ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Циклова комісія інженерія програмного забезпечення

**Курсова робота**

з дисципліни “Об’єктно-орієнтоване програмування”

**на тему:** Автоматизоване робоче місце менеджера банку

Студента III курсу групи П-33

галузі знань 12

спеціальності 121

Цехняк Любомир Васильович

(прізвище та ініціали)

Керівник: Красничук В.В.

(прізвище та ініціали)

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

Члени комісії: Ляшеник А. В.

(підпис)            (прізвище та ініціали)

Смиковчук Т. В.

(підпис)          (прізвище та ініціали)

Красничук В. В.

(підпис)          (прізвище та ініціали)

Коломия 2022

**ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ**

**НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Відділення: ***денне відділення комп’ютерних технологій***

Циклова комісія ***інженерії програмного забезпечення***

Освітньо-кваліфікаційний рівень: ***молодший спеціаліст***

Галузь знань: ***12 "Інформаційні технології"***

Спеціальність: ***121 "Інженерія програмного забезпечення"***

**ЗАТВЕРДЖУЮ**

Голова циклової комісії інженерії програмного забезпечення

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Т.Т.Дуб**

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 року

**З  А  В  Д  А  Н  Н  Я**

**НА КУРСОВУ  РОБОТУ**

Студенту Цехняку Любомиру Васильовичу

1. Тема роботи: Автоматизоване робоче місце менеджера банку.

Керівник роботи: Красничук Вікторія Вікторівна.

2. Термін виконання студентом роботи: 25 червня 2022 року.

3. Вихідні дані до роботи: мова програмування С++.

4. Зміст розрахунково-пояснювальної записки:

Аналіз технічного завдання;

Обґрунтування алгоритму й структури програми;

Розроблення програми;

Тестування програми і результати її виконання.

Дата видачі завдання: 23.02.2022

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Найменування етапів  курсової роботи | Термін виконання етапів роботи | Примітки |
| 1. | Постановка завдання.  Аналіз специфікації вимог.  Визначення структури даних. | 23.02.2022  28.02.2022  11.03.2022 |  |
| 2. | Визначення структури програмних модулів. Розробка алгоритму, визначення форми представлення даних, структури програми. | 20.03.2022  08.04.2022 |  |
| 3. | Програмування.  Розробка документації. | 05.05.2022  12.05.2022 |  |
| 4. | Тестування і верифікація програми. | 17.05.2022 |  |
| 5. | Оформлення пояснювальної записки. | 19.05.2022 |  |
| 6. | Виправлення недоліків та доробка програми.  Представлення роботи до попереднього розгляду. | 19.05.2022 |  |
| 7. | Захист роботи. | 20.06.2022 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Цехняк Любомир Васильович

(підпис студента)                (прізвище, ім’я, по батькові студента)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Красничук Вікторія Вікторівна

(підпис викладача)                (прізвище, ім’я, по батькові викладача)

**ЗМІСТ**

Вступ.........................................................................................................4

1 Аналіз технічного завдання.................................................................5

2 Обґрунтування алгоритму й структури програми.............................6

3 Процес розробки програми..................................................................8

3.1 Розробка системи класів.........................................................8

3.2 Розробка методів....................................................................10

3.3 Створення об'єктів і розробка головної програми..............13

3.4 Опис файлів даних………………………….........................14

4 Тестування програми і результати її виконання..............................16

Висновки.................................................................................................21

Перелік використаної літератури.........................................................22

Додаток Вихідні файли програми з описами методів класів.............23

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

417 КР 121.01.04 ПЗ

Розроб.

*Цехняк Л. В.*

Перевір.

Красничук В.В.

Реценз.

Н. Контр.

Затверд.

*Автоматизоване робоче місце менеджера банку*

Літ.

Аркушів

23

КПК НУ «ЛП»

**ВСТУП**

Метою цієї курсової роботи є створити програмне забезпечення, яке допоможе менеджерам банку в роботі із бухгалтерськими даними, та маніпуляціями над ними.

Важко уявити собі більш сприятливий ґрунт для запровадження комп’ютерних технологій, ніж банківська діяльність. Майже всі завдання, які виникають у процесі роботи банку, піддаються автоматизації. Швидка і безперебійна обробка значних потоків інформації є одним із головних завдань будь-якої великої фінансової організації. Тому необхідна наявність обчислювальної мережі, яка дозволяє обробляти інформаційні потоки, що все збільшуються і збільшуються. Крім того, банки мають достатні фінансові можливості для використання найсучаснішої техніки.

Інтерес до розвитку комп’ютеризованих банківських систем визначається, головним чином, стратегічними інтересами. Як показує практика, інвестиції в такі проекти починають приносити прибуток лише через певний період часу, необхідний для навчання персоналу та адаптації системи до конкретних умов. Вкладаючи кошти в програмне забезпечення, комп’ютерне і телекомунікаційне обладнання та створення бази для переходу до нових обчислювальних платформ, банки, в першу чергу, прагнуть до здешевлення і прискорення рутинної роботи та перемоги в конкурентній боротьбі.

Нові технології допомагають банкам, інвестиційним фірмам та страховим компаніям змінити відносини з клієнтами і знайти нові засоби для отримання прибутку. Аналітики сходяться на думці, що нові технології найбільш активно впроваджують інвестиційні фірми, потім йдуть банки, а останніми їх приймають на озброєння страхові компанії.

**1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ**

Будь-яка система управління здійснює збір та передачу інформації про стан об’єкта, обробку отриманої інформації по заздалегідь сформульованих алгоритмах управління i видачу результатів обробки у вигляді інформації або управлінських дій на об’єкт. Тому застосування автоматизованих робочих місць (АРМ) – прогресивний напрям використання обчислювальної техніки для управлінських працівників.

Даний програмний продукт повинен бути розроблений з використанням основних концепцій об’єктно-орієнтованого підходу (ООП) – класи, успадкування, перевантаження методів і операцій, шаблони методів і класів.

В процесі розробки програми було проведено аналіз відповідної предметної області та дослідження проблем ефективності роботи. На основі цих досліджень були сформовані основні поняття і правила за якими має функціонувати даний програмний продукт.

Було поставлене завдання створити програмне забезпечення, що забезпечить взаємозв’язок між користувачем та даними, та надасть групу можливостей :

* Перегляд даних
* Внесення даних.
* Пошук даних.
* Сортування даних.
* Видалення даних.
* Оформлення кредитів.
* Збір інвестицій в банк.

**2 ОБҐРУНТУВАННЯ АЛГОРИТМУ Й СТРУКТУРИ ПРОГРАМИ**

Дана програма була розроблена в середовищі CodeBlocks і написана на мові високого рівня C++.

В даній програмі будуть використовуватись такі поняття як: клас, шаблони класів, інкапсуляція, STL-бібліотека, Iterator та деякі інші.

**Клас** – це спеціальна конструкція, яка використовується для групування пов'язаних змінних та функцій. При цьому, згідно з термінологією ООП, глобальні змінні класу (члени-змінні) називаються полями даних, а члени-функції називають методами класу. Створений та ініціалізований екземпляр класу називають об'єктом класу. На основі одного класу можна створити безліч об'єктів, що відрізнятимуться один від одного своїм станом (значеннями полів).

**Інкапсуляція** (приховування інформації) – об’єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів (методів). Доступ до стану об'єкта напряму заборонено, і ззовні з ним можна взаємодіяти виключно через заданий інтерфейс (відкриті поля та методи), що дозволяє знизити зв'язність. Таким чином контролюються звернення до полів класів та їхня правильна ініціалізація, усуваються можливі помилки пов'язані з неправильним викликом методу.

