INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

Ingeniería en Sistemas Computacionales

8° Semestre

Alumno: José Félix Alvarado Barrios

NC:16070124

Materia: Prog. Lógica y Funcional

Nombre del trabajo: Mapa conceptual

Docente: ISC Salvador Acevedo Sandoval

Jerez de García Salinas a 7 de febrero de 2020

1. **¿Qué es un paradigma de programación?**

Puede ser definido como un método para resolver un problema o para realizar alguna tarea. Un paradigma de programación es un enfoque para resolver problemas utilizando un lenguaje de programación o se puede decir que es un método para resolver un problema usando herramientas y técnicas que están disponibles siguiendo algún enfoque.

2. **¿Qué paradigmas de programación existen?**

Imperativo.

Declarativo.

Lógico.

Funcional.

Orientado a objetos.

3. **¿Cuáles son las características que definen a cada uno de ellos?**

Paradigma imperativo: Los programas se componen de un conjunto de sentencias que cambian su estado.

Paradigma declarativo: Opuesto al imperativo. Los programas describen los resultados esperados sin mostrar explícitamente los pasos a llevar a cabo para alcanzarlos.

Paradigma funcional: Los programas funcionales se basan en el uso de una o más funciones dentro de las cuales se pueden utilizar funciones creadas anteriormente. Su objetivo es dividir el programa en módulos de forma que cada uno de éstos realice una única función.

Paradigma lógico: El proceso de elaboración de programas está basado en la lógica de primer orden y, a diferencia de los demás paradigmas, especifica qué debe hacer el programa y no cómo hacerlo.

Paradigma orientado a objetos (POO): Las características del paradigma orientado a objetos son encapsulamiento, abstracción, polimorfismo y herencia.

4. **Ejemplos de dichos paradigmas**

Estructurada

```
#include<stdio.h>

#include<math.h>

int cubo(int);

main ()

{

int a;

printf(“Ingrese un entero: ”);

scanf(” %d”, &a);

printf(“El cubo de %d es: %d\n”, a, CalculaCubo(a));
```

Se invoca funcion

```
getch();
}
/* Definicion de la funcion CalculaCubo */
int CalculaCubo (int x)
{
int cubo;
cubo = x*x*x;
return cubo;
}
```

Declarativo.

```
List<Integer> numeros = List.of(18, 6, 4, 15, 55, 78, 12, 9, 8);
int contador = 0;
for(int numero : numeros) {
    if(numero > 10) {
        contador ++;
```

```
    }
}
System.out.println(contador);
```

Imperativo

```
Long result = numeros.stream().filter(num -> num > 10).count();
System.out.println(result);
```

lógico

```
% Hechos:
es_español("Manolo").
es_italiano("Marco").
es_colombiano("Marcelo").

% Reglas:
es_europeo(A) :- es_español(A).
es_europeo(A) :- es_italiano(A).
```

```
es_americano(A) :- es_colombiano(A).

es_terricola(A) :- es_europeo(A).

es_terricola(A) :- es_americano(A).

son_del_mismo_continente(A,B) :- es_europeo(A), es_europeo(B).

son_del_mismo_continente(A,B) :- es_americano(A), es_americano(B).
```

Funcional

```
Object Temporizador {

def unaVezPorSegundo( repite : () >= unit ) {

while (true) { repite(); Thread sleep 1000 }

}

def elTiempoVuela() {

println("el tiempo pasa volando.....")

}

def main(args : Array[String]) {

unaVezPorSegundo( elTiempoVuela )
```

```
}

}
```

Orientada a Objetos

```
class Objeto():
    color = "verde"
    tamanio = "grande"
    aspecto = "feo"
    antenas = Antena()
    ojos = Ojo()
    pelos = Pelo()


    def flotar(self):
        pass
```

5. **Aplicaciones reales donde se implementan dichos paradigmas:**

Diseño de computadoras.

Concurrencia y sistemas paralelos.

Sistemas altamente seguros.

