**ВОЛИНСЬКИЙ КОЛЕДЖ**

**НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ХАРЧОВИХ ТЕХНОЛОГІЙ**

**ЦИКЛОВА ВИПУСКОВА КОМІСІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КУРСОВА РОБОТА**

з дисципліни

**«Об’єктно-орієнтоване програмування»**

на тему:

*«Система обліку відділень банку»*

Студента *ІІІ* курсу групи

ІПЗ-32

Спеціальності:

*Інженерія програмного забезпечення*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник:  *викладач*\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь)

\_\_\_\_\_\_\_\_\_\_\_\_*Демедюк Р.С.* \_\_\_ \_\_\_\_\_

(прізвище та ініціали)

Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії:

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ *Демедюк Р.С. .*\_\_\_\_

(підпис) (прізвище та ініціали)

Луцьк 2021

**ВОЛИНСЬКИЙ КОЛЕДЖ**

**НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ХАРЧОВИХ ТЕХНОЛОГІЙ**

**ЦИКЛОВА ВИПУСКОВА КОМІСІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Освітньо-кваліфікаційний рівень: *молодший спеціаліст*

Спеціальність: *121 Інженерія програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_*

(шифр і назва)

ЗАТВЕРДЖУЮ

голова циклової випускової комісії

інформаційних технологій

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н. Ройко О.Ю.

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 року

**ЗАВДАННЯ**

на курсову роботу студенту

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

ВСТУП

РОЗДІЛ 1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз задачі, виявлення ключових завдань

1.2 Аналіз алгоритмів та програмних засобів для рішення задачі

РОЗДІЛ 2 ПРОЕКТНА ЧАСТИНА

2.1 Загальні відомості про програмне забезпечення

2.2 Функціональна схема взаємодії модулів ПЗ

2.3 Опис окремого модуля

2.4 Опис алгоритму роботи програми

2.5 Використані стандартні бібліотеки

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ОПИС ПРОГРАМНОГО ЗАСОБУ

3.1 Опис середовища розробки

3.2 Опис основних модулів

3.3 Опис розробленого програмного забезпечення

Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_ *Демедюк Р.С..*

(підпис)

Завдання прийняв до виконання: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Вступ

Програмне забезпечення — сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм.

Розробка програмного забезпечення — це процес, спрямований на створення та підтримку працездатності, якості та надійності програмного забезпечення, використовуючи технології, методологію та практики з інформатики, керування проектами, математики, інженерії та інших областей знаннь.

Як й інші традиційні інженерні дисципліни, розробка програмного забезпечення має справу з проблемами якості, вартості та надійності. Деякі програми містять мільйони рядків вихідного коду, які, як очікується, повинні правильно виконуватися в умовах, що змінюються. Складність ПЗ порівнянна з складністю найбільш складних з сучасних машин, таких як літаки.

Завданням курсової роботи є покращення навичок з дисципліни «Об’єктно орієнтоване програмування» та розробка програмного забезпечення інформаційної системи. Для виконання завдання необхідно виконати наступні завдання:

* виконати аналіз завдання;
* проектування програмного забезпечення;
* розробити алгоритм роботи програми;
* написати програмний код який реалізує алгоритм;

# Аналітична частина

## Аналіз задачі, виявлення ключових завдань

Предметною областю використання даного ПЗ є облік відділень банку і також інформації в банку.

В ПЗ надати можливість переглядати і редагувати інформацію про відділення, працівників і клієнтів банку також їх фінансовий стан. В основі моделі даних проекту є таблиці які зберігають дані про відділення, працівників, клієнтів та вклади.

Ключовими завданнями є:

* проектування структури бази даних;
* розробка алгоритмів, користувацького інтерфейсу, редагування та пошуку;
* програмна реалізація алгоритмів на мові C#;
* оформлення супровідної документації (пояснюючої записки).

## Аналіз алгоритмів та програмних засобів для рішення задачі

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Delphi, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

**Інкапсуляція** — один з трьох основних механізмів об'єктно-орієнтованого програмування. Йдеться про те, що об'єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів (методів).

**Поліморфізм** — концепція в програмуванні та теорії типів, в основі якої лежить використання єдиного інтерфейсу для різнотипних сутностей або у використанні однакового символу для маніпуляцій над даними різного типу.

Типи поліморфізму:

* Спеціалізований поліморфізм — коли функції з однаковою назвою реалізовують різну логіку для різних типів вхідних параметрів. Підтримується в багатьох мовах програмування через перевантаження функцій.
* Параметричний поліморфізм — коли код написаний без указування конкретного типу параметрів. В ООП це називається узагальнене програмування. Це основний тип поліморфізму в функційному програмуванні.
* Поліморфізм підтипів — коли під одним ім'ям може використовуватись декілька типів похідних від одного базового. Основний тип поліморфізму в ООП.

**Успадкування** — механізм утворення нових класів на основі використання вже існуючих. При цьому властивості та функціональність батьківського класу переходять до класу нащадка.

**Нащадок** — це більш ніж базовий клас, тому він може використовуватися скрізь, де використовується базовий клас, але не навпаки.

**Об'єкт** – це базове поняття в ООП, це конкретна реалізація, екземпляр класу. Об'єкт складається з трьох частин: стан (змінні стану), методи (операції), ім'я об'єкта.

