ESTRUCTURA DE DATOS

TEMA:
TABLA DE AMORTIZACIÓN

TUTOR:
ING. FERNANDO SOLIS

GRUPO N°5

INTEGRANTES:

JUNNIOR JURADO
THEO ROSERO
YULLIANA ROMAN
ALEX PAGUAY
SANTIAGO SAÑAY

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

CARRERA: ING. EN SOFTWARE	**GUÍA** No. 1	**TIEMPO ESTIMADO:** 1h y 30 min.
ASIGNATURA: Estructura de Datos	**FECHA DE ELABORACIÓN:** **SEMESTRE**: mayo 2021 – Sept. 2021	
TÍTULO: Tabla de Amortización	**DOCENTE:** Ing. Fernando Solís	

OBJETIVO PRINCIPAL

Desarrollar un sistema que permitirá calcular la tabla de amortización del usuario, utilizando el lenguaje de programación C++ el mismo que permitirá agilizar el cálculo del mismo en una entidad financiera.

OBJETIVO ESPECÍFICOS

Realizar el análisis y diseño del sistema para que pueda generar la tabla de amortización compuesta por:

- Cálculo de fechas a pagar.
- Número de cuotas.
- Monto a cancelar dependiendo del interés.
- Generar correo electrónico.
- Manejo de archivos.

DESARROLLO

Se profundizan los conceptos de la programación orientada a objetos en el lenguaje de programación C++. Relacionando el acceso y el control de visibilidad de los miembros de cada clase con las características propias de los paradigmas de orientación a objetos como ocultamiento y encapsulamiento.[1]

Realizando la definición de 16 clases adecuadas, con sus respectivos atributos y métodos creando e inicialización los objetos con sus constructores.

Se trató de simplificar el manejo del programa mediante el tratamiento de excepciones para poder destacar el valor fundamental de las excepciones para “informar errores”, de una manera personalizada.[2]

Utilizar punteros es adecuado e ideal optimizando el recurso de memoria ya que el sistema mismo es el encargado de reservar el espacio necesario para evitar el desperdicio de recursos que es frecuente al utilizar memoria dinámica al no utilizar punteros ni pasar los valores por referencia.

El sistema debe cumplir con varios requisitos funcionales y no funcionales los mismos que fueron tomados en cuenta para el desarrollo y validación del sistema, tratando de dar un programa óptimo para el usuario y al mismo tiempo eficaz para entregar la tabla de amortización adecuado , con las fechas de pago ideales sin tomar en cuenta los fines de semana, no feriados.[3]

Entregando un sistema de amortización genérico para cualquier entidad financiera y adecuado para su uso.

CONCLUSIONES

El uso de memoria dinámica es fundamental para optimizar los recursos de la computadora, para evitar colapsar el sistema operativo.

El uso de listas nos permite un mejor manejo de la información en un versus los vectores convencionales, ya que al manejar nodos permite el acceso a cierta información rompiendo el orden de los factores e ingresando al argumento de la lista sin tener que recorrer toda la lista optimizando el tiempo de trabajo.[4]

Se utilizó el formato de Json para el manejo de la información, aplicando persistencia de datos, y minimizando el riesgo de perder alguna información.

RECOMENDACIONES

Solicito se profundice el estudio en la estructura de datos y en el manejo de memoria dinámica motivó que es fundamental para nuestra carrera en Ingeniería de Software, el buen uso de estos recursos para el desarrollo de sistemas y aplicativos informáticos.

BIBLIOGRAFÍA

[1] B. Cyganek, " C ++ Basics ," in *Introduction to Programming with C++ for Engineers*, Wiley, 2020, pp. 43–225.

[2] Z. Xue and D. B. Thomas, "SynADT: Dynamic Data Structures in High Level Synthesis," in *Proceedings - 24th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2016*, Aug. 2016, pp. 64–71, doi: 10.1109/FCCM.2016.26.

[3] D. Häggander, P. Lidén, and L. Lundberg, "A method for automatic optimization of dynamic memory management in C++," in *Proceedings of the International Conference on Parallel Processing*, 2001, vol. 2001-January, pp. 489–498, doi: 10.1109/ICPP.2001.952096.

[4] L. Séméria, K. Sato, and G. De Micheli, “Resolution of dynamic memory allocation and pointers for the behavioral synthesis from c,” *Proc. -Design, Autom. Test Eur. DATE*, pp. 312–319, 2000, doi: 10.1109/DATE.2000.840289.