Universidad Mariano Gálvez de Guatemala

Ingeniería en sistemas

Curso: Programación I

Estudiantes:

José Ignacio Pereira Quiñonez

1290-22-6800

David Emanuel Castellanos Velásquez

1290-22-6424

Sección: A

Unidad 1

Lenguaje C++:

C++ es un lenguaje de programación de propósito general que se basa en el lenguaje C. Introduce características adicionales, como la programación orientada a objetos, el manejo de excepciones y las plantillas de programación genérica.

Herramientas de desarrollo para C++:

Editores para C++: Algunos editores populares para programar en C++ incluyen Visual Studio Code, Xcode, Code::Blocks, Eclipse CDT y Dev-C++.

Tipos de datos nativos en C++:

C++ ofrece varios tipos de datos nativos, como enteros (int, long, short), números de punto flotante (float, double), caracteres (char), booleanos (bool) y punteros (int *, char *, etc.).

Bibliotecas en C++:

C++ tiene una amplia variedad de bibliotecas estándar, como la biblioteca de entrada/salida (iostream), la biblioteca de manipulación de cadenas (string), la biblioteca de contenedores (vector, list, etc.), la biblioteca de algoritmos (sort, search, etc.) y más. También es posible utilizar bibliotecas externas para ampliar las funcionalidades de C++.

Entrada y salida de flujos:

C++ utiliza la biblioteca iostream para la entrada y salida de flujos de datos. Los flujos de datos básicos son cin (entrada estándar) y cout (salida estándar), que permiten la interacción con el usuario a través de la consola.

Programación estructurada:

La programación estructurada es un enfoque de programación que se basa en la secuencia, la selección (if-else, switch) y la repetición (bucles for, while) de estructuras de control. Es un paradigma que busca la claridad y el control de flujo en el código.

Programación orientada a objetos:

La programación orientada a objetos (POO) es un paradigma de programación que organiza el código en objetos, que son instancias de clases. La POO se basa en conceptos como la abstracción, el encapsulamiento, el polimorfismo y la herencia.

Propiedades de la programación orientada a objetos:

Abstracción: Permite representar objetos del mundo real como entidades en el código.

Encapsulamiento: Permite ocultar los detalles internos de un objeto y exponer solo una interfaz pública para interactuar con él.

Polimorfismo: Permite que los objetos de diferentes clases respondan de manera diferente a la misma función o método.

Herencia: Permite que una clase herede propiedades y comportamientos de otra clase (superclase).

Clase y superclase:

Una clase es una plantilla para crear objetos. Contiene definiciones de propiedades y métodos que los objetos pueden tener.

Una superclase es una clase de la cual se derivan otras clases (subclases). La superclase proporciona las características comunes que se heredan a las subclases.

Sobrecarga de operadores:

La sobrecarga de operadores permite definir el comportamiento de operadores existentes para clases personalizadas. Por ejemplo, se puede sobrecargar el operador "+" para realizar una operación personalizada entre objetos de una clase.

Propiedades y métodos de una clase:

Las propiedades de una clase son variables que almacenan datos relacionados con el objeto. Pueden ser públicas (accesibles desde cualquier parte del programa) o privadas (accesibles solo dentro de la clase).

Los métodos de una clase son funciones que realizan operaciones específicas relacionadas con el objeto. Pueden ser públicos (llamados desde cualquier parte del programa) o privados (llamados solo desde dentro de la clase).

Creación de objetos:

En C++, los objetos se crean a partir de una clase utilizando el operador new. Por ejemplo, si tenemos una clase llamada Persona, podemos crear un objeto persona1 de la siguiente manera: Persona* persona1 = new Persona(); Luego, podemos acceder a las propiedades y métodos del objeto utilizando el operador. (Punto).

Constructores:

Un constructor es un método especial de una clase que se llama automáticamente al crear un objeto. Se utiliza para inicializar las propiedades del objeto. En C++, el constructor tiene el mismo nombre que la clase y no tiene un tipo de retorno específico. Por ejemplo, si tenemos una clase Persona, el constructor se llamará Persona (). También se pueden definir constructores con parámetros para inicializar las propiedades con valores específicos al crear el objeto.

