



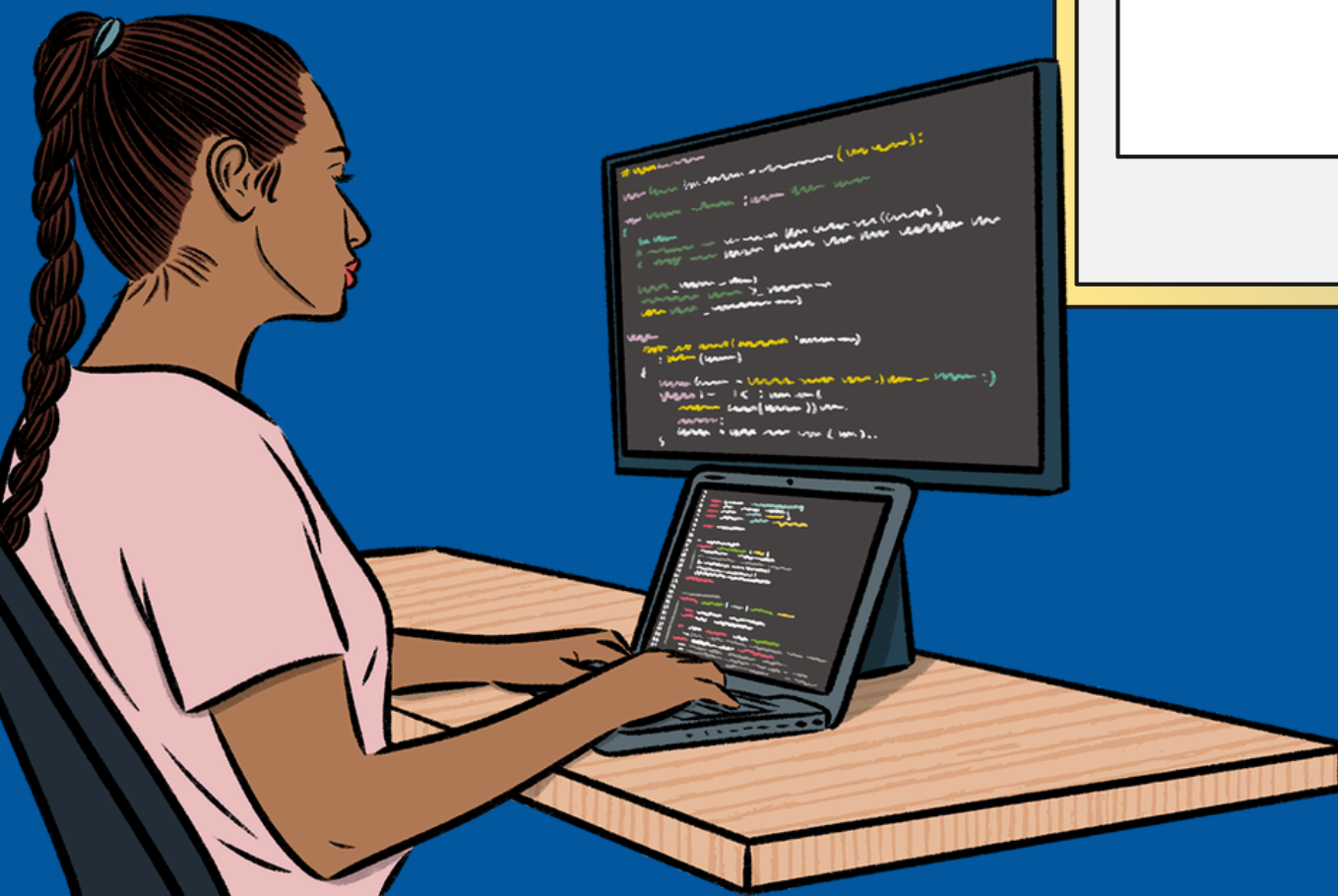
PROGRAMACION AVANZADA

ING. INDUSTRIAL

Esp. Ing.Prof.: Cristina Delia Cruz



TIPOS ESTRUCTURADOS EN C++





REGISTROS EN C++

Un registro en C++ es una estructura de datos que agrupa varios campos (variables) de diferentes tipos bajo una sola entidad, permitiendo acceder y manipularlos de manera organizada. Por ejemplo, puedes tener un registro llamado Empleado que contiene campos como el nombre, el salario, y el ID de empleado.



REGISTROS EN C++

Definición de una Estructura

Para definir un registro en C++, utilizamos la palabra clave **struct** seguida del **“nombre de la estructura”**. Dentro de las llaves {}, declaramos los miembros (variables) que componen la estructura.



