UNRaf

UNIVERSIDAD NACIONAL DE RAFAELA

Bv. J.A. Roca 989 / CP: 2300
Rafaela - Santa Fe - Argentina

T: +54 (03492) **501155**

info@unraf.edu.ar
www.unraf.edu.ar

#IC

Ingeniería en Computación

ALGORITMOS Y ESTRUCTURAS DE DATOS

ALGORITMOS Y ESTRUCTURAS DE DATOS

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

AGENDA

#IC

- Repaso TDA
- TDA
  - Pilas
  - Colas
  - Tablas Hash
- Ejemplos
- Ejercicios

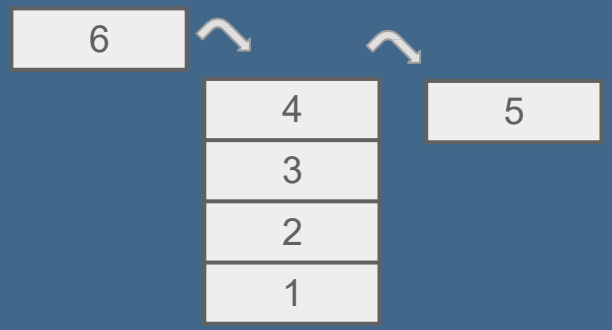
UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

PILAS

#IC

- **Las pilas son colecciones de objetos con reglas específicas para agregar y quitar nuevos objetos.**

- **Las pilas siguen el principio Last in, First out o LIFO.**



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

MÉTODOS EN PILAS

#IC

empty: devuelve True si la pila no contiene ningún elemento

push: agrega un elemento al final de la pila.

pop: elimina y devuelve el elemento del final de la pila

top: devuelve una referencia al último elemento de la pila sin eliminarlo

len: devuelve el número de elementos de la pila

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

IMPLEMENTACIÓN DE PILAS

#IC

```
class Pila:
    def __init__ (self):
        self._data = []

    def len (self):
        return len(self._data)

    def empty(self):
        if len(self._data) == 0:
            return True

    def push(self, e):
        self._data.append(e)

    def top(self):
        if self.empty():
            raise IndexError('Pila vacía')
        return self._data[-1]

    def pop(self):
        if self.empty():
            raise IndexError('Pila vacía')
        return self._data.pop()
```

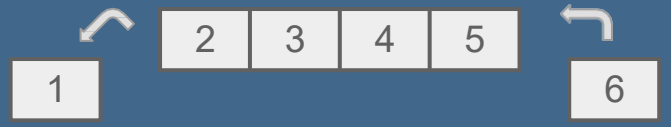
UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

COLAS

#IC

Las colas son colecciones de objetos con reglas específicas para agregar y quitar nuevos objetos pero con una diferencia de las Pilas.

En este caso, el funcionamiento sigue el principio First in, first out o FIFO.



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

MÉTODOS EN COLAS

#IC

- **empty: devuelve True si la cola no contiene ningún elemento.**
- **enqueue: agrega un elemento al final de la cola.**
- **dequeue: elimina y devuelve el primer elemento de la cola.**
- **first: devuelve una referencia al primer elemento de la cola sin eliminarlo.**
- **len: devuelve el número de elementos de la cola.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

IMPLEMENTACIÓN DE COLAS

#IC

```
class Cola:
    def __init__ (self):
        self._data = []

    def len (self):
        return len(self._data)

    def empty(self):
        if len(self._data) == 0:
            return True

    def enqueue(self, e):
        self._data.append(e)

    def dequeue(self):
        if self.empty():
            raise IndexError('Cola vacía')
        return self._data.pop(0)

    def first(self):
        if self.empty():
            raise IndexError('Cola vacía')
        return self._data[0]
```

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

COLAS DOBLES

#IC

Estructura de datos que permiten:

- **añadir**

y

- **eliminar objetos por ambos extremos.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

MÉTODOS EN COLAS DOBLES

#IC

- **add_first (e):** añade un elemento al comienzo de la cola doble.
- **add_last (e):** añade un elemento al final de la cola doble.
- **delete_first (e):** elimina el elemento al comienzo de la cola doble.
- **delete_last (e):** elimina el elemento al final de la cola doble.
- **first():** devuelve, sin eliminar, el primer elemento de la cola doble.
- **last():** devuelve, sin eliminar, el último elemento de la cola doble.
- **empty( ):** devuelve True si la estructura está vacía.
- **len(D):** devuelve el número de elementos en la cola doble.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

