Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI 2611 Algoritmos 1
Abril-julio 2012

# Tarea 6

1.- Dada la siguiente especificación que separa una cantidad total de minutos en días, horas y minutos, proponga usando análisis descendente las acciones globales que llevan a resolver el problema.

```
[
   const
      totalM : entero;
   var
      d, h, m : enteros;
   { totalM ≥ 0 }
   SepararDiasHorasMinutos
   { totalM = d*24*60+h*60+m ∧ 0≤d ∧ 0≤h<24 ∧ 0≤m<60 }
]
```

2.- Dada una secuencia de números enteros verificar si existen dos números en la secuencia que sumen N. Aplique análisis descendente comenzando por un algoritmo que genera cada par de números de la secuencia y comprueba si suman N.

3.- Sean sec1 y sec2 secuencias de caracteres representadas con arreglos. Determinar si sec2 es subsecuencia de sec1. Suponga que sec1 tiene N (≥ 0) elementos, de 0 a N-1, y sec 2 tiene M (≥ 0) elementos, de 0 a M-1. sec2 es subsecuencia de sec1 si y sólo si M≤N y existen p y q enteros con 0≤ p ≤ q ≤ N, p-q = M, y (∀i: 0≤i<M: sec2[i] = sec1[p+i]).

4.- Se quiere escribir una función que reciba un arreglo de enteros de tamaño N y un número entero K, y dependiendo del valor de K calcule: Si K=1 entonces calcula el máximo del arreglo, Si K=2 entonces calcula el promedio del arreglo, se supone que el valor de K es válido cuando se llama a la función.

5.- Dada la siguiente especificación de un procedimiento, escriba el cuerpo del mismo y luego un programa que determine si todos los elementos de un arreglo son diferentes. La postcondición del programa es:

```
{todosdif ≡ (∀i,j: 0≤i,j<N : i≠j ⇒ A[i] ≠ A[j])}

proc diferenteDesdeK(in N:entero; in A:arreglo[0..N) de enteros;
                     in k:entero; out dif:boolean)
  { pre : N>0 ∧ 0≤k<N }
  { post : dif ≡ (∀i: k≤i<N-1 : A[i+1] ≠ A[k]) }
```

6.- Para los siguientes problemas escriba un procedimiento. Tenga en cuenta qué parámetros deben ser de entrada de salida o de entrada-salida. No olvide colocar las pre, post-condición:

a) Contar el número de veces que aparece un elemento `e` en el arreglo `V`
b) Calcular el número de valores diferentes que se encuentran en un arreglo. En este caso escriba una función que dada una posición i del arreglo verifique si su valor aparece en alguna de las posiciones anteriores a i.
c) Dado un elemento `e` y un valor `i`, encontrar la posición de la `i`-ésima ocurrencia del elemento en un arreglo `V`. Ejemplo, si el arreglo es V=<4,5,3,4,1,7,4,8,9> la 3era ocurrencia de 4 en el arreglo está en la posición 6. Si no existe esa ocurrencia devuelva -1.

d) Encontrar el elemento que aparece el mayor número de veces en un arreglo. Para ello, escriba una función que dado un valor diga cuántas veces aparece ese valor en el arreglo. Use esta función para realizar el procedimiento.

e) Dado un polinomio `P` de grado `m`, almacenado en un arreglo, encontrar la evaluación del polinomio para el valor `X`. Escriba una función que calcula el monomio a*x^n, y utilícelo en su programa. Tambien puede utilizar la descomposición: a*x^3+b*x^2+c*x+d = (((a*x + b)*x + c)* x +d, y tener una función que calcula f(z,x,y) = z*x+y . ¿Cuál solución realiza menos multiplicaciones?.

f) Dados dos arreglos `S1` y `S2` de caracteres, devuelva `0` si son iguales, `1` si `S1 > S2` y `-1` si `S1 < S2`. Usando el orden alfanumérico. Puede suponer que los caracteres son comparable es decir 'A' < 'B', 'B' < 'C', y así sucesivamente

g) Dados i y n como entradas, devuelva el cálculo de la sumatoria $\sum_{x=i}^{n} x!$ .

h) Dada una secuencia de n enteros, devuelva la cantidad de números positivos, la cantidad de números negativos y la cantidad de valores nulos que hay en la secuencia.

7.- Realice la corrida en frio de los siguientes programas:

a)

```
[
   var
      a,b,c :enteros ;
   a,b,c:=5,8,3;
   pr(a,b,c);
   pr(7,a+b+c,a);
   pr(a*b,a/b,c);
}
proc pr :(in x :entero,in y :entero,in-out z :entero)
[
      z:=x+y+z;
]
```

b)

```
[
   var
     k,x:entero;
   k:=1;
   x:=2;
   r1(x,x,k);
   r1(k,k,x);
}
proc r1 :(in-out a:entero, in b:entero, out y:entero)
{
   var
      i:entero;
   i:=0;
   a:=2*b;
   y:=r3(i,a,b);
}
func r2:(i:entero,a:entero,k:entero)->entero
{
   var j:entero;
   j:=i+3;
   if (k ≤ 2) -> j:=j+1
   [] (k > 2) -> skip
   fi
   >>j+k+a-i;
}
```

```
func r3:(a:entero, i:entero, k:entero)->entero
{
   int x,y;
   y:=i+k;
   x:=r2(y,a,k);
   y:=r2(2*a,i-1,x));
   >>y
}
```