МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Одеський НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ

Інститут комп’ютерних систем

Кафедра інформаційні технології

КУРСОВА РОБОТА

з дисципліни «АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ»

на тему: “Розробка автоматизованої інформаційної системи

«Банківські вклади»”

Студента І курсу групи АД-192

Березовський В.О.

Керівник

ст. викладач Косенко О.Д.

Загальна оцінка \_\_\_\_\_\_\_\_\_\_\_

Одеса – 2020

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ

Інститут комп’ютерних систем

кафедра інформаційні технології

освітньо-кваліфікаційний рівень бакалавр

за спеціальністю 126 – «Інформаційні технології»

ЗАВДАННЯ НА КУРСОВУ РОБОТУ

З ДИСЦИПЛІНИ АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

студента Березовського Владислава Олександровича

групи АД-192

“Розробка автоматизованої інформаційної системи «Банківські вклади»”

Розробити автоматизовану інформаційну систему засобами мови програмування С++ для роботи з базою даних з вказаними полями, яка б дозволяла надавати користувачеві можливість:

1) вводити дані з клавіатури і заносити їх у базу (файл);

2) видаляти записи з бази (файлу);

3) читати дані з бази (файлу);

4) змінювати записи в базі даних;

5) виконувати пошук потрібної інформації за умовою (вивести список занять у певній аудиторії);

6) упорядковувати записи за умовою (за спаданням номеру аудиторії);

7) виконати розрахунок (кількість занять у кожній групі).

Проект програми повинен бути багато файловим. Реалізувати меню для входу.

ЗМIСТ

[АННОТАЦIЯ 4](#_Toc40961155)

[ANNOTATION 5](#_Toc40961156)

[ВСТУП 6](#_Toc40961157)

[ВИКЛАДЕННЯ ОСНОВНОГО МАТЕРІАЛУ 9](#_Toc40961158)

[1 СТРУКТУРА ПРОЕКТУ 9](#_Toc40961159)

[2 Функції, які виконують поставлені завдання 9](#_Toc40961160)

[2.1 Функція add() 9](#_Toc40961161)

[2.2 Функцiя read() 10](#_Toc40961162)

[2.3 Функцiя edit() 10](#_Toc40961163)

[2.4 Функцiя sort() 11](#_Toc40961164)

[2.5 Функцiя account() 11](#_Toc40961165)

[2.6 Функцiя erase() 12](#_Toc40961166)

[2.7 Функцiя oneYearLater() 12](#_Toc40961167)

[2.8 Функцiя workWithDataBase() 13](#_Toc40961168)

[3 Методи та ідеї, які використані в програмі 14](#_Toc40961169)

[3.2 Ідея про встановлення смуги прокрутки 14](#_Toc40961170)

[4 Підключення бібліотек 14](#_Toc40961171)

[5 Структура Bank 16](#_Toc40961172)

[6 Налаштування консолі 16](#_Toc40961173)

[ВИСНОВОК 17](#_Toc40961174)

[ПЕРЕЛІК ЛІТЕРАТУРИ 18](#_Toc40961175)

[ДОДАТКИ 20](#_Toc40961176)

[Додаток А 20](#_Toc40961177)

[1 Файл Main.cpp 20](#_Toc40961178)

[2 Файл Function.cpp 24](#_Toc40961179)

[Додаток Б 38](#_Toc40961180)

АННОТАЦIЯ

Метою цієї роботи є опис створення додатка «Банківські вклади». В процесі виконання курсової роботи я розробив автоматизовану інформаційну систему «Банківські вклади» використовуючи засобами мови програмування С++ на Visual Studio. Вона призначена для роботи з БД (базою даних) зі вказаними полями, яка б дозволяла надавати користувачеві деякi можливість вводити дані з клавіатури і заносити їх у базу (файл), читати дані з бази, змінювати записи, видаляти їх, виконувати пошук потрібної інформації за умовою (знайти вкладника за номером рахунку), упорядкувати дані за зменшенням суми депозиту, а також виконувати підрахунки суми депозиту через рік. Набагато зручніше користуватися автоматизованою інформаційною системою, аніж звичайною базою даних (незапрограмованою) через можливість використовувати вже готові функції для певних дій, навіть не знаючи, як вони влаштовані, лише знаючи їх назву також, для чого вони були створені. Тож, моя програма розроблена саме завдяки функціям та роботі з файлами у бінарному режимі. Це дає певні прiорiтети: зручний та легкий у використанні інтерфейс та швидкість доступу к даним відповідно. Також розроблено меню, яке призначене для використання усіх можливостей додатку.

ANNOTATION

The purpose of this work is to describe the creation of the application "Bank deposits". In the course of the course work, I developed an automated information system "Bank Deposits" using C ++ programming language on Visual Studio. It is designed to work with a database (database) with the specified fields, which would allow the user to enter some data from the keyboard and enter them into the database (file), read data from the database, change records, delete them, search for the necessary information condition (find a depositor by account number), organize the data by reducing the amount of the deposit, as well as perform calculations of the amount of the deposit in a year. It is much more convenient to use an automated information system than a regular database (unprogrammed) because of the ability to use ready-made functions for certain actions, without even knowing how they are arranged, just knowing their name and why they were created. So, my program is designed thanks to the functions and work with files in binary mode. This gives certain priorities: a user-friendly and easy-to-use interface and data access speed, respectively. There is also a menu that is designed to use all the features of the application.

