|  |
| --- |
|  |
| ESTRUCTURAS DE DATOS PL1 |
|  |

|  |
| --- |
| Nombre: García Ibáñez Pedro DNI: 54890356M  Nombre: Rodríguez Nieto Rodrigo DNI: |

INDICE

[INDICE 1](#_Toc181011661)

[ESPECIFICACION DE LOS TADS 2](#_Toc181011662)

[TADS CREADOS 2](#_Toc181011663)

[DEFINICION DE LOS TADS 2](#_Toc181011664)

[SOLUCIONES ADOPTADAS 3](#_Toc181011665)

[Problema de la implementación dinámica: 3](#_Toc181011666)

[Problema Inicial con el uso de GitHub y GitHub Desktop: 3](#_Toc181011667)

[Problema con la sobre definición de la clase Proceso: 3](#_Toc181011668)

[MÉTODOS DESTACADOS 5](#_Toc181011669)

[Creación de Procesos: 5](#_Toc181011670)

[Pila: 5](#_Toc181011671)

[Colas: 5](#_Toc181011672)

[Listas: 5](#_Toc181011673)

[Gestor: 5](#_Toc181011674)

[COMPORTAMIENTO DEL PROGRAMA 7](#_Toc181011675)

[BIBLIOGRAFIA 8](#_Toc181011676)

ESPECIFICACION DE LOS TADS

En esta primera parte del programa hemos implementado 3 estructuras basadas en tipos abstractos de datos (TADs); la pila, la cola y la lista.

TADS CREADOS

Pila

**espec** PILA[elemento]

**usa** BOOLEANOS

**parámetro formal**

**géneros** elemento

**fparámetro**

**géneros** pila

**fespec**

Cola

**espec** COLA[ELEMENTO]

**usa** BOOLEANOS

**parametro formal**

generos elemento

**fparametro**

**generos** cola

**fespec**

Lista

**espec** LISTA[ELEMENTO]

**usa** BOOLEANOS

**parámetro** **formal**

**géneros** elemento

**operaciones**

(igualdad entre elementos)

\_eq\_ : elemento elemento 🡪 bool

**géneros** lista

**fespec**

DEFINICION DE LAS OPERACIONES

Pila

**operaciones**

pvacia: 🡪pila {Generadora}

insertar: elemento pila 🡪pila {Generadora}

parcial extraer: pila 🡪pila {Modificadora}

parcial cima: pila 🡪elemento {Observadora}

getLongitud: pila 🡪natural {Observadora}

mostrar: cola 🡪elementos? {Observadora}

vaciar: cola 🡪cola {Modificadora}

**var** p: pila; x: elemento

**ecuaciones de definitud**

Def (desapilar (apilar(x,p)) )

Def (cima (apilar(x,p)) )

**ecuaciones**

desapilar (apilar(x,p)) = p

cima (apilar(x,p)) = x

vacía? (pvacía) = T

vacía? (apilar(x,p)) = F

longitud? (pvacía) = 0

longitud? (apilar (x,p)) = longitud? (p) +1

Cola

**operaciones**

cvacía: 🡪 cola {Generadora}

encolarOrdenado: elemento cola 🡪cola {Generadora}

parcial desencolar: cola 🡪cola {Modificadora}

parcial verPrimero: cola 🡪elemento {Observadora}

esVacía: cola 🡪bool {Observadora}

getLongitud: cola 🡪natural {Observadora}

mostrar: cola 🡪elementos? {Observadora}

vaciar: cola 🡪cola {Modificadora}

**var** c: cola; x: elemento

**ecuaciones de definitud**

Def (eliminar (añadir (x,c)))

Def (primero (añadir (x,c)))

**ecuaciones**

eliminar( añadir(x,cvacía) ) = cvacía

vacía?(c)=F ⇒ eliminar( añadir(x,c) ) = añadir( x,eliminar(c) )

primero( añadir(x,cvacía) ) = x

vacía?(c)=F ⇒ primero( añadir(x,c) ) = primero(c)

vacía?( cvacía ) = T

vacía?( añadir(x,c) ) = F

Lista

**Operaciones**

[]: 🡪lista (Generar lista vacia) {Generadora}

(añadir por la izquierda)

\_:\_: elemento lista 🡪 lista {Generadora}

(añadir por la derecha)

\_#\_: elemento lista 🡪 lista {Generadora}

parcial eliminar: natural lista 🡪lista {Modificadora}

parcial getPrimero: lista 🡪 elemento {Observadora}

parcial getUltimo: lista 🡪elemento {Observadora}

getLongitud: lista 🡪natural {Observadora}

contiene: natural lista 🡪bool {Observadora}

vaciar: lista 🡪lista {Modificadora}

cambiarPrioridad: natural natural lista 🡪lista {Modificadora}

**var** x: elemento; l: lista

**ecuaciones de definitud (**los 5 requieren que la lista no este vacia)

Def (prim (x:l))

