Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2022.

PROYECTO 1

202002536 – Angela Gabriela Pinelo Flores

Resumen

Dentro de este ensayo se describe una pequeña implementación de tipos de datos abstractos (TDA) en el Proyecto No.1.

En este proyecto se implementó en especial la lista enlazada doble (descrita posteriormente). Esta estructura como el resto nos permite modelar nuestra propia estructura a un fin específico.

Durante todo el proyecto para poder implementar las estructuras de TDA fue necesaria la Programación Orientada a Objetos ya que esta nos brinda mayor orden y una mejor comprensión para poder comprender el flujo de los datos.

Palabras clave
Flujo de datos, TDA

Listas Enlazadas, Apuntadores

Abstract

Within this essay a small implementation of abstract data types (ADT) in Project No.1 is described.

The double linked list (described later) was especially implemented in this project. This structure, like the rest, allows us to model our own structure for a specific purpose.

Throughout the project, in order to implement the TDA structures, Object Oriented Programming was necessary since it provides us with greater order and better understanding in order to understand the flow of data.

Keywords
Data Flow, pointers, linked lists,, ADT

Introducción

Un tipo de dato abstracto es un conjunto de datos u objetos creado de manera personalizada por un programador para un fin específico.

Un TDA es una abstracción que permite modelar características de un elemento en particular

un tipo de dato abstracto se puede manipular de forma similar a los tipos de datos que están predefinidos dentro del lenguaje de programación, encapsulando más información según se requiera.

La implementación de un tipo de dato abstracto depende directamente del lenguaje de programación que se utilice

En las siguientes páginas se explica la estructura de un tipo de dato abstracto que fue implementada dentro del proyecto

Desarrollo del tema

Para poder comprender la estructura de un tipo de dato abstracto es necesario que conozcamos o tengamos conocimientos acerca de la programación orientada a objetos ya que esto nos permitirá comprender de mejor manera la estructura que se implementa dentro de un TDA.

habiendo tomado en consideración lo anterior se introduce un concepto vital para la comprensión del tema: **Nodo**, un nodo cumple la función de guardar la información de un elemento dentro de nuestra lista y la lista no es más que un conjunto de nodos.

Cuando implementamos una lista debe de haber como mínimo un apuntador, un apuntador hace referencia a una posición en específico en nuestra lista, es decir, podemos tener la cantidad de apuntadores que requiramos por ejemplo al principio y al final también en medio, pero como mínimo debemos de contar con un apuntador.

Los **apuntadores** guardan un espacio en memoria y nos permiten almacenar un nodo de manera temporal haciendo referencia a él y a su información dependiendo del tipo de apuntador que definamos este puede ser el principio o el final de nuestra lista

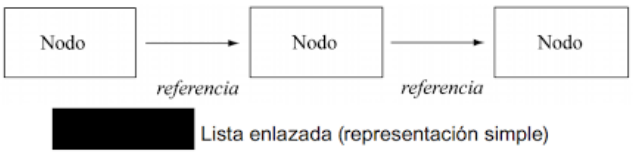


Figura 1. Representación de apuntadores en nodos

Los apuntadores no solo pueden estar en la lista, estos también pueden estar incluidos dentro de nuestros nodos para hacer referencia a un nodo posterior o un nodo anterior.

Existen varios tipos de listas dentro de este proyecto se utilizó la lista enlazada doble.

la lista enlazada doble implementada dentro de este proyecto cuenta con un apuntador a la cabecera de la lista es decir el inicio de la lista y un apuntador hacia el final de nuestra lista, nuestros nodos cuentan con un apuntador hacia el nodo anterior y hacia el nodo

siguiente y estos apuntadores dentro de nuestros nodos es lo que la hace una lista enlazada doble ya que se conecta de principio a fin y de fin a principio.

