Clase Nº 11

Tipos Abstractos de datos (TDA)

Concepto

Desde un primer momento hemos venido trabajando con datos almacenados en variables simples. Posteriormente agregamos el concepto de *estructura de datos,* representado por las listas, tuplas, conjuntos y diccionarios.

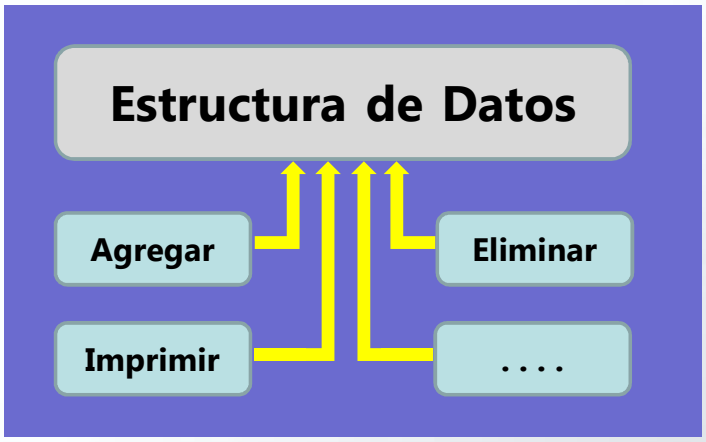
Concepto

- Recordemos que una estructura de datos consiste en datos organizados en algún tipo de estructura, la que le confiere características propias.
- Por ejemplo, la lista organiza los valores en una secuencia, permite duplicados y sus elementos son accesibles en forma directa a través de un subíndice.

Concepto

- Un ***tipo abstracto de datos (TDA)*** es una estructura de datos a la que se le añaden operaciones, como agregar elementos, borrarlos, imprimirlos, etc.
- En un TDA los datos de la estructura sólo pueden ser manipulados a través de estas operaciones, lo que permite mayor control sobre las tareas que se pueden realizar sobre los mismos.

Concepto



Concepto

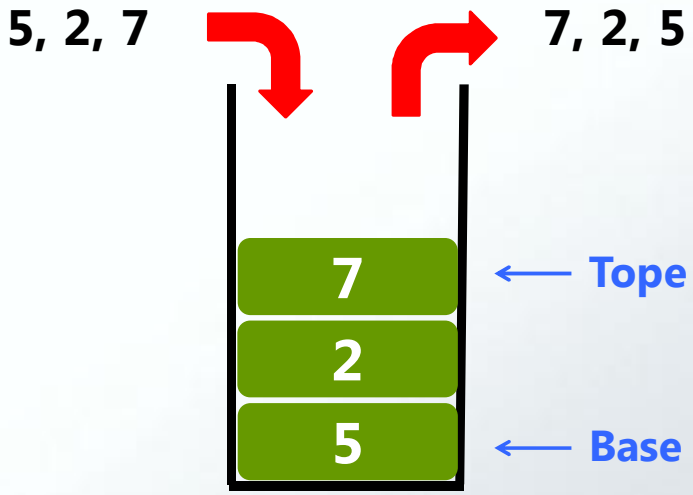
Aunque los TDA serán tratados en detalle en Programación II, en esta materia comenzaremos su estudio con dos TDA básicos:

La *pila* (stack) y la *cola* (queue).

TDA Pila

- Una pila es una estructura de datos en la que los ingresos y egresos de datos se realizan por un único extremo.
- Por esa razón el último elemento ingresado será el primero en salir, lo que se conoce como sistema LIFO (Last In, First Out).

TDA Pila



TDA Pila

TDA Pila

Vamos a implementar cinco operaciones dentro del TDA Pila:

- **inicializarpila(): Crea la pila y la prepara para ser utilizada.**
- **apilar(<pila>, <dato>): Agrega <dato> en la pila <pila>.**

TDA Pila

- **desapilar(<pila>):** Elimina el tope de la pila, el decir el último elemento agregado. Provoca un error si la pila está vacía.
- **tope(<pila>):** Devuelve el valor del tope de la pila, sin eliminarlo. Provoca un error si la pila está vacía.