**Клас** – це якась безліч об'єктів, що мають загальну структуру й загальне поводження.Будь-який конкретний об'єкт є просто екземпляром класу. Що ж не є класом? Об'єкт не є класом, хоча надалі побачимо, що клас може бути об'єктом. Об'єкти, не зв'язані спільністю структури й поводження, не можна об'єднати в клас, тому що по визначенню вони не зв'язані між собою нічим, крім того, що всі вони об'єкти. Класи – серце кожної об’єктно-орієнтованої мови.

**Поле.** Так називається член-змінна, яка має деяке значення. В ООП поля іноді називають даними об'єкта. До поля можна застосовувати кілька модифікаторів залежно від того, як ви збираєтеся це поле використовувати. У число модифікаторів входять static, readonly і const.

Нижче познайомимося з їхнім призначенням і способами їхнього застосування.

**Метод.** Це реальний код, що впливає на дані об'єкта (або поля). Фактично, методи – це функції, які визначають певні дії з даними.

**Властивості.** Їх іноді називають "розумними" полями (smart fields), тому що вони насправді є методами, які клієнти класу сприймають як поля. Це забезпечує клієнтам більший ступінь абстрагування за рахунок того, що їм не потрібно знати, чи звертаються вони до поля прямо або через виклик метода-аксессора.

**Константи.** Як можна припустити, виходячи з ім'я, константа — це поле, значення якого змінити не можна. Нижче обговоримо константи й зрівняємо їх із сутністю за назвою незмінні (readonly) поля.

**Події.** Подія викликає виконання деякого фрагмента коду. Події – невід'ємна частина програмування для Microsoft Windows. Наприклад, події виникають при русі миші, клацанні або зміні розмірів вікна.

**Оператори.** Використовуючи перевантаження операторів С#, можна додавати до класу стандартні математичні оператори, які дозволяють писати більш інтуїтивно зрозумілий код.

**«Microsoft Access»** (повна назва Microsoft Office Access) — [система управління базами даних](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) ,[програма](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), що входить до складу [пакету офісних програм](https://uk.wikipedia.org/wiki/%D0%9E%D1%84%D1%96%D1%81%D0%BD%D0%B8%D0%B9_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82) [Microsoft Office](https://uk.wikipedia.org/wiki/Microsoft_Office). Має широкий спектр функцій, включаючи зв'язані запити, сортування по різних полях, зв'язок із зовнішніми таблицями і базами даних. Завдяки вбудованій мові [VBA](https://uk.wikipedia.org/wiki/VBA), в самому Access можна писати підпрограми, що працюють з старими версіями Microsoft Office Access.

# Проектна частина

## Загальні відомості про програмне забезпечення

Розроблене програмне забезпечення (ПЗ) має назву «Банк» і складається з наступного комплексу компонентів:

* + - Program.cs – основний файл програми, функція main;
    - fLoad.cs – файл коду де міститься завантажувальна форма;
    - fLoad.Designer.cs – файл дизайнера завантажувальної форми;
    - fGeneral.cs – файл коду форми головного меню;
    - fGeneral.Designer.cs – дизайнер головного меню;
    - fBank.cs – файл коду де містяться таблиці «Відділення» і «Працівники»;
    - fBank.Designer.cs – дизайнер форми банку;
    - fClients.cs – файл коду з таблицями «Кліенти» і «Вклади»;
    - fClients.Designer.cs – дизайнер форми кліентів;
    - ReportSuma.cs – файл коду де міститься звіт фінансів кліентів;
    - ReportSuma.Designer.cs – дизайнер форми звіту;

ПЗ працює в середовищі операційної системи Windows, запуск здійснюється з виконавчим файлом Bank.exe. ПЗ є Windows Form додатком на мові C#.

## Функціональна схема взаємодії модулів ПЗ

Основна програма викликає завантажувальну форму і викликає головне меню. Далі залежно від натиснутої кнопки користувачем переходить на форму банку, кліентів або виконується вихід. В кліентах є перехід на форму звіту.

Функціональна схема взаємодії модулів ПЗ показана на рисунку 2.1.

Стартова сторінка є завантажувальною формую програми і завантажує головне меню.

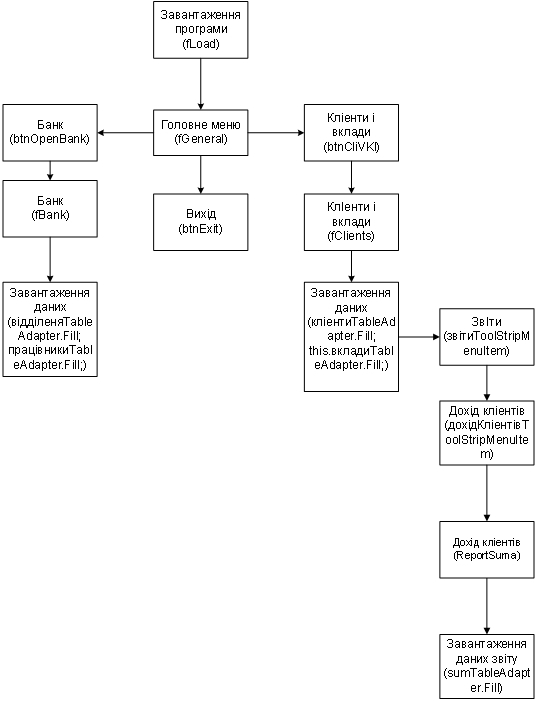


Рисунок 2.1 Функціональна схема взаємодії модулів ПЗ

## Опис окремого модуля

Модуль містить функції і процедури для прив’язки до модуля завантажувального вікна. І використовується як головне меню.