IMPLEMENTACIÓN DE COLAS DOBLES

#IC

```
class ColaDoble:
    def __init__(self):
        self._data = []

    def len(self):
        return len(self._data)

    def empty(self):
        return len(self._data) == 0

    def add_first(self, e):
        self._data.insert(0, e)

    def add_last(self, e):
        self._data.append(e)

    def delete_first(self):
        if self.empty():
            raise IndexError("Cola vacía")
        return self._data.pop(0)
```

```
    def delete_last(self):
        if self.empty():
            raise IndexError("Cola vacía")
        return self._data.pop()

    def first(self):
        if self.empty():
            raise IndexError("Cola vacía")
        return self._data[0]

    def last(self):
        if self.empty():
            raise IndexError("Cola vacía")
        return self._data[-1]
```

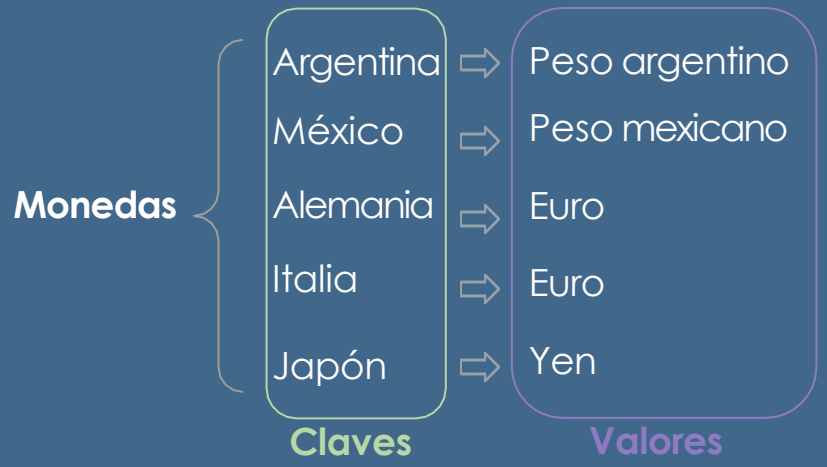
UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

TABLAS HASH

#IC

- **Se basa en el almacenamiento de pares de datos relacionados, denominados clave y valor.**

- **Cada clave es un valor único, que tiene asociado un valor (los valores pueden repetirse).**



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

TABLAS HASH – DICT PYTHON

#IC

- **Para definir una tabla hash se usa la clase dict y se utilizan llaves.**
- **Para acceder a valores dentro de un diccionario se utiliza una sintaxis similar a la de las listas.**
- **Las claves en un diccionario pueden ser cualquier tipo de datos inmutable, la única restricción es que cada clave debe ser un valor único.**

```
monedas = {}
monedas["Argentina"] = "Peso"
monedas["Japon"] = "Yen"
print(monedas["Argentina"])
for m in monedas:
  print(monedas[m])
```

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

VENTAJAS

#IC

- Los valores dentro de un diccionario pueden ser de cualquier tipo, incluidos otros datos abstractos (como diccionarios).

- Al igual que acceder a un elemento en una lista usando un índice, el acceso a un elemento usando su clave se da, en promedio, en tiempo constante.

- Esto es posible gracias al uso de la función hash.

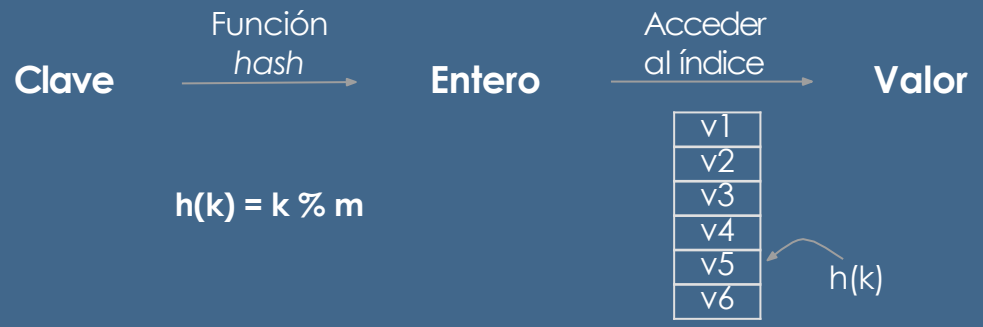
UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

