Algoritmos y Programación I - 75.02
Guía de Ejercicios Nº 2
Programas lineales - Tipos y Constantes

Nota: Los siguientes ejercicios se refieren a programas ANSI-C no modularizados (autocontenidos dentro de la función main()).

Tipos enumerativos y Constantes

1) a) Escriba un programa que defina un tipo enumerativo con los símbolos TRUE y FALSE.
b) Redefina a este tipo enumerativo como tipo "bool".
c) Declare una variable de tipo "bool" y asígnele el símbolo TRUE.
d) Escriba una porción de código que imprima por stdout una cadena de caracteres conforme el valor de la variable anterior (traducción semántica).

2) Idem para un tipo enumerativo "status" compuesto por los símbolos ERROR y OK.

3) a) Defina un tipo enumerativo "tdia" compuesto por 7 símbolos que representen a cada día de la semana.
b) Escriba una porción de código que muestre por stdout el contenido de una variable de tipo tdia (traducción).

4) a) Defina un tipo de dato "tipodato" a partir del tipo nativo "float" (usando `typedef`).
b) Defina un tipo enumerativo "tescala" que contenga los símbolos CELSIUS y FAHRENHEIT.
c) Escriba un fragmento de código que lea del stdin un número representando un valor de temperatura en una de las dos escalas, y la convierta paramétricamente a la otra, mostrando el resultado por stdout.

5) a) Defina un tipo enumerativo "tmes" con 12 símbolos que representen a cada uno de los meses del año.
b) Escriba un fragmento de código que a partir del contenido de una variable de tipo tmes, imprima por pantalla la descripción del mes (traducción).

6) Un ángulo se considera agudo si es menor que 90 grados, obtuso si es mayor que 90 grados y recto si es igual a 90 grados. Escribir un programa que acepte un ángulo y muestre el tipo de ángulo correspondiente al valor introducido en grados. Usar tipos enumerativos para manejar los tres tipos de ángulo.

7) El nivel de grado de los estudiantes que no han terminado la universidad se determina utilizando la siguiente tabla:

Número de créditos	Grado
Menos que 32	Primer año
32 a 63	Segundo año
64 a 95	Tercer año
96 o más	Cuarto año

Usando esta información, escribir un programa que acepte el número de créditos que ha acumulado un estudiante y determine en qué grado se encuentra, mostrando los resultados por pantalla.

9) Cada unidad de disco de un cargamento está marcada con un código del 1 al 4, el cual indica un fabricante, como sigue:

Código de fabricante (Manufacturer ID)	Denominación (Naming)
1	3M Corporation
2	Maxell Corporation
3	Sony Corporation
4	Verbatim Corporation

Escribir un programa que acepte el número de código como dato de entrada y despliegue el nombre correcto del fabricante con base en el valor introducido (usar tipos enumerativos y constantes simbólicas para la

modelización).

10) Escribir un programa que lea un texto desde el teclado y calcule la cantidad de caracteres alfanuméricos, no alfanuméricos y numéricos que se ingresan. El programa debe informar esos valores por pantalla. (Hint: usar isalnum(int caracter), isalpha(int caracter), isdigit(int caracter)).

11) Indicar si el siguiente fragmento de código es correcto o no. Justificar.

```
typedef enum { CORRECTO, INCORRECTO } resultado_t;


int x;
resultado_t proceso = CORRECTO;

if (proceso == CORRECTO) x = 1;
else x = -1;  /* proceso incorrecto */
```

12) Indicar si el siguiente fragmento de código es correcto o no. Justificar.

```
#define CORRECTO   0
#define INCORRECTO 1

int resultado = CORRECTO;

if (resultado == INCORRECTO) ...
```

13) a) Un dispositivo de comunicaciones puede utilizar una velocidad de transferencia de 1200, 2400, 4800, y 9600 baudios. Definir un tipo enumerativo que modelice dicha situación.
b) ¿Conviene utilizar un prefijo para los tokens del tipo enumerativo? ¿Por qué?

14) En relación al ejercicio anterior, dado que internamente los símbolos se representan como un número entero, ¿es posible compilar lo siguiente?

```
typedef enum { 1200, 2400, 4800, 9600 } baudrate_t;
```

Directivas al Preprocesador

15) Escribir una directiva de preprocesador para realizar cada una de las siguientes tareas:
a. Si la constante simbólica TRUE está definida, eliminarla y volverla a definir como 1.
b. Idem pero sin usar la directiva #ifdef .
c. Si la constante simbólica TRUE está definida, eliminarla y volverla a definir como 1, usar la directiva #ifdef.

16) Definir un token "DEBUG", y escribir un fragmento de código que sea compilado (o no) dependiendo de si se está en modo DEBUGing o productivo. ¿Para qué puede servir esta construcción?

17) ¿Cómo se puede parametrizar el código del ejercicio 5), de forma de poder soportar varios idiomas? (usar directivas al preprocesador).