UNIVERSIDAD DE LA FUERZAS ARMADAS
ESPE

ESTRUCTURA DE DATOS

Integrantes: Aguirre Gabriel, Arévalo Dalton, Jiménez Diego, Jacome Andres, Zapata Lizzette

Docente: Ing. Fernando Solís

NRC: 3685

INFORME

TEMA: Aplicación de consola desarrollada con c++ para generar la tabla de amortización de los clientes de una entidad financiera utilizando listas dobles.

1. OBJETIVO GENERAL

Utilizar listas dobles para manejar los datos generados por una entidad financiera mediante el uso de clases para gestionar la emisión de créditos.

2. OBJETIVO ESPECÍFICOS

a. Automatizar las fechas de pago de los créditos considerando que estos solo se pueden realizar en días hábiles.
b. Gestionar apropiadamente las excepciones que el usuario pueda cometer al momento de utilizar la aplicación.
c. Generar un documento PDF que contenga la información detallada del cliente.
d. Generar un correo electrónico único para cada cliente utilizando sus nombres y apellidos.

3. DESARROLLO

3.1 PLANTEAMIENTO DEL PROBLEMA.

Se pide generar un programa en C++ donde conste una tabla de amortización que coste de nombre, apellido, cédula, dirección, teléfono, monto, fecha de pago y el plazo a pagar, esto mediante el uso de listas dobles con sus respectivas validaciones y el manejo de clases, del mismo modo es importante la implementación de pruebas de unidad y modelado del código descrito.

3.2 DESCRIPCIÓN DEL PROCESO.

3.2.1 USO DE CLASES.

En el proceso de desarrollo del proyecto se aplicó el uso de clases para ello es importante tener en claro los conceptos de clases y objetos en c++ una clase es un nuevo tipo de dato que sirve para crear objetos en la programación y así se obtiene una consistencia lógica relacionada con sus miembros. (Aguilar, 2007)

3.2.1.1 main

Es la clase principal del programa se utiliza la cabecera application para generar un variable llamada app de tipo Application la cual invocará el método correr el cual da inicio al programa.

3.2.1.2 application

En esta clase se genera el menú principal del programa utiliza cabeceras como application, menu, menu_option, person_builder, persons_controller, en el constructor de Application se genera un registro de dependencias, y se generan las opciones “Personas”, “Crédito”, “Salir”.

3.2.1.3 controller

En esta clase se va a realizar el control de la clase application se genera un atributo app de Application de tipo puntero, en el constructor de Controller se pasa como parámetro el puntero de app y se prosigue a generar un get_app que retornara app.

3.2.1.4 date

En la clase date debe constar los atributos dia, mes y año para poder hacer uso de sus métodos en la clase cpp respectiva donde se va a desarrollar las validaciones correspondientes como por ejemplo si el dia es laborable o no.

3.2.1.5 menu_option

En esta clase se utilizan las librerías nativas de c++ como son string, functional, vector, map, memory, alli se crean las clases MenuOptionArgumentBase, MenuOptionArgument, MenuOptionArguments y MenuOption donde ocuparemos las clases mencionadas anteriormente.

3.2.1.6 menu

La clase menú sirve como interfaz de usuario para gestionar las opciones que puede seleccionar el cliente. Esta clase hace uso de las librerías windows.h, vector, iostream y la clase menu_option.h mencionada anteriormente. El control de la interfaz es mediante las flechas y los caracteres del teclado.

3.2.1.7 payment

En la clase payment debe incluirse la clase date para poder obtener los métodos de esta clase y aplicarlos al código de pago, junto con el capital y el interés correspondiente que será utilizado con cada persona correspondiente.

3.2.1.8 person

La clase persona funciona como un tipo de dato abstracto que contiene la información de la persona como la cédula, nombre, apellido, dirección, teléfono, y email.

3.2.1.9 person_buider

En esta clase se utilizan las librerías nativas de c++ como son string, map y la cabecera person, en ella se construye los datos de una persona (cédula, nombre, apellido, dirección, teléfono), así como también se generará un email

3.2.1.10 persons_controller

En esta clase se hace uso de las librerías iostream, regex y vector, así como de las cabeceras controller, persons_controller, person_builder, utils y menú, como atributos se crea una lista de tipo persona llamada personas y un Menú de tipo menú en la cual se registrará, mostrará a la persona así también un submenú al cual se podrá ingresar desde la opción “Persona” que se encuentra en el menú principal aquí se encontrará las opciones de “Registrar nueva persona”, “Ver personas”, “Regresar”.

3.2.1.11 linked_list

Esta clase sirve para generar y gestionar listas de forma dinámica. Entre los métodos más relevantes que posee la lista están agregar por el frente, agregar por la cola, insertar en determinada posición, eliminar por índice, buscar, modificar por cada ítem, modificar hasta que se cumpla determinada condición, devolver último nodo, devolver nodo por índice. La clase utiliza polimorfismo y funciones lambda para extender la funcionalidad y así generar un código escalable en el tiempo.

3.2.1.12 node

En esta clase se implementa y crea un nodo el cual no es más que una estructura dinámica que contiene datos y su respectivo puntero con métodos como (set & get) dato, nodo_anterior, nodo_siguiente.

3.2.1.13 utils

En esta clase se hace uso de las librerías string, iostream, limits, algorithm y de las cabeceras linked_list y person, en esta se generará el email y ciertas validaciones que son necesarias para el ingreso de datos al programa como son la validación de la cédula entre otros métodos que son necesarios.

3.2.1.14 pch (PreCompiledHeader)

Su función es hacer caché de todas las librerías que se encuentran, de esa forma compila en menos tiempo el programa.

3.3 MODELADO.

4. CONCLUSIONES

Las listas dobles demuestran ser una estructura de datos óptima para gestionar el manejo de información creciente y cambiante en el tiempo como lo son los datos generados por una entidad financiera emisora de créditos.

Las fechas de pagos tienen que considerar las disposiciones de la sociedad en que se vive por lo que se tiene en cuenta que los pagos no se pueden realizar los días festivos y los fines de semana.

Las interfaces de usuario deben ser capaces de manejar cualquier tipo de situación que le presente el usuario de manera que pueda prevenir errores y de igual manera que sea capaz de corregirlos.

El sistema es capaz de proveer documentación detallada de la información de los clientes registrados en la entidad financiera así como de su estatus crediticio mediante la generación de archivos pdf que contengan esta información. También es capaz de generar correos electrónicos personalizados que consideran que existen nombres parecidos que pueden provocar correos repetidos y cómo solucionarlos.

5. RECOMENDACIONES

Se recomienda utilizar el patrón de diseño Modelo Vista Controlador para separar la lógica del sistema, la interfaz del usuario y las acciones que puede ejecutar el programa.

Se recomienda utilizar los comandos try, catch y through para gestionar el manejo de excepciones en lugar de los condicionales if y else que muestran mensajes de error por consola.

Se recomienda utilizar plantillas y polimorfismo para aumentar la escalabilidad del código haciendo que diferentes tipos de datos abstractos dispongan de los métodos que ya han sido implementados.

6. BIBLIOGRAFÍA

Joyanes, L. A. (2007). *Estructura de datos c++*. Madrid: McGraw - Hill / Interamericana.