FUNCIÓN HASH

#IC

- **El objetivo de esta función es mapear cada clave con un entero.**

- **Si conceptualizamos a los pares clave-valor de un diccionario ubicados dentro de una lista, el objetivo de la función hash calculada sobre la clave es obtener índice para el valor correspondiente.**



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

FUNCIÓN HASH

#IC

- Esta función asigna un entero a cualquier clave que se quiera ingresar.
- A priori, la cantidad de claves que se pueden usar es infinita.
- La lista donde se almacenan los valores no puede ser de tamaño infinito para almacenar todos los posibles índices de todas las claves posibles.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

COLISIONES

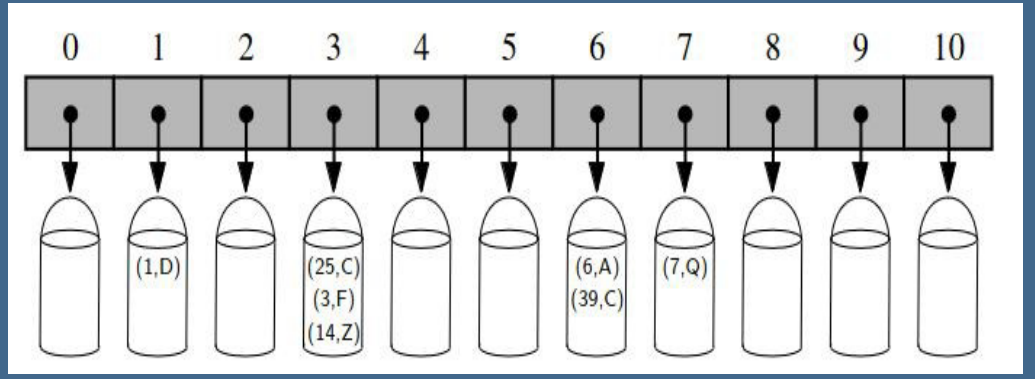
#IC

Supongamos que tenemos los siguientes pares de datos:

1,D 3,F 6,A 7,Q 14,Z 25,C 39,C

Y usando el modulo (%) como función hash para armar una lista de 11 elementos se obtiene lo siguiente:

h(1)=1 h(3)=3 h(6)=6 h(7)=7 h(14)=3 h(25)=3 h(39)=6

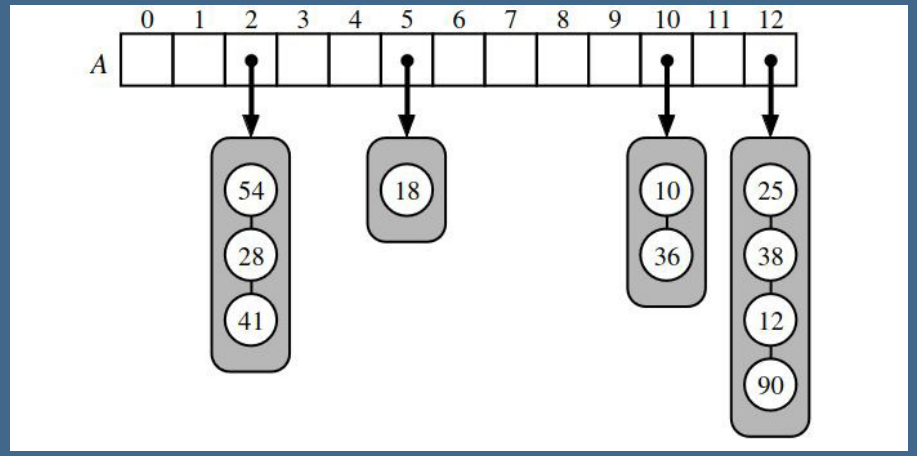


UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN A COLISIONES

#IC

- **Almacenar en el índice de la lista donde se produce la colisión una lista, con valores (k, v).**
- **Si al calcular la función hash el valor dentro de la lista de valores es esta segunda lista, se busca la clave deseada dentro de estos valores. Estrategia conocida como separate chaining o encadenamiento separado.**



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN A COLISIONES

#IC

La mayor desventaja del encadenamiento es la utilización de espacio extra para las listas y de tiempo extra para su recorrido.

Una serie de estrategias alternativas se denomina direccionamiento abierto.