Таблиця 2.1

Структура fGeneral

| **Підпрограма** | **Функціональне**  **призначення** | **Алгоритм функціонування** |
| --- | --- | --- |
| fGeneral() | Клас використовується для ініціалізації компонентів в цьому випадку створення потоку. | Ініціалізація потоку та прив’язка його до певного класу. |
| Thread t = new Thread new ThreadStart() | Використовується для створення потоку. | Створює потік t прив’язує його до класу. |
| t.Start(); | Виконує запуск потоку. | Починає запуск потоку. |
| Thread.Sleep(); | Для затримки потоку на певний час. | Затримує потік на певний час. |
| t.Abort(); | Для закінчення потоку. | Закінчує потік. |
| StartForm() | Клас для перенаправлення на fLoad і прив’язки до потоку. | Робить стартовою формою fLoad. |
| btnExit\_Click() | Вихід з програми | При натисканні виходить з додатку |
| If (MessageBox.Show()) | Виводити повідомлення про вихід. | При натисканні кнопки виходу виводить повідомлення ці потрібно виходити з додатку. |

Опис модулів які містять функції і процедури для перегляду БД і роботи з даними в ній.

Таблиця 2.2

Структура fBank

| **Підпрограма** | **Функціональне**  **призначення** | **Алгоритм функціонування** |
| --- | --- | --- |
| fBank\_Load() | Клас використовується для виводу даних з БД. | Виводить дані з таблиць БД за допомогою метода \*TableAdapter.Fill() |
| btPrevOtd\_Click() | Кнопка для переходу на попередній запис | При натисканні кнопки переходить на попередній запис в табличці Відділеня |
| \*BindingSource.MovePrevious(); | Метод для переходу на попередній запис. | Реалізує перехід на попередній запис |
| btNextOtd\_Click() | Кнопка для переходу на наступний запис | При натисканні кнопки переходить на наступний запис в табличці Відділеня. |
| \*BindingSource.MoveNext(); | Метод для переходу на наступний запис. | Реалізує перехід на наступний запис |
| btAddOtd\_Click() | Кнопка для додавання запису | При натисканні додає запис в табличці Відділеня. |
| \*BindingSource.AddNew(); | Метод для додавання запису | Реалізує додавання нового запису |
| btUpdateOtd\_Click() | Кнопка для оновлення та збереження нових записів | При натискаканні оновлює або зберігає нові записи |
| \*BindingSource.EndEdit(); | Метод для зберігання записів | При натисканні зберігає нові або змінені записи |
| \*TableAdapter.Update(); | Реалізує запит Update | Реалізація SQL запиту Update |
| MessageBox.Show(); | Вивід повідомлень | Виводить певне повідомлення |
| btDelOtd\_Click() | Кнопка видалення записів | При натисканні видаляє рядок з записами в табличці Відділеня |
| \*BindingSource.RemoveCurrent() | Видалення рядків записів | Звертається до рядка і видаляє запис |
| btnPrevWork\_Click() | Кнопка для переходу на попередній запис в Працівниках | При натисканні кнопки переходить на попередній запис в табличці Працівникі |
| btnAddWork\_Click() | Кнопка для додавання запису в Працівниках | При натисканні кнопки переходить на попередній запис в табличці Працівникі |
| btnNextWork\_Click() | Кнопка для переходу на наступний запис в Працівниках | При натисканні кнопки додає новий запис в табличці Працівникі |
| btnUpdateWork\_Click() | Кнопка для оновлення та збереження нових записів в Працівниках | При натискаканні оновлює або зберігає нові записи |
| btnDelWork\_Click() | Видалення рядків записів в Працівниках | При натисканні кнопки додає новий запис в табличці Працівникі |
| RefreshOtd\_Click() | Скидає фільтр пошуку в Відділені | При натисканні скидає фільтр пошуку в Відділені |
| btnSearchOtd\_Click() | Кнопка для пошуку в Відділені | При натисканні виконує пошук в Відділені |
| btnRefreshWorker\_Click() | Скидає фільтр пошуку в Працівниках | При натисканні скидає фільтр пошуку в Працівниках |
| btnSearchWorker\_Click() | Кнопка для пошуку в Працівниках | При натисканні виконує пошук в Працівниках |
| \*BindingSource.Filter | Метод для реалізацій пошуку | В методі реалізується SQL запит LIKE |