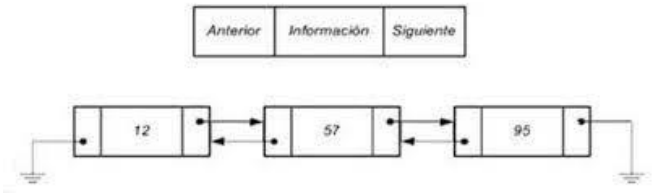


Figura 2. Representación de lista enlazada doble

Dentro del proyecto fueron implementadas 3 listas doblemente enlazadas:

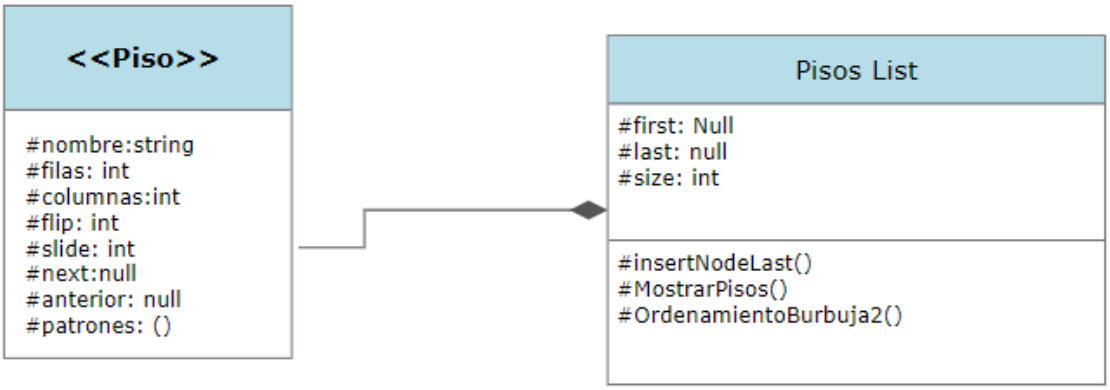
- Lista pisos
- Lista de patrones
- Lista de Casillas

Cada una de estas listas fue realizada con una clase y cada una posee otra clase de tipo nodo que nos funciona de plantilla para todos los nodos implementados dentro de cada una de estas. por eso que la programación orientada a objetos es la utilidad para poder construir un tipo de dato abstracto.

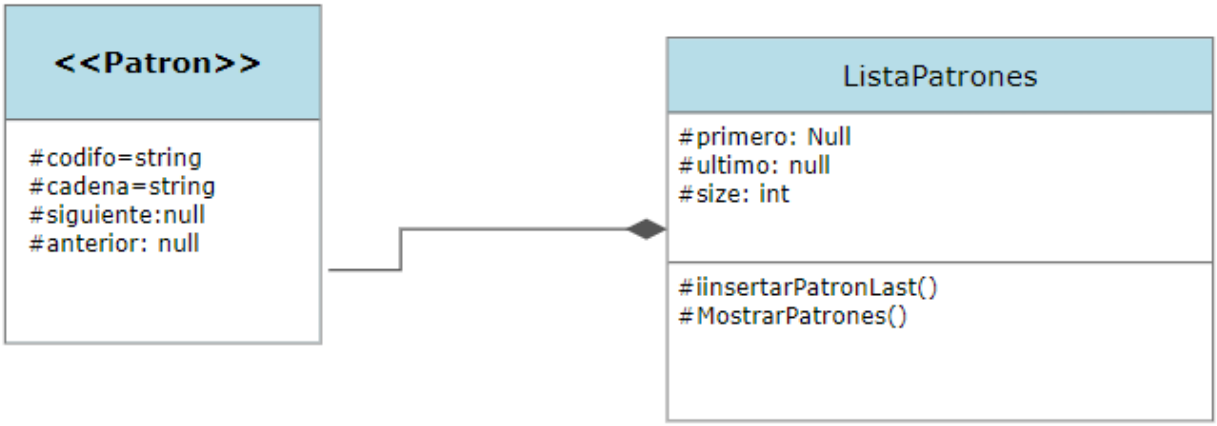
Para comprender de mejor manera la estructura de nuestro proyecto ver en apéndices los diagramas de clase de cada una de nuestras listas y nodos.

DIAGRAMAS

- **Pisos**



- **Patrones**



- **Casillas**