Algoritmos y Estructuras de Datos I - Laboratorio
Proyecto 4

Programación imperativa en C

1. Objetivo

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 del **práctico 3** 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 terminar el ejercicio correspondiente donde se demuestra o deriva el programa escrito en el lenguaje del teórico a partir de su especificación.

  Se debe presentar la derivación o demostración del algoritmo con su resultado final separado, de modo que coincida con el programa escrito en C.

- Se debe chequear que los valores ingresados cumplan la precondición de la especificación del programa.
- Los programas deben ser compilados con las opciones `-Wall -Werror -pedantic` del compilador gcc.

2. Ejercicios y Preguntas

1. Escribir los programas siguientes:

   a) `ejercicio01.c` que lee una variable de tipo `int` en la consola con la función `scanf` y la imprime en la consola con la función `printf`. Este programa tiene sólo una función `main`.

   b) `ejercicio02.c` que lee una variable, la imprime, y vuelve a hacer lo mismo con una segunda variable. Este programa tiene que tener una función `void imprimir(int a)` llamada desde la función `main`.

   c) `ejercicio03.c` con las funciones siguientes:

   - `int main(void)`
   - `void imprimeHola(void)`
   - `void imprimeChau(void)`

   Ese programa tiene que imprimir dos veces "Hola" seguido de dos veces "Chau", llamando a las dos últimas funciones desde el `main`.

   Ayuda: *Se debe entender como corre el flujo de ejecución de este programa leyendo su código fuente.*

   d) `ejercicio04.c` con una función `void compara(int a, int b)`, que imprime tres mensajes distintos según como se comparan sus dos argumentos. El `main` pide dos numeros en la entrada antes de llamar `compara`.

e) `ejercicio05.c` con una función `void holaHasta(int a)`, que dado un `int a`, imprime a veces "Hola!". Si a es igual o inferior a 0, imprimir "Error". (Usar una bucle `while`). El `main` pide un número en la entrada antes de llamar `holaHasta`.

f) `ejercicio06.c` similar al último, pero que usa la función `assert` (ver teórico) para chequear que $x > 0$.

2. Hacer el ejercicio 4 del práctico (intercambio de dos variables) y traducirlo al lenguaje C.
3. Hacer el ejercicio 7.a del práctico (cálculo del mínimo) y traducirlo al lenguaje C.

   Ayuda: *Utilice el enunciado del ejercicio 9 del práctico para traducir a C un* **if** *con dos guardas.*

4. Hacer el ejercicio 7.b del práctico (cálculo del valor absoluto) y traducirlo al lenguaje C.
5. Agregar a los programas en C de los dos ejercicios anteriores todas las anotaciones (pre y poscondiciones y predicados intermedios) de los programa derivados en el práctico con `assert`.

   Ayuda: *Sólo se pueden usar operaciones básicas (* `&&`, `||`, `==`, `>`, *. . .). Pensar cómo se traduce a un predicado el resultado de la función.*

6. Traducir al lenguaje C el ejercicio 5.c) del práctico.
7. Hacer los ejercicios 10.a, 10.b y 10.c del práctico y traducir el resultado al lenguage C. Hacer funciones en C que calculen y devuelvan el resultado (una para cada ejercicio).
8. Utilizar la función del ejercicio anterior que calcula el máximo común divisor para hacer un programa que calcule el mínimo común múltiplo.
9. Traducir el programa del ejercicio 11.b del práctico y encontrar el error con **gdb**.