APRENDERAPROGRAMAR.COM

CONSTANTES SIMBÓLICAS EN C. MACROS CON DEFINE. DIFERENCIA ENTRE CONST Y #DEFINE. EJEMPLO. (CU00531F)

Sección: Cursos
Categoría: Curso básico de programación en lenguaje C desde cero

Fecha revisión: 2031

Resumen: Entrega nº31 del curso básico "Programación C desde cero".

Autor: Mario Rodríguez Rancel

CONSTANTES SIMBÓLICAS Y MACROS

Hasta ahora habíamos visto cómo definir constantes con una sintaxis del tipo const int NOMBRE = 10; Existe otra manera de crear constantes en C basada en la directiva #define. La sintaxis a emplear con C es la siguiente:

```
#define NOMBRECONSTANTE valor
```

La diferencia entre el uso de *const* y el uso de *#define* está en que mediante *const* se declara una constante que tiene un tratamiento asemejable a una variable (por ejemplo, la constante es de un tipo de dato) mientras que mediante *define* se indica que escribir el nombre especificado equivale a escribir el valor, con una correspondencia directa y sin tratamiento análogo al de una variable.

Por ejemplo: *const int JUGADORES = 5;* ... *#define JUGADORES 5*

En la primera declaración se indica que JUGADORES es una constante de tipo *int* mientras que en la segunda se indica que donde aparezca en el código la palabra JUGADORES deberá ser reemplazada por 5 directamente. En general, usar *#define* supone que la compilación sea más rápida al no tener el compilador que realizar el tratamiento y verificaciones propias de variables. Por ello su uso resultará recomendable cuando existan ciertos valores numéricos que tengan un significado especial, valor constante y uso frecuente dentro del código.

Las constantes definidas con *#define* se denominan constantes simbólicas, y algunas de ellas existen de forma predeterminada en el lenguaje.

Otro uso de *#define* es el de definir macros, equivalencias a ejecutar cuando se encuentre la invocación a la macro. Por ejemplo:

```
#include <stdio.h>
#include <stdlib.h>
#define SUMA(x,y) x+y
// Ejemplo aprenderaprogramar.com
int main() {
   int a = 12;   int b = 3;
   printf ("La suma de a y b vale %d", SUMA(a,b));
   return 0;
}
```

En el programa anterior, se define una macro según la cual cuando sea invocada pasando dos valores devolverá la suma de dichos valores. Esta misma tarea puede ser realizada por una función, pero usar macros puede redundar en una mayor eficiencia y velocidad de ejecución al no tener que realizarse el tratamiento propio de las funciones cuando aparece una invocación a la macro.

También existen algunas macros predefinidas del lenguaje. Por ejemplo __LINE__ es una macro que nos devuelve la línea (como número entero) dentro del fichero de código.

EJERCICIO

Responde a las siguientes cuestiones:

a) Usando la macro __LINE__ crea un programa que muestre el mensaje "La línea actual es: nLinea" donde nLinea será el valor de línea correspondiente. Por ejemplo "La línea actual es: 5"

b) Sabiendo que la macro __FILE__ devuelve una cadena de caracteres con la ruta del fichero compilado, ampliar el programa anterior para que además de la línea nos muestre la ruta del fichero con un mensaje del tipo "La línea actual es: nLinea y la ruta del fichero es rutaFich" donde nLinea y rutaFich deberán ser reemplazados por los valores correspondientes.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00532F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=82&Itemid=210