Universidad de San Carlos de Guatemala
Centro Universitario de Occidente División de Ciencias de la Ingeniería
Estructura de Datos
Aux. Yefer Rodrigo Alvarado Tzul
Sección: A

Manual Técnico

Carlos Raúl Alberto López Peláez 202031871

Quetzaltenango,8 de Marzo del 2024

Juego del Solitario

ingresarCartas:

Descripción: Esta función ingresa cartas en una lista doblemente enlazada.
Parámetros:

- lista: Un puntero a la lista doblemente enlazada donde se deben ingresar las cartas.
- catidadCartas: La cantidad total de cartas a ingresar.
- cartas[]: Un arreglo de estructuras Carta que contiene las cartas a ingresar.
- indice: El índice desde el cual comenzar a ingresar las cartas.

Acciones:
Itera a través del arreglo de cartas desde el índice dado hasta la cantidad total de cartas.
Si es la última carta, establece la propiedad ocultar de la carta en false.
Ingresa cada carta en la lista.

IngresarCartasColas:

Descripción: Esta función ingresa cartas en una cola.
Parámetros:

- cola: Un puntero a la cola donde se deben ingresar las cartas.
- catidadCartas: La cantidad total de cartas a ingresar.
- cartas[]: Un arreglo de estructuras Carta que contiene las cartas a ingresar.
- indice: El índice desde el cual comenzar a ingresar las cartas.

Acciones:
Itera a través del arreglo de cartas desde el índice dado hasta la cantidad total de cartas.
Si es la última carta, establece la propiedad ocultar de la carta en false.
Ingresa cada carta en la cola.

ImprimirCartas:

Descripción: Esta función imprime el contenido de varias estructuras de datos (colas, pilas y listas doblemente enlazadas).
Parámetros:

- cola1, cola2: Punteros a las colas a imprimir.
- pila1, pila2, pila3, pila4: Punteros a las pilas a imprimir.
- lista1, lista2, …, lista7: Punteros a las listas doblemente enlazadas a imprimir.

Acciones:
Imprime el contenido de cada estructura de datos en el orden especificado.

cambiarCartasCola:

Descripción: Esta función realiza cambios en las cartas dentro de una cola según ciertas condiciones.

Parámetros:

- cola: Un puntero a la cola de cartas.
- pilaCambio: Un puntero a la pila donde se realizarán los cambios.
- colaCambio: Un puntero a la cola donde se realizarán los cambios.
- listaCambio: Un puntero a la lista doblemente enlazada donde se realizarán los cambios.
- tipo: Un valor entero que indica el tipo de cambio a realizar.

Acciones:
Realiza diferentes acciones según el valor de tipo:

- Si tipo es 1:
- Comprueba si la cola no es nula y si la pila de cambio está vacía.
- Si la carta en la cima de la pila de cambio tiene ciertas propiedades, la inserta en la pila de cambio.
- Si tipo es 2:
- Inserta una carta en la cola de cambio.
- Si tipo es 3:
- Realiza una acción específica en la lista (se requiere completar la lógica).

CambiarCartasLista:

Descripción: Esta función se encarga de mover cartas de una lista doblemente enlazada a una pila, otra cola o una lista doblemente enlazada, según el tipo especificado.

Parámetros:

- lista: Puntero a la lista doblemente enlazada de donde se moverán las cartas.
- pilaCambio: Puntero a la pila donde se moverán las cartas si el tipo es 1.
- colaCambio: Puntero a la cola donde se moverán las cartas si el tipo es 2.
- listaCambio: Puntero a la lista doblemente enlazada donde se moverán las cartas si el tipo es 3.
- tipo: Indica el destino de las cartas:

1) Pila
2) Cola
3) Lista doblemente enlazada

   pos: Posición de la carta en la lista (solo se usa si el tipo es 3).

generarCopiaAnterior:
Descripción: Esta función crea una copia de la carta superior de una estructura de datos en una pila auxiliar.