ВСТУП

Для виконання завдань я обрав мову програмування: С++.

C++ — мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Щоб виконати завдання я використовував:

• Динамічне виділення пам’яті.

• Структура.

• Функції.

• Бінарні файли.

Динамічна пам’ять — це вільна пам’ять, у якій під час виконання програми можна виділяти місце залежно від потреб користувача. Час існування динамічних змінних — від початку створення до кінця програми або до явного звільнення пам’яті. У мові C++ застосовують два способи роботи з динамічною пам’яттю. Перший з них дістався в спадщину від мови С і використовує сукупність функцій malloc(), другий — працює з операціями new та delete. Під час виконання завдань, я використовував другий спосіб використання динамічної пам’яті.

Структура – це , якесь об'єднання різних змінних (навіть з різними типами даних), якому можна присвоїти ім'я. У коді моєї програми завдяки структурі групуються дані про вкладника, у якого буде рахунок, номер рахунку і сума депозиту. Члени структури є відкритими за замовчуванням, на відміну від членів класів.

Функція (в програмуванні) – це фрагмент коду або алгоритм, реалізований на якійсь мові програмування, з метою виконання певної послідовності операцій. Отже, функції дозволяють зробити програму модульною, тобто розділити програму на кілька маленьких підпрограм (функцій), які в сукупності виконують поставлене завдання.

Бінарний файл – це лінійна послідовність байтів, що відповідає внутрішньому поданню даних без поділу на рядки. Робота з бінарними файлами відбувається швидше, ніж з текстовими. Крім того, розмір бінарних файлів менше, ніж розмір текстових файлів. При відкритті файлу з ним зв'язується потік введення-виведення. Інформація, що виводиться, записується в потік, а та, що вводиться, - зчитується з потоку. Щоб правильно прочитати файл, необхідно знати, що в ньому зберігається. У самому файлі немає нічого такого, що могло б вказати, де закінчується одне поле і починається інше.

Виконуючи завдання, я використав такі функції:

* workWithDataBase(vector<Bank>& people) – функция для возобновления работы с базой данных
* add(vector<Bank>& people) – функция для добавления записи в базу данных
* erase(vector<Bank>& people, int n) – функция для удаления определенной записи из базы данных
* read(vector<Bank>& people, int n) – функция для считывания определенной записи из базы данных
* edit(vector<Bank>& people, int n) – фуннкция для изменения определенной записи из базы данных
* account(vector<Bank>& people) – поиск человека по номер счета в банке из базы данных
* sort(vector<Bank>& people) – сортировка по уменьшению депозитного счета из базы данных
* oneYearLater(vector<Bank>& people) – сумма на счетах у клиентов через год после оформления депозита из базы данных

ВИКЛАДЕННЯ ОСНОВНОГО МАТЕРІАЛУ

1. СТРУКТУРА ПРОЕКТУ

Мій проект розбитий на два файли: Main.cpp и Function.cpp. Це зроблено для структуризації функціоналу і читабельності коду. Main.cpp – це головний файл, в якому знаходяться прототипи інших функцій та головна функція main(), а Function.cpp – це функціїi кожна з яких виконує поставлене завдання.

1. Функції, які виконують поставлені завдання
   1. Функція add()

Ця функція використовується для запису нових даних в БД, нові записи створює користувач. При виклику функції в параметр передається загальна кількість записів в БД. Виконуючись функція просить користувача ввести дані нового запису, а саме: прізвище клієнта, номер рахунку, сума депозиту, процентна ставка. У функції реалізована перевірка даних які вводить користувач, наприклад: користувач ввів прізвище клієнта, і тепер йому потрібно ввести номер лицьового рахунку та його баланс після отримання даних від користувача програма перевіряє дані на правильність, а саме якщо користувач ввів від’ємне число або менше 1000, то програма виведе текст: «СУММА ВКЛАДА НА ДЕПОЗИТ НЕ МОЖЕТ БЫТЬ МЕНЬШЕ 1000!». Також програма вкаже на помилку якщо користувач ввів любий символ який не є числом. Це буде повторюватись до тих пір поки користувач не введе вірні, в розумінні програми, дані.

* 1. Функцiя read()

За допомогою цієї функції виконується завдання: «читати дані з бази (файлу)», тобто вивід всіх даних БД. В тілі функції створюється потік ifstream file для зчитування даних з файлу в бінарному режимі. Виконується перевірка на успішне відкриття файлу. Якщо не має помилки в попередньому пункті, виконується наступна перевірка на номер запису, введений користувачем. Якщо номер запису перевищує розмір вектору, який зберігає дані або номер запису є негативним числом, то функція не виконується. Якщо все вірно - буде зчитано та виведено на екран певний запис файлу. Обов'язково закриваємо файл після закінчення його використання для очищення ресурсів й для того, щоб далі не виникало проблем з доступом до файлу.