TDA Pila

- **pilavacia(<pila>):** Devuelve True si la pila está vacía, o False en caso contrario. No informa la cantidad de elementos.

TDA Pila

Vamos a implementar estas operaciones a través de funciones, utilizando una lista de Python como soporte para los datos.

TDA Pila

```
def inicializarpila( ):
    pila = [ ]
    return pila

def apilar(pila, dato):
    pila.append(dato)
```

TDA Pila

```
def desapilar(pila):
    if len(pila)>0:
        pila.pop()
    else:
        raise ValueError("Pila vacía")
```


TDA Pila

```
def tope(pila):
    if len(pila)>0:
        return pila[-1]
    else:
        raise ValueError("Pila vacía")
```

TDA Pila

```
def pilavacia(pila):
    return len(pila)==0
```


Ejemplo 1

Cargar desde el teclado la pila DADA con al menos un elemento. Luego pasar el último elemento (el de la base) de la pila DADA a su primera posición (tope), dejando los restantes elementos en el mismo orden.

```
from pilacola import *

# Cargar DADA con al menos un elemento
dada = inicializarpila()
while pilavacia(dada):
    e = int(input("Ingresá un elemento, o -1 para terminar: "))
    while e!=-1:
        apilar(dada, e)
        e = int(input("Ingresá un elemento, o -1 para terminar: "))
```



```
# Pasamos todos los elementos a AUX1 para obtener la base
aux1 = inicializarpila()
while not pilavacia(dada):
    apilar(aux1, tope(dada))
    desapilar(dada)
# Separamos el tope de AUX1, es decir la base de DADA
baseoriginal = tope(aux1)
desapilar(aux1)
```

```
# Y regresamos los demas elementos a DADA
while not pilavacia(aux1):
    apilar(dada, tope(aux1))
    desapilar(aux1)
# Apilamos lo que era la base de DADA como nuevo tope
apilar(dada, baseoriginal)
# Imprimir DADA
while not pilavacia(dada):
    print(tope(dada), end=" ")
    desapilar(dada)
```


Programa completo

```
from pilacola import *

dada = inicializarpila()
while pilavacia(dada):
    e = int(input("Ingresá un elemento, o -1 para terminar: "))
    while e!=-1:
        apilar(dada, e)
        e = int(input("Ingresá un elemento, o -1 para terminar: "))
aux1 = inicializarpila()
# Pasamos todos los elementos de DADA a AUX1 para obtener su base
while not pilavacia(dada):
    apilar(aux1, tope(dada))
    desapilar(dada)
# Separamos el tope de AUX1, es decir la base de DADA
baseoriginal =  tope(aux1)
desapilar(aux1)
# Y regresamos los demas elementos a DADA
while not pilavacia(aux1):
    apilar(dada, tope(aux1))
    desapilar(aux1)
# Por último, pasamos lo que era la base de DADA como nuevo tope
apilar(dada, baseoriginal)
# Imprimir DADA
while not pilavacia(dada):
    print(tope(dada), end=" ")
    desapilar(dada)
print()
```

Ejemplo de ejecución:

```
Ingresá un elemento, o -1 para terminar:  4
Ingresá un elemento, o -1 para terminar:  7
Ingresá un elemento, o -1 para terminar:  2
Ingresá un elemento, o -1 para terminar: -1
```

4 2 7

TDA Cola

- Una cola es una estructura de datos en la que los ingresos se realizan por extremo, y los egresos por el otro.
- Por esa razón el primer elemento ingresado será también el primero en salir, lo que se conoce como sistema FIFO (First In, First Out).

TDA Cola

5, 2, 7

5, 2, 7

Frente

Fondo

TDA Cola

TDA Cola

Las operaciones a implementar dentro del TDA Cola serán:

- **inicializarcola():** Crea la cola y la prepara para ser utilizada.
- **acolar(<cola>, <dato>):** Agrega <dato> en la cola <cola>.

TDA Cola

- **desacolar(<cola>):** Elimina el frente de la cola, el decir el primer elemento agregado. Provoca un error si la cola está vacía.
- **primero(<cola>):** Devuelve el valor del frente de la cola, sin eliminarlo. Provoca un error si la cola está vacía.

