Práctica No. 10
Macros en C

Introducción
Las macros son secuencias de instrucciones que el usuario de un programa puede escribir y almacenar bajo un nombre para realizar tareas complejas o repetitivas. Las macros se definen especificando: *i*) el nombre de la macro, *ii*) los parámetros de la macro, si los hay y *iii*) el cuerpo de la macro. Antes de comenzar la traducción, el procesador de macros analiza el programa y cada vez que encuentra un nombre de macro lo sustituye por el cuerpo de la macro instanciando los parámetros de la misma, si los hay.

El lenguaje C cuenta con un procesador de macros: el preprocesador. Como vimos al estudiar las etapas de compilación de C, el preproceso es la primera fase de la compilación. El preprocesador es la herramienta de desarrollo en C que nos permite: *i*) Inclusión de archivos, *ii*) Expansión de macros, *iii*) Compilación condicional y *iv*) Control de la numeración de líneas.

Todos los comandos del preprocesador (llamados directivas) operan agregando, borrando o modificando el texto del archivo fuente. Los cambios son temporales y no afectan al archivo fuente en el disco.

Actividades
1.- Consulta las siguientes referencias y contesta las siguientes preguntas:
http://laurel.datsi.fi.upm.es/~rpons/personal/trabajos/curso_c/node88.html
http://garota.fismat.umich.mx/mn1/manual/node13.html

1.1.- Menciona las tareas que realiza el preprocesador:

_____
_____

1.2.- Lista las ventajas que el preprocesamiento aporta a la programación en C:

_____
_____
_____
_____

1.3.- Menciona que tarea nos permite hacer cada una de las siguientes directivas del preprocesador de C:

`#define` _____
`#undef` _____
`#include` _____
`#ifdef` _____

2.- Uso de macros en C
2.1.- Captura el siguiente archivo que define algunas macros, constantes e inclusiones en C:

```
/* ------------------------------------------------------------
 * Nombre del archivo: macros.h
 * Define las variables y macros a utilizar en el prorama preproc1.c
 * ------------------------------------------------------------ */
#define MAX 64                /* Constante MAX */
#define SYSTEM Windows        /* SYSTEM =  Windows */
#define ARREGLO datos[MAX]
#define DOBLE(x) ((x) + (x))

#include <stdio.h>
```

2.2.- Captura el siguiente archivo que usa algunas de las macros.

```
/*-------------------------------------------------------------
 * Nombre del archivo: preproc1.c
 * Ilustra el uso del preprocesasor de C. Utiliza las variables
 * definidas en macros.h y otras macros predefinidas del preprocesados
 * de C.
 * ------------------------------------------------------------ */
#include "macros.h"
int main(void)
{
   int i;

   printf("%s se compila el dia %s a las %s hrs.\n", __FILE__, __DATE__,__TIME__);

   #ifdef ARREGLO
     int ARREGLO;
   #else
     int datos[MAX];
     printf("El arreglo no se definio en macros.h\n");
   #endif

   #ifdef DEBUG
      printf("\n\nVersion de prueba.\n\n");
   #endif

   #ifdef SYSTEM
     printf("En un sistema tipo Windows.\n");
   #endif

   printf("La macro __LINE__ esta ubicada en la linea %d\n",__LINE__);

   for(i=0;i<MAX;i++)
      datos[i]=DOBLE(i);

   for(i=0;i<MAX;i++)
      printf("Datos[%d]=%d\t",i,datos[i]);

   printf("\n");

   return 0;
}
```

2.3.- Desde una ventana DOS ***preprocesa*** el programa anterior tecleando: **`cpp preproc1.c`**
Este comando genera el archivo `preproc1.i` Tamaño `preproc1.i`: _____

3.- Análisis de la salida del preprocesador
Analiza el contenido del archivo `preproc1.i` buscando tu programa al final del mismo y contesta las siguientes preguntas (usa el comando **`more preproc1.i`**):

3.1.- ¿De donde salen todas las líneas de código anteriores a tu programa?

_____
_____

3.2. ¿Qué sucedió con las lineas `#ifdef ... #endif` de tu programa? ¿Porqué?

_____

3.3. ¿Qué sucedió con la macro `DOBLE(x)` de tu programa? ¿Porqué?

_____
_____

3.4.- ¿Qué sucedió con las macros `__FILE__`,`__DATE__` y `__TIME__`?

_____
_____

3.5.- ¿Qué sucedió con la macro `__LINE__`?

_____
_____

4.- Compilación del programa

4.1.- Compila el programa (usa **`tcc preproc1.c`**) lo que debe generar el archivo `preproc1.exe`.

Tamaño `preproc1.exe`: _____

Ejecuta el programa y reporta el resultado de su ejecución:

_____
_____
_____
_____

4.2.- Ahora ***compila*** el programa con el siguiente comando:

```
tcc -DDEBUG preproc1.c
```

Ejecuta nuevamente el programa y comenta diferencias (si las hay) con la salida de la ejecución anterior:

_____
_____
_____
_____

5. Modificaciones al programa

5.1.- Agrega al archivo `macros.h` una macro que calcule la raíz cuadrada[1] de su argumento y agrega también las instrucciones necesarias para utilizarla dentro de un `printf()` en el archivo `preproc1.c`.

Definición de macro en `macro.h`: _____

_____
_____
_____
_____

Instrucciones en `preproc1.c`: _____

_____
_____
_____
_____

Compila y ejecuta el programa modificado y reporta el resultado de su ejecución:

_____
_____
_____
_____

5.2.- Ahora define y agrega al archivo `macros.h` un directiva de compilación condicional que, si está definida la macro DEBUG, agregue al final de cada uno de los dos bloques `for()` la siguiente instrucción:

```
printf("Final bloque for(). Linea: %d\n",__LINE__);
```

1 La función para calcular la raíz cuadrada de x es `sqrt(x)` y requiere la inclusión de la cabecera `math.h`

Definición de macro en `macro.h`: _____

Instrucciones en `preproc1.c`: _____

Compila y ejecuta el programa modificado y reporta el resultado de su ejecución:

5.3.- En el archivo `preproc1.c` cambia la línea

```
#include "macros.h"
```

por

```
#include <macros.h>
```

y recompila el programa con **`tcc preproc1.c`**

¿Qué sucede? ¿Porqué?

6.- Conclusiones y Comentarios