* 1. Функцiя edit()

Функція для зміни даних запису за бажанням користувача. В параметрах вказується кількість записів в БД. Після виклику функції програма «просить» користувача ввести номер лицьового рахунку користувача, запис якого він хоче змінити. Після отримання даних, програма починає «шукати» потрібний номер рахунку серед записів в БД. Далі користувачу пропонуються дані, які він може змінити, а якщо введе невірний пункт то йому виведе “ НЕВЕРНЫЙ ВВОД! ВВЕДИТЕ ЕЩЕ РАЗ: ”. Користувач може змінювати дані БД, але він може це зробить якщо дані будуть, як і у випадку з функцією add(), вірними в розумінні програми.

* 1. Функцiя sort()

Функція для сортування БД за зменшенням суми депозиту. Приймає як параметри посилання на вектор, елемент, з якого треба починати сортування та елемент, до якого сортувати. Саме ж сортування відбувається за методом бульбашки.

Сортує завдяки двох циклів for та функції swap з бібліотеки <algorithm> для обміну записів місцями.

* 1. Функцiя account()

Функція для пошуку даних користувача за номером лицьового рахунку. В параметрі вказуємо кількість записів в БД. Виконуючи цю функцію, програма отримує дані від користувача, та в залежності від отриманих даних може вивести:

1) Прізвище користувача, якщо введений користувачем номер лицьового рахунку є в БД.

2) Номер счета користувача, якщо введений користувачем номер лицьового рахунку є в БД.

3) Сума депозиту користувача, якщо введений користувачем номер лицьового рахунку є в БД.

4) Процентна ставка користувача, якщо введений користувачем номер лицьового рахунку є в БД.

5) Повідомлення в якому йдеться, що такого лицьового рахунку в БД не існує.

* 1. Функцiя erase()

Функція призначена для видалення запису із БД. В параметрах вказується максимальна кількість записів в БД. Функція працює наступним чином: спочатку користувач вводить номер запису, далі програма перевіряє дані які ввів користувач, якщо користувач ввів: букву, любий другий символ який не є цифрою або номер рахунку якого не існує, то програма вкаже на помилку користувача, далі від нього залежить видаляти дані чи вийти в головне меню. Код програми працює так: спочатку приймаються дані від користувача, потім йде перевірка на правильність даних, якщо користувач ввів щось не вірно, програма дасть про це знать. Створюючи потік для запису ofstream file, перезаписуємо у файл за допомогою циклу змінений вектор (виконуємо перевірку на відкриття файлу та закриваємо його після закінчення використання).

* 1. Функцiя oneYearLater()

Функція для підрахунку суми депозиту через рік. Підраховуємо кінцеву вартість за формулою:

Де P сума депозиту, процентна ставка та S баланс користувача через рік.

* 1. Функцiя workWithDataBase()

Дана функція призначена для відновлення роботи з файлом при повторному відкритті програми.

Цей шматочок лістингу допомагає дізнатися скільки записів перебуває в базі даних:

begin = file.tellg();

file.seekg(0, ios::end);

end = file.tellg();

int size = (end - begin) / sizeof(Bank);

Після додаємо порожні записи в вектор, кількість - з файлу:

for (int i = 0; i < size; i++) {

people.push\_back(person);

}

А в кінці заповнюємо порожні записи вектора даними з файлу для подальшої роботи:

for (int i = 0; i < size; i++){

file.read((char\*)&people[i], sizeof(Bank));

}

1. Методи та ідеї, які використані в програмі
   1. Ідея про встановлення смуги прокрутки

Смуга прокрутки - елемент графічного інтерфейсу користувача, що використовується для відображення інформації і елементів інтерфейсу, великих за розміром, ніж використовуваний для їх відображення контейнер.

Частина коду в якому використовується властивість смуга прокрутки:

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(hConsole, &csbi);

csbi.dwSize.Y = 1000;

SetConsoleScreenBufferSize(hConsole, csbi.dwSize);

1. Підключення бібліотек

До складу будь-якої програми на мові програмування C++ включаються заголовки, за допомогою яких стає можливим доступ до додаткового вихідного коду. Підключення цих файлів автоматично робить препроцесор мови C++.

Я включив до свого проекту такі бібліотеки:

#include <iostream>

#include <windows.h>

#include <fstream>

#include <vector>

#include <algorithm>

#include <string>

#include <ctype.h>

using namespace std;

#include <iostream> - заголовок, який визначає стандартні об'єкти потоку введення/виведення.

#include <fstream> - заголовок, що надає доступ до класів файлового потоку.

#include <windows.h> - заголовок, що дає можливість використовувати в своїй програмі функціонал, що надається операційною системою Windows (наприклад, Sleep (time) - функція для штучної затримки наступної дії, робота з мишею (координатами в консолі)).

