Universidad Mariano Gálvez de Guatemala

Programación 1

Tarea final

- Pablo Daniel Reyes López 1290-22-13279
- 21 de mayo del 2023
- Sede antigua Guatemala sección A

Introducción a C++

C++ es un lenguaje de alto nivel diseñado a base de el lenguaje C, C++ se considera un lenguaje multiparadigmas pro poder trabajar los siguientes paradigmas, programación estructurada, programación modular, programación POO, etc.

Editores en C++

Para poder programar en C++, se pueden usar los IDE o entornos de desarrollo integrado, que son programas que ya traen todas las herramientas de C++ para programar. Como lo pueden ser Dev C++ o Visual Studio.

También tenemos los editores de texto, que no traen nada integrado, que se les deben añadir las demás herramientas. Como lo pueden ser Atom o Visual Studio Code.

Tipos nativos de datos

Datos primitivos

Tipos de datos que no se componen a base de otro tipo de datos, que no se pueden descomponer y son únicos

Nombre	**Funcionalidad**
Int	Variable que solo demuestra valores enteros.
Float	Variables numéricas de simbolizan números más complejos a los enteros como decimales
Char	Variable que demuestra un símbolo o carácter n que no puede operarse numéricamente
Bool	Variable que solo puede tener los valores de 0 y 1, simbolizando verdadero y falso

Datos compuestos

Son variables que se representan con campos de variables, los cuales si se pueden descomponer en diferentes campos o variables

Nombre	**Funcionalidad**
Arreglos	Variable que simboliza una cadena de variables primitivas, guardadas en espacio de memoria secuencial
Secuencias	Es un arreglo que puede variar su tamaño durante la ejecución del programa
Estructura	Es un campo que puede agrupar a datos primitivos y complejos en su interior

Bibliotecas

Una biblioteca es un archivo que contiene una cantidad de funcionalidades que podemos añadir a nuestro programa para poder utilizarlas. Las librerías más comunes que podemos encontrar en cualquier programa en C++ son:

#include<iostream>

Contiene parte del STL, que contiene las funcionalidades más básicas de C++.

#include<ostream>

Algoritmo estándar para flujo de salida.

#include<iosfwd>

Contiene declaraciones adelantadas de todas las plantillas de flujos y sus typedefs estándar.

#include<fstream>

Permite la manipulación de archivos dentro del programa, como leer y escribir.

#include<string>

Permite realizar funciones referentes a las cadenas de caracteres, disminuyendo la dificultad cuando se necesita trabajar sobre cadenas tipo char.

#include<math>

Contiene las funcionalidades referentes a las matemáticas.

#include<new>

Maneja memoria dinámica

#include<numeric>

Relativa a operadores numéricos

#include<queue>

Relativo a colas y pilas

#include<stdio>

Contiene funciones macros y manipulación de entradas y salidas.

#include<stdlib>

Contiene las funciones macros de uso general

#include<vector >

Contiene funciones referentes a los vectores o arreglos unidimensionales

#include<forward_list>

Ingresa fácilmente las listas enlazadas simples

#include<list>

Permite enlazar listas doblemente enlazadas

#include<iterator>

Contiene funciones para iterar datos

#include<regex>

Proporciona fácil acceso a expresiones regulares en comparación de patrones

#include<thread>

Útil para la programación multihilos para tener programas con múltiples hilos.

#include<time>

Útil para saber el tiempo exacto que duro la ejecución del programa o de una función.

#include<memory>

Útil para la asignación de memoria y su manipulación, como los punteros inteligentes.

#include<typeinfo>

Mecanismo que identifica el tipo de tiempo de ejecución.

Entradas y salidas de datos

En C++ tenemos diferentes maneras de recibir y mostrar datos, los mas comunes que podemos encontrar son cout y cin. Que donde cout es una función que muestra mensajes y valores de las variables, mientras que cin guarda valores dentro de una variable.

Estas serían sus sintaxis

int A;

cin>>A;

cout<<" valor de A:" <<A<<endl;

técnicas de programación

programación no estructurada: este es un paradigma de la programación mas antiguos que permite a los desarrolladores crear código Turing completo, que es fácil de desarrollar, pero muy difícil de seguir.

Este tipo de programación es bastante inconveniente por ser difícil de depurar errores, de modificar el código, también es muy redundante por no poder tener sentencias, sino que hay que repetir un mismo código varias veces, tampoco tiene comentarios, por lo que no se puede documentar.

programación estructurada: es un tipo de programación que busca la claridad, calidad y tiempo de desarrollo de la creación de un programa añadiendo 3 secuencias

- Secuencial (una línea después de otra)
- Selección (if y switch)
- Iteración (bucles for y while)

De este tipo de programación han surgido varios paradigmas más de la programación como lo puede ser la programación modular, la programación POO o la programación procedimental.