Unidad 2

Operadores lógicos:

Los operadores lógicos en C++ son && (AND), || (OR) y ! (NOT). Se utilizan para combinar expresiones lógicas y evaluar condiciones booleanas.

Estructuras de control:

IF: Se utiliza para ejecutar un bloque de código si se cumple una condición.

IF anidados: Permite tener IF dentro de otro IF, lo que permite condiciones más complejas.

FOR: Se utiliza para repetir un bloque de código un número específico de veces.

WHILE: Permite repetir un bloque de código mientras una condición sea verdadera.

SWITCH: Permite seleccionar una de varias opciones basadas en el valor de una expresión. Los casos dentro de un switch pueden utilizar break para salir del switch o continue para saltar a la siguiente iteración.

Diferencia entre operadores de igualdad y de asignación:

El operador de igualdad (==) se utiliza para comparar si dos valores son iguales.

El operador de asignación (=) se utiliza para asignar un valor a una variable.

Recursividad:

La recursividad es un concepto en el que una función se llama a sí misma. Permite resolver problemas mediante la subdivisión en problemas más pequeños y la combinación de las soluciones.

Manejo de excepciones:

El manejo de excepciones permite detectar y responder a situaciones excepcionales o errores durante la ejecución de un programa. Se utilizan bloques try, catch y throw para manejar y lanzar excepciones.

Procesamiento de archivos:

El procesamiento de archivos en C++ se realiza utilizando flujos de archivo, que son objetos que permiten leer y escribir datos en archivos. Se utilizan las clases ifstream para leer archivos y ofstream para escribir archivos.

Unidad 3

Arreglos y vectores:

Los arreglos y vectores son estructuras de datos que permiten almacenar múltiples elementos del mismo tipo en una secuencia contigua en la memoria.

Los arreglos se declaran especificando el tipo de dato y el tamaño del arreglo, mientras que los vectores son una implementación de arreglos dinámicos en C++ y su tamaño puede cambiar durante la ejecución del programa.

Declaración y creación de arreglos:

Para declarar y crear un arreglo en C++, se utiliza la sintaxis: tipo nombreArreglo[tamaño];

Ejemplos del uso de arreglos:

Almacenar una lista de calificaciones, nombres de estudiantes, datos de temperatura, entre otros.

Arreglos a funciones:

Los arreglos pueden ser pasados como argumentos a funciones en C++. Esto permite manipular o procesar los elementos del arreglo dentro de la función.

Búsqueda de datos en arreglos:

Los algoritmos de búsqueda se utilizan para encontrar la posición o el valor de un elemento en un arreglo. Algunos algoritmos de búsqueda comunes son la búsqueda lineal y la búsqueda binaria.

Ordenamiento de arreglos:

Los algoritmos de ordenamiento se utilizan para reorganizar los elementos de un arreglo en un orden específico. Algunos algoritmos de ordenamiento comunes son el ordenamiento burbuja, el ordenamiento por selección y el ordenamiento por inserción.

Arreglos multidimensionales:

Los arreglos multidimensionales son arreglos que contienen más de una dimensión. Pueden representar estructuras de datos como matrices, tablas y matrices tridimensionales.

Motores de bases de datos:

Los motores de bases de datos son software que permiten almacenar, organizar y recuperar grandes cantidades de datos de manera eficiente. Algunos ejemplos de motores de bases de datos son MySQL, MS SQL Server, PostgreSQL y Oracle.

Comparaciones características:

Los diferentes motores de bases de datos tienen características distintas, como la estructura de almacenamiento, el lenguaje de consulta, la escalabilidad, la seguridad y la capacidad de procesamiento de transacciones. La elección del motor de base de datos depende de los requisitos y las necesidades específicas del proyecto.

Unidad 4

SQL (Structured Query Language) :

Es un lenguaje de programación utilizado para gestionar y manipular bases de datos relacionales. Hay dos tipos principales de comandos en SQL: DML (Data Manipulation Language) y DDL (Data Definition Language).

DML (Data Manipulation Language):

El DML se utiliza para realizar operaciones de manipulación de datos en la base de datos, como la inserción, actualización, eliminación y consulta de registros.

Algunos comandos DML comunes son:

INSERT: Se utiliza para insertar nuevos registros en una tabla.

UPDATE: Se utiliza para modificar los valores de uno o más registros en una tabla.

