Práctico 1: Primeros programas usando Pilas

Objetivos del Práctico:

Al finalizar este práctico se espera que los alumnos

- reconozcan las partes de un programa,
- sean capaces de utilizar las tres estructuras básicas de la programación estructura: secuencia, selección e iteración,
- comprendan la estructura "pila" y sus operaciones asociadas,
- puedan resolver un problema sencillo.

NOTA: Para resolver los ejercicios, usar el *Entorno para Resolución del práctico*. Algunos ejercicios tienen preguntas, pueden contestarlas como comentarios en el ejercicio correspondiente en el entorno.

1) Cargar desde el teclado una pila DADA con 3 elementos. Pasar los dos primeros elementos a la pila CJTO1 y el restante a la pila CJTO2, ambas pilas inicializadas en vacío.

1.1) ¿Importa el orden de las sentencias escritas? Explique.

1.2) ¿Funciona el programa si DADA tiene menos de 3 elementos?. Si funciona explicar por qué, sino modifícalo.

2) Analice la siguiente porción de código

```
.....
        if tope(Pila1) < 10 then
                apilar(Resultado, desapilar(Pila1))
        else
                apilar(OtroResultado, desapilar(Pila1));
```

Suponiendo que las pilas Resultado y OtroResultado se inician vacías muestra como quedarian luego de ejecutar esta porción de código para:

a) Pila1 <base> 6 10 19 <tope>
b) Pila1 <base> 6 10 8 <tope>

2.1) Si el código fuera

```
.....
        apilar (Resultado, desapilar(Pila1));
        apilar (OtroResultado, desapilar(Pila1));
```

y suponiendo que las pilas resultado y OtroResultado están inicialmente vacías muestre cómo quedarian luego de ejecutar esta porción de código para los mismos datos del ejercicio anterior.

2.2) ¿Por qué las dos porciones de código no realizan la misma tarea? ¿Qué función cumple la estructura de control condicional IF? Qué ocurre si Pila1 posee un sólo elemento?

3) Analice la siguiente porción de código y responda.(Pila1, Pila2, y Descarte son pilas)

```
……..
while not pilaVacia(Pila1) do
begin
    apilar (Pila2, desapilar(Descarte))
end;
```

3.1 ¿Cuál es la condición del ciclo?

3.2 ¿Cuándo finaliza el ciclo? ¿Por qué?

3.3 Modifique el código para que el ciclo tenga finalización.

3.4 ¿Qué función realiza este código ahora?

3.5 ¿De qué manera se asegura en un programa que siempre se llegará a un corte dentro del ciclo de repetición condicional?

3.6. ¿Es posible resolver el problema de pasar los datos de una pila a otra sin usar un ciclo de repetición condicional?

4) Realizar un programa para:Cargar desde teclado una pila DADA y pasar a la pila DISTINTOS todos aquellos elementos distintos al valor 8. Los elementos iguales a 8 deben quedar en DADA

5) Se realizó el siguiente programa para el problema: "Cargar por teclado una pila ORIGEN y pasar a la pila DISTINTO todos aquellos elementos que preceden al valor 5 (elementos entre el tope y el valor 5). No se asegura que exista algún valor 5",

```
program PASADISTINTOS;
{Este un programa carga por teclado una pila ORIGEN y pasa a la pila DISTINTO
todos aquellos elementos que preceden al valor 5}
uses estructu;
var Origen, Distinto: pila;
begin
        inicpila(Origen, '');
        inicpila(Distinto, '');
        readpila(Origen);
        if not pilaVacia(Origen) then
                while tope(Origen) <> 5 do
                        apilar (Distinto, desapilar(Origen));
end.
```

a. ¿Resuelve el problema planteado?
b. ¿Cuáles son los errores que encuentra?
c. Reescribir el código para que resuelva adecuadamente el problema planteado.
d. Indicar las componentes del programa.

6) Se realizó el siguiente código para resolver el problema de: Pasar el primer elemento (tope) de la pila DADA a su última posición (base), dejando los restantes elementos en el mismo orden.

Ejemplo: DADA <base> 5 6 7 <tope>

Resultado DADA <base> 7 5 6 <tope>

Funciona correctamente? Si no es asi, corregirlo.

```
Program PasaTope;
{$INCLUDE /usr/Estructu}
{ Dada una pila pasar el TOPE a su ultima posición, el resto queda igual }
var origen, auxTope, auxiliar: pila;
begin
   readpila(origen);
   inicpila(auxTope, '');
   inicpila(auxiliar, '');
   apilar(auxtope,desapilar(origen));
   while not pilavacia(origen) do begin
      apilar (auxiliar, desapilar(origen));
   end;
   while not pilavacia(auxiliar) do begin
      apilar (origen, desapilar(auxiliar));
   end;
   writepila(origen);
end.
```

7) Suponiendo la existencia de una pila LIMITE, pasar los elementos de la pila DADA que sean mayores o iguales que el tope de LIMITE a la pila MAYORES, y los elementos que sean menores a la pila MENORES. ¿Funciona tu solución si DADA está vacía?

8) Dada una pila ORIGEN que se carga desde el teclado, y una pila DESTINO inicialmente vacía, se pretende pasar los datos de ORIGEN a DESTINO pero dejándolos en el mismo orden.

8.1) Modificar el programa anterior para Cargar desde el teclado la pila ORIGEN y que la invierta, es decir, que quede ORIGEN con los mismos elementos pero invertidos

9) Realice un programa para pasar el último elemento (base) de la pila DADA a su primer posición (tope), dejando los restantes elementos en el mismo orden.

Ejemplo: DADA <base> 6 10 9 <tope>

Resultado DADA <base> 10 9 6 <tope>

10) Suponiendo la existencia de una pila MODELO que no esté vacía, eliminar de la pila DADA todos los elementos que sean iguales al tope de la pila MODELO.

11) Codificar un programa que reparta los elementos de la pila POZO en las pilas JUGADOR1 y JUGADOR2 en forma alternativa. Probarlo para una pila POZO con cantidad impares de elementos

12) Suponiendo la existencia de tres pilas ORIGEN, MODELO y REEMPLAZO no vacías, escribir un programa *Reemplazar* que reemplace a todos los elementos de ORIGEN que sean igual al tope de MODELO por el tope de REEMPLAZO.

13) Determinar si la cantidad de elementos de la pila DADA es par. Si es par, pasar el elemento del tope de la pila AUX a la pila PAR y si es impar pasar el tope a la pila IMPAR.

14) Lee el código y determina qué objetivo intenta cumplir este programa.

```
program Incognita;
{$INCLUDE /usr/Estructu}
{ ..... }
var origen, descarte, auxiliar: pila;
begin
   readpila(origen);
   inicpila(auxiliar, '');
   inicpila(descarte, '');
   while not pilavacia(origen) do begin
     if tope(origen) = 2 then
        begin
          apilar (descarte, desapilar(origen));
          apilar (auxiliar, 0)
        end
      else
          apilar(auxiliar, desapilar(origen))
   end;
   while not pilaVacia(auxiliar) do
      apilar(origen, desapilar(auxiliar));
    end;
  writepila(origen);
end.
```