**Шаблони** (англ. template) — засіб мови C++, який призначений для кодування узагальнених алгоритмів, без прив'язки до деяких параметрів: типу даних, розміру буфера та стандартного значення. В C++ можливе створення шаблону функції і шаблону класу. Хоча шаблони надають коротку форму запису ділянки коду, насправді їх використання не скорочує виконуваний код, тому що для кожного набору параметрів компілятор створює окремий екземпляр функції або класу.

**Бібліотека стандартних шаблонів (STL)** ( англ. Standard Template Library ) - набір узгоджених узагальнених алгоритмів , контейнерів , засобів доступу до їх вмісту і різних допоміжних функцій в C ++ .

Бібліотека стандартних шаблонів до включення в стандарт C ++ була сторонньої розробкою, спочатку - фірми HP , а потім SGI . Стандарт мови не називає її «STL», так як ця бібліотека стала невід'ємною частиною мови, проте багато людей до цих пір використовують цю назву, щоб відрізняти її від іншої частини стандартної бібліотеки (потоки введення-виведення ( iostream ), підрозділ Сі та ін. ).

**Итератор** (від англ. Iterator - Нумератор) - інтерфейс , що надає доступ до елементів колекції ( масиву або контейнера ) і навігацію по ним. У різних системах ітератори можуть мати різні загальноприйняті назви. У термінах систем управління базами даних ітератори називаються курсором. У найпростішому випадку ітератором в низькорівневих мовах є покажчик.

**Контейнер list STL**. Двусвязний список, елементи якого зберігаються в довільних комірках пам'яті, на відміну від контейнера vector, де елементи зберігаються в безперервній області пам'яті. Пошук перебором повільніше, ніж у вектора через більшого часу доступу до елементу. Доступ за індексом за{\ Displaystyle O (n)}O (n). У будь-якому місці контейнера вставка і видалення виробляються дуже швидко - за{\ Displaystyle O (1)}O (1).

**3 ПРОЦЕС РОЗРОБКИ ПРОГРАМИ**

3.1 Розробка діаграми класів

Діаграма класів - це тип діаграми та частина уніфікованої мови моделювання (UML), яка визначає та забезпечує огляд та структуру системи з точки зору класів, атрибутів та методів та зв’язків між різними класами.

Він використовується для ілюстрації та створення функціональної схеми системних класів і служить ресурсом розробки системи в межах життєвого циклу розробки програмного забезпечення.

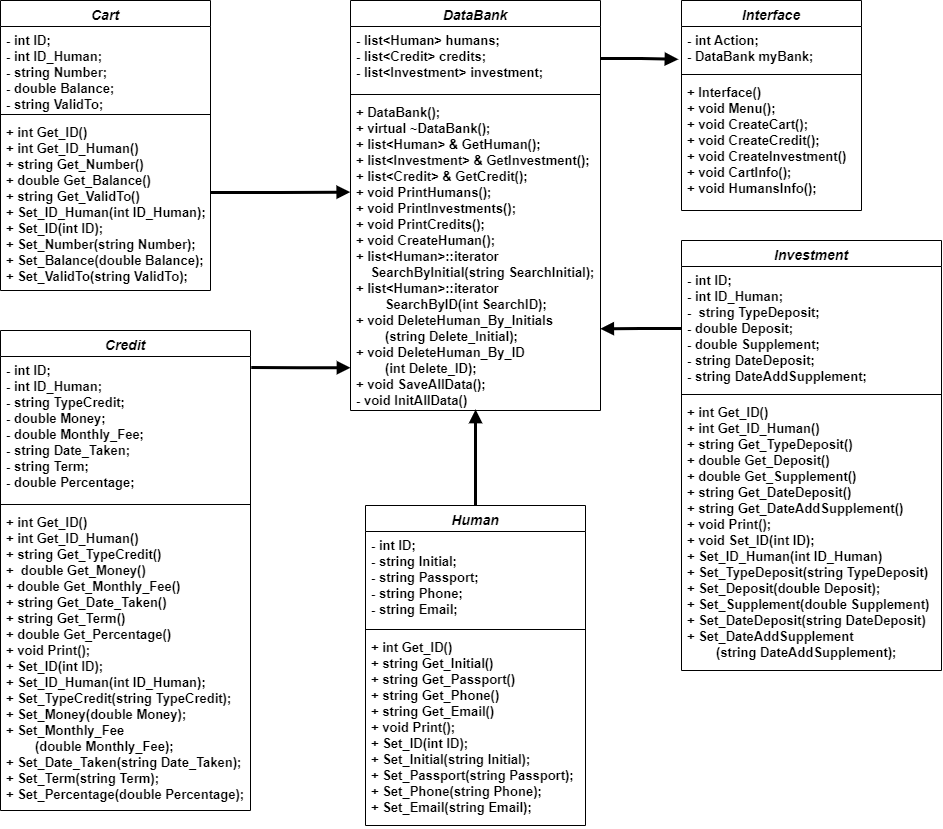


Рис. 1.1. Діаграма класів ПЗ.

Дане програмне забеспечення складається з п’яти класів:

Дані класи описують сутності кредитів, відвідувачів, інвестицій, клас списків та методів для роботи з ними та клас інтерфейс.

* Human – Описує сутність відвідувача банку, який хоче отримати кредит або внести свої кошти, як інветиція.
* Credit – Описує сутність кредиту, та всю необхідну інформацію для відслідковування статусу.
* Cart – Описує сутність кредитної карти.
* Investment – Описує сутність інвестицій та вміщує детальну інформацію.
* DataBank – Описує структуру даних банку, вміщуючи в собі списки попередніх сутностей та надає методи для роботи із ними.
* Interface – Забезпечує взаємодію між користувачем та даними.

**3.2 Розробка методів**

Метод класу **«DataBank» ::SaveAllData()** збарігає всі наявні дані програми в текстову систему бази даних всіх чотирьох класів, які описують сутністі ПО.

void DataBank::SaveAllData()

{

fstream file; bool next = false;

Human human; Credit credit; Investment invest; Cart \_cart;

file.open("File\\Carts.txt", fstream::out);

if(!file.is\_open())

cout<<"Файл File\\Carts.txt не знайдено!"<<endl;

else{ //Людина

for(auto it = cart.begin(); it!=cart.end(); it++){

\_cart = \*it;

if(next == true){file <<endl;}

file << \_cart.Get\_ID();

file <<endl << \_cart.Get\_ID\_Human();

file <<endl << \_cart.Get\_Number();

file <<endl << \_cart.Get\_Balance();

file <<endl << \_cart.Get\_ValidTo();

next = true;

}

}

next = false;

file.close();

. . . . .

Метод класу **«SportswearList» InitAllData()** який при старті програми зчитує з файлової системи всі дані

void DataBank::InitAllData()

{

string \_string;

double \_double = 0;

int \_int = 0;

fstream file;

file.open("File\\Humans.txt", fstream::in);

if(!file.is\_open())

cout<<"Файл File\\Humans.txt не знайдено!"<<std::endl;

else{

Human human;

while(!file.eof()){

file >> \_int;

human.Set\_ID(\_int);

file.ignore();

getline(file, \_string);

human.Set\_Initial(\_string);

getline(file, \_string);

human.Set\_Passport(\_string);

getline(file, \_string);

human.Set\_Phone(\_string);

getline(file, \_string);

human.Set\_Email(\_string);

humans.push\_back(human);

}

file.close();

}

. . . . . .

