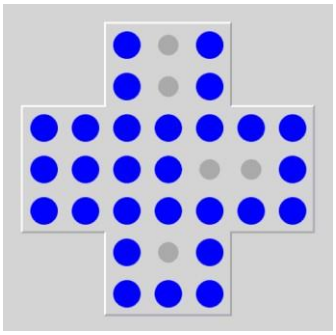
Proyecto 1: SENKU EN C++

Integrantes

- Calderon Solorzano, Greyssi Danuska (201820037)
- Dominguez Salazarcornejo, Sebastian (201820005)
- Guerrero Alejos, Jeremy Rodrigo (201820024)

El conocido juego de tablero “Senku” ha sido implementado por alumnos de la Universidad de Ingeniería y Tecnología (UTEC) en lenguaje C++ y posteriormente subido a la página Github.



Diseñamos y desarrollamos un programa que permita reproducir el juego Senku. Es un juego de tablero con una distribución dependiente del estilo que quieres jugar, inicialmente todos los espacios, representado por los vértices de los cuadrados, están ocupados, excepto el espacio central que se encuentra vacío, los espacios ocupados se representan por O y los espacios vacíos se representado por +.

Elimina las piezas saltando sobre ellas con otra pieza adyacente. El juego acaba cuando sólo queda una.

Requisitos 🚀

Trabajamos este código en el programa C++

Pre-requisitos 💼

En el desarrollo de este código usamos variables estructuradas y arreglos. Generalmente:

ARRAYS

Donde cada elemento se almacena de forma consecutiva en memoria

```
void imprimir(char **matriz,int estilo){
    for (int i = 0; i < estilo; i++) {
        for (int j = 0; j < estilo; j++) {
            cout << matriz[i][j] << " ";
        }
        cout << endl;
    }
    cout<<endl;
}
```

PUNTEROS

Permiten simular el paso por referencia, crear y manipular estructuras dinamicas de datos, tales como listas enlazadas, pilas, colas y árboles. Existen dos operadores: operador de dirección (&) y operador de indirección (*)

```
Ejemplo con operador de indireción:

int validarficha(int f,int c,char **matriz){
    if(matriz[f][c]=='0'){
        return(1);
    }
    else{
        return(0);
    }
}
```

MATRICES

Es juna serie de vectores contenidos uno en el otro (u otros), es decir, es un vector cuyas posiciones son otros vectores

```
En este caso, ya estaban declaradas las matrices anteriormente:

void mover(int validm,int fichaf,int  fichac,int  movf,int movc,char **&matriz){
    char a;
    a=matriz[fichaf][fichac];
    matriz[fichaf][fichac]=matriz[movf][movc];
    matriz[movf][movc]=a;
    if (validm==1){
        matriz[fichaf+1][fichac]='+';
    }
    else if(validm==2){
        matriz[fichaf-1][fichac]='+';
    }
    else if(validm==3){
        matriz[fichaf][fichac+1]='+';
    }
    else if(validm==4){
        matriz[fichaf][fichac-1]='+';
    }
}
```

DINÁMICAS

La memoria dinámica es un espacio de almacenamiento que se puede solicitar en tiempo de ejecución. Además de solicitar espacios de almacenamiento, también

podemos liberarlos (en tiempo de ejecución) cuando dejemos de necesitarlos. Para realizar esta administración de la memoria dinámica, C++ cuenta con dos operadores new y delete. Antes de utilizarlos, debemos incluir el encabezado .

```
En este caso usamos el tipo primitivo(int), podemos observar como usamos el *new*:

if (estilo == 0){
        cout << "Vuelva pronto" << endl;
    }
    else if (estilo == 1){
        estilo=estilo+7;
        matriz = new char*[estilo];
        for(int i = 0; i < estilo; i++){
            matriz[i] = new char[estilo];
        }
```