El ejemplo más simple es el sondeo lineal, donde se ocupa el siguiente elemento de la lista, si el correspondiente está ocupado

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN A COLISIONES

#IC

- En el sondeo lineal, el espacio ocupado por un valor se puede calcular como h(k) + i, siendo i = 0, 1, 2, 3, …

- Otras alternativas para solucionar colisiones son:
  - El sondeo cuadrático, similar al sondeo lineal, pero i toma los valores 12, 22, 32, 42.
  - El doble hasheo, donde se diseña una segunda función hash para asignar un nuevo índice dentro de la lista de claves-valores.

- Todas estas funciones intentan solucionar el problema de las colisiones, pero aumentan la complejidad de la búsqueda dentro de un diccionario.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

DICCIONARIOS

#IC

- Python tiene incorporada la función hash, que permite calcular un valor entero a partir de un elemento inmutable.

- Esta función devuelve números flotantes (incluidos negativos), por lo que la clase dict incluye una segunda función, denominada de compresión, para transformar ese flotante en entero positivo.

- Para definir un diccionario, se utilizan las llaves ({ y }), y los pares clave-valor se separan con dos puntos (:)

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

EJERCICIOS

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 1

#IC

Gestión de Navegación en un Navegador Web

Un navegador web permite ir a páginas anteriores y regresar a páginas recientes. Cuando el usuario pulsa “atrás”, puede volver a la página anterior, y cuando pulsa “adelante”, puede regresar a la última página visitada.

¿Qué estructura de datos utilizarías para gestionar la funcionalidad de “atrás” y “adelante”?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 1

#IC

Usar dos pilas:

- **Una pila para gestionar el historial de páginas anteriores.**

- **Una segunda pila para almacenar las páginas "hacia adelante" cuando se pulsa "atrás".**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 2

#IC

Sistema de Impresión de Documentos

En una oficina, se ha implementado un sistema de impresión de documentos en el que los documentos se envían a la impresora en el orden en que se reciben. La impresora debe procesar los documentos en el mismo orden en que llegaron, sin importar su tamaño.

¿Qué estructura de datos sería más eficiente para gestionar la cola de documentos pendientes de impresión?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 2

#IC

Usar una cola (queue):

- **La cola permite que el primer documento en entrar sea el primero en procesarse (FIFO - First In, First Out).**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 3

#IC

Verificación de Expresiones Matemáticas

Se necesita un programa que verifique si una expresión matemática tiene los paréntesis correctamente balanceados. Por ejemplo, la expresión (a + b) * (c + d) está balanceada, mientras que (a + b) * c + d) no lo está.

¿Qué estructura de datos es adecuada para verificar que todos los paréntesis estén correctamente balanceados?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 3

#IC

Usar una pila (stack):

- **Cada vez que se encuentra un paréntesis de apertura (, se añade a la pila.**
- **Al encontrar un paréntesis de cierre), se elimina un paréntesis de apertura de la pila.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 4

#IC

Línea de Atención al Cliente

En una empresa de telecomunicaciones, los clientes llaman para solicitar asistencia técnica y se deben atender en el orden en que llaman. Cuando un agente está disponible, atiende al primer cliente en espera.

¿Qué estructura de datos utilizarías para organizar a los clientes que esperan ser atendidos?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 4

#IC

Usar una cola (queue):

- **Se agrega cada cliente al final de la cola cuando llama.**
- **Cuando un agente está disponible, se atiende al cliente al frente de la cola.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 5

#IC

Sistema de Deshacer y Rehacer en un Editor de Texto

En un editor de texto, los usuarios deben poder deshacer y rehacer acciones. Cada vez que el usuario realiza una acción (escribe, borra texto, etc.), esta debe poder revertirse si el usuario pulsa “deshacer” y restaurarse si pulsa “rehacer”.

¿Qué estructura(s) de datos utilizarías para gestionar las acciones de “deshacer” y “rehacer”?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 5

#IC

Usar dos pilas:

- **La pila de "deshacer" almacena las acciones en el orden en que se realizan.**
- **Al pulsar "deshacer", se saca la última acción de la pila de "deshacer" y se pasa a la pila de "rehacer".**
- **Al pulsar "rehacer", se saca la última acción de la pila de "rehacer" y se vuelve a realizar.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 6

#IC

Aplicación de Control de Tráfico Aéreo

Un aeropuerto recibe solicitudes de despegue de diferentes aviones. Cada avión debe esperar su turno para despegar según el orden en que llegó la solicitud de despegue.