Метод класу **Credit** для воводу власних даних.

void Credit::Print()

{

cout

<<" Код - "<<ID<<endl

<<" Код людини - "<<ID\_Cart <<endl

<<" Тип кредиту - "<<TypeCredit <<endl

<<" Сума кредиту - "<<Money <<endl

<<" Місячна плата - "<<Monthly\_Fee <<endl

<<" Дата отримання - "<<Date\_Taken<<endl

<<" Термін сплати - "<<Term <<endl

<<" Відсоток - "<<Percentage <<endl<<endl;

}

Метод класу **DataBank PrintCredits()** який виводить всю інформацію про всі зареєстровані кредити.

void SportswearList::PrintListSportsItems()

{

if(sportsItems.empty())

{

std::cout<<" Список товарів порожній !!!"<<std::endl;

}

else{

for(auto it = sportsItems.begin(); it != sportsItems.end(); it++){

SportsItems item = \*it;

item.Print();

}

}

}

Метод класу **DataBank::CreateHuman()** для внесення нових даних про відвідувача банку.

void DataBank::CreateHuman()

{

cout<<" Реєстрація відвідуючого банку в систему"<<endl<<endl;

string \_string;

Human newHuman;

newHuman.Set\_ID(humans.size()+1);

cout<<" Введіть ініціали людини"<<endl<<" >";

getline(cin, \_string);

newHuman.Set\_Initial(\_string);

cout<<" Введіть паспортні дані"<<endl<<" >";

getline(cin, \_string);

newHuman.Set\_Passport(\_string);

cout<<" Введіть номер телефону"<<endl<<" >";

getline(cin, \_string);

newHuman.Set\_Phone(\_string);

cout<<" Введіть електронну пошту"<<endl<<" >";

getline(cin, \_string);

newHuman.Set\_Email(\_string);

humans.push\_back(newHuman);

}

Деструктор для вивільнення пам’яті списків.

DataBank::~DataBank()

{

humans.clear();

credits.clear();

investment.clear();

cart.clear();

}

**3.3 Створення об'єктів і розробка головної програми**

Одні з основних класів даного ПО мають наступні об’єкти класів:

Клас «Interface», який являється мостом між користувачем та данними програми має такий об’єкт класу **myBank**

Клас «DataBank», який містить в собі дані суб’єктів даного ПО містить наступні об’єкти:

* list<Human> humans - Список відвідувачів, які звертаються до банку за певними послугами.
* list<Cart> carts – Список об’єкту Cart, який описує кредитну карту для отримання відвідувачем.
* list<Investment> investment - Список інвестицій відвідувачів в банк. (Вклад власних грошей під відсотки.)
* list<Сredit > credits – Список кредитів банку. Відвідувач може запросити в банку певну суму коштів під відсотки.

**3.4 Опис файлів даних**

Файли даних – це структура текстових файлів, яка формує базу даних ПО та містить в собі всі дані сутностей.

Програма використовує чотири файла:

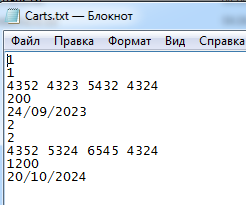


Рис. 2.1. Файл з даними кредитних карт.

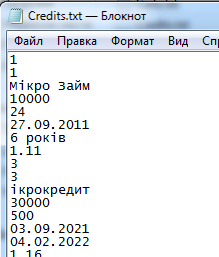


Рис. 2.2. Файл з даними взятих кредитів.

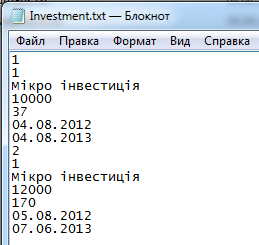


Рис. 2.3. Файл з даними інвестицій відвідувачів в банк.

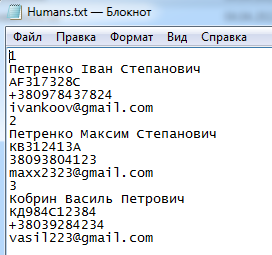


Рис. 2.4. Файл з даними відвідувачів.

**4 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ**

Після запуску програми відкривається консольне вікно «форма головного меню».

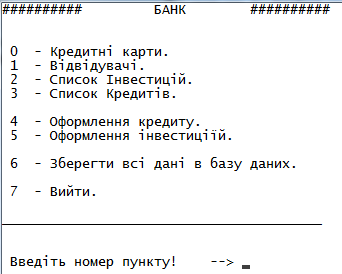


Рис. 4.1. Головне меню додатку банка.

Перший пункт містить підменю з шістьма пунктами для роботи з даними відвідувачів банку.

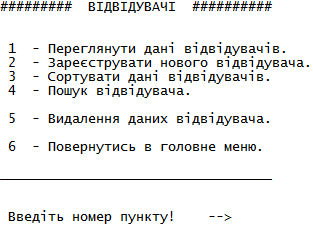


Рис. 4.2. Підменю відвідувачів.

Перший пункт виводить всю інформацію про відвідувачів банку.

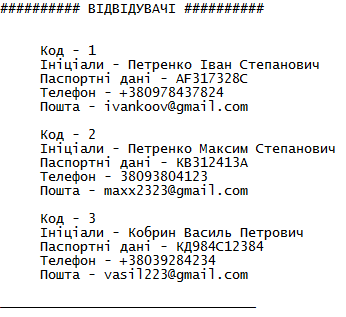


Рис. 4.3. Вивід відвідувачів.

Реєстрація нового відвідувача в системі банку для подальшої роботи з ним.

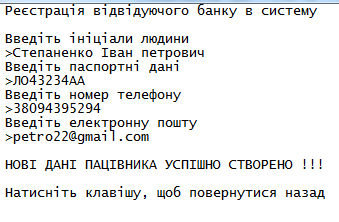


Рис. 4.4. Реєстрація відвідувачів.

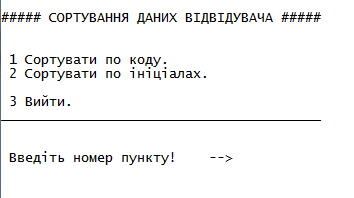


Рис. 4.5. Сортування даних відвідувача.

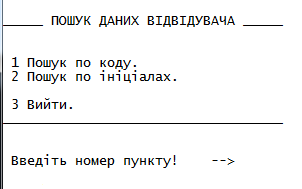


Рис. 4.6. Пошук даних відвідувачів.

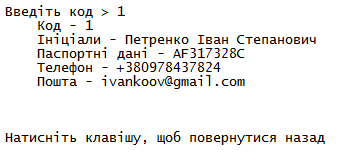


Рис. 4.7. Результат пошуку відвідувача по коду.

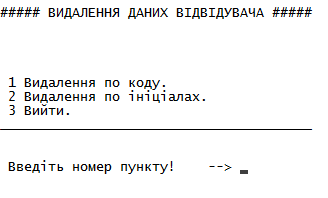


Рис. 4.8. Меню видалення даних відвідувача.

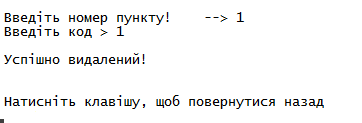


Рис. 4.8. Результат видалення по коду відвідувача.

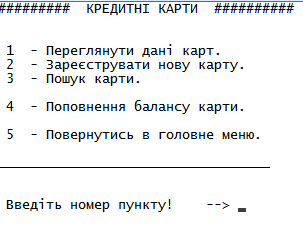


Рис. 4.9. Підменю кредитних карт.

