UNICEN
FAC. CS. EXACTAS

INTRODUCCIÓN A LA PROGRAMACIÓN I

GUÍA DE

TRABAJOS PRÁCTICOS

Práctico 1: Pilas

Objetivos:

Al finalizar este práctico se espera que los alumnos:

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

Codificar los siguientes programas. En el caso de ser necesario se pueden utilizar estructuras auxiliares:

1) Cargar desde el teclado una pila DADA con 5 elementos. Pasar los tres primeros elementos a la pila CJTO1 y los dos restantes a la pila CJTO2, ambas pilas inicializadas en vacío.

2) Cargar desde el teclado la pila ORIGEN e inicializar en vacío la pila DESTINO. Pasar todos los elementos de la pila ORIGEN a la pila DESTINO.

3) Cargar desde teclado una pila DADA y pasar a la pila DISTINTOS todos aquellos elementos distintos al valor 8.

4) Cargar desde el teclado la pila ORIGEN e inicializar en vacío la pila DESTINO. Pasar los elementos de la pila ORIGEN a la pila DESTINO, pero dejándolos en el mismo orden.

5) Cargar desde el teclado la pila DADA. Invertir la pila de manera que DADA contenga los elementos cargados originalmente en ella, pero en orden inverso.

Para los siguientes ejercicios cargar desde teclado o inicializar las pilas según corresponda.

6) Pasar el primer elemento (tope) de la pila DADA a su última posición (base), dejando los restantes elementos en el mismo orden.

7) Pasar el último elemento (base) de la pila DADA a su primera posición (tope), dejando los restantes elementos en el mismo orden.

8) Repartir los elementos de la pila POZO en las pilas JUGADOR1 y JUGADOR2 en forma alternativa.

9) Comparar la cantidad de elementos de las pilas A y B. Si son iguales dejar el tope de la pila AUX en la pila VERDADERO y son distintas en la pila FALSO.

10) Comparar las pilas A y B. Si son iguales dejar el tope de la pila AUX en la pila VERDADERO y si son distintas en FALSO.

11) 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.

12) Suponiendo la existencia de una pila MODELO (vacía o no), eliminar de la pila DADA todos los elementos que existan en MODELO.

13) 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.

14) 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.

15) ¿Cuál es la condición del siguiente ciclo? ¿Cuándo finaliza el ciclo? (Pila1, Pila2, y Descarte son pilas):

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

16) Dado el siguiente código escrito en lenguaje Pascal (*Pila1*, *Aux* y *Result* son pilas):

```
while not pilaVacia(Pila1) do
if tope(Pila1) = 5 then
apilar (Aux, desapilar(Pila1));
apilar (Result, desapilar(Aux));
```

a. ¿Qué es lo que realiza?

b. Identarlo de manera adecuada.

c. Indicar si hay alguna sentencia compuesta.