Def (ult (x:l))

Def (elpri (x:l))

Def (elult (x:l))

Def (eliminar (x:l))

**ecuaciones**

elpri (x:l) = l

eliminar (x:l) = eliminar (x#l)

vacia? (l) = T 🡪 elult (x:l) = []

vacia? (l) = F 🡪 elult (x:l) = x:elult (l)

prim (x:l) = x

vacia? (l) = T 🡪 ult (x:l) = x

vacia? (l) = F 🡪 ult (x:l) = ult (l)

vacia? ([]) = T

vacia? (x:l) = F

SOLUCIONES ADOPTADAS

Problema de la implementación dinámica

Al inicio al implementar la pila, el atributo "valor", que es lo que guarda la pila, se describió como un objeto, Proceso p, esto causó el problema de que no se había implementado dinámicamente, así que se tuvo que cambiar toda la estructura a puntero a objeto proceso "Proceso\* p, lo cual llevo tiempo. Afortunadamente vimos este error pronto. En el desarrollo de la práctica y para el resto de EEDD se hizo bien desde el principio.

Problema Inicial con el uso de GitHub y GitHub Desktop

Tuvimos algunos problemas al usar la herramienta por varios motivos, primero se nos olvidó añadir el .gitignore en la primera interacción, y al añadirlo más tarde se mandaban archivos como el .mk aunque estuvieran añadidos, segundo teníamos repetido varias veces el nombre de PL1 en distintas carpetas, lo que generaba cierta confusión al programa que si se saltaba una carpeta por llevar el mismo nombre no encontraba los archivos. Además, uno de los miembros utilizaba una versión más antigua del codelite por tenerlo del año pasado mientras que el otro tenía el material reciente, por lo que el traspaso de archivos incluía elementos de diferentes versiones que generaba aún más problemas al compilar. Para arreglarlo, el profesor Carlos nos ayudó, y la solución consistió en mandar primero la eliminación de los archivos que no queríamos que se reenviasen, después se revisó la versión y se retocó el programa de compilación para que no diese problemas de versiones, y por último se creó un nuevo workspace con un nombre distinto para reorganizar de nuevo los archivos y que dejase de generar problemas por la estructura de carpetas.

Problema con la sobre definición de la clase Proceso

La clase proceso al ser común a todas las EEDD que se usan producía errores ya que se declaraba en clases en las que ya estaban declaradas por un import a otra clase que ya tenía un import a esa clase, por lo que con tantas clases con un import a Proceso era difícil ver donde estaba el "doble import proceso".

Duda a la hora de implementar los nombres de procesos

Se dudó sobre cómo hacer el atributo nombre de los procesos, se pensó e intentó hacer con array de char, pero tras intentar implementarlo se hizo con simple string.

MÉTODOS DESTACADOS

Creación de Procesos

La creación de procesos tiene los siguientes procesos:

generarNombre(), elige un nombre entre un array de 10 nombres.

generarBool(), devuelve true o false, para el tipo.

generarPrioridad(), según el tipo hace un cálculo u otro.

Pila

Posee las operaciones básicas de las pilas (Insertar, extraer, cima, getLongitud, vaciar…).

Colas

Posee las operaciones básicas de las colas (encolar, desencolar, vaciar, mostrar, ver primero, getLongitud, esVacia), a parte de estas, posee el método encolarOrdenado, que extrae de las pilas e inserta según prioridad y tipo en su cola correspondiente.

Listas

Posee las operaciones básicas de las listas( insertarInicio, insertarFinal,eliminar,mostrar,getLongitud, getPrimero, getUltimo) a parte de estos, posee los métodos para satisfacer las características de la práctica, estos son:

Eliminar(int PID)-> elimina según un PID introducido por teclado.

Mayor prioridad->muestra el proceso en la lista con mayor prioridad.

Menor prioridad-> lo mismo que el anterior pero con menor prioridad.

busquedaNombre(string nombre)->muestra los procesos creados por un nombre introducido por consola.

cambiarPrioridad(int PID, int Prioridad) -> busca si hay un proceso con el PID introducido y en caso de que exista se le cambia la prioridad.

Gestor

Los métodos de la clase Gestor son básicamente llamadas a otros métodos ya mencionados de la estructura de datos que sea, para realizar la operación que se introduzca por teclado.

COMPORTAMIENTO DEL PROGRAMA

BIBLIOGRAFIA