Tecnólogo en Informática - Principios de Programación - curso 2007

Práctico 14: Problemas

Ejercicio 1
Escribir la función **cantidadOcurrencias** que tiene el siguiente cabezal:

```
int cantidadOcurrencias(int num, int arr[], int tamArr)
```

Esta función retorna la cantidad de veces que aparece **num** en **arr**. Los números del arreglo **arr** están ordenados en forma creciente.
Si **num** no está en **arr**, la función debe reotornar el valor 0.
La solución implementada **debe tomar ventaja del ordenamiento del arreglo** para no recorrer celdas innecesariamente.

Ejercicio 2
Implementar la función:

```
int function EnQueFila(char car, int matriz[][], int cantCol, int cantFil)
```

Esta función se comporta de la siguiente manera:

- Si el carácter **car** aparece en una sola fila, retorna el índice de esa fila.
- Si no, devuelve 0 (si **car** aparece en más de una fila o no aparece en ninguna).

Ejercicio 3
Se cuenta con las siguientes constantes y los siguientes tipos para representar oraciones y palabras de largo variable:

```
#define MAX_OR 20
#define MAX_PAL 10

typedef struct {
       char palabra[MAX_PAL];
       int largo_pal;
} TPalabra;

typedef struct {
       TPalabra oracion[MAX_OR];
       int largo_or;
} TOracion;
```

a) Escribir una función que, dadas dos palabras, devuelva el valor 1 si son iguales y el valor 0 si no lo son. La función deberá tener el siguiente cabezal:

```
int palabras_iguales(Tpalabra pal1,pal2)
```

b) Escribir una función que, dadas una oración y una palabra, devuelva la cantidad de apariciones de la palabra en la oración. La función deberá tener el siguiente cabezal:

```
int cantidad_ocurrencias(Toracion ora, Tpalabra pal)
```

c) Escribir una función que, dada una oración y un número entero positivo n, devuelva la primera palabra de la oración de largo n. La función deberá tener el siguiente cabezal:

```
Tpalabra palabra_N(Toracion ora, int n)
```

Nota: Asumir que todas las oraciones y palabras dadas tienen al menos un elemento.

Ejercicio 4
Escribir un programa que lea de la entrada un número **n** entero mayor o igual que 0 y un número **x** real distinto de 0, y despliegue el valor de la siguiente suma:

$$\text{suma} = 1 + x + \frac{x2}{2!} + \frac{x3}{3!} + \text{--------} + \frac{xn}{n!}$$

El programa sólo puede hacer sumas, multiplicaciones y divisiones. La cantidad de multiplicaciones no puede ser mayor que el doble de **n** y la cantidad de divisiones no puede ser mayor que **n**. No se pueden utilizar funciones auxiliares ni invocar funciones de ninguna biblioteca estándar, salvo la de entrada y salida.

Ejercicio 5

Sea la siguiente estructura de datos que representa polinomios con coeficientes reales de grado <= **N**.

```
#define N 7
typedef struct {
	float coefs[N];
	int grado;
} Polinomio;
```

Si el polinomio es

$P(x) = c0 + c1 * x + ... + cn * x^n$

entonces en esta estructura el arreglo **coefs** contiene en la posición **i** el coeficiente **ci** y en el campo **grado** el valor **n**.

Implemente una función que recibe dos polinomios y retorna el polinomio suma de los anteriores.

Ejercicio 6

Se consideran las siguientes definiciones:

```
#define N 6

typedef struct {
	float x,y;
} Punto;

typedef struct {
	Punto arrPuntos[N];
	int cantPuntos;
} Puntos;

typedef struct {
	Punto unExtremo,otroExtremo;
} Segmento;
```

Escribir la siguiente función:

```
Segmento minimoSegmento(Puntos pts)
```

Esta función recibe en **pts** un conjunto de puntos y retorna el segmento más corto de todos los segmentos no nulos, cuyos extremos pertenecen a **pts**. Si hay más de un segmento de largo mínimo retorna uno cualquiera. Se supone que los puntos son todos distintos.