Таблиця 2.3

Структура fСlients

| **Підпрограма** | **Функціональне**  **призначення** | **Алгоритм функціонування** |
| --- | --- | --- |
| fClients\_Load() | Клас використовується для виводу даних з БД. | Виводить дані з таблиць БД за допомогою метода \*TableAdapter.Fill() |
| btPrevCli\_Click() | Кнопка для переходу на попередній запис | При натисканні кнопки переходить на попередній запис в табличці Кліенти |
| \*BindingSource.MovePrevious(); | Метод для переходу на попередній запис. | Реалізує перехід на попередній запис |
| btNextCli\_Click() | Кнопка для переходу на наступний запис | При натисканні кнопки переходить на наступний запис в табличці Кліенти |
| \*BindingSource.MoveNext(); | Метод для переходу на наступний запис. | Реалізує перехід на наступний запис |
| btAddCli\_Click() | Кнопка для додавання запису | При натисканні додає запис в табличці Кліенти. |
| \*BindingSource.AddNew(); | Метод для додавання запису | Реалізує додавання нового запису |
| btUpdateCli\_Click() | Кнопка для оновлення та збереження нових записів в Кліентах | При натискаканні оновлює або зберігає нові записи в Кліентах |
| \*BindingSource.EndEdit(); | Метод для зберігання записів | При натисканні зберігає нові або змінені записи |
| \*TableAdapter.Update(); | Реалізує запит Update | Реалізація SQL запиту Update |
| MessageBox.Show(); | Вивід повідомлень | Виводить певне повідомлення |
| btDelCi\_Click() | Кнопка видалення записів | При натисканні видаляє рядок з записами в табличці Кліенти |
| \*BindingSource.RemoveCurrent() | Видалення рядків записів | Звертається до рядка і видаляє запис |
| btPrevVkl\_Click() | Кнопка для переходу на попередній запис в Вкладах | При натисканні кнопки переходить на попередній запис в табличці Вклади |
| btAddVkl\_Click | Кнопка для додавання запису в Вкладах | При натисканні кнопки переходить на попередній запис в табличці Вклади |
| btNextVkl\_Click | Кнопка для переходу на наступний запис в Вкладах | При натисканні кнопки додає новий запис в табличці Вклади |
| btUpdateVkl\_Click | Кнопка для оновлення та збереження нових записів в Вкладах | При натискаканні оновлює або зберігає нові записи в Вкладах |
| btDelVkl\_Click | Видалення рядків записів в Вкладах | При натисканні кнопки додає новий запис в табличці Вклади |
| RefreshCli\_Click | Скидає фільтр пошуку в Кліентах | При натисканні скидає фільтр пошуку в Кліентах |
| btnSearhCli\_Click | Кнопка для пошуку в Кліентах | При натисканні виконує пошук в Кліентах |
| btRefreshVkl\_Click | Скидає фільтр пошуку в Вкладах | При натисканні скидає фільтр пошуку в Вкладах |
| btSearchVkl\_Click | Кнопка для пошуку в Вкладах | При натисканні виконує пошук в Вкладах |
| \*BindingSource.Filter | Метод для реалізацій пошуку | В методі реалізується SQL запит LIKE |

Таблиця 2.4

Структура ReportSuma

| **Підпрограма** | **Функціональне**  **призначення** | **Алгоритм функціонування** |
| --- | --- | --- |
| ReportSuma\_Load() | Клас використовується для виводу даних з БД. | Виводить дані з запиту БД за допомогою метода sumTableAdapter.Fill() |

## Опис алгоритму роботи програми

1. Запуск програми здійснюються за допомогою виконавчого файлу Bank.exe.

На початку програми виконується завантажувальне вікно.

«*Головне меню*». Відображає початкове меню, з набором клавіш «Розпочати роботу з БД», «Вихід». Для виконання необідної функції потрібно натиснути відповідну кнопку на головному меню. При натисканні клавіші «Вихід» програма завершує роботу .

2. «*Банк*». Головне меню містить команди для роботи з файлом бази даних.

Попередній. Повертає на наступний запис.

Додати*.* Створює новий рядок для вводу даних з клавіатури.

Наступний. Переходить на наступний запис.

Зберегти. Зберігає дані в рядку після їх змін і додавання в рядок.

Видалити. Видалити рядок з таблиці.

Оновити. Скидає фільтр.

Пошук. Здійснює пошук по існуючій базі даних.

5. Вихід. Завершує роботу програми.

## Використані стандартні бібліотеки

Системні бібліотеки:

*using System*- Простір імен System містить фундаментальні і базові класи, які визначають поширені значущі і посилальні типи даних, події і їх обробники, інтерфейси, атрибути і обробку винятків.

*System.Data* - Простір імен System.Data містить класи для доступу до даних з різних джерел і для управління цими даними. Простір імен верхнього рівня і кілька дочірніх просторів імен утворюють архітектуру ADO.NET і постачальники даних ADO.NET. Наприклад, доступні постачальники для SQL Server, Oracle, ODBC і OleDB. Інші дочірні простору імен містять класи, використовувані моделлю EDM ADO.NET і службами даних WCF.

*System.Drawing* - Простір імен System.Drawing містить типи, що підтримують базові графічні функції GDI +. Дочірні простору імен підтримують більш складні функції двомірної і векторної графіки, додаткові функції обробки зображень, а також служби, пов'язані з друком і типографією. Дочірнє простір імен також містить типи, які розширюють логічні і графічні можливості призначеного для користувача інтерфейсу під час розробки.

*System.Text* - Простір імен System.Text містить типи для роботи з кодуваннями символів і для управління рядками. Дочірній простір імен дозволяє обробляти текст з використанням регулярних виразів.

*System.ComponentModel* - Попередньо представляє базову реалізацію інтерфейсу IComponent і робить можливим спільне використання об'єктів різними додатками.

*System.Threading.Tasks* - Передоставляє типи, які узагальнюють роботу за написанням паралельного та асинхронного коду. Основні типи: Task, що представляє асинхронну операцію, яку можна очікувати і змінити, і Task <TResult>, що представляє собою завдання, яке може визначити значення. Клас TaskFactory пропонує статистичні методи для створення завдань, а клас TaskScheduler забезпечує інфраструктуру планування потоків за умовою.