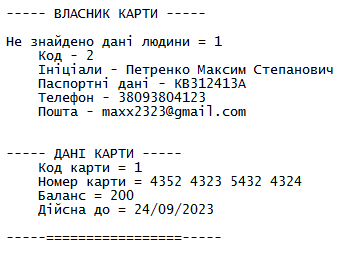


Рис. 4.10. Вивід інформації про кредитні карти.

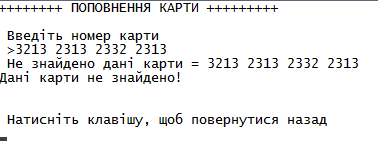


Рис. 4.11. Невдале поповнення рахунку.

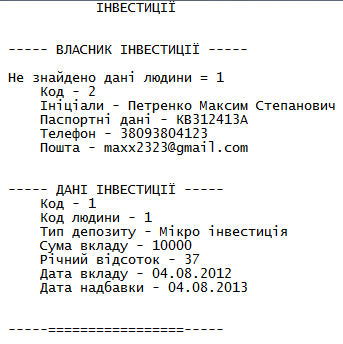


Рис. 4.11. Вивід даних інвестицій.

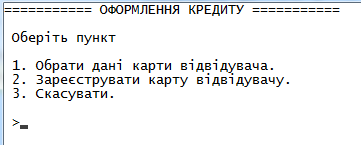


Рис. 4.12. Оформлення кредиту.

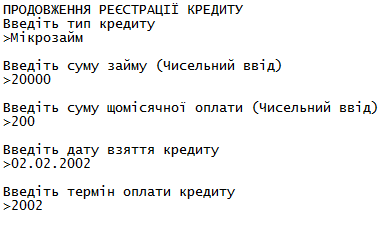


Рис. 4.13. Оформлення кредиту завершено.

**ВИСНОВКИ**

Під час виконання курсової роботи отримав навички у програмуванні мовою програмування с++, роботою з бібліотекою контейнерів STL та виводом в файл та з нього в програму. Реалізував програмний продукт за допомогою об’єктно-орієнтованого програмування.

Сам процес виконання та технології розробки програмного продукту мені дуже сподобались та дали мотивацію в подальшому розвитку в сфері програмування.

**ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Дейтл Н., Уимз Ч., Хедингтон М. Программирование на С++. Пер. С англ. - М.:ДМК, 2000. - 672 с.

2. Страуструп Б. Язык программирования С++. Специальное издание.-М., СПб.:”Издательство БИНОМ” – „Невский Диалект”, 2001г.-1099 с.

3. Бегун А.В. Технологія програмування: об`єктно-орієнтований підхід: Навч.- метод. Посібник для самост. вивч. дисц. -К.:КНЕУ, 2000.-200 с.

4. Р. Лафоре. Объектно-ориентированное программирование в С++. 4-е

5. Седжвик Р. Фундаментальные алгоритмы на С++. –К.: Изд-во ДиаСофт», 2001. -688с

издание. Издательство: Питер. Серия: Классика computer science, 2005.- 928 с.

6. Шилдт Г. Полный справочник по С++. 4-е издание. - М.: Изд. Дом «Вильямс», 2003.–800с.

**ДОДАТОК А**

#include <iostream>

#include "Interface.h"

using namespace std;

int main()

{

system("chcp 1251 > null");

Interface UI;

UI.Menu();

return 0;

}

#ifndef DATABANK\_H

#define DATABANK\_H

#include <conio.h>

#include <ctime>

#include <fstream>

#include <list>

#include "Cart.h"

#include "Human.h"

#include "Credit.h"

#include "Cart.h"

#include "Investment.h"

class DataBank

{

public:

DataBank();

virtual ~DataBank();

list<Human> & GetHuman();

list<Investment> & GetInvestment();

list<Credit> & GetCredit();

list<Cart> & GetCart();

void PrintHumans();

void PrintInvestments();

void PrintCredits();

void PrintCarts();

void CreateHuman();

list<Human>::iterator SearchByInitial(string SearchInitial);

list<Human>::iterator SearchByID(int SearchID);

list<Cart>::iterator SearchCartByNumber(string Number);

list<Cart>::iterator SearchCartByID(int SearchID);

void DeleteHuman\_By\_Initials(string Delete\_Initial);

void DeleteHuman\_By\_ID(int Delete\_ID);

void Edit\_DataC(list<Cart>::const\_iterator it, Cart& edit\_obj);

void SaveAllData();

private:

list<Cart> cart;

list<Human> humans;

list<Credit> credits;

list<Investment> investment;

void InitAllData();

};

#endif // DATABANK\_H

#include "DataBank.h"

DataBank::DataBank()

{

InitAllData();

}

DataBank::~DataBank()

{

humans.clear();

credits.clear();

investment.clear();

cart.clear();

}

list<Human>& DataBank::GetHuman()

{

return \*& humans;

}

list<Investment>& DataBank::GetInvestment()

{

return \*& investment;

}

list<Credit>& DataBank::GetCredit()

{

return \*& credits;

}

void DataBank::PrintHumans()

{

if(humans.empty())

std::cout<<" Список людей порожній!"<<std::endl;

else{

for(auto it = humans.begin(); it != humans.end(); it++){

Human human = \*it;

human.Print();

}

}

}

void DataBank::PrintInvestments()

{

if(investment.empty())

std::cout<<" Список інвестицій порожній!"<<std::endl;

else{

for(auto it = investment.begin(); it != investment.end(); it++){

Investment inv = \*it;

cout<<endl<<" ----- ВЛАСНИК ІНВЕСТИЦІЇ -----"<<endl<<endl;

(\*SearchByID(inv.Get\_ID\_Human())).Print();

cout<<endl<<" ----- ДАНІ ІНВЕСТИЦІЇ -----"<<endl;

inv.Print();

cout<<endl<<" -----=================-----"<<endl;

}

}

}

void DataBank::PrintCredits()

{

if(credits.empty())

std::cout<<" Список кредитів порожній!"<<std::endl;

else{

for(auto it = credits.begin(); it != credits.end(); it++){

Credit credit = \*it;

cout<<endl<<" ----- ВЛАСНИК КРЕДИТУ -----"<<endl<<endl;

(\*SearchByID(credit.Get\_ID\_Cart())).Print();

cout<<endl<<" ----- ДАНІ КРЕДИТУ -----"<<endl;

credit.Print();

cout<<endl<<" -----=================-----"<<endl;

}

}

}

void DataBank::CreateHuman()

{

cout<<" Реєстрація відвідуючого банку в систему"<<endl<<endl;

string \_string;

Human newHuman;

newHuman.Set\_ID(humans.size()+1);

cout<<" Введіть ініціали людини"<<endl<<" >";

getline(cin, \_string);

newHuman.Set\_Initial(\_string);

cout<<" Введіть паспортні дані"<<endl<<" >";

getline(cin, \_string);

newHuman.Set\_Passport(\_string);

cout<<" Введіть номер телефону"<<endl<<" >";

getline(cin, \_string);

newHuman.Set\_Phone(\_string);

cout<<" Введіть електронну пошту"<<endl<<" >";

getline(cin, \_string);

newHuman.Set\_Email(\_string);

humans.push\_back(newHuman);

}

list<Human>::iterator DataBank::SearchByInitial(string SearchInitial)

{

auto null = humans.begin();

if(!humans.empty()){

for(auto it = humans.begin(); it != humans.end(); it++){

Human searchHuman = \*it;

if(searchHuman.Get\_Initial() == SearchInitial){

return it;

}

}

cout<<" Не знайдено дані людини = "<<SearchInitial<<endl;

return null;

}

else {

cout<<" Дані відсутні!"<<endl;

return null;

}

}

list<Human>::iterator DataBank::SearchByID(int SearchID)