No hemos usado ninguna librería que no venga dentro del C++

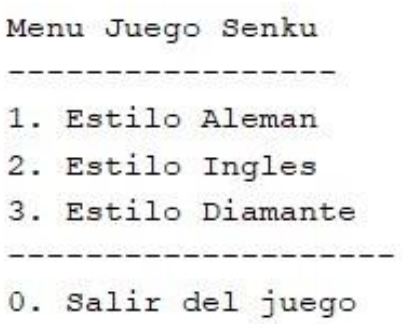
Intrucciones de uso

Para abrir el juego

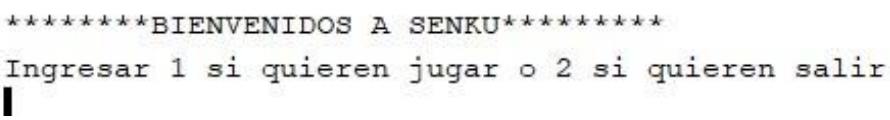
- Seleccionar el código del juego
- Abrir CLion o un programa que permita C++
- Pega el código y corra el programa

Para el juego

1) Se debe escoger cual de los tres estilos implementados se desea jugar (Alemán, Inglés, Diamante)



2) Se deberá presionar 1 si se desea jugar o 2 para salir del juego.



3) El usuario deberá ingresar la fila y la columna de una ficha (‘0’) y del mismo modo, la fila y columna de la ubicación a donde desea mover esa ficha, que deberá ser un espacio vacío (‘+’). Esto será válido solo si hay una ficha de por medio.

```
Si quieres parar el juego en cualquier momento colocar 9 en todos los parametros
  1 2 3 4 5 6 7
1       0 0 0
2       0 0 0
3 0 0 0 0 0 0 0
4 0 0 0 + 0 0 0
5 0 0 0 0 0 0 0
6       0 0 0
7       0 0 0

Ingresa ficha a mover:
Fila:1
 Columna:1
 Ficha invalida
Ingresa ficha a mover:
Fila:
```

4) Cuando se efectúe el movimiento, la ficha que está entre la ficha escogida y el lugar escogido, deberá eliminarse.

```
Ingresa ficha a mover:
Fila:2
 Columna:4
 Ficha valida
Ingresar fila a mover:4
 Ingresar columna a mover:4
 Movimiento valido
  1 2 3 4 5 6 7
1       0 0 0
2       0 + 0
3 0 0 0 + 0 0 0
4 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0
6       0 0 0
7       0 0 0
```

5) El objetivo principal es eliminar todas las fichas hasta que quede una sola en el tablero.
6) En caso desee salir del juego, deberá ingresar el numero 12 en filas y columnas.

Reglas del juego

- Al inicio el jugador mueve una ficha a un espacio vacío, obligadamente comiendo a otra ficha.
- Para comer una ficha se debe saltar sobre ella.
- La ficha comida se retira del tablero.
- Las fichas se mueven en forma horizontal y vertical y sólo salta sobre una ficha.

- No pueden moverse en diagonal.
- Gana si logra dejar sólo una ficha en el tablero. Pierde si no tiene más movimientos posibles.

Partes del código

1. Main (main.cpp)
   En el main se tiene:
   - La impresión del menu de entrada
   - Se inicializa la matriz de punteros principal (matriz), la cual tendra cierta proporcion dependiendo del estilo de juego elegido.
   - Finalmente se deriva a la funcion menu, la cual se encuentra en declarar funciones.
2. Declaración de funciones (declarar funciones.h)
   En este archivo se tiene los siguientes headers:

```
void imprimir(char **matriz,int estilo);
void hacertablero(char **&matriz,int estilo,int a);
int validarficha(int estilo,int f,int c,char **matriz);
int validarmovimiento(int estilo,int fichaf,int  fichac,int  movf,int movc,char
**matriz);
void mover(int validm,int fichaf,int  fichac,int  movf,int movc,char **&matriz);
void menu(char **&matriz,int estilo,int a);
```

3. Definición de funciones (definir funciones.cpp)
   En este archivo se encuentran la definición de cada una de las funciones declaradas en el archivo h.