*System.Threading* - Створює та контролює потік, задає пріоритет і повертає статус.

# Реалізація та опис програмного засобу

## Опис середовища розробки

### Visual Studio 2019

Програмний продукт розроблено в середовищі Visual Studio 2010.

Visual Studio 2019 - інтегроване середовище розробки на C#.

Можливості Visual Studio 2019:

Ви можете створювати:

* додатки та ігри, які виконуються на платформі Windows.
* веб-сайти і веб-служби на основі ASP.NET, JQuery, AngularJS і інших популярних платформ;
* додатки для самих різних платформ і пристроїв, включаючи, але не обмежуючись: Office, Sharepoint, Hololens, Kinect і "Інтернету речей";
* ігри і графічні додатки для різних пристроїв Windows, включаючи Xbox, з підтримкою DirectX.

За замовчуванням Visual Studio забезпечує підтримку C #, C і C ++, JavaScript, F # і Visual Basic. Visual Studio добре працює і інтегрується зі сторонніми додатками, наприклад Unity і Apache Cordova, за допомогою розширень Набір засобів Visual Studio для Unity і інструментів Visual Studio для Apache Cordova відповідно. Ви також можете самостійно розширити

Visual Studio, створивши власні інструменти для виконання спеціалізованих завдань.

Проект, що відкрився представляє з себе базовий віконний додаток на C#.

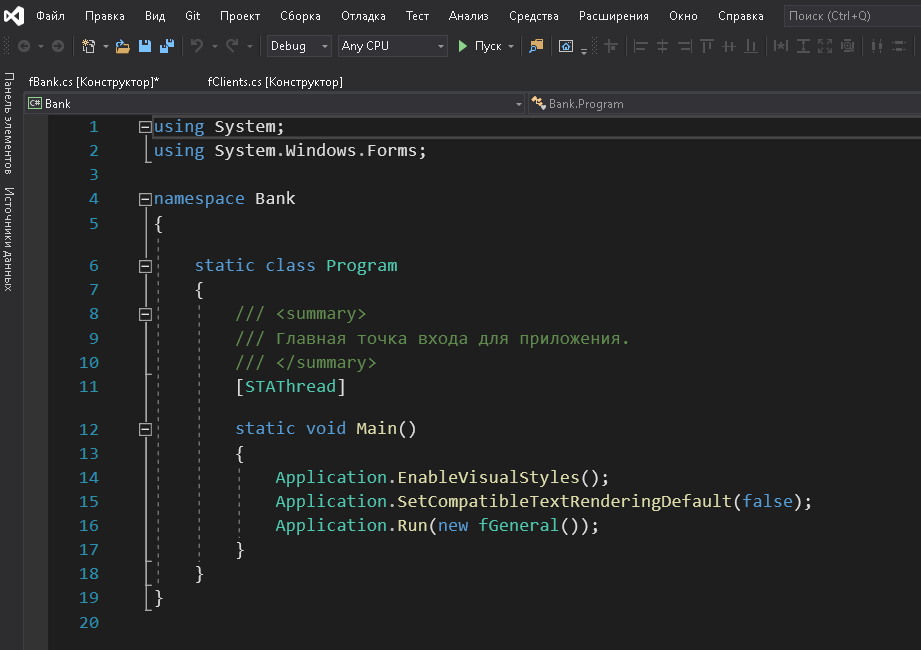


Рисунок 3.1 Вікно середовища MicrosoftVisual Studio

Проект, що відкрився представляє з себе базовий віконний додаток на C#.

Знову скористаємося панеллю інструментів і запустимо проект - тиснемо кнопку "Запуск". Далі запускається програма та можна починати роботу з нею.

## 3.2 Опис основних модулів

### Реалізація головного меню програми

Блок-схема алгоритму, який реалізує головне меню наведено на рисунку.

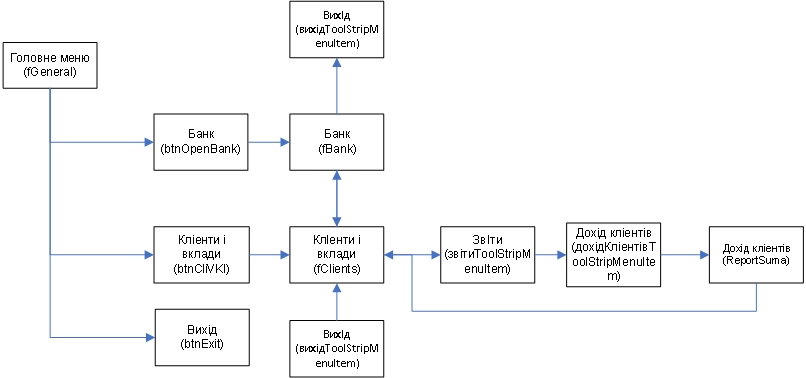


Рисунок 3.1 Блок-схема алгоритму головного меню

## Опис розробленого програмного забезпечення

Після запуску програми на екрані монітору з’являється завантажується стартове вікно