Programación procedimental: consiste en un grupo de funciones que tienen que ser repetidas varias veces, que cuando se tenga la necesidad de utilizarse se puedan llamar.

Programación modular: este tipo de programación busca desglosar el código general en varias funciones o módulos, los cuales realizan una tarea definida que se realizaran cuando se llamen de la función principal.

Programación POO

La programación POO o Programación Orientada a Objetos es n paradigma de la programación que se basa en la idea de utilizar las iteraciones de los objetos para el desarrollo de aplicaciones.

Se basa en la idea de que el mundo esta lleno de objetos y que estos nos pueden ayudar a solucionar nuestros problemas.

Conceptos básicos

- **Objeto:** informalmente se puede denominar a un objeto como algo que existe en la vida real, pero en POO es una instancia que se basa en una clase en POO, que contienen atributos y métodos utilizados en una clase.
- **Clase:** una clase es como un molde donde se definen los atributos y métodos que pueden tener un objeto,
- **Atributos:** son variables que se definen dentro de una clase que pueden tener los objetos que utilizamos en el código.
- **Método:** son funciones que se definen dentro de una clase que realiza las acciones que puede hacer un objeto.

Pilares de POO

- **Abstracción:** es el proceso de encontrar los atributos y métodos adecuados para la definición de una clase, ignorando todos los demás atributos que pueden existir en un objeto, pero que no son útiles para el desarrollo.
- **Encapsulamiento:** consiste en ocultar métodos y atributos de las demás clases, permitiendo que el programa pueda trabajar de manera más sencilla al reducir la complejidad de un programa, por ocultar los atributos y métodos de una clase.
- **Polimorfismo:** determina que puede existir varios métodos en diferentes clases con el mismo nombre, pero que estos métodos puedan realizar diferentes acciones porque los diferencia las clases.
- **Herencia:** consiste en que una clase obtiene todos los métodos y atributos de otra clase, cómo funciona la herencia genética en la vida real.

  En la herencia existes 2 términos, la superclase y la subclase, la superclase es aquella clase que hereda sus características a otra clase, mientras que la subclase en aquella que recibe toda la herencia de la superclase.

Sobrecarga de operadores

La sobrecarga de operadores aumenta la capacidad del compilador en la pprogramación POO porque permite realizar operaciones entre variables que vengan de distintos objetos, y que esas variables sean de distintos tipos, su declaración es como cualquier función, pero se le llama operador (símbolo que necesite) (variable 1, variable 2…)

```
operador (símbolo que necesite) (variable 1, variable 2…) {

instrucción;

}
```

Creación de objetos

Para crear un objeto ya debemos tener definida una clase, con sus atributos y métodos,

Lo que generalmente se hace es declarar los objetos en la función main que se vería más o menos así

```
NombreClase NombreObjeto = NombreClase (atributos…)

NombreObjeto Función ();
```

Constructores

Los constructores son funciones especiales en programación POO, los cuales siempre son ejecutados cuando se inicia la clase, que generalmente se usan para inicializar los atributos de la clase. El constructor se crea con el nombre de la clase, su lógica para la declaración es una función normal donde el tipo es el nombre de la función, que recibe sus atributos sin alguna limitación, y que realiza una acción definida entre corchetes.

Operadores Lógicos

Un operador lógico es un símbolo que se usa en las condiciones, los cuales nos indican si hay una condición or ||, and &&, y not ¡

Estructuras de control

-if: esta función permite realizar una acción, dependiendo que la condición que generalmente se utiliza con operadores lógicos, esta función tiene su contraparte que es else, esta contraparte realiza una acción cuando la condición de if no se cumpla, esta sentencia se puede repetir en una condición ya escrita de if, a esto se le conoce como if anidado, es como un if que se cumple, mientras que otro se haya cumplido.

- for: esta sentencia lo que hace es repetir una instrucción que se definió en su interior, la cual parara solo hasta que su condición se convierta en falsa, esta sentencia como las demás, se pueden anidar.

- while: esta sentencia se podría decir que es lo opuesto a for, esta sentencia repite una instrucción que se definió en su interior hasta que su condición se vuelva verdadera. También a esta sentencia se le puede añadir do while, la cual garantiza que la acción dentro de while se realice una vez.

- swtch: esta sentencia es parecida al if, pero tiene la capacidad de operar con varias condiciones, que, si una de las condiciones se cumple, realizara una acción de acuerdo a el caso, útil para realizar menús.

Entre las sentencias útiles para la realización de swith tenemos break y continue,

Break termina una sentencia, independientemente de si la sentencia pudiera seguir

Continue da la instrucción de continuar con una sentencia, aunque haya una instrucción que pare la acción.