REGISTROS EN C++

Registros/Estructuras

Los registros, también llamados estructuras (**struct**) en C++, son una forma de agrupar varios datos relacionados bajo un mismo nombre.

A diferencia de los arreglos, que solo pueden contener elementos del mismo tipo, una estructura puede agrupar diferentes tipos de datos (por ejemplo, un número entero, un carácter y un número con decimales) en una sola unidad. Cada uno de estos datos individuales dentro de la estructura se conoce como miembro o campo. Piensa en una estructura como una ficha o un formulario que contiene información completa sobre una sola entidad, como un empleado, un producto o un cliente. Por ejemplo, en lugar de tener variables separadas para el nombre, el apellido y la edad de una persona, puedes agruparlas todas en una única estructura llamada Persona.



REGISTROS EN C++

Acceso a los Miembros de la Estructura

Como se ve en el ejemplo 1, los miembros de una estructura se acceden usando el operador de punto (.). Es decir, si tienes una variable de tipo Empleado, puedes acceder a sus miembros usando `nombre_variable.miembro`.



REGISTROS EN C++

Estructuras y Funciones

También puedes pasar estructuras a funciones para trabajar con los datos. Pueden pasarse por valor o por referencia, dependiendo de si deseas modificar el registro original o solo trabajar con una copia.



REGISTROS EN C++

Uso de Arreglos de Estructuras

En C++, también puedes tener arreglos de estructuras, lo que te permite manejar varios registros del mismo tipo.



REGISTROS EN C++

Características de las Estructuras en C++:

1. Tipos heterogéneos: Puedes almacenar diferentes tipos de datos bajo el mismo registro (entero, cadena, flotante, etc.).
2. Memoria contigua: Los miembros de una estructura se almacenan en posiciones de memoria contiguas.
3. Pasar a funciones: Las estructuras pueden pasarse por valor o por referencia a funciones.
4. Acceso mediante el operador punto: Se accede a los miembros utilizando el operador ".".



REGISTROS EN C++

*En C++, los registros se implementan mediante estructuras (**struct**), lo que permite agrupar diferentes tipos de datos bajo una misma entidad lógica. Son útiles cuando quieres manejar datos relacionados, como la información de una persona, un producto, o un empleado, de manera estructurada y más organizada.*



REGISTROS EN C++

Tamaño de una estructura

El tamaño de una estructura es la cantidad de memoria que el programa necesita para guardar todos sus miembros.

*Imagina una estructura llamada **Paquete** para un almacén. Esta estructura tiene dos miembros:*

- ***Un código de producto (int)***, que ocupa 4 bytes.
- ***El peso del paquete (double)***, que ocupa 8 bytes.

*Aunque la suma lógica es 12 bytes, al usar **sizeof(Paquete)**, el resultado podría ser **16 bytes**. Esto se debe a un concepto llamado "**relleno**" (**padding**), donde el compilador alinea los datos en la memoria para que el procesador pueda leerlos más rápido y eficientemente. En este caso, el compilador podría **añadir 4 bytes** de espacio extra después del int para que el double se inicie en una dirección de memoria que sea un múltiplo de 8, lo cual mejora el rendimiento de la CPU.*

*En resumen, **el tamaño de la estructura es la suma de los tamaños de todos sus campos, más el espacio adicional que el compilador agrega por cuestiones de optimización.***



EJEMPLO 1: DECLARACIÓN BÁSICA Y USO DEL OPERADOR PUNTO (.)

Crear un programa para registrar la información básica de una máquina en planta.

Objetivo: Comprender la definición de una estructura (struct) con campos heterogéneos y el acceso a sus miembros mediante el operador punto.

Requisitos:

1. Definir una estructura llamada Maquina con los campos: id (entero), nombre (cadena de caracteres) y activa (booleano).
2. Declarar una variable de este tipo en el main, asignar valores directamente a sus campos y mostrarlos por pantalla usando el operador '.'.



EJEMPLO 2: CARGA INTERACTIVA DE DATOS EN UN REGISTRO

Permitir que el operador ingrese los datos de un lote de producción mediante la consola.

Objetivo: Implementar la lectura dinámica de datos introducidos por el usuario para almacenarlos dentro de los campos de un registro.

Requisitos:

1. Utilizar la estructura LoteProduccion con campos de tipo entero y flotante.
2. Solicitar al usuario mediante cin los valores correspondientes a cada campo y posteriormente imprimirlos formateados.

EJEMPLO 3: ARREGLO DE ESTRUCTURAS (LISTADO DE EMPLEADOS O ESTACIONES)

Administrar un conjunto de estaciones de trabajo dentro de una línea de montaje utilizando arreglos de registros.