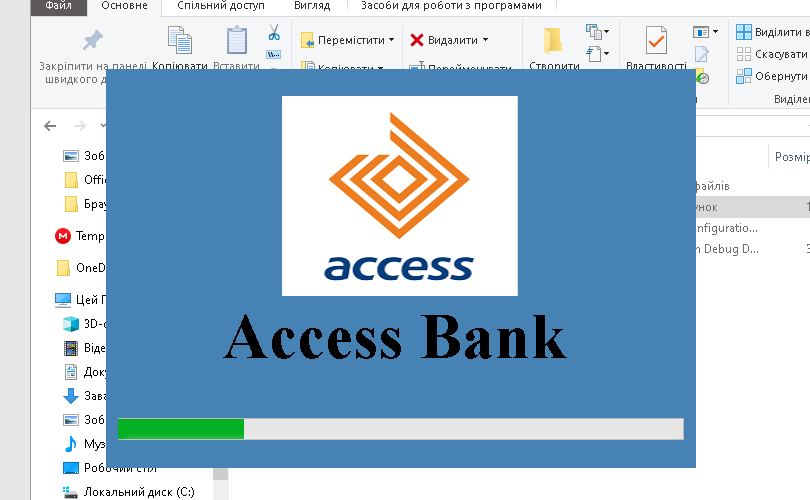


Рисунок 3.2 Стартове вікно програми

Після завантаження програми з’являється головне меню на якому знаходяться три кнопки дві для переходу до даних і роботи з ними. Також клавіша для виходу.

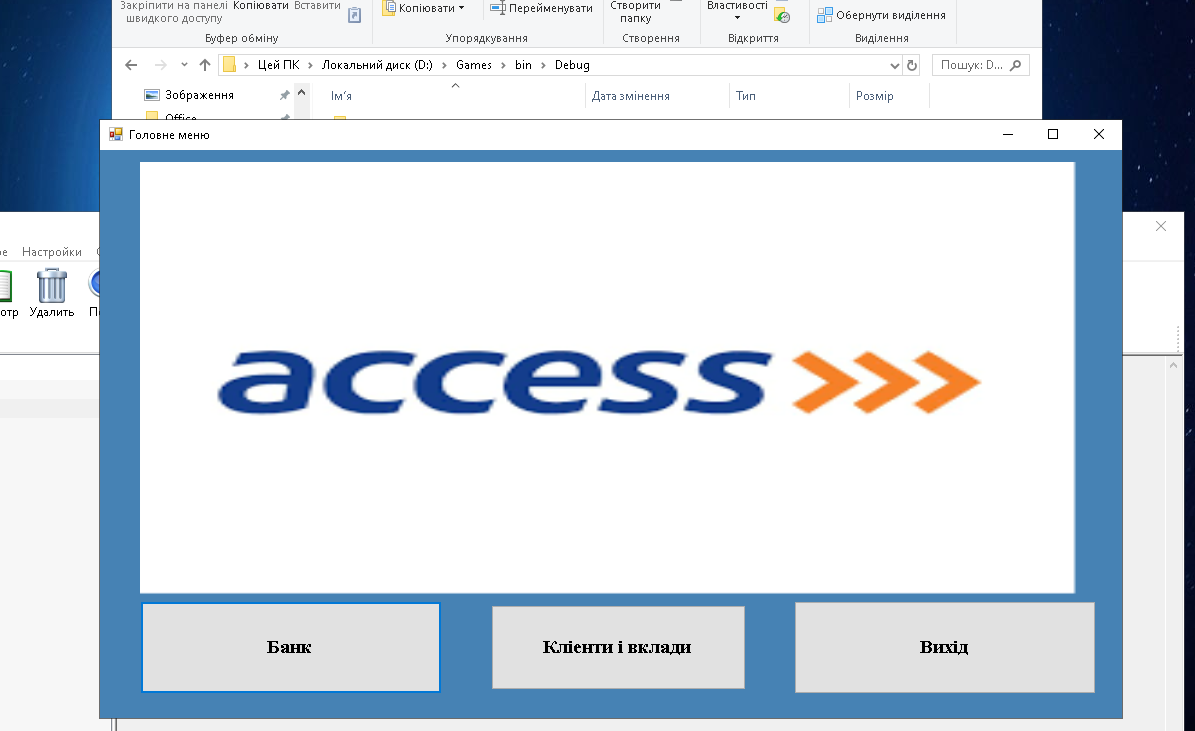


Рисунок 3.3 Вікно головного меню.

Натиснувши клавішу Банк потрапим на форму з відділенями і працівниками.