{

auto null = humans.begin();

if(!humans.empty()){

for(auto it = humans.begin(); it != humans.end(); it++){

Human searchHuman = \*it;

if(searchHuman.Get\_ID() == SearchID){

return it;

}

}

cout<<" Не знайдено дані людини = "<<SearchID<<endl;

return null;

}

else {

cout<<" Дані відсутні!"<<endl;

return null;

}

}

void DataBank::Edit\_DataC(list<Cart>::const\_iterator it, Cart& edit\_obj)

{

cart.insert(it, edit\_obj);

cart.erase(it);

}

void DataBank::DeleteHuman\_By\_Initials(string Delete\_Initial)

{

auto it = SearchByInitial(Delete\_Initial);

Human deleteHuman = \*it;

if(deleteHuman.Get\_Initial() != Delete\_Initial){

cout<<endl<<" Видалення не вдалося!"<<endl;

}

else {

humans.erase(it);

cout<<endl<<" Успішно видалений!"<<endl;

}

}

void DataBank::DeleteHuman\_By\_ID(int Delete\_ID)

{

auto it = SearchByID(Delete\_ID);

Human deleteHuman = \*it;

if(deleteHuman.Get\_ID() != Delete\_ID){

cout<<endl<<" Видалення не вдалося!"<<endl;

}

else {

humans.erase(it);

cout<<endl<<" Успішно видалений!"<<endl;

}

}

void DataBank::SaveAllData()

{

fstream file; bool next = false;

Human human; Credit credit; Investment invest; Cart \_cart;

file.open("File\\Carts.txt", fstream::out);

if(!file.is\_open())

cout<<"Файл File\\Carts.txt не знайдено!"<<endl;

else{ //Людина

for(auto it = cart.begin(); it!=cart.end(); it++){

\_cart = \*it;

if(next == true){file <<endl;}

file << \_cart.Get\_ID();

file <<endl << \_cart.Get\_ID\_Human();

file <<endl << \_cart.Get\_Number();

file <<endl << \_cart.Get\_Balance();

file <<endl << \_cart.Get\_ValidTo();

next = true;

}

}

next = false;

file.close();

file.open("File\\Humans.txt", fstream::out);

if(!file.is\_open())

cout<<"Файл File\\Humans.txt не знайдено!"<<endl;

else{ //Людина

for(auto it = humans.begin(); it!=humans.end(); it++){

human = \*it;

if(next == true){file <<endl;}

file << human.Get\_ID();

file <<endl << human.Get\_Initial();

file <<endl << human.Get\_Passport();

file <<endl << human.Get\_Phone();

file <<endl << human.Get\_Email();

next = true;

}

}

next = false;

file.close();

file.open("File\\Credits.txt", fstream::out);

if(!file.is\_open())

cout<<"Файл File\\Credits.txt не знайдено!"<<endl;

else{

for(auto it = credits.begin(); it!=credits.end(); it++){

credit = \*it;

if(next == true){file <<endl;}

file << credit.Get\_ID();

file <<endl << credit.Get\_ID\_Cart();

file <<endl << credit.Get\_TypeCredit();

file <<endl << credit.Get\_Money();

file <<endl << credit.Get\_Monthly\_Fee();

file <<endl << credit.Get\_Date\_Taken();

file <<endl << credit.Get\_Term();

file <<endl << credit.Get\_Percentage();

next = true;

}

}

file.close();

next = false;

file.open("File\\Investment.txt", fstream::out);

if(!file.is\_open())

cout<<"Файл File\\Investment.txt не знайдено!"<<endl;

else{

for(auto it = investment.begin(); it!=investment.end(); it++){

invest = \*it;

if(next == true){file <<endl;}

file << invest.Get\_ID();

file <<endl << invest.Get\_ID\_Human();

file <<endl << invest.Get\_TypeDeposit();

file <<endl << invest.Get\_Deposit();

file <<endl << invest.Get\_Supplement();

file <<endl << invest.Get\_DateDeposit();

file <<endl << invest.Get\_DateAddSupplement();

next = true;

}

}

file.close();

}

list<Cart>& DataBank::GetCart()

{

return cart;

}

void DataBank::PrintCarts()

{

if(cart.empty())

std::cout<<" Список карт порожній!"<<std::endl;

else{

for(auto it = cart.begin(); it != cart.end(); it++){

Cart tmpCart = \*it;

cout<<endl<<" ----- ВЛАСНИК КАРТИ -----"<<endl<<endl;

(\*SearchByID(tmpCart.Get\_ID\_Human())).Print();

cout<<endl<<" ----- ДАНІ КАРТИ -----"<<endl;

tmpCart.Print();

cout<<endl<<" -----=================-----"<<endl;

}

}

}

list<Cart>::iterator DataBank::SearchCartByNumber(string Number)

{

auto null = cart.begin();

if(!cart.empty()){

for(auto it = cart.begin(); it != cart.end(); it++){

Cart searchCart = \*it;

if(searchCart.Get\_Number() == Number){

return it;

}

}

cout<<" Не знайдено дані карти = "<<Number<<endl;

return null;

}

else {

cout<<" Дані відсутні!"<<endl;

return null;

}

}

list<Cart>::iterator DataBank::SearchCartByID(int SearchID)

{

auto null = cart.begin();

if(!cart.empty()){

for(auto it = cart.begin(); it != cart.end(); it++){

Cart searchCart = \*it;

if(searchCart.Get\_ID() == SearchID){

return it;

}

}

cout<<" Не знайдено дані карти = "<<SearchID<<endl;

return null;

}

else {

cout<<" Дані відсутні!"<<endl;

return null;

}

}

void DataBank::InitAllData()

{

string \_string;

double \_double = 0;

int \_int = 0;

fstream file;

file.open("File\\Humans.txt", fstream::in);

if(!file.is\_open())

cout<<"Файл File\\Humans.txt не знайдено!"<<std::endl;

else{

Human human;

while(!file.eof()){

file >> \_int;

human.Set\_ID(\_int);

file.ignore();

getline(file, \_string);

human.Set\_Initial(\_string);

getline(file, \_string);

human.Set\_Passport(\_string);

getline(file, \_string);

human.Set\_Phone(\_string);

getline(file, \_string);

human.Set\_Email(\_string);

humans.push\_back(human);

}

file.close();

}

file.open("File\\Credits.txt", fstream::in);

if(!file.is\_open()){

cout<<"Файл File\\Credits.txt не знайдено!"<<std::endl;

}

else{

Credit credit;

while(!file.eof()){

file >> \_int;

credit.Set\_ID(\_int);

file >> \_int;

credit.Set\_ID\_Cart(\_int);

file.ignore();

getline(file, \_string);

credit.Set\_TypeCredit(\_string);

//file.ignore();

file >> \_double;

credit.Set\_Money(\_double);

file >> \_double;

credit.Set\_Monthly\_Fee(\_double);

file.ignore();

getline(file, \_string);

credit.Set\_Date\_Taken(\_string);

getline(file, \_string);

credit.Set\_Term(\_string);

file.ignore();

file >> \_double;

credit.Set\_Percentage(\_double);

file.ignore();

credits.push\_back(credit);

}

file.close();

}

file.open("File\\Investment.txt", fstream::in);

if(!file.is\_open()){

cout<<"Файл File\\Investment.txt не знайдено!"<<std::endl;