¿Qué estructura de datos utilizarías para gestionar la cola de aviones que esperan para despegar?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 6

#IC

Usar una cola (queue):

- **Cada avión que solicita despegue se añade al final de la cola.**
- **El primer avión en la cola será el siguiente en despegar cuando la pista esté libre.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 7

#IC

Gestión de una Playlist en una Aplicación de Música

En una aplicación de música, los usuarios pueden añadir canciones a la cola de reproducción. La aplicación debe reproducir las canciones en el orden en que fueron añadidas. Además, los usuarios pueden mover una canción al principio de la cola en cualquier momento.

¿Qué estructura de datos utilizarías para gestionar la cola de reproducción y permitir cambios de posición?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 7

#IC

Usar una lista doblemente enlazada:

- **Las canciones se añaden al final de la lista en el orden en que fueron seleccionadas.**
- **Si una canción se mueve al principio, el enlace del nodo de la canción se ajusta para colocarse al inicio.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 8

#IC

Sistema de Control de Ingreso de Empleados

En una empresa, los empleados ingresan y salen de forma secuencial por la entrada principal. Un sistema de control debe registrar el orden de entrada y salida de los empleados para verificar la hora de ingreso y egreso de cada uno.

¿Qué estructura de datos utilizarías?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 8

#IC

Usar una cola (queue):

- **Los empleados se registran en el orden en que ingresan y salen.**
- **Cada empleado que ingresa se añade al final de la cola, y al salir, se remueve del frente de la cola.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 9

#IC

Guardado de Estados de Juego en una Aplicación

Un videojuego permite a los usuarios guardar diferentes estados de su progreso. Cuando el jugador decide cargar un estado guardado, debe poder seleccionar el último guardado o cualquier estado anterior.

¿Qué estructura de datos utilizarías?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 9

#IC

Usar una pila (stack):

- **Cada vez que el jugador guarda el estado, este se apila en la estructura de guardados.**
- **Al cargar, puede seleccionar el último estado guardado o "deshacer" los avances uno por uno.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 10

#IC

Realización de Encuestas en Línea

Una empresa de investigación debe enviar una serie de preguntas a los participantes en una encuesta en línea. Las preguntas deben aparecer una tras otra, permitiendo que el usuario avance o regrese a una pregunta anterior.

¿Qué estructura de datos utilizarías?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 10

#IC

Usar una lista doblemente enlazada:

- **Cada pregunta es un nodo en la lista, y el usuario puede moverse adelante o atrás en la secuencia.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 11

#IC

Procesamiento de Datos en un Sensor de Temperatura

Un sensor de temperatura mide continuamente la temperatura ambiente y almacena los últimos 60 valores (uno por segundo) en un sistema. Cuando se alcanza el límite, se sobrescriben los datos más antiguos.

¿Qué estructura de datos utilizarías?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 11

#IC

Usar una cola circular:

- **La cola circular almacena los 60 valores más recientes, sobrescribiendo el valor más antiguo con el más nuevo.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

EJERCICIO 12

#IC

Gestión de un Carrito de Compras en Línea

Un sistema de tienda en línea debe implementar un carrito de compras para cada usuario. El carrito debe permitir, añadir productos, eliminar productos, actualizar la cantidad de un producto ya añadido, mostrar todos los productos en el orden en que fueron añadidos. Cada producto tiene una identificación única (ID) y una cantidad específica seleccionada por el usuario.

¿Qué estructura(s) de datos utilizarías para gestionarlo?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

SOLUCIÓN PROPUESTA EJERCICIO 12

#IC

Usar una lista enlazada y un diccionario (hash table) con cada producto como un nodo o entrada en el diccionario:

- **Lista enlazada: Permitiría que los productos se añadan en el orden en que se ingresaron y que el usuario recorra el carrito fácilmente para ver su contenido.**

- **Diccionario (hash table): Con el ID del producto como clave y la cantidad como valor, junto con la información adicional como nombre y precio.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

UNRaf

UNIVERSIDAD
NACIONAL DE
RAFAELA

Bv. J.A. Roca 989 / CP: 2300
Rafaela - Santa Fe - Argentina

T: +54 (03492) **501155**

info@unraf.edu.ar
www.unraf.edu.ar