Parámetros:

- cola: Puntero a la cola de donde se copiará la carta (si el tipo es 1).
- pila: Puntero a la pila de donde se copiará la carta (si el tipo es 2).
- lista: Puntero a la lista doblemente enlazada de donde se copiará la carta (si el tipo es 3).

- pilaCopia: Puntero a la pila auxiliar donde se creará la copia.
- tipo: Indica la estructura de datos de donde se copiará la carta:

1) Pila
2) Cola
3) Lista doblemente enlazada

Acciones:

- Valida si la estructura de datos está vacía: Si la estructura de datos está vacía, la función retorna sin realizar ninguna acción.
- Copia la carta:
- Cola: Se utiliza la función mostrarPosValor para obtener la carta en la posición indicada y se inserta en la pila auxiliar.
- Pila: Se utiliza la función mostrarPosValor para obtener la carta en la cima de la pila y se inserta en la pila auxiliar.
- Lista doblemente enlazada: Se utiliza la función consultar para obtener la carta en la posición indicada y se inserta en la pila auxiliar.

Menú Principal del solitario

```
std::cout << "*******************************" << std::endl;
    std::cout << "*          Solitario          *" << std::endl;
    std::cout << "*******************************" << std::endl;

    while (pila1->contador() < 13 && pila2->contador() < 13 && pila3->contador() < 13 && pila4->contador() < 13 ) {

        imprimirCartas(cola1, cola2, pila1, pila2, pila3, pila4, lista1, lista2, lista3, lista4, lista5, lista6,lista7);
        int seleccion;

        cout << "1. Colas " << endl;
        cout << "2. Barajas " << endl;
        cout << "3. volver " << endl;
        cout << " Ingrese un numero " << endl;
        cin >> seleccion;

        switch (seleccion) {
            case 1:
                imprimirCartas(cola1, cola2, pila1, pila2, pila3, pila4, lista1, lista2, lista3, lista4, lista5, lista6,lista7);
                cout << " A seleccionado Colas " << endl;
                cout << "1. Cola 1 " << endl;
                cout << "2. Cola 2 " << endl;
                cout << " Ingrese un numero " << endl;
                int colas;
                cin >> colas;
                switch (colas) { ...
                }
                break;
            case 2:
                imprimirCartas(cola1, cola2, pila1, pila2, pila3, pila4, lista1, lista2, lista3, lista4, lista5, lista6,lista7);
                cout << " A seleccionado Barajas " << endl;
                cout << "1. Baraja 1 " << endl;
                cout << "2. Baraja 2 " << endl;
                cout << "3. Baraja 3 " << endl;
                cout << "4. Baraja 4 " << endl;
                cout << "5. Baraja 5 " << endl;
                cout << "6. Baraja 6 " << endl;
                cout << "7. Baraja 7 " << endl;
                cout << " Ingrese un numero " << endl;
                int pilas;
                cin >> pilas;
                switch (pilas) { ...
                }
                break;
            case 3:
                cout<<"Restaurando" << endl;
                    cola1->borrar();
                    cola2->borrar();
                    restaurarCopia(cola1, pila1, lista1, pilaCopia[1], 1);
                    restaurarCopia(cola2, pila1, lista1, pilaCopia[2], 1);

                break;
            default:
                break;
        }
    }
```

Descripción general:
Es el inicio del programa solitario donde se encuentra toda la jugabilidad principal,permite al usuario mover cartas entre diferentes pilas y barajas, y también ofrece la posibilidad de restaurar el juego a un estado anterior.

Componentes principales:

Funciones:

ImprimirCartas()`: Muestra las cartas en las pilas, barajas y listas.
restaurarCopia()`: Restaura el juego a un estado anterior a partir de una copia.

Variables:

colal y cola2: Objetos que representan las dos colas del juego.
pilal` a `pila4`: Objetos que representan las cuatro pilas del juego.
listal` a `lista7`: Objetos que representan las siete listas del juego.
seleccion Variable que almacena la opción seleccionada por el usuario.
opciones de colas y barajas respectivamente.