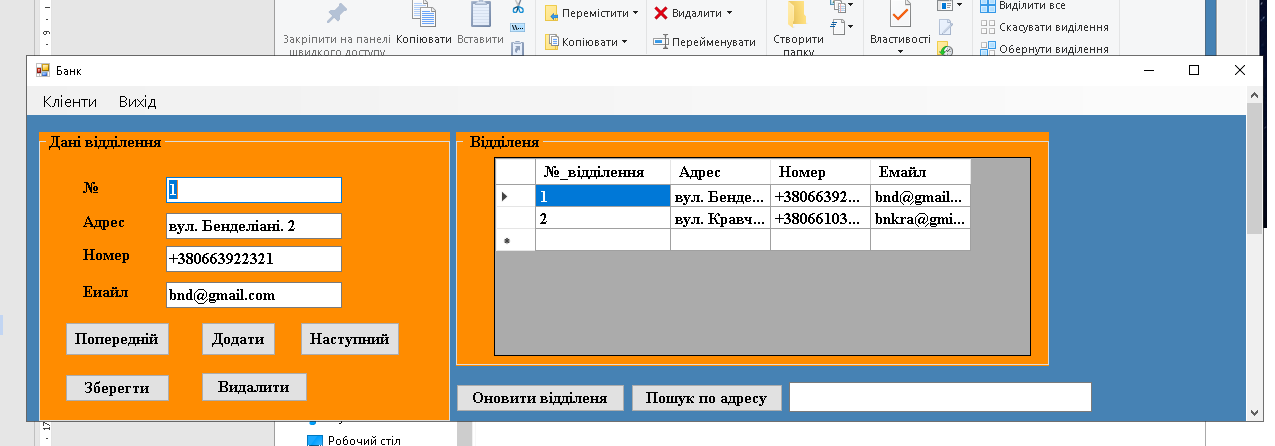


Рисунок 3.4 Вікно Банку з таблицею Відділеня.

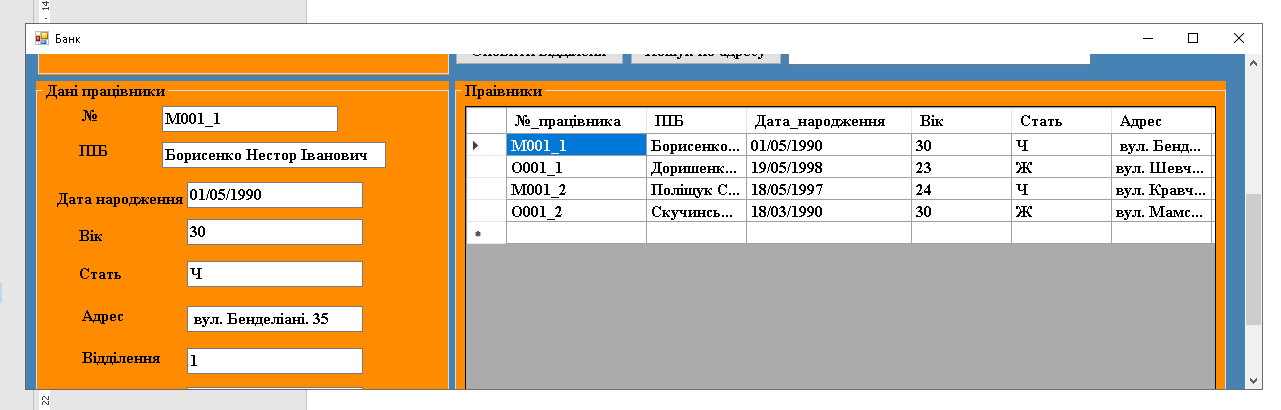


Рисунок 3.5 Вікно Банку з таблицею працівників

Натиснувши пункт меню Кліенти перейдем на їх форму там знаходяться таблиці з даними кліентів і вкладів.

