Macros en C

Eva Lucrecia Gibaja Galindo

Dpto. Informática y Análisis Numérico

Macros en C

- La directiva #define define un *alias*, es decir una sustitución de texto (por ejemplo, constantes). Podemos poner parámetros a esas sustituciones, que se comportan entonces como si de una pseudo-función se tratara.
- En una macro con argumentos, los argumentos se sustituyen en el texto de reemplazo, y a continuación la macro se *expande*, es decir, en el programa el texto de reemplazo reemplaza al identificador y a la lista de argumentos. Ni los comentarios ni las cadenas serán afectadas.
- Las macros:
  - Permiten insertar código en el programa directamente, evitando la sobrecarga de invocar a una función (pasar parámetros a la pila, realizar un salto, recibir parámetros, etc.).
  - Permiten realizar cálculos durante la compilación. Así en el ejemplo que nos ocupa el compilador le da directamente el valor adecuado a la variable “a”, en lugar de insertar instrucciones para que se evalúe cada vez que se use.
- Estructura:

```
#define nombreMacro (parametros sin tipo) texto
```

Macros en C

- La macro consta de una única línea.
  - Para especificar varias líneas, situar una \ al final de la primera línea y continuar en la siguiente.
  - Para añadir más líneas, proceder de igual forma.
- Ejemplo:

```
#define PI 3.14
#define AREA_CIRCULO(x) (PI * (x) * (x))
void main()
{ int a;
   a = AREA_CIRCULO(3);
}
```

- Durante la compilación la macro se expande a:
  - a = 3.14 * (3) * (3).

Macros en C

- Otros ejemplos de macros:
  - `#define MAX(A,B) ((A)>(B)?(A):(B))  /* Macro para el máximo */`
  - `#define MIN(A,B) ((A)>(B)?(B):(A))  /* Macro para el mínimo */`
  - `#define CUBO(A) ((A)*(A)*(A))  /* Macro para el cubo */`
  - `#define SUMA(A) ((A)+(A))  /* Macro para la suma */`

- Macros equivocadas:
  - `#define CUBO_MAL(A) A*A*A`
    `a=CUBO_MAL(1+5);`
    - Resultado obtenido: a= 1+5*1+5*1+5=1+5+5+5=16.
    - Resultado esperado: (1+5)*(1+5)*(1+5)=6*6*6=216.
  - `#define SUMA_MAL(A) (A)+(A)`
  - `a=5*SUMA_MAL(1);`
    - Resultado obtenido: 5*1+1=6;
    - Resultado esperado: 5*(1+1)=10.

- Para prevenir estos problemas:
  - Incluir un paréntesis en torno a cada variable en una macro.
  - Incluir un paréntesis adicional en torno a la totalidad de la expresión.