17) Para el ejercicio “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”, se realizó el siguiente programa:

```
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.

18) Dado el siguiente ciclo (*Pila1*, *Pila2* y *Descarte* son pilas):

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

a. ¿Cuál es la condición del ciclo?

b. ¿Cuáles son los posibles estados de ambas pilas al finalizar el ciclo?

Práctico 2: Filas

Objetivos:

Al finalizar este práctico se espera que los alumnos:

- comprendan la estructura "fila" y sus operaciones asociadas,
- reconozcan las principales diferencias que existen entre las pilas y las filas,
- puedan decidir, dependiendo del problema, qué estructuras utilizar para resolverlo.

1) ¿Cuáles son las operaciones que pueden realizarse sobre pilas? ¿Y sobre filas?

2) ¿Cuál es la principal diferencia que existe entre estas estructuras?

3) Se tienen las siguientes estructuras:

Pilas: Pila1 y Pila2

Filas: Fila1 y Fila2.

Para las siguientes acciones, indicar cómo quedan los elementos en la estructura destino (¿quedan en el mismo orden en el que estaban en la estructura original?)

a. Pasar todos los elementos de Pila1 a Pila2
b. Pasar todos los elementos de Pila1 a Fila1
c. Pasar todos los elementos de Fila1 a Pila1
d. Pasar todos los elementos de Pila1 a Fila2

Codificar los siguientes programas leyendo desde teclado o inicializando las pilas según corresponda. En caso de ser necesario se pueden utilizar estructuras auxiliares.

4) Pasar los elementos de la fila ORIGEN a la pila DESTINO.

5) Comparar el ejercicio anterior con los ejercicios 2, 4 y 5 del Práctico 1. ¿Con cuál guarda semejanza conceptual (realiza la misma tarea) y a cuál algoritmo se parece más? ¿Por qué?

6) Invertir el orden de la fila DADA usando una pila auxiliar.

7) Cargar la fila DATOS y pasar el primer elemento (tope) de la fila DATOS a su última posición (base), dejando los restantes elementos en el mismo orden. Finalmente imprima la fila por pantalla.

8) Cargar por teclado la fila DATOS y pasar el último elemento (base) de la fila DATOS a su primera posición (tope), dejando los restantes elementos en el mismo orden. Finalmente imprima la fila por pantalla.

9) Comparar los dos ejercicios anteriores con el ejercicio 6 y 7 del Práctico 1.

10) Cargar PILA1 y PILA2 y luego imprimir PILA1 ó PILA2 según un valor ingresado por el usuario en la pila VALOR.

11) Cargar una estructura (pila o fila) MODELO con un solo elemento y otra estructura ORIGEN. Modificar ORIGEN tal que si existe un elemento igual al de MODELO se lo ubique en el tope. Imprima ORIGEN.

12) Ubicar el tope de la estructura DADA debajo del primer elemento de valor 12, dejando los demás elementos en el mismo orden. Si el elemento de valor 12 es el tope, dejarlo como estaba. Se supone que al menos hay un elemento de valor de 12.

13) Concatenar dos pilas de modo que la que posee menos elementos quede abajo; si ambas tienen la misma cantidad de elementos, cualquiera puede quedar abajo. ¿Convendría resolver el problema con filas en lugar de pilas? Justifique su respuesta.

14) Una pila ORIGINAL y otra COPIA contienen los mismos elementos aunque en distinto orden. Redisponer los elementos de COPIA para que resulte una reproducción de ORIGINAL.

15) Indicar si una pila PARTE está incluida en otra pila GRANDE (o sea, que una porción de GRANDE es igual a PARTE con los mismos elementos en el mismo orden). La forma de indicarlo será dejando vacía PARTE si efectivamente está incluida.

16) La fila DADA se encuentra ordenada. Insertar el tope de la pila ELEMENTO en DADA de tal forma que continúe ordenada.

17) En el ejercicio 18 del Práctico 1, ¿cuáles serían los posibles estados de las pilas al salir del ciclo si el operador lógico fuera “OR”?.

18) Usando el código escrito para el punto 16, verificar el comportamiento del algoritmo con los siguientes ejemplos de datos:

FILA: [ultimo] ..... [primero]

a. DADA: 13 9 7 5 3
   ELEMENTO: {vacía}

b. DADA: {vacía}
   ELEMENTO: 5

c. DADA: 13 9 7 5 3
   ELEMENTO: 2

d. DADA: 13 9 7 5 3
   ELEMENTO: 18

e. DADA: 13 9 7 5 3
   ELEMENTO: 8

Práctico 3: Modularización y Procedimientos

Objetivos:

Al finalizar este práctico se espera que los alumnos:

- comprendan el concepto de parámetros, la diferencia entre parámetros formales y actuales y las distintas formas de pasaje de parámetros,
- utilicen correctamente los parámetros
- comprenda el concepto de procedimiento y la utilidad de los mismos como herramienta para la estructuración de un programa,

Para cada uno de los ejercicios realizar el diagrama de estructura y codificar la solución.

1) Leer los ejercicios 4 y 5 del Práctico 1, identificar un procedimiento común a cada uno de ellos, asignarle un nombre representativo con sus respectivos parámetros, codificar el procedimiento y las soluciones. {pasar en igual orden e invertir}

2) Realizar un procedimiento que elimine los elementos repetidos de una pila DADA.

3) Resolver el ejercicio 14 del Práctico 2 {redisponer copia según original} realizando sucesivos llamados a un procedimiento que implemente la tarea especificada en el ejercicio 11 {si existe uno igual a modelo ponerlo en el tope} del Práctico 2.

4) Codificar un procedimiento para resolver el ejercicio 11 del Práctico 1 {eliminar según tope de modelo}. Luego con este procedimiento resolver los ejercicios 12 del práctico 1 {eliminar todos los de modelo} y 2 de este práctico {eliminar los repetidos}.

5) Dadas dos pilas A y B que simulan conjuntos (cada conjunto no tiene elementos repetidos), realizar un procedimiento que calcule en la pila C la operación de unión.

6) Intercalar dos filas ordenadas en forma creciente (ORDENADA1 y ORDENADA2) dejando el resultado en una fila también ordenada en forma creciente (ORDENADAFINAL).

7) Para el ejercicio anterior (6), verificar el comportamiento del algoritmo utilizando los siguientes ejemplos de datos:

FILA: [ultimo] ..... [primero]

a. ORDENADA1: 22 19 8 5 2
   ORDENADA2: {vacía}

b. ORDENADA1: {vacía}
   ORDENADA2: 22 19 8 5 2

c. ORDENADA1: 22 19 8 5 2
   ORDENADA2: 13 12 11 5 3

8) El procedimiento PasaElemPila que se muestra a continuación se encarga de pasar los elementos de PilaOrigen a PilaDestino:

