




14 DE MARZO DE 2019

RELACIÓN 1. OCULTACIÓN DE INFORMACIÓN EN ADA

SISTEMAS DE TIEMPO REAL

LAURA AGUILERA CHECA
IGNACIO AGUILERA GÓMEZ
GRADO EN INGENIERÍA INFORMÁTICA
Universidad de Almería



Contenido

1. Autores	2
2. Introducción	2
3. Actividades a realizar	2
3.1. Ejercicio 1	2

1. Autores

Apellidos	Aguilera	Checa
Nombre	Laura	
Titulación	Grado de Ingeniería Informática	

Apellidos	Aguilera	Gómez
Nombre	Ignacio	
Titulación	Grado de Ingeniería Informática	

2. Introducción

Con estos ejercicios se pretende realizar un acercamiento del alumno a la programación en ADA. Para ello, se plantea un ejercicio sencillo de utilización de paquetes en ADA. Dicho ejercicio deberá de ser remitido al profesor a través del módulo de Tareas de la plataforma Aula Virtual antes de la finalización del plazo de envío indicado. El formato de entrega de los ejercicios deberá de ser un documento PDF en el que se incluyan los comentarios y el código del programa realizados.

3. Actividades a realizar

3.1. Ejercicio 1 – ESTRUCTURAS BÁSICAS DE DATOS

Escribir un programa en ADA con las siguientes características:

1. Genere una cola con 10 datos usando el paquete Cola.
2. Extraiga y muestre por pantalla los datos introducidos en la cola anterior

En primer lugar, creamos el paquete Cola.ads y .adb correspondiente. El paquete constará con tres métodos: Vacía, Poner y Quitar.

```
package Cola is
    --Elemento de rango [-100, 100]
    type Elemento is range -100..100;
    --Devuelve true si está vacía y false si no lo está
    function Vacía return Boolean;
    --Añade el elemento indicado en la cola
    procedure Poner(E:Elemento);
    --Elimina el elemento indicado de la cola
    procedure Quitar(E: out Elemento);
end Cola;
```

En cuanto al cuerpo del paquete, se iniciará usando dos variables: Enlace y Nodo, donde Enlace es un puntero al Nodo y Nodo consta de un contenido de tipo Elemento (declarado anteriormente en cola.ads) y una variable Siguiente de tipo Enlace. También se usarán las variables Primero y Último de tipo Enlace, inicialmente nulas.

```
type Nodo;
--Enlace es un puntero al Nodo
type Enlace is access Nodo;
type Nodo is record
    Contenido: Elemento;
    Siguiente: Enlace;
end record;

--Se inicializa el primer y último enlace a null
Primero, Ultimo : Enlace :=null;
```

El método Vacía de tipo boolean devuelve true si la cola está vacía y false si contiene algún elemento. Será necesario para implementar los métodos Poner y Quitar. Para ello, basta con comprobar que el primer elemento de la cola es no nulo.

```
--Comprueba si la cola está vacía comprobando el primer elemento
function Vacía return Boolean is
begin
    return Primero = null;
end Vacía;
```

Por otra parte, el método Poner sirve para añadir un elemento nuevo en la cola. Para ello, se le pasa un objeto E de tipo Elemento y se crea una nueva variable Nuevo de tipo Enlace, a la que se añade el elemento E en Contenido. Por último, se cambia el puntero del último elemento para que apunte a Nuevo y se sustituye el último por este, cuyo puntero apunta a nulo. De este modo, se añade un nuevo elemento al final de la cola.

```
--Añade el elemento a la cola
procedure Poner(E:Elemento) is
    Nuevo : Enlace;
begin
    Nuevo := new Nodo;
    Nuevo.Contenido:=E;
    Nuevo.Siguiente := null;
    if Vacía then
        Primero := Nuevo;
    else
        Ultimo.Siguiente := Nuevo;
    end if;
    Ultimo := Nuevo;
end Poner;
```

El método Quitar, por otra parte, elimina el primer elemento de la cola y lo devuelve. Para ello, crea una nueva variable denominada Viejo que guarda el contenido del elemento a eliminar, que ocupa la posición de primero. A continuación, se sustituye el valor del primero por el siguiente elemento. Si no hay más elementos, la cola se declara vacía.

```
--Quita el primer elemento de la cola y devuelve dicho elemento
procedure Quitar(E: out Elemento) is
    Viejo: Enlace;
begin
    Viejo := Primero;
    E := Viejo.Contenido;
    Primero := Viejo.Siguiente;
    if Primero = null then Ultimo := null;
    end if;
end Quitar;
```

La clase main, por otra parte, contiene el código a ejecutar, que accede a el paquete Cola. En primer lugar, se ejecuta un bucle en el que se introducen enteros del 1 al 10 en la cola haciendo uso del método Poner(E).

```
with Cola;  
use Cola;  
with Ada.Text_IO, Ada.Float_Text_IO;  
use Ada.Text_IO, Ada.Float_Text_IO;  
  
procedure main is  
  E: Elemento;  
  Eliminado: Integer := 0;  
  
begin  
  --Insertar números del 1 al 10 en la cola  
  for i in 1..10 loop  
    E := Elemento(i);  
    Poner(E);  
  end loop;  
  
  --Eliminar los elementos de la cola y mostrarlos por pantalla  
  while Vacía = false loop  
    --Se elimina el elemento y se guarda en la variable Eliminado  
    Quitar(Elemento(Eliminado));  
    --Se imprime la variable por pantalla  
    Put_Line(Integer'Image(Eliminado));  
  end loop;  
  
end main;
```

A continuación, eliminaremos dichos elementos y los mostraremos por pantalla mediante un bucle while que se ejecuta hasta que la cola quede vacía. Primero, se ejecuta el método Quitar y se guarda su resultado en la variable Eliminado y, a continuación, se imprime el valor de dicha variable por pantalla, mostrando el siguiente resultado:

```
C:\GNAT\2018\bin\obj\main.exe  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
[2019-03-16 12:34:45] process terminated successfully, elapsed time: 03.77s
```

De este modo la cola queda de nuevo vacía y los elementos mostrados en pantalla.