Objetivo: Manejar múltiples registros del mismo tipo agrupados en una estructura matricial o lineal unidimensional.

Requisitos:

1. Definir una estructura Estacion (ID y nombre del operador a cargo).
2. Declarar un arreglo unidimensional de tamaño 3 y cargarlo mediante un bucle for.



EJEMPLO 4: PASO DE ESTRUCTURAS A FUNCIONES POR VALOR

Enviar los datos de un producto industrial a una función para mostrarlos de forma modular, sin alterar el original.

Objetivo: Enseñar el pasaje de parámetros utilizando estructuras bajo la modalidad "por valor".

Requisitos:

1. Crear una función auxiliar `mostrarProducto(Producto p)` que reciba la estructura copiada.
2. Llamar a dicha función desde el main.



EJEMPLO 5: PASO DE ESTRUCTURAS A FUNCIONES POR REFERENCIA

Desarrollar una función que incremente el sueldo de un operario directamente sobre su registro original.

Objetivo: Aplicar el paso de estructuras por referencia utilizando el operador ampersand (&) para modificar los datos originales.

Requisitos:

1. Definir una estructura Empleado (nombre y salario).
2. Crear una función aumentarSalario(Empleado &e, float porcentaje) que modifique el campo de salario de manera permanente.



EJEMPLO 6: USO DE OPERADOR SIZEOF Y ANÁLISIS DE RELLENO (PADDING)

Calcular y comparar el tamaño en memoria que ocupa una estructura industrial y comprender el concepto de alineación de memoria (padding).

Objetivo: Explicar cómo el compilador ajusta los bytes en memoria para optimizar la lectura de la CPU.

Requisitos:

1. Definir una estructura `PaqueteAlmacen` con un `int` y un `double`.
2. Imprimir el valor devuelto por la función `sizeof(PaqueteAlmacen)` y analizar el espacio de relleno.



EJEMPLO 7: BÚSQUEDA DE UN REGISTRO DENTRO DE UN ARREGLO DE ESTRUCTURAS

Buscar una herramienta específica dentro del inventario del pañol de planta ingresando su código identificador.

Objetivo: Recorrer un arreglo de estructuras aplicando una condición de búsqueda lineal.

Requisitos:

1. Declarar un arreglo de 3 elementos de tipo Herramienta.
2. Solicitar al usuario un código a buscar y recorrer el arreglo comparando el campo correspondiente.



EJEMPLO 8: CÁLCULO MATEMÁTICO Y ESTADÍSTICO CON REGISTROS

Calcular el promedio de eficiencia de una flota de vehículos de transporte industrial a partir de un arreglo de registros.

Objetivo: Procesar campos numéricos internos de múltiples estructuras para arrojar métricas de rendimiento.

Requisitos:

1. Definir una estructura Vehiculo con patente y consumo de combustible por kilómetro.
2. Implementar un bucle para acumular los valores y calcular la media aritmética.



EJEMPLO 9: ESTRUCTURAS ANIDADAS (REGISTROS DENTRO DE REGISTROS)

Gestionar los datos de envíos industriales donde cada encomienda posee una estructura interna de ubicación (dirección).

Objetivo: Utilizar el concepto de estructuras anidadas para modelar objetos complejos del mundo real.

Requisitos:

1. Definir una estructura interna Ubicacion (ciudad y código postal).
2. Definir una estructura externa Envio que contenga como uno de sus campos a la estructura Ubicacion.
3. Acceder a los campos anidados usando doble operador punto (envio.ubicacion.ciudad).



EJEMPLO 10: SISTEMA MODULAR INTEGRADO DE CONTROL DE CALIDAD

Desarrollar un sistema de menú interactivo basado en registros y funciones para auditar lotes de producción industrial.

Objetivo: Integrar el uso de estructuras, arreglos, funciones y validación mediante bucles en un proyecto completo.

Requisitos:

1. Crear una estructura Auditoria (ID de pieza, defecto detectado y estado aprobado/rechazado).
2. Diseñar un menú interactivo con do-while que permita: 1. Registrar auditoría de piezas, 2. Mostrar listado de piezas auditadas, 3. Salir.
3. Modularizar cada opción mediante funciones independientes.

¿PREGUNTAS?

```
    }  
    render() {  
      return (  
        <React.Fragment>  
          <div className="py-5">  
            <div className="container">  
              <Title name="our" title="product">  
                <div className="row">  
                  <ProductConsumer>  
                    {(value) => {  
                      console.log(value)  
                    }}  
                  </ProductConsumer>  
                </div>  
              </div>  
            </div>  
          </React.Fragment>  
        )  
      )  
    }  
  }  
}
```




MUCHAS
GRACIAS!!!

Nos Vemos en la Siguiente Clase