#include <vector> - це контейнер, який впорядковує елементи даного типу у вигляді лінійної послідовності. Він забезпечує швидкий довільний доступ до будь-якого елементу і дозволяє динамічно додавати елементи в послідовність і видаляти їх. Vector є найпридатнішим типом контейнера для послідовності, коли на першому місці стоїть продуктивність довільного доступу.

#include <algorithm> - визначає набір функцій, спеціально призначених для використання в діапазонах елементів (функція swap для обміну двох елементів місцями).

#include <string> - цей заголовок визначає тип рядків і набір функцій перетворення для роботи з ними.

#include <ctype.h> - заголовок, що містить оголошення функцій для класифікації і перетворення окремих символів (функція isdigt для перевірки символу на утримання числа в ньому).

using namespace std; - явна вказівка простору імен.

1. Структура Bank

Далі наголошуємо структуру Bank, яка відповідає темі роботи «Банківські вклади» з полями name - прізвище, numnerOfAccount – номер счета, sumOfDeposit – сума депозиту, persentOfDeposit – вiдсоток депозиту.

struct Bank {

char name[40];

int numberOfAccount;

double sumOfDeposit;

int percentOfDeposit;

};

1. Налаштування консолі

В процесі написання коду я вирішив налаштувати колір, мову і міжнародні стандарти організації:

setlocale(0, "ru");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("mode con cols=58 lines=35"); //задаем консоли определенный размер

system("color F1"); //задаем белый цвет фона и синий шрифта

ВИСНОВОК

Виконуючи дану курсову роботу було розроблено БД «Банківські вклади». Перше що слід відмітити, так це те що під час виконання курсової роботи, я закріпив весь пройдений матеріал за останній рік. Також я отримав досвід в програмуванні БД, створенні алгоритмів та формул за якими написав код.

Найбільшу увагу приділив таким темам, як «Потоки» та «Робота з файлами». Розібрався, як працює додаток з великою кількістю функцій, використовуючи файли. Реалізував збереження даних після закінчення роботи певної функції та перед відкриттям додатку повторно.

Також було використано такі теми, як «Динамічні структури даних», а саме – «Вектори». Завдяки їхнього устрою було набагато простіше реалізувати збереження даних без додаткового самостійного виділення пам’яті та її очищення. Меню було зроблено за допомогою механізму використання миші в консольному додатку.

Особливості моєї програми полягають в реалізації БД і гнучкого меню для користувачів. Завдяки вектором ми можемо з легкістю працювати з даними, а бінарні файли допомагають швидко скористатися даними.

Моя програма призначена в банківській сфері відповідно моєму варіанту і користувачам, які побічно стосуються цієї сфери.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Сергеев А., Терен А. Программирование в Microsoft Visual C++ 2005. Самоучитель. - М.:Диалектика, 2006. – С. 20-60.

2. Прата С. Язык программирования C. Лекции и упражнения, 5-е издание. - М.:Вильямс, 2006. – С.256-281.

3. Шилдт Г. Полный справочник по С, 4-е издание. - М.: Вильямс, 2007. – С. 26-601.

4. Страуструп Б. Язык программирования C++. - М.: Радио и связь, 2013. - 350 c.

5. Кёниг Э., Му Б. Эффективное программирование на C++. – М.: Диалектика-Вильямс, 2016. – С.402.

6. Мейерс С. Эффективный и современный С++: 42 рекомендации по использованию C++11 и C++14. – К.: Диалектика-Вильямс, 2020. – С.322

7. Stack Overflow на русском – сайт вопросов и ответов для программистов. [Электронный ресурс], Режим доступа: <https://ru.stackoverflow.com/>

8. Microsoft Docs - Справочник по языку C++. [Электронный ресурс], Режим доступа: [https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-language reference?view=vs-2019](https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-language%20reference?view=vs-2019)

9. YouTube канал #SimpleCode. [Электронный ресурс], Режим доступа: <https://www.youtube.com/channel/UCtLKO1Cb2GVNrbU7Fi0pM0w>

10. DariaErmacs – Интернет страничка человека с большим стажем программиста на С++, который делиться сикретами: <https://dariaemacs.com/>

11. CppStudio - Основы программирования на языках Си и C++ для начинающих. [Электронный ресурс], Режим доступа: <http://cppstudio.com/>

12. proginfo - язык программирования C++ для начинающих. [Электронный ресурс], Режим доступа: <https://proginfo.ru/>

12. cplusplus - Standard C++ Library reference. [Электронный ресурс], Режим доступа: <http://www.cplusplus.com/reference/>

13. Ravesli - онлайн-ресурс о программировании для начинающих и более продвинутых программистов. [Электронный ресурс], Режим доступа: <https://ravesli.com/>

14. CyberForum - С++ для начинающих. [Электронный ресурс], Режим доступа: <https://www.cyberforum.ru/cpp-beginners/>

ДОДАТКИ

Додаток А

Лістинг програмного коду:

1. Файл Main.cpp

#include <iostream>

#include <windows.h>

#include <fstream>

#include <algorithm>

#include <vector>

#include <string>

#include <ctype.h>

using namespace std;

struct Bank { //структура, согласно заданию курсовой

char name[40];

int numberOfAccount;

double sumOfDeposit;

int percentOfDeposit;

