PREGRADO

UNIDAD 1 | INTRODUCCIÓN A ESTRUCTURAS DE DATOS

PILAS Y COLAS

SEMANA 5

S1AS10SI393 | Fundamentos de Sistemas de Información

Al finalizar la unidad, el estudiante implementa aplicaciones en entorno visual teniendo en cuenta la Programación orientada a objetos y haciendo un uso de estructura de datos

AGENDA

- Definición.
- Clase Stack (Push. Pop. Peek. IsEmpty).
- Clase Queue (Enqueue, Dequeue. Peek. IsEmpty)

Definición de una cola

```
class persona
{
    public string nombre;
    public string apellido;
    public int edad;
};
0 referencias
class Program
{
    0 referencias
    static void Main()
    {
        Queue<string> clientesEspera = new Queue<string>();
        Queue<persona> colaPersona = new Queue<persona>();

        clientesEspera.Enqueue("Juan");
        clientesEspera.Enqueue("Ana");
        clientesEspera.Enqueue("Pepe");
        clientesEspera.Enqueue("Maria");
        clientesEspera.Enqueue("Sonia");

        colaPersona.Enqueue(new persona { nombre = "Juan", apellido = "Perez", edad = 20 });
        colaPersona.Enqueue(new persona { nombre = "Ana", apellido = "Torres", edad = 21 });
        colaPersona.Enqueue(new persona { nombre = "Pepe", apellido = "Veliz", edad = 19 });
        colaPersona.Enqueue(new persona { nombre = "Maria", apellido = "Salas", edad = 22 });
        colaPersona.Enqueue(new persona { nombre = "Sonia", apellido = "Caceres", edad = 21 });
    }
}
```

- Son estructuras utilizadas muy a menudo como herramientas de programación de tipo FIFO (First in-First out).
- Uso común:
  - ❖ Administración de procesos
  - ❖ Impresión en impresoras
  - ❖ Sistemas de atención al cliente

El método **Enqueue()** encola un elemento a una estructura de colas

Recorrido de una Cola

Se puede recorrer sin modificarla, empleando el bucle foreach

```
Queue<string> clientesEspera = new Queue<string>();
Queue<persona> colaPersona = new Queue<persona>();

clientesEspera.Enqueue("Juan");
clientesEspera.Enqueue("Ana");
clientesEspera.Enqueue("Pepe");
clientesEspera.Enqueue("Maria");
clientesEspera.Enqueue("Sonia");

colaPersona.Enqueue(new persona { nombre = "Juan", apellido = "Perez", edad = 20 });
colaPersona.Enqueue(new persona { nombre = "Ana", apellido = "Torres", edad = 21 });
colaPersona.Enqueue(new persona { nombre = "Pepe", apellido = "Veliz", edad = 19 });
colaPersona.Enqueue(new persona { nombre = "Maria", apellido = "Salas", edad = 22 });
colaPersona.Enqueue(new persona { nombre = "Sonia", apellido = "Caceres", edad = 21 });


foreach (string cliente in clientesEspera)
{
    Console.WriteLine(cliente);
}

foreach (persona cliente in colaPersona)
{
    Console.WriteLine($"{cliente.nombre} " +
                      $"{cliente.apellido} " +
                      $"({cliente.edad} años)");
}
```

Métodos de las colas → Count(), Peek()

```
colaPersona.Enqueue(new persona { nombre = "Juan", apellido = "Perez", edad = 20 });
colaPersona.Enqueue(new persona { nombre = "Ana", apellido = "Torres", edad = 21 });
colaPersona.Enqueue(new persona { nombre = "Pepe", apellido = "Veliz", edad = 19 });
colaPersona.Enqueue(new persona { nombre = "Maria", apellido = "Salas", edad = 22 });
colaPersona.Enqueue(new persona { nombre = "Sonia", apellido = "Caceres", edad = 21 });

Console.WriteLine($"la cola tiene: {colaPersona.Count()} personas en espera");
```

El método **Count(),** devuelve el numero de elementos que están en la cola

```
clientesEspera.Enqueue("Juan");
clientesEspera.Enqueue("Ana");
clientesEspera.Enqueue("Pepe");
clientesEspera.Enqueue("Maria");
clientesEspera.Enqueue("Sonia");

Console.WriteLine($"el turno de atencion es para {clientesEspera.Peek()} ");
```

```
colaPersona.Enqueue(new persona { nombre = "Juan", apellido = "Perez", edad = 20 });
colaPersona.Enqueue(new persona { nombre = "Ana", apellido = "Torres", edad = 21 });
colaPersona.Enqueue(new persona { nombre = "Pepe", apellido = "Veliz", edad = 19 });
colaPersona.Enqueue(new persona { nombre = "Maria", apellido = "Salas", edad = 22 });
colaPersona.Enqueue(new persona { nombre = "Sonia", apellido = "Caceres", edad = 21 });

Console.WriteLine($"el turno de atencion es para {colaPersona.Peek().nombre} " +
                  $"{colaPersona.Peek().apellido}");
```

El método **Peek(),** devuelve el primer elemento de la Cola.