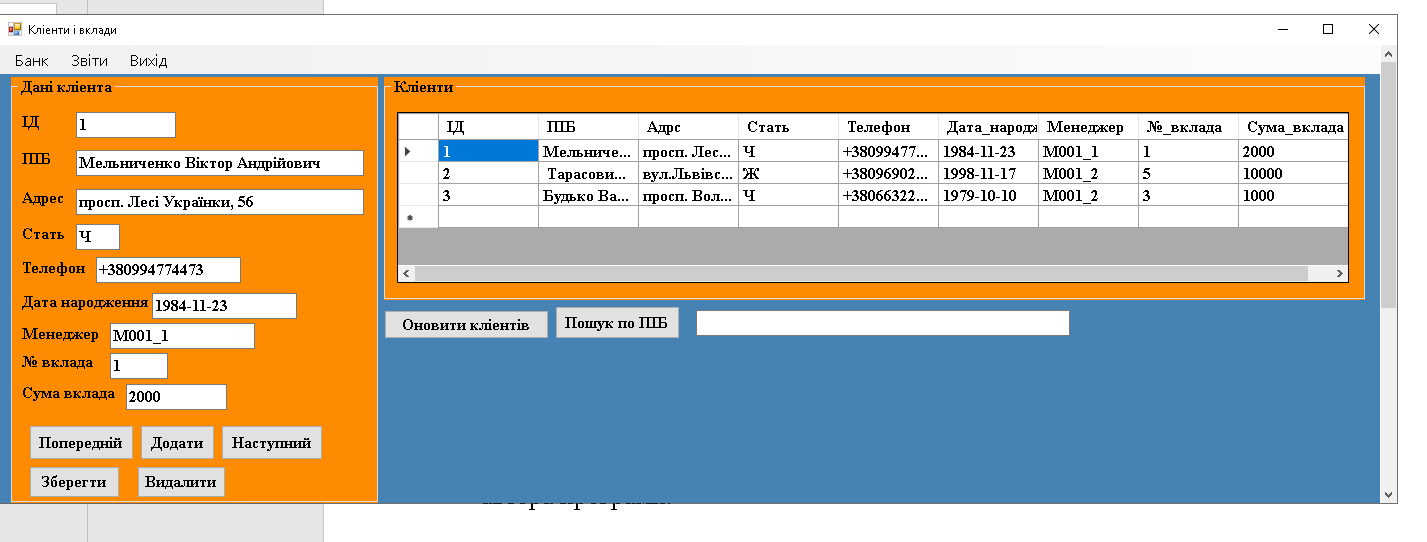


Рисунок 3.6 Вікно Кліентів з таблицею кліентів

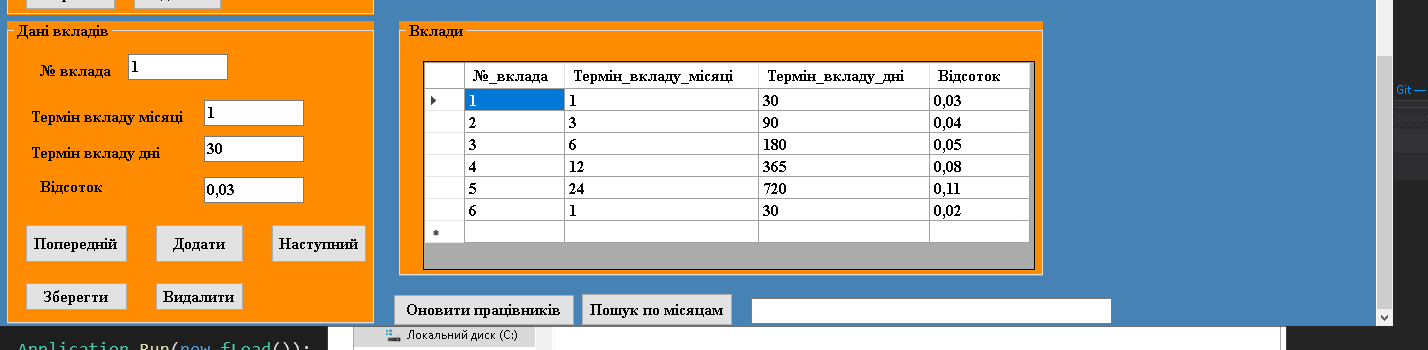


Рисунок 3.7 Вікно Кліентів з таблицею вкладів

І остання форма на яку ми можемо перейти натиснувши пункт меню Звіти – це форма зі звітом по фінансовому становищу кліентів.

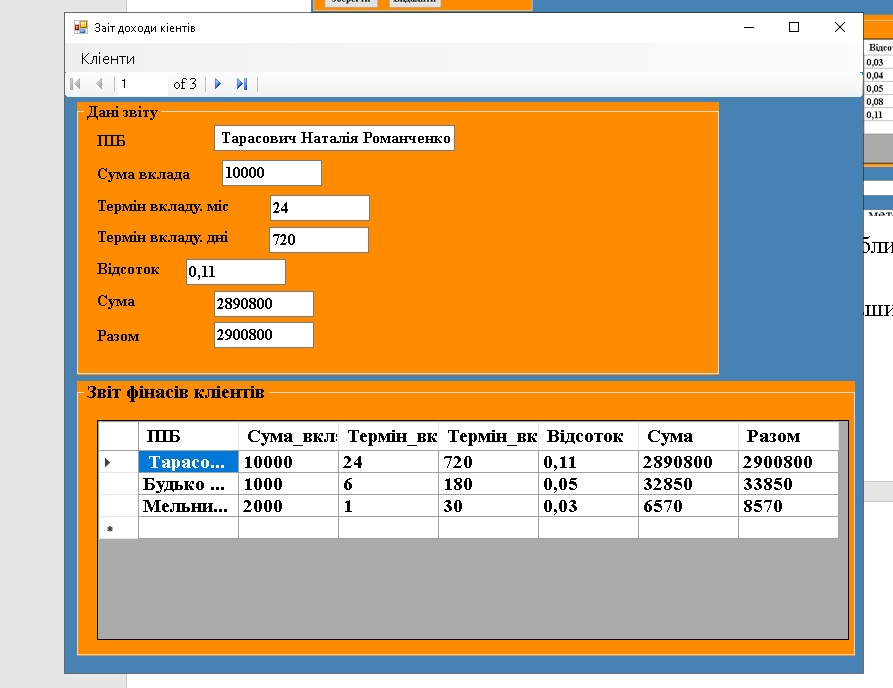


Рисунок 3.8 Вікно звіту по фінансам

Натиснувши пункт меню Кліенти вернемось до кліентів і натиснувши Вихід вийдемо з програми.

Висновки

В даній курсовій роботі розроблено і описано систему обліку «Облік відділень банку». Для розробки програми вибрано мову C# (середовище Visual Studio). Програма розроблена із застосуванням методики структурного та об’єктно орієнтованого програмування. Вибір алгоритмічної мови C# для реалізації поставленої задачі повністю виправдало себе. В процесі відладки програми зроблено висновок про можливість її вдосконалення - реалізацію деяких підзадач можна було б оформити у вигляді незалежних програмних модулів, які б можна було надалі використати при розробці інших подібних задач економічного характеру.

Програма відкомпільована та відлагоджена з використанням набору тестових даних. Результат тестування та прогнозовані вихідні дані повністю співпали, тому можна зробити висновок про можливість використання розробленої програми на практиці.

У ході розробки програмного продукту були практично вивчені такі поняття, як інкапсуляція, поліморфізм, клас, члени і методи класу.Також отриманий досвід роботи у програмному середовищі Visual Studio були вивчені нові компоненти та програмно реалізовані специфічні події пов’язані з ними.