Programación avanzada
Primer Programa: evalua

Prof. Miguel A. Pizaña 19 de Noviembre de 2007

Objetivo. Hacer un programa que evalúe expresiones aritméticas y convierta expresiones infijas a postfijas.

Expresiones aritméticas. Las expresiones aritméticas que el programa debe aceptar son todas las expresiones aritméticas que pueden construirse usando: dígitos, paréntesis, suma, resta, multiplicación división y exponenciación. Las expresiones aritméticas pueden contener espacios y el programa debe ignorarlos. Ver ejemplos en la sección de formas de invocación.

Simplificaciones. Para simplificar la tarea de programación, el programa sólo está obligado a reconocer los números del 0 al 9 en las expresiones aritméticas que lee. Se otorgará puntaje extra, si es capaz de leer números enteros arbitrarios, siempre y cuando cumpla también con los requisitos obligatorios. Sin embargo, el cálculo debe hacerse en punto flotante de doble presición y debe imprimirse usando "%g" en el printf. El programa no está obligado a reconocer el "menos unario" (por ejemplo en expresiones como: "-3" o "5+(-3)"). Se otorgará puntaje extra si el programa puede procesar el menos unario correctamente (y distingirlo de la resta) siempre y cuando cumpla también con los requisitos obligatorios.

Formas de invocación. El programa debe correr en GNU/linux y admitir varias formas de invocación: Con la opción "-h" debe desplegar un mensaje de ayuda al usuario y salir (en este caso no debe procesar la expresión aritmética aunque estuviera presente). Con la opción "-p" debe convertir la expresión aritmética a una expresión postfija (y no evaluar). Caso contrario debe evaluar la expresión aritmética para obtener un resultado numérico. La expresión aritmética en sí, debe leerse de la misma linea de comandos (después de la opción "-p" en su caso) y si no se encuentra ahí debe leerse de la entrada estándar (stdin). En ningún caso el programa funcionará de manera interactiva (preguntando al usuario la expresión, por ejemplo). Algunos ejemplos se muestran a continuación (en negritas lo que escribe el usuario, seguido de lo que contesta el programa):

```
echo 2+3/2 | evalua
3.5
echo 4*(2+3) | evalua -p
4 2 3 + *
evalua 4+ 2*          5
14
evalua -p 3*(5+4)^2
3 5 4 + * 2 ^
evalua -h
<poner aquí una descripción de lo que hace el programa
y sus formas de invocación>
```

Buen estilo. El programa debe cumplir con todos los principios de la buena programación que se estudiaron en clase. En particular, el programa debe estar modularizado, el código debe ser claro y estar bien documentado. Una parte muy importante de la calificación depende de satisfacer estos requerimientos.

Robustez. El programa debe ser robusto y no caerse o presentar fallos de segmentación. En particular debe resistir el embate de expresiones aritméticas mal formadas emitiendo en todos los casos el mensaje de error pertinente, liberando toda la memoria que se haya usado y terminando la ejecución con `exit(1)`. También debe resistir en embate de expresiones aritméticas arbitrariamente grandes (en la medida que la memoria de la computadora lo permita). Cuando hayan ocurrido errores de cualquier tipo el programa debe terminar con `exit(1)`, caso contrario debe terminar con `exit(0)`. Una parte muy importante de la calificación depende de satisfacer estos requerimientos

Forma de entrega. Deberán entregar un sobre con los nombres de los integrantes del equipo (máximo 4) que contenga: (1) un disquete con los archivos .c y .h que conforman su programa y el Makefile. (2) El código impreso de todos los archivos de su programa (trate de no usar demasiado papel). El sobre deberá tener aderida por fuera la forma de evaluación. La fecha de entrega es **3 de Diciembre de 2007**. Se descontará un punto de calificación por cada día hábil (= día de clase) de entrega tardía, hasta un máximo de 3 puntos. Para este efecto, la hora de corte de la entrega es la hora en la que acaba la clase.