Métodos de las colas → Dequeue()

```
colaPersona.Enqueue(new persona { nombre = "Juan", apellido = "Perez", edad = 20 });
colaPersona.Enqueue(new persona { nombre = "Ana", apellido = "Torres", edad = 21 });
colaPersona.Enqueue(new persona { nombre = "Pepe", apellido = "Veliz", edad = 19 });
colaPersona.Enqueue(new persona { nombre = "Maria", apellido = "Salas", edad = 22 });
colaPersona.Enqueue(new persona { nombre = "Sonia", apellido = "Caceres", edad = 21 });

colaPersona.Dequeue();
Console.WriteLine($"el turno de atencion es para {colaPersona.Peek().nombre} " +
                  $"{colaPersona.Peek().apellido}");
Console.WriteLine($"la cola tiene: {colaPersona.Count()} personas en espera");
```

El método **Dequeue(),** retira el primer elemento de la cola.

Se puede comprobar si la cola está vacía o esta llena empleando el método **Count()**

```
class persona
{
    public string nombre;
    public string apellido;
    public int edad;

    1 referencia
    public bool IsEmpty(Queue<persona> cola)
    {
        return cola.Count == 0;
    }
    1 referencia
    public bool IsFull(Queue<persona> cola, int capacidad)
    {
        return cola.Count >= capacidad;
    }
};
```

```
if (objPersona.IsEmpty(colaPersona))
    Console.WriteLine("la cola esta vacia");
else
{
    if (objPersona.IsFull(colaPersona, capacidadCola))
        Console.WriteLine("la cola esta llena");
    else
        Console.WriteLine("la cola aun tiene espacio");
}
```

Practica lo aprendido

Existe un estacionamiento que tiene un sólo carril que aloja hasta 5 carros. Los autos llegan por el extremo sur del estacionamiento y salen por el extremo norte del mismo.

Si llega un cliente para recoger un carro que no está en el extremo norte, se sacan todos los automóviles de ese lado, se retira el auto y los otros coches se restablecen en el mismo orden que estaban. Cada vez que sale un auto, todos los autos del lado sur se mueven hacia adelante para que en todas las ocasiones todos los espacios vacíos estén en la parte sur del estacionamiento.

Escriba un programa que lea un grupo de líneas de ingreso. Cada línea contiene una ``A'' para las llegadas y una ``D'' para las salidas y un número de placa. Se supone que los carros llegan y salen en el orden especificado en la entrada.

El programa debe imprimir (en la terminal estándar) un mensaje cada vez que entra o sale un auto. Cuando llega un carro, el mensaje debe especificar si hay espacio o no para él en el estacionamiento. Si no hay espacio, el carro espera hasta que hay espacio o hasta que se lee una línea de salida para el auto. Cuando queda disponible espacio, debe imprimirse otro mensaje. Cuando salga un coche, el mensaje debe incluir la cantidad de veces que se movió el auto dentro del estacionamiento, incluyendo la salida misma, pero no la llegada. Este número es 0 si el carro sale de la fila de espera.

Definición de una Pila

- Son estructuras utilizadas muy a menudo como herramientas de programación de tipo LIFO (Last in- First out).

- Se apilan elementos y se sacan en orden inverso
- Uso común:
  - Control de ejecución (stack de llamadas)
  - Deshacer acciones en editores
  - Evaluación de expresiones

```
class persona
{
    public string nombre;
    public string apellido;
    public int edad;
};
0 referencias
class Program
{
    0 referencias
    static void Main()
    {
        Stack<int> pila = new Stack<int>();
        Stack<persona> pilaPersonas = new Stack<persona>();

        pila.Push(1);
        pila.Push(2);
        pila.Push(3);
        pila.Push(4);
        pila.Push(5);

        pilaPersonas.Push(new persona { nombre = "juan", apellido = "lopez", edad = 20 });
        pilaPersonas.Push(new persona { nombre = "ana", apellido = "perez", edad = 21 });
        pilaPersonas.Push(new persona { nombre = "pepe", apellido = "torres", edad = 20 });
        pilaPersonas.Push(new persona { nombre = "maria", apellido = "salas", edad = 19 });
        pilaPersonas.Push(new persona { nombre = "sonia", apellido = "valle", edad = 19 });

    }
}
```

