Examen 1 Teoria

Autor: Fabian Sander Hangen C07336

Correr cada programa:

Calculadora en C++: En la terminal se corre "g++ .\calculadora.cpp -o .\calculadora.exe" se crea el ejecutable y se tiene que ejecutar con el comando ".\calculadora.exe". Puede ser que si se corre el programa en linux el correr el programa sea diferente, e porgrama se desarrolla en Windows con los comandos dichos y corre de manera correcta.

Convertidor de cadena en Python: Hay que correr el comando "py cadena.py". Puede ser que si se corre el programa en linux el correr el programa sea diferente, le porgrama se desarrolla en Windows con los comandos dichos y corre de manera correcta.

Preguntas y Respuestas

¿Explique la diferencia entre paso por valor y paso por referencia en C++?

En el lengiuaje de programación C++, pasar algo por valor implica que se copie o que se cree una copia del valor en la variable que se pasa, hay que recalcar que si se pasa por valor, se crea una copia en una nueva ubicación de memoria, esta copia no afecta al valor original. Si se pasa por referencia, significa que se pasa la dirección de la memoria de esta variable que se quiere. Por lo que, si se pasa un valor por referencia, el mas mínimo cambio que se le haga a la variable copiada, se le va a hacer a la variable original, ya que se pasa la referencia de la ubicación, esta práctica es utilizada ya que se puede ahorrar memoria y recursos.

¿Qué es la recursividad y cuándo es útil utilizarla en la programación?

El recursividad es un concepto que se puede ver en muchos lenguajes de programación, este consiste en que una funcion se llame a si misma dentro de esta, un ejemplo muy utilizado para ejemplificar la recursividad es la de un número factorial. La recursividad es útil usarla cuando se quiere dividir un problema muy grande en problemas pequeños que tengan las mismas características pasos. Esta se utiliza por ejemplo en factoriales, busqueda de estructuras de datos o realizar algoritmos de ordenación que ya hemos visto.

¿Cuál es la diferencia principal entre listas y tuplas en Python? ¿Cuándo es preferible utilizar cada una?

En el lenguaje de programacion Python, las listas y las tuplas pueden llegar a verse muy similares, ya que ambas coleccionan elementos. La diferencia mas grande entre estas dos estructuras es que las listas son editables (mutables) y las tuplas no. Esto quiere decir que las listas pueden cambiar elementos de su interior y las tuplas no. Normalmente se utilizan las listas cuando se espera o quiere que los elementos cambien con el tiempo o programa, de lo contrario, se pueden utilizar las tuplas.

Explique el concepto de herencia en la programación orientada a objetos.

La herencia en la OOP es un concepto donde se "heredan" caracteristicas y metodos de una clase a la otra. Esta es utilizada ya que ayuda a la reutilización de código y para crear una atmósfera de relación entre clases. Un ejemplo muy usado, y que tambien lo vimos en clase, es el de la clase "Animal", la cual puede tener sus

propiedades como "nombre" y "edad", asemas se puede crear una clase "Perro" que hereda las propiedades, esta clase perro puede tener sus propias "características" como "raza". Hay que mencionar que hay tipos de herencia, como la herencia "pública", la "protegida" y la "privada".

¿Qué es la comprensión de listas (list comprehension) en Python? Dé un ejemplo de su uso y explique cómo puede simplificar el código.

La compresión de listas en el lengiuaje de programacóon de Python es una forma ahorrativa y utilizada de comprimir código en una sola línea. EL profesor explicó en clase que esta práctica es característica de los programadores de Python ya que ahorra tiempo y espacio, aunque puede ser confusa para los programadores principiantes.

Ejemplo:

En este cas se comprimio el total de 8 lineas de código en 3 lineas. En este caso se toma ventaja de que los if y los else son comprimibles, por eso es que se ahorra espacio y esta práctica es tan utilizada.

Codigo:

```
Ejemplo:

numeros = [1, 2, 3, 4, 5]
nueva_lista = []
for numero in numeros:
    if numero % 2 == 0:
        nueva_lista.append(numero * 2)
    else:
        nueva_lista.append(numero * 3)
print(nueva_lista)

Nuevo:

numeros = [1, 2, 3, 4, 5]
nueva_lista = [numero * 2 if numero % 2 == 0 else numero * 3 for numero in numeros]
print(nueva_lista)
```

Explique el concepto de plantillas (templates) en C++. ¿Cuál es su utilidad y cómo se utilizan en la programación genérica?

Las plantillas en C++ son características que se utilizan en la programación con C++, ya que ahorra espacio y tiempo a la hora de programar con funciones parecidas. Se sabe que las plantillas son un recurso que se utiliza, este permite que las clases y las funciones sean utilizados con tipos de datos genericos para que una funcion o clase real pueda utilizar su "molde", esto permite la reutilización del código y la rapidez de programación. Un ejemplo de esta práctica es al utilizar muchos tipos de datos como flotantes, enteros y strings, en este caso ahorra tiemo en la creacion de la función o clase.

¿Qué es la sobrecarga de operadores en C++? Dé un ejemplo de cómo se puede sobrecargar un operador y cómo puede mejorar la legibilidad y la usabilidad del código.

La sobrecarga de operadores en C++ permite que los operadres se comporten de una manera distinta, dependiendo del deseo del creador o usuario. Esto depende de los tipos de los operandos. ESta práctica

permite que el código sea mas fácil de entender y utilizar y permite redefinir el comportamiento de los operadores estándar (como +, -, *, /,). Esta práctica ayuda a que las clases persobnalizadas sean utilizadas de manera intuitiva para el operador.

Ejemplo:

En este caso se sobrecarga el operador +, ya que se suman los elementos de de clase "Box".

Codigo:

```cpp
#include <iostream>

class Box {
private:
    int content;

public:
    Box(int content = 0) : content(content) {}

    // Sobrecargar el operador +
    Box operator+(const Box& b) {
        return Box(this->content + b.content);
    }

    void print() const {
        std::cout << "Box(" << content << ")\n";
    }
};

int main() {
    Box b1(10);
    Box b2(20);
    Box b3 = b1 + b2;  // Esto es posible gracias a la sobrecarga del operador +

    b3.print();  // Imprime: Box(30)

    return 0;
}
```

Explique la diferencia entre una función y un método en Python. ¿Cómo se accede a un método de una clase y qué es el argumento self?

En el lenguaje de programación de Python, la funcion es un pedazo de codigo que realiza una accione en específica, esta depende de los parámetros que se le ingresen, esta es reutilizabe pero ademas es indempendidnte de las clases y de los objetos. Un método es parecida a una función. El método es una función que esta relacionado con las clases y los objetos. Los métodos se acceden a través de una instancia de una clase, los métodos modifican los objetos a los que estan asociados. Para acceder a un método de una clse, primero se tiene que crear una instancia de la clase y luego se tiene que llamar al método. El argimento "self" es una referencia al objeto que esta llamando al método. El programa normalmente llama el objeto

como el primer argumento del método. El "self" es un método especial, este se utiliza para dar referencia de la clase que se utiliza.