

Proyecto 4

Algoritmos y Estructuras de Datos I Laboratorio

14 de noviembre de 2011

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:

- Antes de escribir el programa en lenguaje C, derivar o demostrar el algoritmo en el lenguaje del teórico a partir de su especificación. 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 `-pedantic -Wall -std=c99` 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:

$$\{a = A \wedge b = B\}$$

S

$$\{res = \max. A. B\}$$

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 (día, 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:

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

S

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

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 determina si algún elemento de un arreglo es igual a la suma de los elementos que lo preceden (ejemplo 19.8 del libro).
8. Derive y programe en C el programa que calcula la mayor potencia de 2 menor o igual que cierto número mayor que cero (ejercicio 18.2 del libro¹).
9. 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 *: Discutir e implementar alguna forma de comparar el tiempo de ejecución de este programa con el anterior.
10. Derive y programe en C el programa que calcula el cubo usando solamente sumas (ejemplo 19.6 del libro).
11. Derive y programe en C el programa que calcula fibonacci (ejemplo 19.7 del libro).
Punto *: Existe alguna implementación más eficiente de fibonacci ($\mathcal{O}(\log n)$)? (sin usar funciones de números reales). Si la hay, impleméntela.
Punto *: Discutir e implementar alguna forma de comparar el tiempo de ejecución de este programa con el anterior.
12. Derive y programe en C el programa que calcula la suma del segmento de suma máxima (ejemplo 19.9 del libro).
13. Derive y programe en C el programa que determina si algún elemento en un arreglo es positivos (ejercicio 19.3 del libro).
14. Derive y programe en C el programa que dice si los paréntesis en un string están balanceados (ejemplo 20.3 del libro).

Punto *: 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 los ejercicios anteriores con arreglos según la respuesta.

¹En el libro, la precondition y el invariante contienen un error de tipeo. Escriba los correctos.