6. **Ventajas y desventajas de cada paradigma.**

Imperativa

Ventajas:

Su relativa simplicidad y facilidad de implementación de los compiladores e intérpretes.

La capacidad de reutilizar el mismo código en diferentes lugares en el programa sin copiarlo.

Una forma fácil seguir la pista de flujo del programa.

La capacidad de ser muy modular o estructurado.

Necesita menos memoria solamente.

Desventajas:

Los datos son expuestos a la totalidad del programa, así que no hay seguridad para los datos.

Dificultad para relacionarse con los objetos del mundo real.

Difícil crear nuevos tipos de datos reduce la extensibilidad.

Se da importancia a la operación de datos en lugar de los datos mismos.

Declarativos

Desventajas:

No puede resolver cualquier problema dado, solo un subconjunto de problemas para los que el intérprete o compilador fue diseñado.

El proceso de interpretación es más lento por que lleva acabo una fase de interpretación extra.

Ventajas:

La solución de un problema se puede realizar a un nivel de abstracción más alto.

Permite utilizar variables para almacenar valores intermedios.

Una de las ventajas más significativas es que se le indica a la computadora que tarea se realizará en lugar de como.

Funcional

Ventajas:

Altos niveles de abstracción.

Código declarativo y comprensible.

La evaluación perezosa.

Mayor probabilidad de aplicar expansión en línea.

Optimizaciones a partir de la utilización de funciones puras.

Desventajas:

Dificultad inicial para producir buen código.

Generación de grandes cantidades de short-lived garbage.

Menor eficiencia en el uso de CPU comparados con su contraparte imperativa.

Lógica

Ventajas

Descripciones independientes de la implementación (unificación semántica).

Puede mejorarse la eficiencia modificando el componente de control sin tener que modificar la lógica del algoritmo.

Base de conocimiento fácilmente escalable.

Relaciones multipropósito.

Expresión simple y precisa de los problemas.

Generación rápida de prototipos e ideas complejas.

Sencillez en la implementación de estructuras complejas.

Potencia.

Desventajas

Dependiendo del problema a solucionar, la implementación y el motor de inferencia, puede llegar a ser extremadamente ineficiente.

Pocas y muy específicas áreas de aplicación.

Existen muy pocas herramientas de depuración, en su mayoría poco efectivas.

En problemas reales, es poco utilizado.

Si el programa no contiene suficiente información para responder una consulta la respuesta puede ser una que se preste para malentendidos.

Inferencia limitada por su base de conocimiento.

Orientado a Objetos

Ventajas:

Los componentes se pueden reutilizar.

Facilidad de mantenimiento y modificación de los objetos existentes.

Una estructura modular clara se puede obtener, la cual no revelará el mecanismo detrás del diseño.

Se proporciona un buen marco que facilita la creación de rica interfaz gráfica de usuario aplicaciones (GUI).

Se acopla bien a la utilización de bases de datos, debido a la correspondencia entre las estructuras.

Desventajas:

Limitaciones del programador.

No hay una forma única de resolver los problemas.

Se requiere una documentación amplia para determinar la solución planteada.

7. **¿Qué es una función (desde el punto de vista matemático)?**

En matemáticas, se dice que una magnitud o cantidad es función de otra si el valor de la primera depende exclusivamente del valor de la segunda.

De manera más abstracta, el concepto general de función, aplicación o mapeo se refiere en matemáticas a una regla que asigna a cada elemento de un primer conjunto un único elemento de un segundo conjunto.

8. **¿Qué es la Programación Funcional?**

El paradigma de programación funcional pretende eliminar los cambios de estado, para ello se basa principalmente en la composición de funciones, la transparencia referencial y las funciones puras.

El paradigma funcional separa las estructuras de datos y las funciones que operan sobre ellas. Los programas se construyen mediante la composición de funciones, de manera que una función realiza su trabajo llamando a otras funciones cada vez más simples hasta alcanzar las primitivas del lenguaje.