};

//прототипы функций, т.к. они находятся в другом файле

void workWithDataBase(vector<Bank>& people); //функция для возобновления работы с базой данных

void add(vector<Bank>& people); //функция для добавления записи в базу

void erase(vector<Bank>& people, int n); //функция для удаления определенной записи из базы

void read(vector<Bank>& people, int n); //функция для считывания определенной записи из базы

void edit(vector<Bank>& people, int n); //фуннкция для изменения определенной записи

void account(vector<Bank>& people); //поиск человека по номер счета в банке

void sort(vector<Bank>& people); //сортировка по уменьшению депозитного счета

void oneYearLater(vector<Bank>& people); //сумма на счетах у клиентов через год после оформления депозита

int menu() {

cout << " 1. ВВОД ДАННЫХ В БД " << endl;

cout << " 2. УДАЛЕНИЕ ЗАПИСИ ИЗ БД " << endl;

cout << " 3. ИЗМЕНЕНИЕ ЗАПИСИ В БД " << endl;

cout << " 4. РАСПЕЧАТКА ДАННЫХ БД " << endl;

cout << " 5. ВКЛАДЧИК ПО НОМЕРУ СЧЕТА " << endl;

cout << " 6. УПОРЯДОЧИТЬ ПО УМЕНЬШЕНИЮ СУММЫ ДЕПОЗИТА " << endl;

cout << " 7. ПОДСЧИТАТЬ СУММУ ДЕПОЗИТА ЧЕРЕЗ ГОД " << endl;

cout << " 8. ВЫХОД " << endl;

int choice;

while (true) {

cin >> choice;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "ВЫ ВЫБРАЛИ НЕВЕРНЫЙ ВАРИАНТ!\n";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

return choice;

}

}

}

int main() {

setlocale(0, "ru");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("mode con cols=58 lines=35"); //задаем консоли определенный размер

system("color F1"); //задаем белый цвет фона и синий шрифта

CONSOLE\_SCREEN\_BUFFER\_INFO csbi; //возвращаем полосу прокрутки

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(hConsole, &csbi);

csbi.dwSize.Y = 1000;

SetConsoleScreenBufferSize(hConsole, csbi.dwSize);

vector<Bank> people;

workWithDataBase(people);

int n;

int choice;

do {

cout << endl;

choice = menu();

if (people.size() == 0) {

switch (choice) {

case 1:

add(people);

break;

default:

cout << "В БАЗЕ ЕЩЕ НЕТ ЗАПИСЕЙ!" << endl;

}

}

else {

switch (choice){

case 1:

add(people);

break;

case 2:

cout << "ВВЕДИТЕ НОМЕР ЗАПИСИ, КОТОРУЮ НУЖНО УДАЛИТЬ: ";

while (true) {

cin >> n;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nВВЕДИТЕ НОМЕР ЗАПИСИ, КОТОРУЮ НУЖНО УДАЛИТЬ: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

erase(people, n - 1);

break;

}

}

break;

case 3:

cout << "ВВЕДИТЕ НОМЕР ЗАПИСИ В БАЗЕ, КОТОРУЮ НУЖНО ИЗМЕНИТЬ: ";

while (true) {

cin >> n;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nВВЕДИТЕ НОМЕР ЗАПИСИ В БАЗЕ, КОТОРУЮ НУЖНО ИЗМЕНИТЬ: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

edit(people, n - 1);

break;

}

}

break;

case 4:

cout << endl << "ВСЕ ЗАПИСИ БАЗЫ: " << endl << endl;

for (int i = 0; i < people.size(); i++){

cout << i + 1 << ". ";

read(people, i);

cout << endl;

}

break;

case 5:

account(people);

break;

case 6:

sort(people);

break;

case 7:

oneYearLater(people);

break;

case 8:

exit(0);

break;

default:

cout << "ВЫ ВЫБРАЛИ НЕВЕРНЫЙ ВАРИАНТ!\n";

break;

}

}

} while (choice != 8);

return 0;

}

1. Файл Function.cpp

#include <iostream>

#include <windows.h>

#include <fstream>

#include <vector>

using namespace std;

struct Bank {

char name[40];

int numberOfAccount;

double sumOfDeposit;

int percentOfDeposit;

};

void workWithDataBase(vector<Bank>& people) { //функция для возобновления работы с файлом при повторном открытии приложения

fstream file("bankAccounts.txt", ios::binary | ios::app);

streampos begin, end;

begin = file.tellg();

file.seekg(0, ios::end);

end = file.tellg();

int size = (end - begin) / sizeof(Bank); //узнаем, сколько записей в базе

file.close();

Bank person;

for (int i = 0; i < size; i++) {

people.push\_back(person); //добавляем пустые записи в вектор, количество - из файла

}

file.open("bankAccounts.txt", ios::binary | ios::in);

for (int i = 0; i < size; i++){

file.read((char\*)&people[i], sizeof(Bank)); //заполняем пустые записи вектора данными из файла для дальнейшей работы

