# Laboratorio 03

***Competencias para desarrollar***

Aplicar los conceptos relacionados a fundamentos de c++ para microprocesadores y su funcionamiento.

***Instrucciones***

Esta actividad se realizará individualmente. Al finalizar los períodos de laboratorio o clase, deberá entregar este archivo en formato PDF en la actividad correspondiente en Canvas.

**Parte 01**

**(20 puntos, 2 puntos cada respuesta) Investiga y responda el significado de los siguientes operadores o conceptos en c++**

1. std:
   1. Espacio de nombres estándar en C++.
   2. Contiene las definiciones de las librerías estándar de C++ (e.g., `std::cout`, `std::vector`).
2. :: (Operador de resolución de ámbito):
   1. Se usa para acceder a una función o variable dentro de un espacio de nombres o clase específica.
   2. Ejemplo: `std::cout` accede a `cout` dentro del espacio de nombres `std`.
3. >> (Operador de extracción):
   1. Se utiliza con `std::cin` para extraer (leer) datos desde la entrada estándar.
   2. Ejemplo: `std::cin >> variable;`
4. << (Operador de inserción):
   1. Se utiliza con `std::cout` para insertar (escribir) datos en la salida estándar.
   2. Ejemplo: `std::cout << "Hola, mundo!";`
5. \* (Operador de desreferencia o multiplicación):
   1. Como operador de desreferencia, se usa para acceder al valor apuntado por un puntero.
   2. Como operador aritmético, se usa para la multiplicación.
   3. Ejemplo: `int \*ptr; \*ptr = 10;` (desreferencia), `int resultado = a \* b;` (multiplicación).
6. & (Operador de dirección o bitwise AND):
   1. Como operador de dirección, se usa para obtener la dirección de una variable.
   2. Como operador bitwise AND, se usa para realizar operaciones AND a nivel de bits.
   3. Ejemplo: `int \*ptr = &variable;` (dirección), `int c = a & b;` (bitwise AND).
7. Conversions explícitas (Type casting explícito):
   1. Proceso de convertir una variable de un tipo a otro usando castings explícitos.
   2. Ejemplo: `int x = (int)3.14;` o `int x = static\_cast<int>(3.14);`.
8. Puntero:
   1. Variable que almacena la dirección de memoria de otra variable.
   2. Ejemplo: `int \*ptr = &variable;`.
9. Variable estática\*\*:
   1. Variable cuya duración de almacenamiento dura toda la ejecución del programa.
   2. Dentro de una función, conserva su valor entre llamadas.
   3. Ejemplo: `static int contador;`.
10. Variable dinámica\*\*:
    1. Variable cuya memoria se asigna en tiempo de ejecución.
    2. Se utiliza `new` para asignar y `delete` para liberar.
    3. Ejemplo: `int \*ptr = new int; delete ptr;`.

**Parte 02**

**(70 puntos): Utilizando C++, completa los siguientes enunciados.**

1. **Sube tu archivo .cpp** con los programas solicitados.
2. **Adjunta capturas de pantalla** de cada programa en ejecución, mostrando resultados visibles.
3. **(10 puntos)** Escribe un programa que pida al usuario ingresar un número entero entre 0 y 35. Si el número es menor o igual a 9, el programa debe mostrar el número; de lo contrario, debe mostrar A para 10, B para 11, C para 12,..., y Z para 35.

**Requisito:** usa el operador de conversión, static\_cast<char>(), para números ≥ 10.

Texto

Descripción generada automáticamente

1. **(40 puntos)** Escribe un programa en C++ que realice las siguientes tareas utilizando arreglos:
2. **Entrada de** Datos: Pide al usuario que ingrese el número de elementos del arreglo (máximo 100). Luego, generar números aleatorios enteros para llenar el arreglo.
3. **Operaciones en el Arreglo:**
   * Encuentra el número más grande y el más pequeño en el arreglo.
   * Calcula el promedio de los números en el arreglo.
   * Cuenta cuántos números en el arreglo son mayores que el promedio.
   * Ordena el arreglo en orden ascendente.
4. **Salida de Datos:** Imprime el número más grande, el más pequeño, el promedio de los números, la cantidad de números mayores que el promedio, y el arreglo ordenado.

Texto

Descripción generada automáticamente

1. **(40 puntos)** Vas a escribir un programa guiado en C++ que realiza las siguientes tareas utilizando punteros:
2. **Entrada de Datos:**
   * Solicita al usuario ingresar dos números enteros.
3. **Operaciones con Punteros:**
   * Usa punteros para intercambiar los valores de los dos números.
   * Calcula la suma y el producto de los dos números utilizando punteros.
4. **Salida de Datos:**

* Imprime los números después del intercambio, la suma, el producto, y el mayor número.

**Paso 1: Declaración de Variables y Punteros**

1. Declara dos variables enteras para almacenar los números que el usuario ingresará.
2. Declara dos punteros que apunten a estas variables.

int num1, num2;

int \*p1, \*p2;

1. Asigna a cada puntero la dirección de las variables correspondientes.

p1 = &num1;

p2 = &num2;

**Paso 2: Solicitar los Números al Usuario**

1. Usa cin para leer los números desde la entrada estándar.

cout << "Ingrese el primer número: ";

cin >> num1;

cout << "Ingrese el segundo número: ";

cin >> num2;

**Paso 3: Intercambio de Valores Usando Punteros**

1. Implementa una función intercambiar que reciba dos punteros y use una variable temporal para intercambiar los valores a los que apuntan.

void intercambiar(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

1. Llama a la función intercambiar pasando los punteros p1 y p2.

intercambiar(p1, p2);

**Paso 4: Calcular Suma y Producto Usando Punteros**

1. Implementa una función calcularSuma que reciba dos punteros y devuelva la suma de los valores a los que apuntan.

int calcularSuma(int \*a, int \*b) {

return \*a + \*b;

}

1. Implementa una función calcularProducto que reciba dos punteros y devuelva el producto de los valores a los que apuntan.

int calcularProducto(int \*a, int \*b) {

return \*a \* \*b;

}

1. Llama a estas funciones y guarda los resultados.

int suma = calcularSuma(p1, p2);

int producto = calcularProducto(p1, p2);

**Paso 5: Imprimir los Resultados**

1. Imprime los números después del intercambio, la suma, el producto y el mayor número.

cout << "Después del intercambio:" << endl;

cout << "Primer número: " << num1 << endl;

cout << "Segundo número: " << num2 << endl;

cout << "La suma de los números es: " << suma << endl;

cout << "El producto de los números es: " << producto << endl;

1. Explica con tus palabras cómo se pueden usar los punteros para acceder y modificar directamente los datos en memoria.

Texto

Descripción generada automáticamente

Los punteros usados para poder acceder y modificar directamente los datos me memoria se pueden usar para alterar directamente el valor de la variable sin cambiarlo manualmente, con el puntero almacenamos la variable a la cual estamos apuntando, y al querer cambiar el valor del puntero, cambiamos el valor de la variable guardada de una forma “violenta”. (chambea joaquín)