```
procedure PasaElemPila (….. PilaOrigen, …… PilaDestino: pila);
begin
while not(pilaVacia(PilaOrigen)) do
    apilar (PilaDestino, desapilar(PilaOrigen));
end;
```

Si se invoca a este procedimiento con la pila Origen (cuyos elementos fueron ingresados por teclado) y con la pila Destino (inicializada en vacío):

```
PasaElemPila (Origen, Destino);
```

a. Indicar el estado de ambas pilas luego de la invocación al procedimiento si los parámetros fueran pasados de la siguiente manera:
   i. procedure PasaElemPila (var PilaOrigen, PilaDestino: pila);
   ii. procedure PasaElemPila (var PilaOrigen: pila; PilaDestino: pila);
   iii. procedure PasaElemPila (PilaOrigen: pila; var PilaDestino: pila);
   iv. procedure PasaElemPila (PilaOrigen, PilaDestino: pila);
b. Si el objetivo es pasar los elementos de la pila origen a la pila destino, ¿cuál de las opciones anteriores es la adecuada y por qué?.

9) Las operaciones ReadPila, WritePila, ReadFila y WriteFila son implementadas como procedimientos. Para cada uno de ellos, definir sus encabezados indicando los parámetros y el tipo de los mismos:

a. procedure ReadPila (¿……?);
b. procedure WritePila (¿……?);
c. procedure ReadFila (¿……?);
d. procedure WriteFila (¿……?);

Práctico 4: Método de Desarrollo

Objetivos:

Al finalizar este práctico se espera que los alumnos:

- puedan plantear estrategias para resolver problemas (siguiendo la forma de trabajo dada en la teoría),
- comprendan la utilidad del diagrama de estructura,
- desagreguen problemas utilizando diagramas de estructura,
- usen diagramas de estructura y procedimientos en la resolución de problemas.
- incorporen las cuplas al diagrama de estructura.

1) Para el siguiente problema: “Separar de la pila DADA el mayor de sus elementos (incluso suponiendo que puede haber más de uno), colocándolo en la pila MAYOR”, identificar y describir cada uno de los pasos de la forma de trabajo especificada en la teoría (comprender el problema, estudiar las situaciones posibles, plantear las estrategias posibles, etc.).

Para cada uno de los ejercicios, plantear la estrategia a utilizar (en pseudo-código), realizar el diagrama de estructura y luego codificar (con uno o varios procedimientos según corresponda).

2) Verificar si una pila DADA es capicúa.

3) Dividir la pila DADA a la mitad dejando el resultado en MITAD1 y MITAD2, respetando el orden en ambas partes.

4) Ordenar la fila DADA en forma descendente, usando un procedimiento que resuelva el problema del ejercicio 1 de este práctico {separar el mayor}.

5) Separar de la pila DADA el menor de sus elementos (incluso suponiendo que puede haber más de uno), colocándolo en la pila MENOR. DADA debe conservar el orden original.

6) Dada la fila ORIGEN, separar en la fila SECMASLARGA la secuencia más larga de elementos no nulos de ORIGEN (esto quiere decir que las secuencias están separadas por ceros). ¿Con cuántos procedimientos trabajó en este caso? Justificar dicha elección.

7) Suponer un juego de cartas en el que en cada mano se reparten dos cartas por jugador. Un jugador gana la mano cuando la suma de sus cartas es mayor que las del contrario y al hacerlo coloca todas las cartas (las de él y las de su rival) en su pila de puntos. En caso de empate (y para simplificar) siempre gana el jugador1.

Simular la ejecución del juego de tal manera que dada una pila MAZO (con un número de elementos múltiplo de cuatro) distribuya las cartas en las pilas PUNTOSJUG1 y PUNTOSJUG2 como si éstos hubieran jugado. Utilizar las pilas auxiliares que crea conveniente.

8) Dada la fila ORIGEN ordenarla en forma ascendente por método de **selección** dejando el resultado en la fila DESTINO.

9) Dada la fila ORIGEN ordenarla en forma ascendente por método de **inserción** dejando el resultado en la fila ORIGEN.

10) Pensar el ejercicio 6 de este práctico {separar la sec. más larga} como si fuera el ejercicio 1 de este práctico {separar el mayor}.

