**IMPLEMENTACIÓN DEL JUEGO**

**“EL SOLITARIO”**

PRIMER PROYECTO DE ESTRUCTURA DE DATOS



Integrantes:

* Ferro Palomino, Gianfranco Augusto
* Ipanaqué Casquina, Ingrid Fransheska
* Hidalgo Chavez, Daniel Alfredo
* Huacac Chiri, Claudia Esteffani
* Aguirre Janampa, Cristian Fernando

**INTRODUCCIÓN**

La programación estructurada desempeña un papel de elevada importancia en la informática, al igual que en el aprendizaje de todo aquel que estudia programación, y como toda disciplina presenta un sinfín de problemas que por si mismos suponen un desafío en el avance de esta. Para un óptimo entendimiento de esta disciplina se requiere de sólidos conocimientos en lógica, capacidad de abstracción y de la buena práctica de ordenamiento de datos.

El proyecto que mostraremos a continuación está basado en el lenguaje de programación C, ya que éste lenguaje es multiplataforma y funciona en diferentes sistemas operativos, donde pondremos en práctica conceptos como estructuras, arreglos, pilas y muchos conceptos más aprendidos a lo largo de nuestras clases necesarios para la implementación de nuestro proyecto.

**OBJETIVOS**

* Programar el juego “Solitario”.
* Desarrollo y aplicación de conocimientos de estructuración de datos, arreglos y pilas en lenguaje C.
* Desarrollo de una interfaz gráfica.
* Obtener nuevos conocimientos de la estructuración de datos a través de la investigación y el desarrollo del proyecto.

**MARCO TEÓRICO**

* **Pila**

Una pila (stack) es un tipo fundamental de lista (Tipo de dato abstracto fundamental) en la que todas las inserciones y supresiones de elementos solo se pueden realizar desde una única posición, el extremo, la cima o tope de la pila (TOP)

Las tres operaciones más importantes un una pila son:

PUSH: Meter, poner, apilar. Operación que inserta un elemento en la cima de la pila.

POP: Sacar, quitar, desapilar. Operación que elimina el elemento que ocupaba la posición cima. El elemento que había sido insertado inmediatamente antes que el eliminado pasara a ser ahora la nueva posición cima.

TOPE: Operación que devuelve el elemento que ocupa en ese instante la posición cima pero sin extraerlo.

* **Estructuras**

Las estructuras conocidas generalmente con el nombre de registros, representan un tipo de datos estructurado. Se utilizan para resolver problemas que involucran tipos de datos estructurados, heterogéneos.

“Una estructura es una colección de elementos finita y heterogénea”

Finita, porque se puede determinar el número de componentes y heterogénea porque todos los elementos pueden ser de tipos de datos diferentes. Cada componente de la estructura se denomina campo y se identifica con un nombre único.

Ejemplo:

typedef struct Alumno {

char num\_carne [11];

char nombre\_alumno [60];

int edad;

char direccion[60];

}Alum;

**DESARROLLO**

El desarrollo del proyecto se definirá de la siguiente manera:

La aplicación que se desarrollará en el lenguaje de programación C, es la simulación de un juego llamado “El solitario”, lo cual es una aplicación en el dominio de pilas como una estructura de datos.

**-Distribución de las cartas:**

El total de pilas que desarrollaremos serán 13 en total, de las cuales están distribuidas de la siguiente manera:

* **7 pilas de juego**
* **4 pilas de salida**
* **1 pila de reserva**
* **1 pila de descarte**

El juego consta de una baraja de 52 cartas de las cuales distribuiremos 28 de las cartas en **las 7 pilas de juego** de la siguiente manera:

* En la primera pila 1 carta
* En la segunda pila 2 cartas

Y así sucesivamente hasta tener 7 cartas en la séptima pila. La condición es que en las 7 pilas, siempre la última carta este **visible o abierta**, como se ve en la figura de abajo: 

Las cartas restantes son colocadas **cerradas** en otra pila llamada **pila de reserva.**

Por otro lado tenemos, la **pila de descarte**, la cual no tendrá ninguna carta al comienzo del juego y estará a la espera de cartas **abiertas** de la **pila de reserva.**

**IMPLEMENTACIÓN DE LAS PILAS:**

* **Mediante vectores:**

Una pila puede implementarse mediante vectores, en cuyo caso la dimensión

es fija. En este caso se dirá que es una implementación estática. Cuando lo

implementamos mediante un vector hay que tener en cuenta que el tamaño de la pila

no puede exceder el número de elementos del array.

