5. Trabajo Práctico 5 - Punteros y arreglos

5.1. Ejercicio 1

1. ¿Qué función cumple el operador &?
2. ¿Por qué cada vez que se ejecuta el programa tienen direcciones distintas?
3. ¿De qué depende la cantidad de bytes que ocupa la dirección de una variable?
4. ¿Hay alguna relación entre el tipo de dato de una variable y el tamaño de su dirección?

```c
#include <stdio.h>

int main (void) {
    int varIntA, varIntB;
    char varCharA, varCharB;

    printf ("varIntA = %08x;  %p\r\n", varIntA, &varIntA);
    printf ("varIntB = %08x;  %p\r\n", varIntB, &varIntB);
    printf ("varCharA = %08x; %p\r\n", varCharA, &varCharA);
    printf ("varCharB = %08x; %p\r\n", varCharB, &varCharB);

    return (0);
}
```

5.2. Ejercicio 2

Implemente un programa que utilizando un puntero de tipo **unsigned char** imprima byte a byte una variable de tipo **unsigned int** en hexadecimal que ha sido inicializada con el número 0x12345678, junto a la dirección de memoria de cada byte. ¿El byte menos significativo del unsigned int, en qué dirección se encuentra respecto al byte más significativo? **Nota:** Investigue y desarrolle el concepto de **endianness**.

5.3. Ejercicio 3

Implemente una función que realice las cuatro operaciones básicas entre dos números de tipo float y retorne el resultado por referencia en otra variable float. La operación se indicará en un parámetro adicional. El prototipo de la función es el siguiente:

int calculo(float operadorA, float operadorB, float* resultado, char operacion);

Donde:

- operandoA y operandoB son números con los cuales se debe realizar la operación.
- operación: La operación matemática a realizar:
  1. '+': Realiza la suma.
  2. '-': Realiza la resta.
  3. '*': Realiza el producto.
  4. '/': Realiza la división.
- resultado: Dirección de la variable donde se debe almacenar el resultado de la operación.
- La función retorna EXITO (0) si se pudo realizar el cálculo, y ERROR (-1) si ocurrió algún error.

Importante: Para la implementación utilice estructura de selección switch.

5.4. Ejercicio 4

Dada la siguiente declaración de un array:

int v[5] = {32, 12, 15, 89, 6};

Asumiendo que el sistema operativo nos asigna la dirección **0xbfff0000** para el comienzo del mismo, responda:

1. ¿Qué dirección de memoria contendrá el valor de v[2]?
2. ¿Qué dirección de memoria resultará de resolver v+3?
3. Indique cómo accedería al valor 89 dentro del vector, por cualquier método que conozca.
4. Podemos recorrer el vector utilizando post-incremento (v++)? Justifique su respuesta.
5. ¿Recibiríamos algún error durante la compilación o el linkeo si tratáramos de acceder al contenido de v+10? En caso de que sí se pide justificar la respuesta. **Nota: No se admiten respuestas del tipo: "La consola imprime violación de segmento".**

5.5. Ejercicio 5

Implemente una función que realice las cuatro operaciones básicas entre un vector y un número de tipo float y retorne el resultado (modificando por referencia el propio vector). La operación se indicará en un parámetro adicional. El prototipo de la función es el siguiente:

int calculo_vector(float vector[], unsigned int largo, float numero, char operacion);

Donde:

- Cada elemento del vector y número son los números con los cuales se debe realizar la operación.
- largo: Largo del vector.
- operación: La operación matemática a realizar:
  1. '+': Realiza la suma.
  2. '-': Realiza la resta.
  3. '*': Realiza el producto.
  4. '/': Realiza la división.
- La función retorna EXITO (0) si se pudo realizar el cálculo, y ERROR (-1) si ocurrió algún error.

Importante: Para la implementación utilice estructura de selección switch.

5.6. Ejercicio 6

Implemente una función que obtenga varias características estadísticas del contenido de un vector. El prototipo de la función es el siguiente:

int estadisticas_vector(float vector[], unsigned int largo, float* resultado, int caracteristica);

Donde:

- vector: Vector numérico del cual se obtendrán las estadísticas.
- largo: Largo del vector.
- característica: La característica estadística a obtener:
  1. Devuelve el valor medio (promedio).
  2. Devuelve la mediana.
  3. Devuelve el mínimo.
  4. Devuelve el máximo.
- resultado: Dirección de la variable donde se debe almacenar el resultado obtenido.
- La función retorna EXITO (0) si se pudo obtener la característica, y ERROR (-1) si ocurrió algún error.

Importante: Para la implementación utilice estructura de selección switch.

Ayuda

Cálculo de la mediana:

- Ordenamos los datos de menor a mayor.
- Si el vector tiene un número impar de elementos, la mediana es el valor del elemento central. Ejemplo: [1, 2, 2, 3, 5, 5, 11, 12, 14] -> Mediana = 5
- Si el vector tiene un número par de elementos, la mediana es la media (promedio) entre los dos elementos centrales. Ejemplo: [2, 5, 9, 10, 17, 26] -> Mediana = 9,5

5.7. Ejercicio 7

Implemente una función que calcule la derivada discreta (por diferencias finitas) del contenido de un vector de enteros. El prototipo de la función es el siguiente:

void calcular_derivada_discreta(int vectorIn[], int vectorOut[], unsigned int largo);

Ayuda

Cálculo de la derivada discreta:

$$\frac{\partial f}{\partial x}[x] = F[x] - F[x-1]$$

Nota: Asigne un valor 0 (cero) al primer elemento del vector de salida.