a. ¿qué procedimientos necesita? {extraer secuencias y comparar secuencias}
b. definir sus encabezados (incluyendo los parámetros).
c. codificar la solución en base a los procedimientos definidos.
d. codificar los procedimientos definidos.

11) Codificar un procedimiento que elimine las secuencias de igual longitud en una pila (tome como modelo el ejercicio 2 del Práctico 3).

12) Codificar un procedimiento que elimine las secuencias iguales en una pila, usando otros procedimientos.

13) Codificar un procedimiento que ordene descendentemente las secuencias de una pila según la longitud de dichas secuencias.

14) Invertir el orden de una fila utilizando sólo filas auxiliares.

Práctico 5: Variables

Objetivos:

Al finalizar este práctico se espera que los alumnos:

- comprendan el concepto de variable y la operación de asignación,
- reconozcan los distintos tipos predefinidos de Pascal,
- utilicen variables y distingan el alcance de las mismas.

Realizar los diagramas de estructura y codificar los procedimientos que realicen:

1) Intercambiar los valores de dos variables enteras. ¿Cuáles serían las diferencias si las variables fueran de tipo caracter?

2) Sumar los N primeros números naturales.

3) Se tiene una fila CHEQUES, donde cada elemento representa el importe de un cheque, y otra SOCIOS, donde cada elemento representa un socio según su número de carnet. Determinar cuánto recibe cada socio si se distribuye equitativamente el total del dinero.

4) Dadas tres variables enteras A1, M1, D1, que representan una fecha, y otras tres A2, M2 y D2, que representan otra. Colocar la variable booleana Resultado en True o False si la primer fecha es menor que la segunda.

5) Con las mismas variables del problema anterior, informar la cantidad de días entre dos fechas (suponga que todos los meses tienen 30 días y los años 360 días). Además, definir al menos 3 casos de testeo con valores para ambas fechas y el resultado que espera tener en cada caso. Realizar el seguimiento del código con cada par de valores y verificar si se obtiene el resultado esperado.

6) Colocar en una variable la posición en la fila ORIGINAL de un elemento dado.

7) Dada la variable entera Posición, eliminar de la pila Secuencia el elemento que se encuentre en dicha posición.

8) Dadas dos pilas (ORIGINAL y POSICIONES), eliminar de ORIGINAL todos los elementos que POSICIONES indica, usando los procedimientos anteriores. Note que al eliminar un elemento los restantes disminuyen en uno su ubicación. Contemple este caso para que los corrimientos no afecten su algoritmo.

9) En base a una pila DADA (que contiene al menos un elemento), generar una pila RESULTADO donde cada elemento será la suma de los valores de DADA ubicados en una posición menor o igual al mismo. El tope corresponde a la posición 1. Por ejemplo, en la posición 3 de RESULTADO, estará la suma de las posiciones 1, 2 y 3 de DADA.

10) Separar la pila DADA en secuencias no nulas, lo más largas posibles pero de tal forma que los elementos de cada secuencia no sumen más de 20. Suponga que ningún elemento es mayor a 20.

11) Separar la pila DADA en dos mitades de tal forma que la diferencia entre la suma de cada mitad sea mínima.

12) Codificar el procedimiento ExtraerSecuencia que recibe en el parámetro DADA la pila de la que extrae, y retorna en la pila NUEVA la primer secuencia de elementos no nulos de DADA. Usando el procedimiento ExtraerSecuencia, codificar el procedimiento CantidadSecuencias que recibe como parámetro la pila DADA, y que devuelve en el parámetro entero Cantidad el número de secuencias de la pila DADA.

13) Generar una pila RESULTADO cuyos elementos se correspondan con la cantidad de elementos que tiene cada secuencia en la pila DADA. En el tope de RESULTADO debe estar el promedio de la cantidad de elementos de las secuencias encontradas.

14) El siguiente programa informa si la pila Dada tiene más de 10 elementos:

```
program MAYOR10;
var
    Dada, Aux: pila;
    cant: integer;
begin
    inicPila(Aux, '');
    readPila(Dada);
    while not pilaVacia (Dada) do
    begin
        cant:=cant+1;
        apilar(Aux, desapilar(Dada));
    end;
    if cant > 10 then
        writeln ('La cantidad de elementos de la pila es mayor a 10')
    else
        writeln ('La cantidad de elementos de la pila es menor o igual a 10');
end.
```

a. ¿Hace lo pedido? En caso de que haya errores, señalarlos.

b. ¿Es necesario "mirar" todos los elementos de la pila?

c. Modificar el programa anterior de modo que no recorra toda la pila si no es necesario.

Práctico 6: Funciones

Objetivos:

Al finalizar este práctico se espera que los alumnos:

- comprendan el concepto de función,
- distingan las diferencias entre las funciones y los procedimientos y
- utilicen funciones de manera adecuada.

Realizar el diagrama de estructura y codificar las funciones o procedimientos que realicen:

1) Obtener el valor de la multiplicación de dos enteros. Usar sólo sumas.

2) Dados dos enteros A y B, obtener A^B (A elevado a la B), utilizando el procedimiento/función del ejercicio 1.

3) Verificar si un número es par o impar, devolviendo true o false, respectivamente.

4) Dadas dos variables booleanas, indicar el resultado de la operación lógica determinada por un caracter también dado, siendo este "Y" o "0".

5) Dada una pila, devolver el promedio de todos sus elementos.

6) Dado un número, calcular su factorial. Ej: fact(4) = 4*3*2*1 = 24.

7) Dada una fila, devolver el valor que resulte de calcular la suma de todos sus elementos.

8) Dado un valor entero, codificar la función que lo busque en la pila Dada. Si el valor entero existe en Dada, retornar la posición en la que lo encontró, y si no existe, retornar -1. Testear la función con los siguientes casos:

PILA: [Tope] .... [Ultimo]

a. DADA: {vacía}
b. DADA: 3 5 8 7 3 4
   elemento: 9
c. DADA: 8 4 3 22 9
   elemento: 8
d. DADA: 8 4 3 22 9
   elemento: 3

9) Dada una pila, devolver el número que se repite más veces.

10) Dada una fila donde cada elemento tiene un solo dígito (0 a 9), retornar el valor entero formado por la concatenación de todos los elementos de la fila.

Ejemplo: fila = < 1 2 4 0 9 > → 12409

Luego codifique el procedimiento inverso (dado un número entero, retornar la fila de dígitos cuya concatenación es igual al entero).

11) Dada una fila, generar otra con unos y ceros para indicar si el elemento que se encuentra en la misma posición es par o impar.

12) Calcular el Máximo Común Divisor de dos enteros no negativos, basándose en las siguientes fórmulas matemáticas:

$$\text{MCD}(X, X) = X$$
$$X < Y \Rightarrow \text{MCD}(X, Y) = \text{MCD}(Y, X)$$
$$X > Y \Rightarrow \text{MCD}(X, Y) = \text{MCD}(X - Y, Y)$$

Utilizando la función MCD anterior, calcular el MCD de todos los elementos de una fila dada, basándose en que:

$$\text{MCD}(X, Y, Z) = \text{MCD}(\text{MCD}(X, Y), Z)$$

13) Para cada una de las operaciones definidas sobre filas:

- FILAVACIA
- PRIMERO
- AGREGAR
- EXTRAER
- READFILA
- WRITEFILA

a. Indicar si es una función o un procedimiento.

b. Definir sus encabezados indicando los parámetros, el tipo de los mismos y el valor de retorno en el caso de las funciones.

Práctico 7: Arreglos

Objetivos:

Al finalizar este práctico se espera que los alumnos:

- comprendan la noción de arreglo (como tipo estructurado),
- puedan realizar operaciones con arreglos,
- entiendan las ventajas que presentan y
- realicen algoritmos de búsqueda, inserción, eliminación y ordenamiento en arreglos.

1) Realizar un procedimiento que cargue caracteres (no más de 50) desde el teclado en un arreglo. El fin de la carga se detecta por el ingreso de un carácter "*".

2) Realizar una función que sume los elementos reales de un arreglo de dimensión 100.

3) Realizar una función que indique si un elemento dado se encuentra en un arreglo de caracteres.

4) Realizar un procedimiento / función (según corresponda) que inserte un caracter en un arreglo ordenado alfabéticamente, conservando el orden.

5) Realizar un procedimiento / función (según corresponda) que obtenga el máximo caracter de un arreglo dado. ¿Cómo cambiaría la solución si el arreglo estuviera ordenado?

6) Realizar un procedimiento que dado un arreglo de N caracteres, devuelve un arreglo de N-1 valores booleanos, tal que cada valor de este último arreglo corresponde al resultado de la comparación de los pares de valores consecutivos del primer arreglo. El valor booleano es True si el primer carácter del par es menor o igual que el siguiente, y False si es mayor que el segundo.

Ejemplo: N = 5

4	2	8	8	6

Resultado:

false	true	true	false

7) Realizar un procedimiento / función (según corresponda) que determine si un arreglo es capicúa.

8) Realizar un procedimiento / función (según corresponda) que invierta los elementos de un arreglo.

9) Ordenar un arreglo según los siguientes métodos:

a. Selección.
b. Burbujeo.
c. Inserción.

10) Realizar un procedimiento que tome un arreglo VALORES y genere el arreglo POSNUEVAS de manera que estas posiciones permitan acceder a los elementos de valores en forma ordenada ascendente.

Ejemplo: VALORES: [f b n x] → POSNUEVAS: [2 1 3 4]

11) Definir las estructuras que permitan almacenar pilas de enteros (de hasta 50 elementos) con arreglos, e implemente las funciones y los procedimientos para su manipulación:

- PILAVACIA
- TOPE
- APILAR
- DESAPILAR
- READPILA
- WRITEPILA

Sugerencia: Utilizar la posición cero del arreglo para almacenar la cantidad de elementos de la pila.

12) Si un arreglo puede ser de cualquier tipo, ¿cuán dificultoso sería cambiar las definiciones y procedimientos para tener pilas de booleanos, de caracteres, etc.?

13) Definir las estructuras que permitan almacenar filas de enteros (de hasta 50 elementos) con arreglos, e implemente las funciones y los procedimientos para su manipulación:

- FILAVACIA
- PRIMERO
- AGREGAR
- EXTRAER
- READFILA
- WRITEFILA

14) El procedimiento Inicializar se realizó con el objetivo de asignarle el valor 0 a cada uno de los N elementos enteros de un arreglo. Se tienen las siguientes versiones de dicho procedimiento:

a.