Definiremos, antes de implementar una pila mediante vectores, el número

máximo de elementos que podremos meter en el apila. Será n

La declaración de pila es:

Elemento = E;

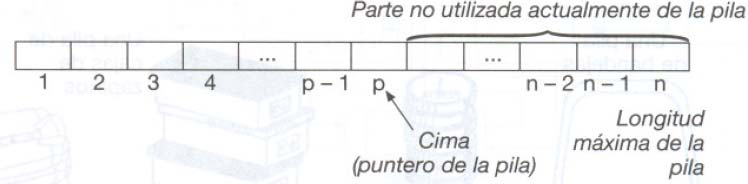
Pila = registro de

cima: numérico;

arreglo: vector [1...n] de elemento:

fin registro;

Luego la pila será:



Donde cima corresponde a la posición en el vector del último elemento insertado en la lista, o sea que a de apuntar en todo momento al elemento tope. Así si pila.cima=01 entenderemos que la pila esta vacía, mientras que si pila.cima = n la pila estará llena. Esta implementación no es apropiada cuando no se conoce el número máximo de elementos que puede tener una pila ya que podría quedarse corta por faltarle posiciones, y se produciría un error, o por pasarse de larga, es decir, utilizar mucha más memoria de la estrictamente necesaria y desaprovecharla. De todas formas, esta implementación tiene una gran ventaja, que es la sencillez y rapidez con la que se programa y se realizan las operaciones. Por ultimo tenemos 4 **pilas de salida**, la cual estará inicialmente vacía y a la espera de poder llenarse con las condiciones que detallaremos más adelante.

**-Movimientos admitidos de las cartas**

* En las **pilas de juego** solo se pueden abrir cartas que formen una secuencia decreciente consecutiva y de colores alternados, por ejemplo si en el tope de la pila se encuentra el 7 de trébol, solo se puede colocar sobre él un 6 de oro o de corazones.
* Todas las cartas abiertas de la **pila de juego** pueden ser movidas como una unidad y colocadas en otra pila desde que cumplen la regla anterior.
* Cada **pila de juego** debe tener por lo menos una carta abierta, de modo que si una carta abierta fue removida, entonces la carta del tope debe ser abierta.
* Cuando una **pila de juego** queda vacia ella solo puede recibir o un rey que esta abierta en alguna **pila de juego**(conjuntamente con las cartas sobre él) o la carta que se encuentra en el tope de la **pila de descarte.**
* Siempre que un As aparece en el tope de una **pila de juego o de descarte** ella debe ser pasada directamente a una de las **pilas de salida.**
* En cada una de las pilas de salida las cartas son colocadas abiertas, formando una secuencia consecutiva creciente del mismo naipe. No está permitido remover las cartas de la **pila de salida.** Solo una carta del tope de la **pila de descarte** o de la **pila de juego** debe ser colocada en la **pila de salida.**
* Las cartas de la **pila de reserva** pueden ser removidas abiertas hacia la **pila de descarte** uno a la vez.
* Las cartas de la **pila de descarte** solo pueden ser removidas una a la vez. Siempre que la **pila de descarte** quede vacía una carta de la **pila de reserva** se transfiere a ella.
* Para empezar el juego, mover una carta de la **pila de reserva** a la **pila de descarte**
* Si no es posible realizar ningún movimiento transferir una carta de la **pila de reserva** a la **pila de descarte.**
* El juego termina cuando no es posible realizar ningún movimiento y la **pila de reserva** quede vacía.

**-Características del programa**

* El juego puede efectuarse varias veces que el usuario desee.
* Las 13 pilas usadas para el programa están almacenadas en un vector de 60 posiciones, con las operaciones de apilar, desapilar, pilavacia, etc.
* Las cartas están divididas en 4 tipos: corazones, espadas, diamantes y tréboles, para tipo de carta hay una secuencia de números: A(as), 2, 3, 4, 5, 6, 7, 8, 9, 10, 111, 12, 13. Asi como su color: negro o rojo y también si carta esta visible o no.

**REFERENCIAS**

* The art of computer programming, Knuth Donald, editorial: Addison-Wesley vol. 1.