}

else{

Investment invest;

while(!file.eof()){

file >> \_int;

invest.Set\_ID(\_int);

file >> \_int;

invest.Set\_ID\_Cart(\_int);

file.ignore();

getline(file, \_string);

invest.Set\_TypeDeposit(\_string);

file >> \_double;

invest.Set\_Deposit(\_double);

file >> \_double;

invest.Set\_Supplement(\_double);

file.ignore();

getline(file, \_string);

invest.Set\_DateDeposit(\_string);

getline(file, \_string);

invest.Set\_DateAddSupplement(\_string);

investment.push\_back(invest);

}

file.close();

}

file.open("File\\Carts.txt", fstream::in);

if(!file.is\_open()){

cout<<"Файл File\\Carts не знайдено!"<<std::endl;

}

else{

Cart \_cart;

while(!file.eof()){

file >> \_int;

\_cart.Set\_ID(\_int);

file >> \_int;

\_cart.Set\_ID\_Human(\_int);

file.ignore();

getline(file, \_string);

\_cart.Set\_Number(\_string);

file >> \_double; file.ignore();

\_cart.Set\_Balance(\_double);

getline(file, \_string);

\_cart.Set\_ValidTo(\_string);

cart.push\_back(\_cart);

}

file.close();

}

}

#ifndef INVESTMENT\_H

#define INVESTMENT\_H

#include<string>

#include<iostream>

using namespace std;

class Investment

{

public:

Investment(int ID = 0, int ID\_Cart = 0, string TypeDeposit = "",

double Deposit = 0, double Supplement = 0, string DateDeposit = "", string DateAddSupplement = "");

int Get\_ID(){return ID;}

int Get\_ID\_Human(){return ID\_Cart;}

string Get\_TypeDeposit(){return TypeDeposit;}

double Get\_Deposit(){return Deposit;}

double Get\_Supplement(){return Supplement;}

string Get\_DateDeposit(){return DateDeposit;}

string Get\_DateAddSupplement(){return DateAddSupplement;}

void Print();

void Set\_ID(int ID);

void Set\_ID\_Cart(int ID\_Cart);

void Set\_TypeDeposit(string TypeDeposit);

void Set\_Deposit(double Deposit);

void Set\_Supplement(double Supplement);

void Set\_DateDeposit(string DateDeposit);

void Set\_DateAddSupplement(string DateAddSupplement);

private:

int ID;

int ID\_Cart;

string TypeDeposit;

double Deposit;

double Supplement;

string DateDeposit;

string DateAddSupplement;

};

#endif // INVESTMENT\_H

#include "Investment.h"

Investment::Investment(int ID, int ID\_Cart, string TypeDeposit, double Deposit, double Supplement, string DateDeposit, string DateAddSupplement)

{

this->ID = ID;

this->ID\_Cart = ID\_Cart;

this->TypeDeposit = TypeDeposit;

this->Deposit = Deposit;

this->Supplement = Supplement;

this->DateDeposit = DateDeposit;

this->DateAddSupplement = DateAddSupplement;

}

void Investment::Print()

{

cout

<<" Код - "<<ID<<endl

<<" Код людини - "<<ID\_Cart <<endl

<<" Тип депозиту - "<<TypeDeposit <<endl

<<" Сума вкладу - "<<Deposit <<endl

<<" Річний відсоток - "<<Supplement <<endl

<<" Дата вкладу - "<<DateDeposit <<endl

<<" Дата надбавки - "<<DateAddSupplement<<endl<<endl;

}

void Investment::Set\_ID(int ID)

{

this->ID = ID;

}

void Investment::Set\_ID\_Cart(int ID\_Cart)

{

this->ID\_Cart = ID\_Cart;

}

void Investment::Set\_TypeDeposit(string TypeDeposit)

{

this->TypeDeposit = TypeDeposit;

}

void Investment::Set\_Deposit(double Deposit)

{

this->Deposit = Deposit;

}

void Investment::Set\_Supplement(double Supplement)

{

this->Supplement = Supplement;

}

void Investment::Set\_DateDeposit(string DateDeposit)

{

this->DateDeposit = DateDeposit;

}

void Investment::Set\_DateAddSupplement(string DateAddSupplement)

{

this->DateAddSupplement = DateAddSupplement;

}

#ifndef CART\_H

#define CART\_H

#include<string>

#include<iostream>

using namespace std;

class Cart

{

public:

Cart(int ID = 0, int ID\_Human = 0, string Number = "", double Balance=0, string ValidTo="");

int Get\_ID(){return ID;}

int Get\_ID\_Human(){return ID\_Human;}

string Get\_Number(){return Number;}

double Get\_Balance(){return Balance;}

string Get\_ValidTo(){return ValidTo;}

void Set\_ID\_Human(int ID\_Human);

void Set\_ID(int ID);

void Set\_Number(string Number);

void Set\_Balance(double Balance);

void Set\_ValidTo(string ValidTo);

void Print();

private:

int ID;

int ID\_Human;

string Number;

double Balance;

string ValidTo;

};

#endif // CART\_H

#include "Cart.h"

Cart::Cart(int ID, int ID\_Human, string Number, double Balance, string ValidTo)

{

this->ID = ID;

this->ID\_Human = ID\_Human;

this->Number = Number;

this->Balance = Balance;

this->ValidTo = ValidTo;

}

void Cart::Set\_ID\_Human(int ID\_Human)

{

this->ID\_Human = ID\_Human;

}

void Cart::Set\_ID(int ID)

{

this->ID = ID;

}

void Cart::Set\_Number(string Number)

{

this->Number = Number;

}

void Cart::Set\_Balance(double Balance)

{

this->Balance = Balance;

}

void Cart::Set\_ValidTo(string ValidTo)

{

this->ValidTo = ValidTo;

}

void Cart::Print()

{

cout<<" Код карти = "<< ID<<endl

<<" Номер карти = "<< Number<<endl

<<" Баланс = "<< Balance<< endl

<<" Дійсна до = "<< ValidTo<<endl;

}

#ifndef INTERFACE\_H

#define INTERFACE\_H

#include "DataBank.h"

#include "Credit.h"

#include "Investment.h"

#include "Human.h"

class Interface

{

public:

Interface(){}

void Menu();

void CreateCart();

void CreateCredit();

void CreateInvestment();

void CartInfo();

void HumansInfo();

private:

int Action;

DataBank myBank;

};

#endif // INTERFACE\_H

#include "Interface.h"

void Interface::HumansInfo()

{

while(true){

cout

<<"######### ВІДВІДУВАЧІ ##########"<<endl<<endl<<endl

<<" 1 - Переглянути дані відвідувачів."<<endl

<<" 2 - Зареєструвати нового відвідувача."<<endl

<<" 3 - Сортувати дані відвідувачів. "<<endl

<<" 4 - Пошук відвідувача."<<endl<<endl

<<" 5 - Видалення даних відвідувача."<<endl<<endl

<<" 6 - Повернутись в головне меню."<<endl<<endl

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<" Введіть номер пункту! --> ";

cin>>Action;

switch(Action){

case 1 :{

system("cls");

cout<<endl<<endl

<<"########## ВІДВІДУВАЧІ ##########"<<endl<<endl<<endl;

myBank.PrintHumans();

cout

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

case 2 :{

system("cls");

cin.ignore();

myBank.CreateHuman();

cout<<endl<<" НОВІ ДАНІ ПАЦІВНИКА УСПІШНО СТВОРЕНО !!!";

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

case 3 :{

bool work\_sort = true;

while(work\_sort){

system("cls");

cout<<endl<<endl

<<"##### СОРТУВАННЯ ДАНИХ ВІДВІДУВАЧА #####"<<endl<<endl<<endl

<<" 1 Сортувати по коду."<<endl

<<" 2 Сортувати по ініціалах."<<endl<<endl

<<" 3 Вийти."<<endl

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<" Введіть номер пункту! --> ";

cin>>Action;

switch(Action){

case 1: {

myBank.GetHuman().sort([](Human &ob1, Human &ob2){

return ob1.Get\_ID() < ob2.Get\_ID();