último elemento

primer elemento

Recorrido de una pila

```
foreach (int num in pila)
{
    Console.WriteLine(num);
}

foreach (persona objPersona in pilaPersonas)
{
    Console.WriteLine($"{objPersona.nombre} " +
                      $"{objPersona.apellido} " +
                      $"({objPersona.edad} años)");
}
```

Se puede recorrer, pero no sacar elementos empleando el bucle foreach

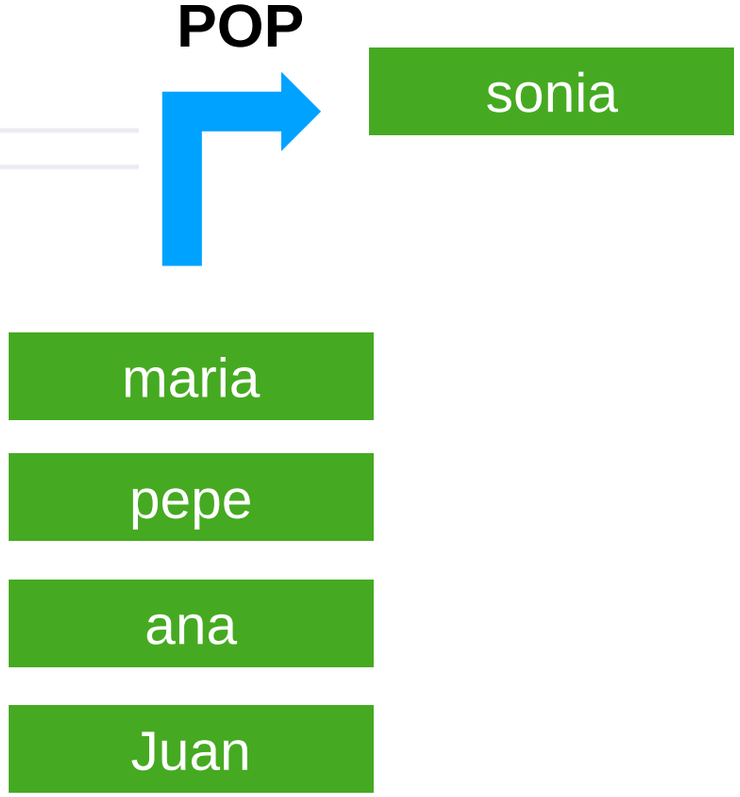
Métodos de las pilas

Pop() → Remueve y devuelve el elemento superior

```
pilaPersonas.Push(new persona { nombre = "juan", apellido = "lopez", edad = 20 });
pilaPersonas.Push(new persona { nombre = "ana", apellido = "perez", edad = 21 });
pilaPersonas.Push(new persona { nombre = "pepe", apellido = "torres", edad = 20 });
pilaPersonas.Push(new persona { nombre = "maria", apellido = "salas", edad = 19 });
pilaPersonas.Push(new persona { nombre = "sonia", apellido = "valle", edad = 19 });
```

```
pilaPersonas.Pop();

foreach (persona objPersona in pilaPersonas)
{
    Console.WriteLine($"{objPersona.nombre} " +
                      $"{objPersona.apellido} " +
                      $"({objPersona.edad} años)");
}
```



Métodos de las pilas

```
pilaPersonas.Push(new persona { nombre = "juan", apellido = "lopez", edad = 20 });
pilaPersonas.Push(new persona { nombre = "ana", apellido = "perez", edad = 21 });
pilaPersonas.Push(new persona { nombre = "pepe", apellido = "torres", edad = 20 });
pilaPersonas.Push(new persona { nombre = "maria", apellido = "salas", edad = 19 });
pilaPersonas.Push(new persona { nombre = "sonia", apellido = "valle", edad = 19 });
```

```
while (pilaPersonas.Count() > 0)
{
    Console.WriteLine($"{pilaPersonas.Peek().nombre} {pilaPersonas.Peek().apellido}");
    pilaPersonas.Pop();
}
Console.WriteLine($"cantidad final de pila: {pilaPersonas.Count()} elementos");
```

Al igual que en las colas, **Count()** devuelve el numero de elementos de la pila, el método **Peek()** devuelve el elemento de la pila que esta en la cima de esta.

En el ejemplo se recorre la pila, desapilando cada elemento

PREGRADO

Ingeniería de Sistemas de Información

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

UPC
Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco
Lima 33 - Perú
T 511 313 3333
https://www.upc.edu.pe

exígete, innova

LAUREATE
INTERNATIONAL
UNIVERSITIES