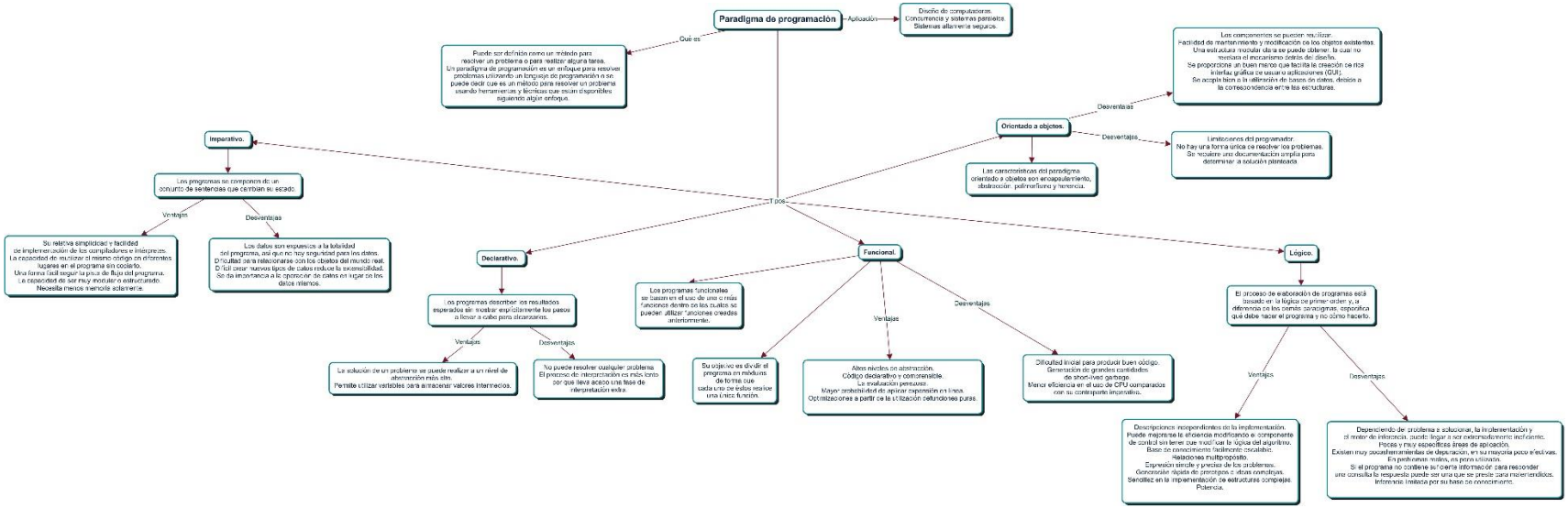
9. **¿Qué es una expresión matemática?**

Es una ecuación que relaciona constantes o variables matemáticas y que se expresa mediante una igualdad matemática.

Una expresión matemática es una combinación de símbolos que pueden designar números (constantes), variables, operaciones, símbolos de agrupación y otros signos de puntuación.

10. ¿Qué es la reducción (simplificación o evaluación) de expresiones matemáticas?

La evaluación de una expresión algebraica consiste en sustituir el o los valores proporcionados de las variables, para encontrar el valor numérico de la expresión.