Flujo del programa:

1. Se muestran las cartas en las pilas, barajas y listas.
2. Se presenta al usuario un menú con tres opciones:
    - Mover cartas entre colas.
    - Mover cartas entre barajas.
    - Restaurar el juego a un estado anterior.
3. El usuario selecciona una opción del menú.
4. Se ejecuta la acción correspondiente a la opción seleccionada.
5. Se repiten los pasos 1 a 4 hasta que el usuario decida salir del programa.

Cola

```
//
// Created by carlosl on 27/02/24.
//

#ifndef SOLITARIOENCONSOLA_COLA_H
#define SOLITARIOENCONSOLA_COLA_H
#include "string"
#include "../Nodo/Nodo.h"
#include "../Pila/Pila.h"

using namespace std;


class Cola {
private:
    Nodo *raiz;
    int contadorNum;
public:
    int numCola = 0;
    Cola();
    ~Cola();
    void insertar(Carta x);
    Carta extraer();
    void imprimir();
    void borrar();
    int contador();
    Nodo * devolverRaiz();
    Carta mostrarValor();
    void mostrarPosValor(Pila *pilacopia);
};


#endif //SOLITARIOENCONSOLA_COLA_H
```

Descripción general:

Este código define una clase Cola que representa una estructura de datos tipo cola. La cola implementa el orden FIFO (primero en entrar, primero en salir).

Atributos:

raiz: Puntero al primer nodo de la cola.
numCola: Número identificador de la cola.

Métodos:

Cola::Cola(): Constructor por defecto. Inicializa la cola vacía (raiz = nullptr).

Cola::insertar(Carta x): Inserta un nuevo elemento (Carta x) al final de la cola.

Cola::imprimir(): Imprime los elementos de la cola en orden FIFO. Se alternan la carta real y una carta oculta ("-").

Cola::extraer(): Extrae y devuelve el primer elemento de la cola. Si la cola está vacía, devuelve un valor vacío {}.

Cola::~Cola(): Destructor. Elimina todos los nodos de la cola.

Cola::contador(): Cuenta el número de elementos en la cola.

Cola::devolverRaiz(): Devuelve el puntero al primer nodo de la cola.

Cola::mostrarValor(): Devuelve el valor del primer elemento de la cola sin extraerlo. Si la cola está vacía, devuelve un valor vacío {}.

Cola::mostrarPosValor(Pila *pilacopia): genera una copia de la cola utilizando una pila auxiliar pilacopia.

Cola::borrar(): Elimina todos los elementos de la cola.

Lista Doblemente Enlazada

```
//
// Created by carlosl on 27/02/24.
//
#ifndef SOLITARIOENCONSOLA_LISTADOBLEMENTEENLAZADA_H
#define SOLITARIOENCONSOLA_LISTADOBLEMENTEENLAZADA_H
#include "../Carta/Carta.h"
#include "../Nodo/Nodo.h"
#include "../Pila/Pila.h"

class ListaDoblementeEnlazada {
private:
    Nodo *raiz;
public:
    int numLista = 0;
    ListaDoblementeEnlazada();
    ~ListaDoblementeEnlazada();
    int cantidad();
    void insertar(int pos, Carta x);
    void ingresar( Carta x);
    Carta extraer(int pos);
    void borrar(int pos);
    void borrarTodo();
    void intercambiar(int pos1, int pos2);
    bool vacia();
    void imprimir();
    bool existe(string x);
    Nodo * devolverRaiz();
    Carta mostrarValor();
    void mostrarPosValor( Pila *pilacopia);
    Carta consultar(int pos);
    int cartasSeleccionables();
};


#endif //SOLITARIOENCONSOLA_LISTADOBLEMENTEENLAZADA_H
```

Descripción general:

Este código define una clase ListaDoblementeEnlazada que representa una estructura de datos tipo lista doblemente enlazada. La lista permite almacenar elementos de tipo Carta y ofrece operaciones básicas como insertar, eliminar, buscar e imprimir.

