos[miPila.tope] == 'c'){
                            miPila.valor = 0;
                        }
                        cout << "dato: "<<miPila.datos[miPila.tope]<<endl;
                        std::cout << miPila.pop() << std::endl;
                    }
                    if (torre == 2){
```

```
                    if (miPila2.datos[miPila2.tope] == 'a'){
                        miPila.valor = 2;
                    }
                    else if(miPila2.datos[miPila2.tope] == 'b'){
                        miPila.valor = 1;
                    }
                    else if(miPila2.datos[miPila2.tope] == 'c'){
                        miPila.valor = 0;
                    }
                    cout << "dato: "<<miPila2.datos[miPila2.tope]<<endl;
                    std::cout << miPila2.pop() << std::endl;
                }
                if (torre == 3){
                    if (miPila3.datos[miPila3.tope] == 'a'){
                        miPila.valor = 2;
                    }
                    else if(miPila3.datos[miPila3.tope] == 'b'){
                        miPila.valor = 1;
                    }
                    else if(miPila3.datos[miPila3.tope] == 'c'){
                        miPila.valor = 0;
                    }
                    cout << "dato: "<<miPila3.datos[miPila3.tope]<<endl;
                    std::cout << miPila3.pop() << std::endl;
                }
                int torr;
                cout <<"En que pila quiere meter el disco
?:"<<endl;cin>>torr;
                if (torr == 1 && miPila.valor == 2){
                    miPila.push('a');
                }
                else if (torr == 1 && miPila.valor == 1){
```

```
    miPila.push('b');
}
else if (torr == 1 && miPila.valor == 0){
    miPila.push('c');
}
if (torr == 2 && miPila.valor == 2){
     miPila2.push('a');
}
else if (torr == 2 && miPila.valor == 1){
     miPila2.push('b');
}
else if (torr == 2 && miPila.valor == 0){
     miPila2.push('c');
}
if (torr == 3 && miPila.valor == 2){
     miPila3.push('a');
}
else if (torr == 3 && miPila.valor == 1){
     miPila3.push('b');
}
else if (torr == 3 && miPila.valor == 0){
     miPila3.push('c');
}
miPila.imprimir_pila();
miPila2.imprimir_pila2();
miPila3.imprimir_pila3();
if (miPila.vacia() && miPila2.vacia()){
        system("cls");
    cout << "Ganaste Felicidades" <<endl;
        system("pause");
        return;
}
```

```
        }break;

        case 4:
            return;
            break;

        default:
        cout<<"La opcion: "<<opc1<<"No existe"<<endl;
        }
        }while(opc!=4);
}break;
/*case 2: {
cout<<"Que posicion deseas eliminar :"<<endl;cin>>pos;
mi_lista.elimina(pos);
}break;
case 3: {
cout<<"Que posicion deseas consultar :"<<endl;cin>>pos;
mi_lista.recupera(pos);
}break;

case 4: {
cout<<"Que dato deseas buscar :"<<endl;cin>>dato;
mi_lista.localiza(dato);
}break;

case 5: {
mi_lista.imprimir();
}break;

case 6: {
mi_lista.anular();
}break;*/
```

```
        case 4:break;

        default:
            cout<<"La opcion: "<<opc<<"No existe"<<endl;
      }
}
```