Paradigma de programación
Aplicación
Diseño de computadoras.
Concurrencia y sistemas paralelos.
Sistemas altamente seguros.
Qué es
Puede ser definido como un método para resolver un problema o para realizar alguna tarea.
Un paradigma de programación es un enfoque para resolver problemas utilizando un lenguaje de programación o se puede decir que es un método para resolver un problema usando herramientas y técnicas que estan disponibles siguiendo algún enfoque.
Tipos
Imperativo.
Los programas se componen de un conjunto de sentencias que cambian el estado.
Ventajas
Su relativa simplicidad y facilidad de implementación de los compiladores e intérpretes.
La capacidad de realizar el mismo código en diferentes lugares en el programa sin costeto.
Una forma fácil seguir la pista de flujo del programa.
La capacidad de ser muy modular o estructurado.
Necesita menos memoria solamente.
Desventajas
Los datos son expuestos a la falsedad del programa, así que no hay seguridad para los datos.
Dificultad para relacionarse con los objetos del mundo real.
Difícil crear nuevos tipos de datos reduce la escencialidad.
Se da importancia a la operación de datos en lugar de los datos mismos.
Orientado a objetos.
Las características del paradigma orientado a objetos son encapsulamiento, abstracción, polimorfismo y herencia.
Desventajas
Limitaciones del programador.
No hay una forma única de resolver los problemas.
Se requiere una documentación amplia para determinar la solución planteada
Desventajas
Los componentes se pueden reutilizar.
Facilidad de mantenimiento y modificación de los objetos existentes.
Una estructura modular de la se puede obtener, la cual no revelan el mecanismo detrás del diseño.
Se proporciona un buen marco que facilita la creación de una interfaz gráfica de usuario aplicaciones (GUI).
Se acepta bien a la utilización de bases de datos, debido a la correspondencia entre las estructuras.
Declarativo.
Los programas describen los resultados esperados sin mostrar explícitamente los pasos a llevar a cabo para alcanzarlos.
Ventajas
La solución de un problema se puede realizar a un nivel de abstracción más alto.
Permite utilizar variables para almacenar valores intermedios.
Desventajas
No puede resolver cualquier problema.
El proceso de interpretación es más lento porque lleva acabo una fase de interpretación extra.
Funcional.
Los programas funcionales se basan en el uso de uno o más funciones dentro de los cuales se pueden utilizar funciones creadas anteriormente.
Su objetivo es dividir el programa en módulos de forma que cada uno de ellos realice una única función.
Ventajas
Altos niveles de abstracción.
Código declarativo y comprensible.
La evaluación perezosa.
Mayor probabilidad de aplicar ejecución en línea.
Optimizaciones a partir de la utilización de funciones puras.
Desventajas
Difícultad lineal para producir buen código.
Generación de grandes cantidades de short-lives garbage.
Menor eficiencia en el uso de CPU comparados con su contraparte imperativo.
Lógico.
El proceso de elaboración de programas está basado en la lógica de primer orden y, al diferencia de los demás paradigmas, especifica qué debe hacer el programa y no cómo hacerlo.
Ventajas
Descripciones independientes de la implementación.
Puede mejorarse la eficiencia modificando el componente de control sin tener que modificar la lógica del algoritmo.
Base de conocimiento fácilmente creable.
Relaciones multipropósito.
Expresión concisa y precisa de los problemas.
Generación rápida de prototipos o ideas complejas.
Sencillez en la implementación de estructuras complejas.
Potencia.
Desventajas
Dependiendo del problema a solucionar, la implementación y el motor de inferencia, puede llegar a ser extremadamente ineficiente.
Pocas y muy específicas áreas de aplicación.
Existen muy pocas herramientas de depuración, en su mayoría poco efectivas.
En ocasiones reales, es poco utilizado.
Si el programa no contiene suficiente información para responder una consulta la respuesta puede ser una que se presta para malentendidos.
Inferencia limitada por su base de conocimiento.

Referencias

Cifuentes, D., & Zaldua, A. (2017). *Programación Funcional*. Obtenido de Programación Funcional: http://ferestrepoca.github.io/paradigmas-de-programacion/progfun/funcional_teoria/index.html

Olaya, F. (9 de Septiembre de 2014). *Paradigma Imperativo*. Obtenido de Paradigma Imperativo: https://es.slideshare.net/JFREDYOLAYARAMOS/paradigma-imperativo-39302522

Programación Lógica. (s.f.). Obtenido de Programación Lógica: https://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/lang.html

Programación Orientada a Objetos. (s.f.). Obtenido de Programación Orientada a Objetos: https://objetosweb.wordpress.com/2016/05/18/ventajas-y-desventajas-de-la-programacion-orientada-a-objetos/