}

}

void add(vector<Bank>& people) { //функция для добавления новой записи в базу

Bank person;

string emptyLine = "";

cout << "ФАМИЛИЯ ВКЛАДЧИКА: ";

cin.getline(person.name, sizeof(person.name));

while (person.name == emptyLine) { //делаем проверку на ввод пустой строки пользователем

cout << "ЭТО ПОЛЕ НЕ МОЖЕТ БЫТЬ ПРОПУЩЕНО!\nФАМИЛИЯ ВКЛАДЧИКА: ";

cin.getline(person.name, sizeof(person.name));

}

cout << "НОМЕР СЧЕТА: ";

while (true) {

cin >> person.numberOfAccount;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nНОМЕР СЧЕТА: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

break;

}

}

cout << "CУММА ДЕПОЗИТА: ";

while (true) {

cin >> person.sumOfDeposit;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nCУММА ДЕПОЗИТА: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

while (person.sumOfDeposit < 1000) { //делаем проверку на ввод пустой строки пользователем

cout << "СУММА ВКЛАДА НА ДЕПОЗИТ НЕ МОЖЕТ БЫТЬ МЕНЬШЕ 1000!\nCУММА ДЕПОЗИТА: ";

cin >> person.sumOfDeposit;

}

break;

}

}

cout << "ПРОЦЕНТНАЯ СТАВКА: ";

while (true) {

cin >> person.percentOfDeposit;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nПРОЦЕНТНАЯ СТАВКА: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

while (person.percentOfDeposit < 0 || person.percentOfDeposit > 35) { //делаем проверку на ввод пустой строки пользователем

cout << "ПРОЦЕНТКАЯ СТАВКА НЕ МОЖЕТ БЫТЬ ОТРИЦАТЕЛЬНОЙ ИЛИ БОЛЬ-\nШЕ 35%!\nПРОЦЕНТНАЯ СТАВКА: ";

cin >> person.percentOfDeposit;

}

break;

}

}

people.push\_back(person);

ofstream file("bankAccounts.txt", ios::binary | ios::out | ios::app);

if (!file) {

cout << "ФАЙЛ НЕ БЫЛ ОТКРЫТ"; // Проверка открытия файла

}

else {

file.write((char\*)&person, sizeof(Bank)); //Записываем объекты в открытый нами файл

}

file.close();

}

void erase(vector<Bank>& people, int n) { //функция для удаления записи из базы

if (n >= 0 && n < people.size()) { //проверка на корректный ввод пользователя данных

auto iterator = people.begin(); //создаем итератор для возможности использовать функцию erase у вектора

people.erase(iterator + n); //удаляем запись из вектора

ofstream file("bankAccounts.txt", ios::binary | ios::out);

if (!file) {

cout << "ФАЙЛ НЕ БЫЛ ОТКРЫТ"; // Проверка открытия файла

}

else {

for (int i = 0; i < people.size(); i++){

file.write((char\*)&people[i], sizeof(Bank)); //Записываем изменненый вектор в открытый нами файл

}

}

file.close();

cout << "ЗАПИСЬ " << n + 1 << " УДАЛЕНА\n";

}

else {

cout << "НЕТ ТАКОЙ ЗАПИСИ\n";

}

}

void read(vector<Bank>& people, int n) { //функция для чтения записи из файла

ifstream file("bankAccounts.txt", ios::binary);

if (!file) {

cout << "ФАЙЛ НЕ БЫЛ ОТКРЫТ"; // Проверка открытия файла

}

else {

if (n >= 0 && n < people.size()) { //проверка на корректный ввод пользователя данных

Bank lesson;

file.seekg(n \* sizeof(Bank));

file.read((char\*)&lesson, sizeof(Bank)); //считываем данные записи из файла

cout << "ФАМИЛИЯ ВКЛАДЧИКА: " << people[n].name << endl; //выводим на экран

cout << "НОМЕР СЧЕТА: " << people[n].numberOfAccount << endl;

cout << "CУММА ДЕПОЗИТА: " << people[n].sumOfDeposit << endl;

cout << "ПРОЦЕНТНАЯ СТАВКА: " << people[n].percentOfDeposit << endl;

}

else {

cout << "НЕТ ТАКОЙ ЗАПИСИ\n";

}

}

file.close();

}

void edit(vector<Bank>& people, int n) { //функция для изменения записи

if (n >= 0 && n < people.size()) { //проверка на корректный ввод пользователя данных

cout << "\nДАННЫЕ ДО ИЗМЕНЕНИЯ: \n\n";

read(people, n);

Bank person;

string emptyLine = "";

person = people[n];

int choice = 0;

cout << "\nКАКОЕ ПОЛЕ НУЖНО ИЗМЕНИТЬ?\n1. ФАМИЛИЯ ВКЛАДЧИКА\n2. НОМЕР СЧЕТА\n3. CУММА ДЕПОЗИТА\n4. ПРОЦЕНТНАЯ СТАВКА\n";

while (true) {

cin >> choice;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nВВЕДИТЕ ЕЩЕ РАЗ: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

break;

