Laboratorio N°2
[24 de Marzo del 2016]
Algoritmos y Lenguaje de Programación en C

Objetivos Generales:

- **I.** Familiarizarse con los tipos de datos básicos del lenguaje C.
- **II.** Conocer los operadores matemáticos y su precedencia.
- **III.** Utilizar variables y operadores para resolver problemas.

Tipos de datos:

Como se vio en clases, y se comentó en el primer laboratorio, existen distintos tipos de datos en el lenguaje C, cada uno de ellos se utiliza para distintas situaciones. Una característica importante de los tipos de datos es su tamaño en bytes, ya que nos permite tener mejor control de ellos para evitar fallos como overflow o underflow. Un operador unario que permite medir el tamaño en relación al tipo de dato "char", es el operador **sizeof**. Por ejemplo:

```c
#include <stdio.h>

int main(void)
{
    printf("%d\n", sizeof(char));
    return 0;
}
```

Ejercicio 1:

Utilice el operador **sizeof** y escriba un programa (en lenguaje C) que muestre el tamaño de los siguientes tipos de datos: **int, float, double, long**.

Variables y Operadores

Una variable es un nombre que se le da a un espacio en memoria que nuestros programas pueden manipular. Cada variable tiene un tipo específico que determina su tamaño en bytes, el rango de valores que puede almacenar y el conjunto de operaciones que se le pueden aplicar.

En C, para declarar una variable, se debe especificar su nombre y tipo, por ejemplo:

```c
#include <stdio.h>

int main(void)
{
    int a = 5;
    float b = 2.25;

    printf("La variable a = %d es un entero, y b = %.2f es flotante", a, b);
    return 0;
}
```

Cabe destacar que es buena práctica poner nombres representativos a las variables, es decir, que al leer su nombre, se pueda inferir de manera simple cuál será su utilidad. Por otro lado, se debe considerar el rango de valores que tomará una variable, debido a que pueden ocurrir problemas como *overflow* o *underflow*.

Ejercicio 2:

Copie el siguiente programa, compile y ejecute:

```c
#include <stdio.h>
#include <limits.h>

int main(void)
{
    int a;
    a = INT_MAX;

    printf("El mayor entero posible es: %d\n", a);
    a++;

    printf("Si se suma 1 al mayor entero posible, se obtiene %d\n", a);

    return 0;
}
```

¿Son correctos los resultados mostrados en pantalla?

Obs: En el programa se utilizó "a++", que es un operador incremental, y es equivalente a escribir a = a + 1. Más información respecto a operadores en C se puede encontrar en el siguiente enlace: http://www.tutorialspoint.com/cprogramming/c_operators.htm

Ejercicio 3:

Escriba un programa que permita calcular el área en cm^2 de la circunferencia, del rectángulo y la suma de ellas. Pida al usuario que ingrese mediante el teclado valores para el diámetro de la circunferencia (hasta 100 cm), para el largo y ancho del rectángulo (hasta 500 cm y 200 cm respectivamente). Por último, despliegue en pantalla el resultado para estos valores.

Ejercicio 4:

El campo eléctrico producido por una vara con densidad de carga lineal λ, de largo L, a una distancia a como se muestra en la figura 1, está dado por:

$$\vec{E} = \lambda k_e \left(\frac{1}{a} - \frac{1}{a+L}\right)[\hat{\imath}]$$

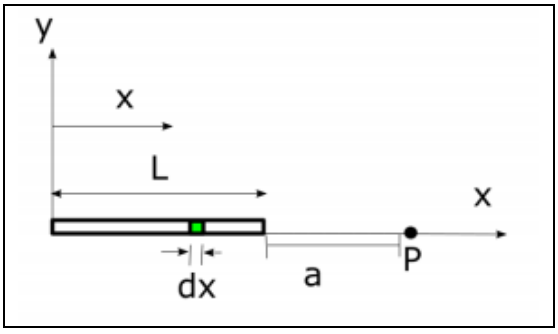

Figura 1: Visualización del enunciado.

Asumiendo que $L > 0$, $a > 0$, $k_e = 9 \cdot 10^9$, escriba un programa en C que realice lo siguiente:

1. Pida al usuario que ingrese mediante el teclado valores para λ, a y L.
2. Guarde en una variable el resultado del cálculo de E.
3. Muestre en pantalla el resultado.

Ejercicio 5:

Declare 4 variables flotantes w, x, y, z; asígneles valores por teclado y despliegue en pantalla el resultado de las siguientes operaciones matemáticas:

a) w*x/y + z/y
b) (w*x/y + z)/y
c) w*x/++y + z/y

¿Se obtienen los mismos resultados?

El operador módulo

El operador módulo es bastante usado en diferentes áreas, como por ejemplo: generar números aleatorios entre cierto rango, lógica interna de un reloj digital, resetear la posición de una partícula, mostrar una variable por medio de imágenes en una interfaz gráfica, entre otras. Supongamos que queremos mostrar por medio de imágenes una variable, cuyos valores se encuentran entre 0 y 9999. Si lo hiciéramos con una imagen por valor, necesitaríamos 10000 imágenes, lo cual sería poco práctico. Sin embargo, utilizando el operador módulo, reducimos esa cantidad a sólo 4, extrayendo las unidades de mil, centenas, decenas, y unidades. A continuación se muestra un código que realiza lo enunciado:

```c
#include <stdio.h>

int main()
{
    int a=1234;
    printf("unidad %i\n", a % 10);
    printf("decena %i\n",(a / 10) % 10);
    printf("centena %i\n",(a / 100) % 10);
    printf("unidad de mil %i\n",(a / 1000) % 10);

    return 0;
}
```

Ejercicio 6:

Intente entender la lógica del código, por qué funciona como funciona, qué es lo que se está haciendo internamente.