DELETE: Se utiliza para eliminar registros de una tabla.

SELECT: Se utiliza para recuperar registros de una o más tablas.

DDL (Data Definition Language):

El DDL se utiliza para definir la estructura y el esquema de la base de datos. Permite crear, modificar y eliminar objetos de la base de datos, como tablas, índices, vistas y restricciones.

Algunos comandos DDL comunes son:

CREATE: Se utiliza para crear objetos en la base de datos, como tablas, vistas e índices.

ALTER: Se utiliza para modificar la estructura de los objetos existentes en la base de datos.

DROP: Se utiliza para eliminar objetos de la base de datos, como tablas, vistas e índices.

TRUNCATE: Se utiliza para eliminar todos los registros de una tabla sin afectar su estructura.

Unidad 5

Memoria dinámica:

La memoria dinámica es un área de memoria que se reserva y se administra durante la ejecución de un programa. Permite asignar y liberar memoria según sea necesario.

Declaración e inicialización de punteros:

Los punteros son variables que almacenan direcciones de memoria. Se declaran utilizando el tipo de dato seguido de un asterisco (*). Ejemplo: int *ptr;

Arrays de punteros:

Un array de punteros es un arreglo que contiene elementos que son punteros a otros tipos de datos. Puede utilizarse para almacenar múltiples direcciones de memoria.

Aritmética de punteros:

Los punteros se pueden manipular mediante operaciones aritméticas como suma, resta e incremento/decremento. Esto permite acceder a diferentes ubicaciones de memoria.

Operador Sizeof:

El operador sizeof se utiliza para determinar el tamaño en bytes de un tipo de dato o una variable. Puede ser útil al trabajar con asignación de memoria dinámica.

Relación entre apuntadores y arreglos:

En C++, los arreglos y los punteros tienen una relación estrecha. Un puntero puede apuntar al primer elemento de un arreglo, y los elementos de un arreglo pueden ser accedidos utilizando aritmética de punteros.

Apuntadores a funciones:

Los punteros a funciones son variables que almacenan direcciones de funciones. Permiten llamar a una función a través del puntero, lo que puede ser útil en situaciones como callbacks o programación dinámica.

Asignación de memoria dinámica:

La asignación de memoria dinámica se realiza utilizando el operador new en C++. Permite reservar memoria en tiempo de ejecución y devuelve un puntero al bloque de memoria asignado.

Introducción a listas enlazadas (Conceptos):

Las listas enlazadas son estructuras de datos dinámicas que consisten en nodos enlazados, donde cada nodo contiene un elemento y un puntero al siguiente nodo. Permiten almacenar y manipular datos de manera eficiente.

Unidad 6

Introducción a la estructura de datos en C++:

La estructura de datos se refiere a la forma en que se organizan y se almacenan los datos en la memoria de una computadora. En C++, puedes implementar diferentes estructuras de datos utilizando clases, punteros y asignación dinámica de memoria.

Clases auto referenciadas:

Las clases auto referenciadas son aquellas en las que un miembro de la clase es un puntero a un objeto de la misma clase. Esto permite crear estructuras de datos recursivas, como las listas enlazadas.

Listas enlazadas:

Las listas enlazadas son estructuras de datos dinámicas que constan de nodos enlazados. Cada nodo contiene un elemento y un puntero al siguiente nodo. Las listas enlazadas pueden ser simples, dobles o circulares, y permiten la inserción y eliminación eficientes de elementos en cualquier posición.

Asignación dinámica de memoria y estructura de datos:

La asignación dinámica de memoria en C++ se realiza utilizando el operador new. Puedes asignar memoria para crear objetos de estructuras de datos como listas enlazadas, pilas y colas, lo que permite una gestión flexible de la memoria durante la ejecución del programa.

Pilas:

Una pila es una estructura de datos en la que los elementos se agregan y eliminan siguiendo el principio de "último en entrar, primero en salir" (LIFO, Last In First Out). Los elementos se agregan en la parte superior de la pila y solo se pueden acceder o eliminar el elemento superior.

Colas:

Una cola es una estructura de datos en la que los elementos se agregan al final y se eliminan desde el frente, siguiendo el principio de "primero en entrar, primero en salir" (FIFO, First In First Out). Los elementos se agregan al final de la cola y se accede o elimina el elemento del frente.