}

}

//делаем такие же провери для полей, как и при добавлении записи в базу

switch (choice) {

case 1:

cout << "ФАМИЛИЯ ВКЛАДЧИКА: ";

cin.getline(person.name, sizeof(person.name));

while (person.name == emptyLine) { //делаем проверку на ввод пустой строки пользователем

cout << "ЭТО ПОЛЕ НЕ МОЖЕТ БЫТЬ ПРОПУЩЕНО!\nФАМИЛИЯ ВКЛАДЧИКА: ";

cin.getline(person.name, sizeof(person.name));

}

break;

case 2:

cout << "НОМЕР СЧЕТА: ";

while (true) {

cin >> person.numberOfAccount;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nНОМЕР СЧЕТА: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

break;

}

}

break;

case 3:

cout << "CУММА ДЕПОЗИТА: ";

while (true) {

cin >> person.sumOfDeposit;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nCУММА ДЕПОЗИТА: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

while (person.sumOfDeposit < 1000) { //делаем проверку на ввод пустой строки пользователем

cout << "СУММА ВКЛАДА НА ДЕПОЗИТ НЕ МОЖЕТ БЫТЬ МЕНЬШЕ 1000!\nCУММА ДЕПОЗИТА: ";

cin >> person.sumOfDeposit;

}

break;

}

}

break;

case 4:

cout << "ПРОЦЕНТНАЯ СТАВКА: ";

while (true) {

cin >> person.percentOfDeposit;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nПРОЦЕНТНАЯ СТАВКА: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

while (person.percentOfDeposit < 0 || person.percentOfDeposit > 35) { //делаем проверку на ввод пустой строки пользователем

cout << "ПРОЦЕНТКАЯ СТАВКА НЕ МОЖЕТ БЫТЬ ОТРИЦАТЕЛЬНОЙ ИЛИ БОЛЬ-\nШЕ 35%!\nПРОЦЕНТНАЯ СТАВКА: ";

cin >> person.percentOfDeposit;

}

break;

}

}

break;

default:

cin.ignore();

break;

}

auto iterator = people.begin();

\*(iterator + n) = person; //изменяем запись с помощью итератора

ofstream file("bankAccounts.txt", ios::binary | ios::out);

if (!file) {

cout << "ФАЙЛ НЕ БЫЛ ОТКРЫТ"; // Проверка открытия файла

}

else {

for (int i = 0; i < people.size(); i++){

file.write((char\*)&people[i], sizeof(Bank)); //перезаписываем объекты в открытый нами файл

}

}

file.close();

//выведем измененную запись

cout << endl << "ПОСЛЕ ИЗМЕНЕНИЯ:\n";

read(people, n);

}

else {

cout << "НЕТ ТАКОЙ ЗАПИСИ\n";

}

}

void account(vector<Bank>& people) {

bool found = false;

int numberOfAccount;

cout << "НОМЕР СЧЕТА: ";

while (true) {

cin >> numberOfAccount;

// Проверка на предыдущее извлечение

if (cin.fail()){ // если предыдущее извлечение оказалось неудачным,

cin.clear(); // то возвращаем cin в 'обычный' режим работы

cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера

cout << "НЕВЕРНЫЙ ВВОД!\nНОМЕР СЧЕТА: ";

}

else {

cin.ignore(32767, '\n'); // удаляем лишние значения

for (int i = 0; i < people.size(); i++) {

if (numberOfAccount == people[i].numberOfAccount){

found = true;

break;

}

}

if (found == true) {

for (int i = 0; i < people.size(); i++){

if (numberOfAccount == people[i].numberOfAccount){

cout << "ФАМИЛИЯ ВКЛАДЧИКА: " << people[i].name << endl; //выводим на экран

cout << "НОМЕР СЧЕТА: " << people[i].numberOfAccount << endl;

cout << "CУММА ДЕПОЗИТА: " << people[i].sumOfDeposit << endl;

cout << "ПРОЦЕНТНАЯ СТАВКА: " << people[i].percentOfDeposit << endl;

}

}

}

else {

cout << "НЕТ ВКЛАДЧИКА С ТАКИМ НОМЕРОМ СЧЕТА\n";

break;

}

}

}

}

void sort(vector<Bank>& people) {

for (int i = 0; i < people.size() - 1; i++) {

for (int j = 0; j < people.size() - i - 1; j++) {

if (people[j].sumOfDeposit < people[j + 1].sumOfDeposit) {

// меняем элементы местами

swap(people[j], people[j + 1]);

}

}

}

for (int i = 0; i < people.size(); i++) {

read(people, i);

cout << endl;

}

}

void oneYearLater(vector<Bank>& people) {

for (int i = 0; i < people.size(); i++) {

people[i].sumOfDeposit = people[i].sumOfDeposit + people[i].sumOfDeposit \* (double)((double)people[i].percentOfDeposit / 100.0);

}

cout << "ЧЕРЕЗ ГОД: " << endl;

for (int i = 0; i < people.size(); i++) {

read(people, i);

cout << endl;