TDA Cola

- **colavacia(<cola>): Devuelve True si la cola está vacía, o False en caso contrario. No informa la cantidad de elementos.**

TDA Cola

Vamos a implementar estas operaciones a través de funciones, utilizando una lista de Python como soporte para los datos.

TDA Cola

```
def inicializarcola( ):
    cola = [ ]
    return cola

def acolar(cola, dato):
    cola.append(dato)
```


TDA Cola

```
def desacolar(cola):
    if len(cola)>0:
        cola.pop(0)
    else:
        raise ValueError("Cola vacía")
```

TDA Cola

```
def primero(cola):
    if len(cola)>0:
        return cola[0]
    else:
        raise ValueError("Cola vacía")
```


TDA Cola

```
def colavacia(cola):
    return len(cola)==0
```

Ejemplo 2

Cargar desde el teclado la cola DADA con al menos un elemento. Luego pasar el último elemento de DADA a la primera posición, dejando los restantes elementos en el orden original.


```
from pilacola import *

# Cargar DADA con al menos un elemento
dada = inicializarcola()
while colavacia(dada):
    e = int(input("Ingresá un elemento, o -1 para terminar: "))
    while e!=-1:
        acolar(dada, e)
        e = int(input("Ingresá un elemento, o -1 para terminar: "))
```

```
# Hacemos una copia de DADA
aux = inicializarcola()
copia = inicializarcola()
while not colavacia(dada):
    acolar(aux, primero(dada))
    desacolar(dada)
while not colavacia(aux):
    acolar(copia, primero(aux))
    acolar(dada, primero(aux))
    desacolar(aux)
```



```
# Ahora eliminamos un elemento de COPIA
desacolar(copia)
# Y rotamos DADA tantas veces como indique COPIA
# Al tener un elemento menos, el fondo quedará como frente
while not colavacia(copia):
    acolar(dada, primero(dada))
    desacolar(dada)
    desacolar(copia)
```

```
# Imprimir DADA
print()
while not colavacia(dada):
    print(primero(dada), end=" ")
    desacolar(dada)
print()
```


Programa completo

```
from pilacola import *

# Cargar DADA con al menos un elemento
dada = inicializarcola()
while colavacia(dada):
    e = int(input("Ingresá un elemento, o -1 para terminar: "))
    while e!=-1:
        acolar(dada, e)
        e = int(input("Ingresá un elemento, o -1 para terminar: "))
# Hacemos una copia de DADA
aux = inicializarcola()
copia = inicializarcola()
while not colavacia(dada):
    acolar(aux, primero(dada))
    desacolar(dada)
while not colavacia(aux):
    acolar(copia, primero(aux))
    acolar(dada, primero(aux))
    desacolar(aux)
# Ahora eliminamos un elemento de COPIA
desacolar(copia)
# Y rotamos DADA tantas veces como indique COPIA
# Al tener un elemento menos, el fondo quedará como frente
while not colavacia(copia):
    acolar(dada, primero(dada))
    desacolar(dada)
    desacolar(copia)
# Imprimir DADA
print()
while not colavacia(dada):
    print(primero(dada), end=" ")
    desacolar(dada)
print()
```

Ejemplo de ejecución:

```
Ingresá un elemento, o -1 para terminar:  4
Ingresá un elemento, o -1 para terminar:  5
Ingresá un elemento, o -1 para terminar:  0
Ingresá un elemento, o -1 para terminar:  1
Ingresá un elemento, o -1 para terminar: -1
```

1 4 5 0

ATENCIÓN

Los TDA sólo puden ser manipulados a través de las funciones primitivas implementadas dentro de los mismos.

No es posible acceder a la lista que sirve como soporte.

ATENCIÓN

```
elementos = len(dada)

for dato in dada:
    [ ... ]

print(dada)
```

INCORRECTO

Ejercitación

- **Práctica 9: Completa**

Trabajo Práctico 9
Ejercitación por equipos

Tomar el número del grupo y calcular el resto de dividirlo por 3.

- **Resto 0: Ejercicios 1, 5 y 10**
- **Resto 1: Ejercicios 3, 7 y 8**
- **Resto 2: Ejercicios 2, 4 y 9**

© Lic. Ricardo Thompson