```
procedure Inicializar (var Arr: arrEnteros);
i: integer
begin
    I := 1;
    while (i <= N) do
        Arr[i] := 0
end;
```

b.

```
procedure Inicializar (var Arr: arrEnteros);
i: integer
begin
    i:= 1;
    while (i <= N) do
    begin
        Arr[i] := 0;
        i := i + 1
    end;
end;
```

c.

```
procedure Inicializar (Arr: arrEnteros);
i: integer
begin
    i:= 1;
    while (i <= N) do
        Arr[i] := 0;
        i := i+1
end;
```

d.

```
procedure Inicializar (var Arr: arrEnteros; i:integer);
begin
    i:= 1;
    while (i <= N) do
    begin
        Arr[i] := 0;
        i := i+1
    end;
end;
```

Analizar cada una de las versiones presentadas e indicar si realizan la tarea pedida. En cada caso, mencionar los errores que encuentre.

15) La siguiente función retorna verdadero si un número dado (nro) se encuentra en un arreglo de enteros y falso en caso contrario:

```
Function Existe (Arr: tipoArr; nro: integer): boolean;
Var
    esta: boolean;
    i: integer;
begin
    i:=1;
    while (Arr[i] <> nro) and (i <= N) do
        i := i + 1;
    if Arr[i] = nro then
        esta := true;
    Existe := esta
end;
```

a. ¿Esta función realiza lo especificado? Listar los casos de testeo que cree que son necesarios para realizar esta prueba (ejemplos de valores para los parámetros *Arr* y *nro*). Por cada uno de ellos indicar el resultado esperado de la ejecución de la función. Realizar el seguimiento del código con cada caso y verificar si el resultado coincide con el esperado.

b. En caso de tener errores, identificarlos y corregirlos.

16) Se presenta otra versión de la función especificada anteriormente:

```
Function Existe (Arr: tipoArr; nro: integer): boolean;
Var
    esta: boolean;
    i: integer;
begin
    i:=1;
    while (i <= N) do
        if Arr[i] = nro then
            begin
                esta := true;
                i := N + 1
            end;
        else
            i:=i+1
    Existe:=esta
end;
```

¿Qué puede decir acerca de esta solución?

Practico 8: Matrices

Objetivos:

Al finalizar este práctico se espera que los alumnos puedan trabajar con arreglos de una y más dimensiones.

1) Definir una matriz de números enteros que contenga a lo sumo 5 columnas y 5 filas. Realizar un procedimiento que posibilite el ingreso de datos para la misma. ¿En que cambiaría si la matriz fuera de otro tipo (otras dimensiones y/u otro tipo de componentes?

2) Realizar un procedimiento que dado como parámetro la matriz definida en el ejercicio anterior, la convierta en la matriz identidad (contiene unos en la diagonal y ceros en el resto de sus posiciones).

3) Definir una matriz de 10 x 10 enteros y realizar los siguientes procedimientos o funciones:

a. Cargar la matriz - Para ello se puede usar una variación del ejercicio 1.
b. Sumar una columna dada.
c. Sumar una fila dada.
d. Calcular el promedio de una fila dada.
e. Calcular el promedio de una columna dada.

4) Dadas las matrices enteras A de $m \times n$ y B de $r \times t$, realizar las siguientes operaciones de matrices, teniendo en cuenta que en algunas operaciones el tamaño de las matrices guarda cierta relación. Por Ejemplo, en la operación iii, n debe ser igual a t (deben tener la misma cantidad de columnas).

a. Intercambiar dos filas dadas
b. Intercambiar dos columnas dadas
c. Sumar filas dadas: A(i,k) := A(i,k) + B(j,k), suma de las filas i de A y j de B.
d. Sumar columnas dadas: A(k,i) := A(k,i) + B(k,j), suma de las columnas i de A y j de B.
e. Multiplicar una fila por un factor constante: A(i,k) := A(i,k) · factor.
f. Multiplicar una columna por un factor constante: A(i,k) := A(i,k) · factor.

5) Dada una matriz A de $m \times n$ y una B de $n \times r$, obtener una matriz C de $m \times r$, que contenga el producto de las dos primeras sabiendo que:

$$C(i, j) := \sum_{k=1}^{n} A(i, k) \cdot B(k, j)$$

6) Verificar que una matriz cuadrada de 10 elementos de lado es palíndrome (capicúa en todas sus filas y columnas).

7) Dada una matriz de $m \times n$ enteros, ordenar cada una de sus filas.

8) Dada una matriz de $m \times n$ caracteres, ordenar sus filas alfabéticamente como si fueran palabras.

9) Dada la matriz Lluvias definida como:

```
type
    dias = 1..31;
    meses = (ene,feb,mar,abr,may,jun,jul,ago,sep,oct,nov,dic);
var
    Lluvias : array [meses,dias] of integer; {se suponen todos los meses de 31 dias}
```

Calcular: lluvia total caída y el promedio diario. Modularizar realizando los procedimientos y funciones correspondientes.

10) Se tiene la matriz Temperatura que contiene las temperaturas mínima y máxima que se registraron para cada día del año 2004. Esta matriz tiene las siguientes dimensiones: [1..12, 1..30, 1..2]. Las filas se corresponden con los meses, las columnas con los días (sólo se consideran meses de 30 días) y en la tercera dimensión se almacena la temperatura mínima (en la primera posición) y la máxima (en la segunda). Estas temperaturas mínimas y máximas se registran para cada día del año.

a. Realizar el diagrama de estructura y codificar un procedimiento o función que, dada la matriz Temperatura, muestre los meses ordenados en forma creciente teniendo en cuenta las temperaturas máximas.

b. Realizar el diagrama de estructura y codificar un procedimiento o función que, dada la matriz Temperatura, genere el arreglo BajoCero de 12 elementos. En cada componente se debe almacenar un valor TRUE si en ese mes hubo alguna temperatura bajo cero y FALSE en caso contrario.

c. Realizar el diagrama de estructura y codificar un procedimiento o función que, a partir del arreglo BajoCero muestre los meses con temperaturas bajo cero.

d. Realizar el diagrama de estructura (completo) y codificar el programa principal que lea la matriz Temperatura e invoque los módulos codificados en los ejercicios 1, 2 y 3. Incluir todas las declaraciones necesarias.