Atributos:

raiz: Puntero al primer nodo de la lista.
numLista: Número identificador de la lista.

Métodos:

ListaDoblementeEnlazada(): Constructor por defecto. Inicializa la lista vacía (raiz = nullptr).

~ListaDoblementeEnlazada(): Destructor. Elimina todos los nodos de la lista.

borrarTodo(): Elimina todos los nodos de la lista.

cantidad(): Cuenta el número de elementos en la lista.

cartasSeleccionables(): Cuenta el número de cartas que no están ocultas en la lista.

insertar(int pos, Carta x): Inserta un nuevo elemento (Carta x) en la posición pos de la lista.

ingresar(Carta x): Inserta un nuevo elemento (Carta x) al inicio de la lista.

extraer(int pos): Extrae y devuelve el elemento en la posición pos de la lista. Si la posición es inválida, devuelve un valor vacío {}.

borrar(int pos): Elimina el elemento en la posición pos de la lista.

intercambiar(int pos1, int pos2): Intercambia los elementos en las posiciones pos1 y pos2 de la lista.

vacia(): Devuelve true si la lista está vacía, false en caso contrario.

devolverRaiz(): Devuelve el puntero al primer nodo de la lista.

imprimir(): Imprime los elementos de la lista, alternando la carta real y una carta oculta ("-").

existe(string x): Busca un elemento con la carta x en la lista. Devuelve true si lo encuentra, false en caso contrario.

mostrarValor(): Devuelve el valor del primer elemento de la lista sin extraerlo.

consultar(int pos): Devuelve el valor del elemento en la posición pos de la lista.

mostrarPosValor(Pila *pilacopia): genera una copia de la lista utilizando una pila auxiliar pilacopia.

Pila

```
//
// Created by carlosl on 27/02/24.
//

#ifndef SOLITARIOENCONSOLA_PILA_H
#define SOLITARIOENCONSOLA_PILA_H


#include "../Carta/Carta.h"
#include "../Nodo/Nodo.h"

class Pila {

    Nodo *raiz;
    int ordenamiento = 0;
public:
    int numPila = 0;
    Pila();
    ~Pila();
    void insertar(Carta x);
    void ingresar(Carta x);
    Carta extraer();
    bool revisar(string dato);
    void imprimir();
    void borrar();
    int contador();
    Carta mostrarValor();
    void mostrarPosValor(Pila *pila);
    Nodo * devolverRaiz();
};


#endif //SOLITARIOENCONSOLA_PILA_H
```

Descripción general:

Esta clase implementa una pila utilizando una estructura de datos ligada. La pila permite insertar, extraer, revisar y mostrar elementos.

Atributos:

raiz: Puntero al primer nodo de la pila.
ordenamiento: Variable que se utiliza para controlar el orden de inserción de los elementos.
numPila: Número de la pila (se utiliza para identificarla en caso de que existan multiples).

Métodos:

Pila(): Constructor de la clase. Inicializa la variable raiz a nullptr.

insertar(Carta x): Inserta un elemento en la pila. El elemento se inserta al principio de la pila si su valor coincide con el valor de la variable ordenamiento.

ingresar(Carta x): Inserta un elemento en la pila. El elemento se inserta al principio de la pila si su valor es diferente de 0.

revisar(string dato): Revisa si un elemento con el nombre dado se encuentra en la pila.

imprimir(): Imprime los elementos de la pila.

extraer(): Extrae un elemento de la pila. El elemento que se extrae es el que se encuentra al principio de la pila.

Pila::~Pila(): Destructor de la clase. Elimina todos los nodos de la pila.

devolverRaiz(): Devuelve un puntero al primer nodo de la pila.

contador(): Devuelve el número de elementos que hay en la pila.

mostrarValor(): Devuelve el valor del elemento que se encuentra al principio de la pila.

mostrarPosValor(Pila pilacopia): Copia los elementos de la pila actual a la pila pilacopia.

borrar(): Elimina todos los elementos de la pila.