});

cout<<endl<<" ДАНІ БУЛИ ПОСОРТОВАНІ ПО КОДУ !!!";

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

break;}

case 2: {

myBank.GetHuman().sort([](Human &ob1, Human &ob2){

return ob1.Get\_Initial() < ob2.Get\_Initial();

});

cout<<endl<<" ДАНІ БУЛИ ПОСОРТОВАНІ ПО ІНІЦІАЛАХ !!!";

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

break;}

default: {

work\_sort = false;

system("cls");

break;}

}

}

break;}

case 4 :{

bool \_search = true;

while(\_search){

system("cls");

cout<<endl<<endl

<<"\_\_\_\_\_ ПОШУК ДАНИХ ВІДВІДУВАЧА \_\_\_\_\_"<<endl<<endl<<endl

<<" 1 Пошук по коду."<<endl

<<" 2 Пошук по ініціалах."<<endl<<endl

<<" 3 Вийти."<<endl

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<" Введіть номер пункту! --> ";

cin>>Action;

switch(Action){

case 1:{

system("cls");

int search\_ID;

cout<<endl<<" Введіть код > ";

cin>>search\_ID;

Human it = \*myBank.SearchByID(search\_ID);

if(it.Get\_ID() == search\_ID){

it.Print();

}

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

case 2:{

system("cls");

cin.ignore();

string search\_Initial;

cout<<endl<<" Введіть ініціали > ";

getline (cin, search\_Initial);

Human it = \*myBank.SearchByInitial(search\_Initial);

if(it.Get\_Initial() == search\_Initial){

it.Print();

}

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

default :{

\_search = false;

system("cls");

break;}

}

}

system("cls");

break;}

case 5 :{

bool \_delete = true;

while(\_delete){

system("cls");

cout<<endl<<endl

<<"##### ВИДАЛЕННЯ ДАНИХ ВІДВІДУВАЧА #####"<<endl<<endl<<endl

<<""<<endl<<endl

<<" 1 Видалення по коду."<<endl

<<" 2 Видалення по ініціалах."<<endl

<<" 3 Вийти."<<endl

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<" Введіть номер пункту! --> ";

cin>>Action;

switch(Action){

case 1:{

int delete\_ID;

cout<<" Введіть код > ";

cin>>delete\_ID;

myBank.DeleteHuman\_By\_ID(delete\_ID);

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

case 2:{

cin.ignore();

string delete\_Initial;

cout<<" Введіть ініціали > ";

getline (cin, delete\_Initial);

myBank.DeleteHuman\_By\_Initials(delete\_Initial);

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

default :{

\_delete = false;

system("cls");

break;}

}

}

system("cls");

break;}

default :{

system("cls");

return;

break;}

}

}

}

void Interface::Menu()

{

while(true){

cout

<<"########## БАНК ##########"<<endl<<endl<<endl

<<" 0 - Кредитні карти."<<endl

<<" 1 - Відвідувачі."<<endl

<<" 2 - Список Інвестицій. "<<endl

<<" 3 - Список Кредитів."<<endl<<endl

<<" 4 - Оформлення кредиту."<<endl

<<" 5 - Оформлення інвестиціїй."<<endl<<endl

<<" 6 - Зберегти всі дані в базу даних."<<endl<<endl

<<" 7 - Вийти."<<endl<<endl

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<" Введіть номер пункту! --> ";

cin>>Action;

switch(Action){

case 0:{

system("cls");

CartInfo();

system("cls");

break;}

case 1:{

system("cls");

HumansInfo();

system("cls");

break;}

case 2:{

system("cls");

cout<<" IНВЕСТИЦІЇ"<<endl<<endl;

myBank.PrintInvestments();

cout<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

break;}

case 3:{

system("cls");

cout<<" КРЕДИТИ"<<endl<<endl;

myBank.PrintCredits();

cout<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

break;}

case 4:{

system("cls");

CreateCredit();

system("cls");

break;}

case 5:{

system("cls");

CreateInvestment();

break;}

case 6:{

system("cls");

myBank.SaveAllData();

cout<<endl<<endl<<" Дані успішно збережені!!!"<<endl;

cout<<endl<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

break;}

default : {

system("cls");

cout<<endl<<endl<<"Програма завершила свою роботу !"<<endl<<endl;

return;

break;}

}

}

}

void Interface::CreateCredit()

{

int \_int = 0, tmpAction; double \_double = 0;

string \_string;

system("cls");

Credit tempC;

tempC.Set\_ID(myBank.GetHuman().size() + 1);

cout<<"=========== ОФОРМЛЕННЯ КРЕДИТУ ==========="<<endl<<endl

<<" Оберіть пункт"<<endl<<endl

<<" 1. Обрати дані карти відвідувача."<<endl

<<" 2. Зареєструвати карту відвідувачу."<<endl

<<" 3. Скасувати."<<endl<<endl<<" >";

cin>>tmpAction;

switch(tmpAction){

case 1: {

if(myBank.GetCart().size() == 0){

cout<<" Список карт відвідувачів порожній, додайте їх ! "<<endl

<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

return;

}

myBank.PrintCarts();

while(true){

cout<<endl<<" Введіть код карти відвідувача"<<endl<<" >";

cin>>tmpAction;

if(tmpAction <= myBank.GetCart().size() && tmpAction > 0)

break;

else{

cout<<endl<<" Ви введли невірний код, спробуйте ще раз"<<endl;

}

}

tempC.Set\_ID\_Cart(tmpAction);

break;}

case 2: {

system("cls"); cin.ignore();

CreateCart();

system("cls");

tempC.Set\_ID\_Cart(myBank.GetCart().size());

break;}

default: {

cout<<" ОФОРМЛЕННЯ СКАСОВАНО! "<<endl

<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

return;}

}

system("cls");

cout<<endl<<" ПРОДОВЖЕННЯ РЕЄСТРАЦІЇ КРЕДИТУ "<<endl

<<" Введіть тип кредиту"<<endl<<" >"; cin.ignore();

getline(cin, \_string);

tempC.Set\_TypeCredit(\_string);

cout<<endl<<" Введіть суму займу (Чисельний ввід)"<<endl<<" >";

cin>>\_double;

tempC.Set\_Money(\_double);

cout<<endl<<" Введіть суму щомісячної оплати (Чисельний ввід)"<<endl<<" >";

cin>>\_double;

tempC.Set\_Monthly\_Fee(\_double); cin.ignore();

cout<<endl<<" Введіть дату взяття кредиту"<<endl<<" >";

getline(cin, \_string);

tempC.Set\_Date\_Taken(\_string);

cout<<endl<<" Введіть термін оплати кредиту"<<endl<<" >";

getline(cin, \_string);

tempC.Set\_Term(\_string);

cout<<endl<<" Введіть відсоток доплати (Чисельний ввід)"<<endl<<" >";

cin>>\_double;

tempC.Set\_Percentage(\_double);

cin.ignore();

myBank.GetCredit().push\_back(tempC);

system("cls");

cout<<endl<<endl<<" КРЕДИТ УСПІШНО ОФОРМЛЕНИЙ!!!"<<endl;

cout<<endl<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

}

void Interface::CreateInvestment()

{

int \_int = 0, tmpAction; double \_double = 0;

string \_string;

system("cls");

