Proyecto 4

Algoritmos y Estructuras de Datos I
Laboratorio

15 de noviembre de 2010

El objetivo del proyecto es desarrollar programas en lenguaje C en base al formalismo visto en el teórico de la materia. La idea general es derivar o demostrar los programas y traducirlos al lenguaje C, agregando las partes correspondientes a la entrada/salida con las herramientas que nos brinda este último lenguaje. Para hacerlo hay que tener en cuenta:

- La implementación en lenguaje C debe hacerse traduciendo el resultado de la derivación o demostración en el formalismo del teórico al lenguaje C, tal como se vio en el teórico del laboratorio. Para aprobar el proyecto se debe presentar la derivación o demostración del algoritmo con su resultado final separado, de modo que coincida con el programa escrito en C.
- No escribir una línea de código en la computadora hasta que no haya demostrado lo que se pide en cada ejercicio y se tenga el programa escrito en el lenguaje del teórico.
- Los programas deben tomar los datos de entrada del usuario y mostrar los resultados en pantalla.

  Recordar que la entrada debe chequear el cumplimiento de la precondición.
- Los programas deben ser compilados con las opciones `-ansi -pedantic` del compilador gcc. Al hacerlo no debe aparecer ningún mensaje de error o de alerta.
- Recordar que la entrada debe chequear el cumplimiento de la precondición en la especificación del programa.
- No usar ninguna variable global.

Ejercicios

1. Escriba en lapiz y papel la definición funcional de la función máximo entre dos números (`max`). Luego derive el programa imperativo (en lapiz y papel y en el lenguaje del teórico) en base a la siguiente especificación:

   $$\{a = A \wedge b = B\}$$
   $$\mathrm{S}$$
   $$\{res = \texttt{max}.\, A.\, B\}$$

   Al final, traduzca el programa al lenguaje C agregando la entrada/salida.

2. Escriba en lapiz y papel la definición funcional de la función `edad` que dadas dos fechas representadas por 3 enteros (dia, mes y año), la función devuelve los años transcurridos entre la primera y la segunda. Luego derive el programa imperativo (en lapiz y papel y en el lenguaje del teórico) en base a la siguiente especificación:

$$\{d1 = D1 \land m1 = M1 \land a1 = A1 \land d2 = D2 \land m2 = M2 \land a2 = A2\}$$

$$\mathrm{S}$$

$$\{res = \texttt{edad}.\, D1.\, M1.\, A1.\, D2.\, M2.\, A2\}$$

3. Derive y programe en C el programa que calcula el cociente y resto de la división (ejemplo 19.1 del libro).

4. Derive y programe en C el programa que calcula el máximo común divisor (hecho en el teórico y en ejemplo 17.6 del libro).

5. Derive y programe en C el programa que calcula la exponenciación (ejercicio 18.1, punto (a) del libro).

6. Derive y programe en C el programa que calcula la exponenciación de manera eficiente (ejercicio 18.1, punto (b) del libro).

7. Derive y programe en C el programa que calcula la mayor potencia de 2 menor o igual que cierto número (segundo ejercicio, punto (b) del capítulo "Introducción al cálculo de programas imperativos").

8. Derive y programe en C el programa que calcula el cociente y resto de la división de forma mejorada (ejemplo 19.3 del libro).

   Punto $\star$: Discutir e implementar alguna forma de comparar el tiempo de ejecución de este programa con el anterior.

9. Derive y programe en C el programa que calcula el cubo usando solamente sumas (ejemplo 19.6 del libro).

10. Derive y programe en C el programa que calcula fibonacci (ejemplo 19.7 del libro).

    Punto $\star$: Existe alguna implementación más eficiente de fibonachi ($\mathcal{O}(\log n)$)? (sin usar funciones de números reales). Si la hay, impleméntela.

    Punto $\star$: Discutir e implementar alguna forma de comparar el tiempo de ejecución de este programa con el anterior.

11. Derive y programe en C el programa que calcula el mínimo común múltiplo (ejercicio 19.1 del libro).

12. Derive y programe en C el programa del ejercicio 19.2.1 del libro.

13. Derive y programe en C el programa que determina si algún elemento de un arreglo es igual a la suma de los elementos que lo preceden (ejemplo 19.8 del libro).

14. Derive y programe en C el programa que calcula la suma del segmento de suma máxima (ejemplo 19.9 del libro).

15. Derive y programe en C el programa que determina si algún elemento en un arreglo es positivos (ejercicio 19.3 del libro).

16. Derive y programe en C el programa que dice si los paréntesis en un string están balanceados (ejemplo 20.3 del libro).

    Punto $\star$: Al momento de programar los algoritmos anteriores se puede definir el tamaño del arreglo como una constante `N` (definida mediante `#define`). ¿Se puede hacer que este tamaño sea decidido por el usuario? Lea la info page de libc donde se habla de arreglos, responda a la pregunta y lea si la posible solución tiene algún inconveniente. Programar todos los ejercicios anteriores según la respuesta.

17. **Punto $\star$:** Hacer todos lo programas anteriores en funciones separadas. La entrada y salida de valores también deben estar en funciones separadas.