Proyecto 1

Algoritmos y Estructuras de Datos I
Laboratorio

10 de marzo de 2011

El proyecto consiste en realizar dos implementaciones del TAD Número Complejo en el lenguaje C.

Para hacer los programas tener en cuenta:

- Para implementar el TAD hay que utilizar estructuras en C y archivos separados según se indique.
- Cuando se creen "headers" (archivos `.h`) hacer la cerradura `ifndef` como se explicó en el teórico.
- Compilar todos los archivos y linkear el programa con las opciones `-pedantic`, `-Wall` y `-std=c99`. Ver que no haya mensajes de error o advertencias.
- No usar ninguna variable global.
- Para obtener `B+` todos los ejercicios y puntos estrella deben estar hechos en forma correcta.
- Para obtener `B` todos los ejercicios deben estar hechos en forma correcta.

El proyecto consta de las siguientes partes:

1. Escribir un archivo separado para los números reales llamado real.h e implementarlo como un sinónimo del tipo `long double`.
2. Escribir en archivos separados la definición del tipo complejo como un par parte real y parte imaginaria, su constructor y demás funciones (TAD) siguiendo el esquema de los archivos:
   - En el archivo complejo.h definir el tipo como una estructura y escribir los prototipos. El archivo debe quedar como el que figura en el apéndice A.
   - En el archivo complejo.c, incluir el archivo complejo.h e implementar todas sus funciones.
3. Escribir en otro archivo el bloque *main* que implemente una interfase con el usuario donde el mismo pueda cargar 2 números complejos `x` e `y` por teclado y muestre por pantalla el valor:

```
a * x - b * y + c
```

con

```
a = 2 + i * 3
b = 8 + i
c = 5 + i * 7
```

fijos.

4. Derive y programe en C el programa que calcula la exponencial (ejercicio 18.1, punto (a) del libro de Algoritmos I). La función debe estar programada en archivos separados (`.h` y `.c`) y su signatura debe ser

   ```
   compl expcompl(compl x, int y);
   ```

   Programar una interfase con el usuario para probar la función.

5. Después de terminar este programa, hacer otra implementación donde el número complejo se represente por su módulo y su ángulo. Para hacer esto, cambiar únicamente el archivo complejo.c y la definición de la estructura en el archivo complejo.h.

6. Utilizar las mismas interfaces con el usuario de los items anteriores para probar esta nueva implementación.

7. Comparar el tiempo que tardan ambas implementaciones para calcular $(3 + i(-6))^{5000}$, 100000 veces.

   Ayuda: *Se puede hacer un bloque main que invoque 100000 veces la función* `expcompl` *mediante un bucle. Para ver el tiempo se puede usar el comando de consola* `time` *(ver su man page).*

   Si hay diferencias analizar sus causas.

Punto ⋆ 1 *Si se cambia la definición del tipo* `real` *por* `float` *(cambiando el tipo sinónimo solamente) el programa compila con advertencias. Encuentre las causas de las mismas y busque e implemente una manera para poder cambiar la definición de aquel tipo sin estos problemas.*

Punto ⋆ 2 *Con el archivo complejo.h del apéndice ¿se ocultó la implementación del TAD? Si no es así, ¿se le ocurre alguna manera de hacerlo?*
Si la última respuesta es afirmativa programe el TAD de esa manera y vuelva a hacer el proyecto con el mismo.

Apéndice. complejo.h

```
#ifndef COMPLEJO_H
#define COMPLEJO_H

#include <real.h>
```

```
struct scompl {
  real rl, img;
};

typedef struct scompl compl;

compl
compl_create(const real rl, const real img);
/*
DESC: Constructor del tipo
*/

compl
compl_suma(const compl a, const compl b);
/*
DESC: Suma.
*/

compl
compl_resta(const compl a, const compl b);
/*
DESC: Resta.
*/

compl
compl_prod(const compl a, const compl b);
/*
DESC: Multiplicacion.
*/

void
compl_print(const compl c);
/*
DESC: Imprime en pantalla c con formato a + i b.
*/

compl
compl_leer(void);
/*
DESC: Lee por teclado en el formato a + i b y construye.
*/

#endif /* COMPLEJO_H */
```