Investment tempI;

tempI.Set\_ID(myBank.GetCredit().size() + 1);

cout<<"=========== ВКЛАД ГРОШЕЙ (ІНВЕСТИЦІЯ) ==========="<<endl<<endl

<<" Оберіть пункт"<<endl<<endl

<<" 1. Обрати дані карти відвідувача."<<endl

<<" 2. Зареєструвати карту для відвідувача."<<endl

<<" 3. Скасувати."<<endl<<endl<<" >";

cin>>tmpAction;

switch(tmpAction){

case 1: {

if(myBank.GetCart().size() == 0){

cout<<" Список карт порожній, додайте їх ! "<<endl

<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

return;

}

myBank.PrintCarts();

while(true){

cout<<endl<<" Введіть код відвідувача"<<endl<<" >";

cin>>tmpAction;

if(tmpAction <= myBank.GetCart().size() && tmpAction > 0)

break;

else{

cout<<endl<<" Ви введли невірний код, спробуйте ще раз"<<endl;

}

}

tempI.Set\_ID\_Cart(tmpAction);

break;}

case 2: {

system("cls"); cin.ignore();

CreateCart();

system("cls");

tempI.Set\_ID\_Cart(myBank.GetCart().size());

break;}

default: {

cout<<" ОФОРМЛЕННЯ СКАСОВАНО! "<<endl

<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

return;}

}

system("cls");

cin.ignore();

cout<<endl<<" ПРОДОВЖЕННЯ РЕЄСТРАЦІЇ ІНВЕСТИЦІЇ "<<endl

<<" Введіть тип інвестиції"<<endl<<" >";

getline(cin, \_string);

tempI.Set\_TypeDeposit(\_string);

cout<<endl<<" Введіть суму депозиту (Чисельний ввід)"<<endl<<" >";

cin>>\_double;

tempI.Set\_Deposit(\_double);

cout<<endl<<" Введіть суму річної надбавки (Чисельний ввід)"<<endl<<" >";

cin>>\_double;

tempI.Set\_Supplement(\_double); cin.ignore();

cout<<endl<<" Введіть дату інвестиції"<<endl<<" >";

getline(cin, \_string);

tempI.Set\_DateDeposit(\_string);

cout<<endl<<" Введіть дату наступного нарахування надбавки"<<endl<<" >";

getline(cin, \_string);

tempI.Set\_DateAddSupplement(\_string);

myBank.GetInvestment().push\_back(tempI);

system("cls");

cout<<endl<<endl<<" ІНВЕСТИЦІЯ УСПІШНО ЗАРЕЄСТРОВАНА!!!"<<endl;

cout<<endl<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

}

void Interface::CartInfo()

{

while(true){

cout

<<"######### КРЕДИТНІ КАРТИ ##########"<<endl<<endl<<endl

<<" 1 - Переглянути дані карт."<<endl

<<" 2 - Зареєструвати нову карту."<<endl

<<" 3 - Пошук карти."<<endl<<endl

<<" 4 - Поповнення балансу карти."<<endl<<endl

<<" 5 - Повернутись в головне меню."<<endl<<endl

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<" Введіть номер пункту! --> ";

cin>>Action;

switch(Action){

case 1 :{

system("cls");

cout<<endl<<endl

<<"########## КРЕДИТНІ КАРТИ ##########"<<endl<<endl<<endl;

myBank.PrintCarts();

cout

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

case 2 :{

system("cls");

cin.ignore();

CreateCart();

cout<<endl<<" НОВІ ДАНІ КАРТИ УСПІШНО СТВОРЕНО !!!";

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

case 3 :{

bool \_search = true;

while(\_search){

system("cls");

cout<<endl<<endl

<<"\_\_\_\_\_ ПОШУК ДАНИХ КАРТИ \_\_\_\_\_"<<endl<<endl<<endl

<<" 1 Пошук по коду."<<endl

<<" 2 Пошук по номеру карти."<<endl<<endl

<<" 3 Вийти."<<endl

<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl<<endl

<<" Введіть номер пункту! --> ";

cin>>Action;

switch(Action){

case 1:{

system("cls");

int search\_ID;

cout<<endl<<" Введіть код > ";

cin>>search\_ID;

Cart it = \*myBank.SearchCartByID(search\_ID);

if(it.Get\_ID() == search\_ID){

it.Print();

}

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

case 2:{

system("cls");

cin.ignore();

string search\_;

cout<<endl<<" Введіть номер карти > ";

getline (cin, search\_);

Cart it = \*myBank.SearchCartByNumber(search\_);

if(it.Get\_Number() == search\_){

it.Print();

}

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

default :{

\_search = false;

system("cls");

break;}

}

}

system("cls");

break;}

case 4: {

string tmp;

system("cls"); cin.ignore();

cout<<"++++++++ ПОПОВНЕННЯ КАРТИ +++++++++"<<endl<<endl

<<" Введіть номер карти "<<endl<<" >";

getline(cin, tmp);

auto it = \*myBank.SearchCartByNumber(tmp);

Cart cc = it;

if(cc.Get\_Number() != tmp)

cout<<"Дані карти не знайдено!"<<endl;

else{

double tmpD;

cout<<" Введіть суму поповнення"<<endl<<" >";

cin>>tmpD;

cc.Set\_Balance(it.Get\_Balance() + tmpD);

myBank.Edit\_DataC(myBank.SearchCartByNumber(tmp), cc);

cout<<"Успішно поповнено рахунок!"<<endl;

}

cout<<endl<<endl<<" Натисніть клавішу, щоб повернутися назад"<<endl;

getch();

system("cls");

break;}

default :{

system("cls");

return;

break;}

}

}

}

void Interface::CreateCart()

{

int \_int = 0, tmpAction; double \_double = 0;

string \_string;

system("cls");

Cart tempC;

tempC.Set\_ID(myBank.GetCart().size() + 1);

cout<<"=========== ОФОРМЛЕННЯ КАРТИ ==========="<<endl<<endl

<<" Оберіть пункт"<<endl<<endl

<<" 1. Обрати дані зареєстрованого відвідувача."<<endl

<<" 2. Зареєструвати відвідувача."<<endl

<<" 3. Скасувати."<<endl<<endl<<" >";

cin>>tmpAction;

switch(tmpAction){

case 1: {

if(myBank.GetHuman().size() == 0){

cout<<" Список відвідувачів порожній, додайте їх ! "<<endl

<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

return;

}

myBank.PrintHumans();

while(true){

cout<<endl<<" Введіть код відвідувача"<<endl<<" >";

cin>>tmpAction;

if(tmpAction <= myBank.GetHuman().size() && tmpAction > 0)

break;

else{

cout<<endl<<" Ви введли невірний код, спробуйте ще раз"<<endl;

}

}

tempC.Set\_ID\_Human(tmpAction);

break;}

case 2: {

system("cls"); cin.ignore();

CreateCart();

system("cls");

tempC.Set\_ID\_Human(myBank.GetHuman().size());

break;}

default: {

cout<<" ОФОРМЛЕННЯ СКАСОВАНО! "<<endl

<<endl<<" Натисніть клавішу, щоб продовжити!"<<endl;

getch();

system("cls");

return;

}

}

system("cls");

cin.ignore();

cout<<endl<<" ПРОДОВЖЕННЯ РЕЄСТРАЦІЇ КАРТИ "<<endl

<<" Введіть номер карти"<<endl<<" >";

getline(cin, \_string);

tempC.Set\_Number(\_string);

cout<<endl<<" Введіть, карта дійсна до:"<<endl<<" >";

getline(cin, \_string);

tempC.Set\_ValidTo(\_string);

myBank.GetCart().push\_back(tempC);

system("cls");

}