METODOLOGÍA DE LA PROGRAMACIÓN

Grado en Ingeniería Informática
Primer curso. Segundo cuatrimestre
Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

NOMBRE Y APELLIDOS: **VÍCTOR MONSERRAT VILLATORO.**
GRUPO: **8.**

1.- ENUNCIADO.

Un polinomio es una expresión algebraica de la forma:

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_1 x + a_0$$

A cada $a_i x^i$ se le denomina monomio, siendo a_i el coeficiente del monomio e i el exponente del monomio. Se denomina polinomio a la suma algebraica de varios monomios.

Un polinomio se puede representar como una lista enlazada. El primer nodo de la lista representa el primer monomio del polinomio, el segundo nodo el segundo monomio del polinomio, y así sucesivamente. Cada nodo representa un monomio del polinomio y tiene como campo dato el coeficiente del monomio (a) y el exponente (e).

Escribe un programa que permita:

• Crear un polinomio. El programa preguntará al principio cuántos monomios tendrá el polinomio .
• Obtener una tabla de valores de un polinomio para valores de x = 0.0, 0.5, 1.0, 1.5, ..., 5.0
• Para el polinomio (1) tendríamos la siguiente salida:
(x = 0.0, 3), (x = 0.5, 4), (x = 1.0, 5), (x = 1.5, 6), ..., (x = 5.0, 13)
• Eliminar del polinomio el monomio con exponente E .

Implementa para ello las siguientes funciones:

Función evaluaPolinomio. Evalúa el polinomio para un valor concreto de x.

2.- DATOS DE ENTRADA.

Parámetros formales:

- Nombre: **polinomio.**
- Significado: **puntero que apunta al primer monomio del polinomio.**
- Tipo de dato: **struct monomio * (puntero a estructura).**
- Restricciones: **el número de monomios (nodos) tiene que ser positivo y no mayor que n.**

- Nombre: **x.**
- Significado: **valor para evaluar el polinomio.**
- Tipo de dato: **float (real).**

- Entrada: **por teclado, valor pedido al usuario.**

Parámetros locales:

- Nombre: **aux.**
- Significado: **puntero auxiliar que apunta al primer monomio del polinomio.**
- Tipo de dato: **struct monomio * (puntero a estructura).**
- Restricciones: **el número de monomios (nodos) tiene que ser positivo y no mayor que n.**

- Nombre: **valor.**
- Significado: **valor del polinomio en un punto x.**
- Tipo de dato: **float (real).**
- Salida: **Se devuelve el valor a la función principal y se muestra por pantalla.**

3.- RESULTADOS o SALIDA.

- Nombre: **valor.**
- Significado: **valor del polinomio en un punto x.**
- Tipo de dato: **float (real).**
- Salida: **Se devuelve el valor a la función principal y se muestra por pantalla.**

4.- DATOS AUXILIARES (OPCIONAL).

- Nombre: **aux.**
- Significado: **puntero auxiliar que apunta al primer monomio del polinomio.**
- Tipo de dato: **struct monomio * (puntero a estructura).**
- Restricciones: **el número de monomios (nodos) tiene que ser positivo y no mayor que n.**

5.- DESCRIPCIÓN DEL ALGORITMO.

Se recorrerá la lista de monomios evaluando cada uno individualmente con el valor pedido al usuario que se introducirá por teclado, se sumarán todos los valores y se obtiene la evaluación del polinomio completo.

Ejemplo: $5x^3+3x^2+2x+4$.

Con x = 2 → $5*2^3+3*2^2+2*2+4 = 60$.

6.- PSEUDOCÓDIGO DEL ALGORITMO.

```
Función evaluaPolinomio (polinomio: puntero a estructura lista, x: real)
Inicio
    puntero a aux ← NULO
    valor ← 0
    aux ← polinomio
    Mientras aux ≠ NULO
        valor ← valor + aux (coeficiente) * x^aux (exponente)
        aux ← aux (siguiente)
    Finmientras
    Devolver valor
Finfunción
```

7.- CÓDIGO DEL ALGORITMO EN LENGUAJE C.

```
void evaluaPolinomio (struct monomio *polinomio, float x)
{
	struct monomio *aux = NULL;
	float valor=0;

	aux = polinomio;
	while (aux != NULL)
	{
		valor = valor + aux->coeficiente * pow (x, aux->exponente);
		aux = aux->siguiente;
	}
	return (valor);
}
```

1.- ENUNCIADO.