Diferencia entre operadores de igualdad y asignación

Los operadores de asignación son los que le dan un valor a una variable, que demuestra explícitamente que una variable se le otorga ese valor, y entre esos operadores tenemos únicamente a =.

Mientras que los operadores de aguadad son los que nos dan la posibilidad de saber que una condición se cumple.

Recursividad

La recursividad en si es la resolución de problemas, a base de la repetición de una misma acción, en programación la recursividad se refiere a la capacidad de una función a llamarse a sí misma, tenemos 2 tipos de recursividad, la recursividad directa e indirecta.

- Recursividad directa: capacidad de la función de llamarse a si misma dentro del cuerpo de la misma función, que se puede parar con un ciclo de repetición para evitar la recursión infinita,
- Recursividad indirecta: es un ciclo de llamadas de funciones el cual, termina en el punto inicial de la función que empezó a llamar, este tipo de función también se debe restringir la recursión con ciclos, para que no termine en recursión infinita.

Manejo de excepciones

Las excepciones en programación nos ayudan en los momentos en que algo en el programa salga mal, como inicializar un objeto que no debería inicializarse, que la función consiga un valor que no debería tener o que retorne un valor que no debería dar, por lo que tenemos las funciones try, trown y catch, que están especializadas en capturar errores que se han inicializado, try realiza la acción con posible error, trown retorna un valor si se da el caso de que la función de un error extraño, y catch atrapa el valor de trown para realizar una acción, como enviar un mensaje de error.

Procesamiento de archivos

Los archivos son una herramienta que nos permite guardar los valores de variables dentro de nuestras unidades de almacenamiento, como un disco duro, por ejemplo, los archivos suelen ser utilizados para tener un registro de ciertas cosas, que se pueden modificar en algún momento, gracias a la librería de fstream, que trae varias herramientas para modificar un archivo, desde abrirlo, cerrarlo, cambiar datos que ya se encuentra, cambiar el modo de escritura del archivo, etc.

Arreglos y vectores

En programación un arreglo es una secuencia de datos del mismo tipo de que se guardan en una ¨línea¨ de datos secuenciales de la memoria. Que generalmente se declaran de esta manera.

Tipo Nombre variable [tamaño de vector];

Que el tamaño del vector muestra cuantos cupos de variables se pueden ingresar en el vector, aunque realmente el numero no demuestra realmente el valor del tamaño del vector, porque la cadena empieza en la posición 0, por lo que la cantidad de espacios en un vector seria de n+1.

El uso que podríamos darle a un vector seria por ejemplo las notas de un estudiante, las cuales se almacenarían dentro de un vector, para no tener que declarar varias variables para ingresar nota por nota, aunque también se le pueden dar usos más complejos.

Para pasar vectores por funciones es bastante fácil, lo que necesitamos es tener una función, que reciba el vector sin tamaño, y a la par una variable que nos de su tamaño, de esta manera

MuestraArreglo(ar_numeros,10); parte de la llamada de función

void MuestraArreglo(**int** valores[], **int** tamaño) los parámetros de la función

búsqueda de valores en un vector

para mostrar en si el valor de un vector lo más fácil es solamente mostrarlo de esta manera

cout <<" vector:" <<NombreVector[PosicionDeseada] << endl;

pero también se puede hacer que el usuario pida el valor que quiera ver y mostrarlo de esta manera

int PosicionDeseada;

cin >> PosicionDeseada;

cout <<" vector:" <<NombreVector[PosicionDeseada] << endl;

ordenamiento de vectores

para ordenar un vector según qué tan grande sea un número, se han creado varios métodos, pero el más sencillo que tenemos es el método burbuja, el cual permite elevar los valores más pequeños al inicio del vector, como burbujas, mientras que los valores más grandes van a las últimas posiciones del vector, como piedras que se hunden, (también se puede realizar de manera contraria).

Algoritmo de búsqueda

En este algoritmo se usa para encontrar un registro, este recibe u valor de ID, que el programa revisara si entre los registros hay un valor igual al ingresado, buscando uno por uno, para que se pueda mostrar, si no hay ningún registro, el programa termina de ejecutar sin mostrar nada.

Bases de datos

Oracle: esta base de datos perteneciente a la compañía Oracle es un servicio que tiene 2 variantes, una es la base de datos gratuita en la cual cualquier persona puede gestionar datos de manera limitada, y la variante de membresía, la cual tiene todas las opciones de la potencia de la base de datos sin limitaciones,

MYSQL: esta base de datos perteneciente a Microsoft es una base de datos de uso gratuito que es código cerrado, que permite la gestión de base de datos.

PostgreSQL: esta base de datos esta gestionada por una comunidad que la actualiza y modifica por ser de código abierto.

DDL y DML

El lenguaje de definición de datos es el causante de la existencia de estos tipos de instrucciones que usan las bases de datos para realizar sus acciones, que permiten el almacenamiento de datos en alguna zona que pueda almacenarlos como un disco duro.

- DDL: se usan para alterar la estructura de las tablas en la base de datos, como también los objetos de esta.

  create crea un esquema en la base de datos

  alter modifica la estructura de la base de datos.

  Drop borra objetos de la base de datos.

  Trunkate elimina todos los registros de la base de datos, incluyendo los espacios de almacenamiento.

- DML: se usa para la manipulación y consulta de los datos almacenados en la base de datos, los cuales realizan las operaciones básicas de un CRUD, Create, Read, Update, Delete.

  Select muestra los datos

  Insert añade nuevos datos a la base de datos.

  Update modifica datos ya existentes

  Delete borra algún registro.

Memoria dinámica

La memoria dinámica es un concepto el cual consiste en que se declare un espacio de la memoria de algún sitio para una variable, el cual siempre estará disponible para ella.

Para declarar una variable dinámica es necesaria usar punteros, los cuales se declaran con el símbolo *, de esta manera.

Tipo NombreVariable;

Tipo Puntero*;

Puntero = &NombreVariable;

Importante un puntero debe ser del mismo tipo de la variable que haga referencia, y para utilizar el valor de la variable debemos usar la variable puntero. Y para hacer referencia con un puntero, se debe igualar a la dirección de la variable con el símbolo &.

Con los punteros vectores pasa algo interesantes, es que cuando vemos su dirección de memoria, podemos observar la secuencia de la memoria, porque podemos ver la separación entre cada espacio de memoria entre los espacios de un vector.

Para declarar un puntero que a su vez sea un vector debemos hacer lo siguiente

Tipo Puntero*;

Puntero = new Tipo [tamañoVector];

Aritmética de punteros: la declaración de punteros se realiza por medio del símbolo * después del identificador de la variable, mientras que para saber la dirección de memoria de una variable se usa el símbolo &.

Sizeof: esta función como lo dice en su nombre puede ver la cantidad de bytes o almacenamiento que utiliza una variable primitiva o compleja, devolviéndonos un valor entero relacionada con el almacenamiento.

Relación entre punteros y apuntadores.

Ambos se relacionan por la memoria dinámica, porque ambos son esenciales para su aplicación, uno pudiendo darnos una variable dinámica, la cual no cambiara el valor hasta que la variable que haga referencia cambie

Mientras que los apuntadores nos ayudan para que un puntero pueda tener una variable a la cual hacer referencia.

Usar un puntero para una función.

Para usar un puntero por una función es bastante fácil, debemos tener un puntero haciendo referencia a una variable y hacer lo siguiente

Void función (*puntero); como una variable normal, pero con puntero

Estructuras de datos

Una estructura de datos es una estructura que nos permite almacenar datos de diferentes maneras, en si están en todas partes, por ejemplo, un numero entero es una estructura de datos bastante pequeña que no notamos su estructura en sí.

Clase auto referenciada

Una clase auto referenciada es una clase que tiene la capacidad de hacer referencia a un objeto del mismo tipo de la clase, utilizando punteros, este concepto también puede utilizarse en las estructuras de datos para hacer estructuras con objetos que hagan referencia a la misma estructura.

Listas enlazadas

Es una lista de manera dinámica, la cual consta de 2 tipos de variables en un nodo, una variable de cualquier tipo, y un puntero, ese puntero apunta a la siguiente variable en la lista, así hasta terminar con la lista, este tipo de lista puede crecer y decrecer dependiendo de la cantidad de nodos que pueda tener.

Pilas

Las pilas son estructuras de datos que consisten en el primero en el ultimo en entrar, primero en salir, las cuales como dice esa frase, el ultimo dato ingresado en la pila, será el primero en poder salir, es como una pila de platos de porcelana que queremos desmontar, primero debemos sacar el que este más arriba o último en entrar para desmontar con seguridad, imagina que sacas el plato de en medio, la pila se rompería, pero en nuestro programa de pila simplemente no funcionaria esa instrucción.

Cola

Las colas son estructuras de datos que consisten en primero en entrar, primero en salir, como lo dice su nombre, el primer dato que entra es el dato que debe salir para que los demás puedan hacer lo mismo, como una cola en cualquier negocio, donde el que llego antes, recibe su servicio y se va, y luego pasa el siguiente, pero si le dan el servicio al de en medio habría indignación en la vida real, pero en nuestro programa simplemente no funcionaría esa instrucción.