Para mover los contenedores de mercancía de un importante puerto comercial se utiliza una filosofía basada en pilas. De este modo, el contenedor más abajo en la pila fue el primero que se apiló, y, para moverlo, es necesario mover a otra pila todos los contenedores que hay encima de él. Cada contenedor de mercancía está identificado por un código entero, X. Por motivos de seguridad, como mucho se pueden apilar N contenedores en una misma pila. De este modo, si la pila no está llena, entonces se puede apilar un nuevo contenedor. Si se desea sacar un contenedor de código X, entonces:

• Se deben desapilar previamente los contenedores encima de él colocándolos en una nueva pila auxiliar.
• Se extrae el contenedor X y se vuelven a introducir los contenedores extraídos previamente. Codifica un programa que, utilizando las funciones push (apilar), pop (desapilar) y vacia que están implementadas en la biblioteca pilas.a, permita gestionar una pila de contenedores con la siguiente funcionalidad:

Función para conocer si un contenedor de código X está en la pila.

2.- DATOS DE ENTRADA.

Parámetros formales:

- Nombre: **contenedores.**
- Significado: **puntero que apunta al primer contenedor.**
- Tipo de dato: **struct pila * (puntero a estructura).**
- Restricciones: **el número de contenedores (nodos) tiene que ser positivo y no mayor que N.**

- Nombre: **n.**
- Significado: **código a comprobar si está en la pila.**
- Tipo de dato: **int (entero).**
- Entrada: **por teclado, valor pedido al usuario.**

Parámetros locales:

- Nombre: **encontrado.**
- Significado: **parámetro que adquirirá el valor de 1 cuando encuentre un contenedor con ese mismo código.**

- Tipo de dato: **int (entero).**
- Restricciones: **valdrá 0 hasta que el contenedor sea encontrado.**
- Salida: **el valor será devuelto a la función principal y se mostrará por pantalla al usuario si se encontró o no el contenedor.**

- Nombre: **X.**
- Significado: **variable donde iremos almacenando el número del contenedor que vayamos desapilando.**
- Tipo de dato: **int (entero).**

- Nombre: **aux.**
- Significado: **estructura auxiliar para ir apilando los contenedores desapilados.**
- Tipo de dato: **struct pila * (puntero a estructura).**

3.- RESULTADOS o SALIDA.

- Nombre: **encontrado.**
- Significado: **parámetro que adquirirá el valor de 1 cuando encuentre un contenedor con ese mismo código.**
- Tipo de dato: **int (entero).**
- Restricciones: **valdrá 0 hasta que el contenedor sea encontrado.**
- Salida: **el valor será devuelto a la función principal y se mostrará por pantalla al usuario si se encontró o no el contenedor.**

4.- DATOS AUXILIARES (OPCIONAL).

- Nombre: **aux.**
- Significado: **estructura auxiliar para ir apilando los contenedores desapilados.**
- Tipo de dato: **struct pila * (puntero a estructura).**

5.- DESCRIPCIÓN DEL ALGORITMO.

Se recorrerá la pila de contenedores desapilándolos y apilándolos en una pila auxiliar. Se comprobará uno a uno el código de cada contenedor para comprobar si es el mismo que ha sido introducido por el usuario. Una vez encontrado se irán apilando los contenedores de la pila auxiliar a la original. Por último, se devolverá a la función principal si ha sido o no encontrado el contenedor.

Ejemplo: código a comprobar 35682.

Códigos de la pila: 68230, 24519, 35682, 15024.

Se devolverá un 1, ya que ha sido encontrado.

6.- PSEUDOCÓDIGO DEL ALGORITMO.

```
Función buscarContenedor (contenedores: puntero a estructura pila, n: entero)
Inicio
    encontrado ← 0
    aux ← NULO
    Mientras contenedores ≠ NULO
        X ← Invocar función desapilar (contenedores)
        Si X = n
```

```
                    Entonces encontrado ← 1
                Finsi
                Invocar función apilar (aux, X)
            Finmientras
            Mientras aux ≠ NULO
                X ← Invocar función desapilar (aux)
                Invocar función apilar (contenedores, X)
            Finmientras
            Devolver encontrado
        Finfunción
```

7.- CÓDIGO DEL ALGORITMO EN LENGUAJE C.

```
int buscarContenedor (struct pila **contenedores, int n)
{
        int encontrado=0, X;
        struct pila *aux;

        aux = NULL;

        while (*contenedores != NULL)
        {
                X = pop (contenedores);
                if (X == n)
                {
                        encontrado = 1;
                }
                push (&aux, X);
        }
        while (aux != NULL)
        {
                X = pop (&aux);
                push (contenedores, X);
        }
        return (encontrado);
}
```