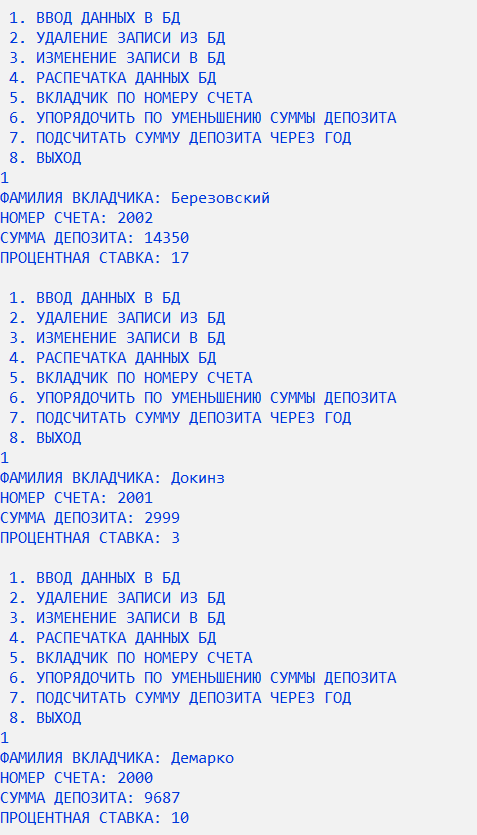
}

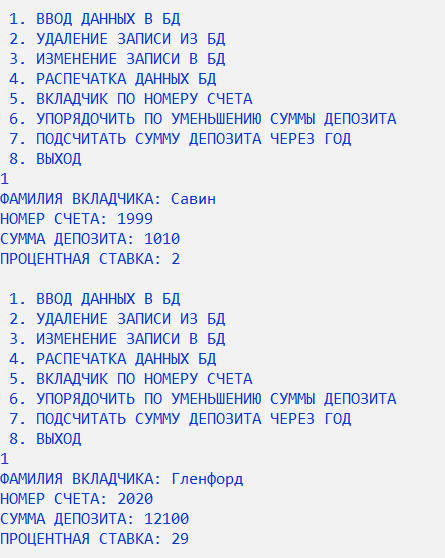
}

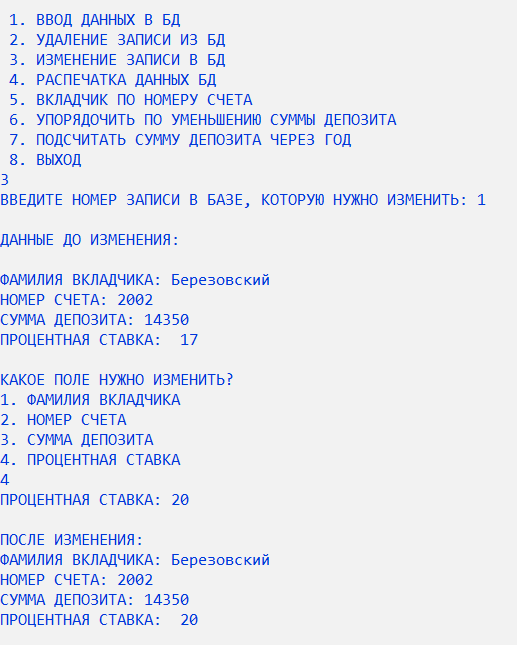
Додаток Б

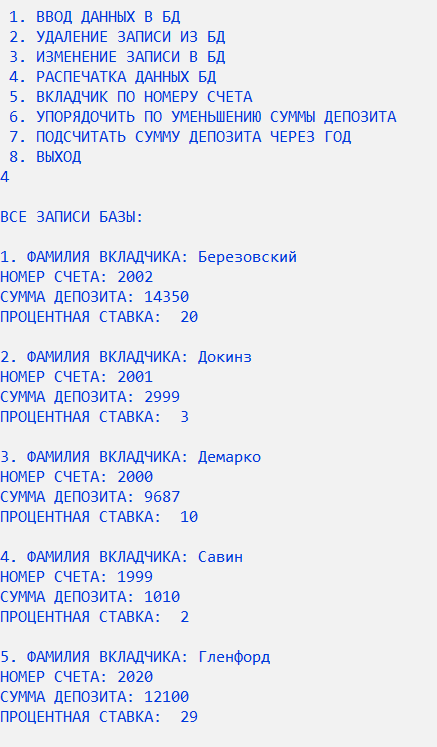
Відображення додатку в консолі:

Вводимо п'ять записів в базу даних:

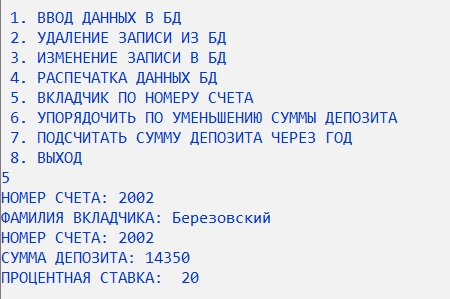


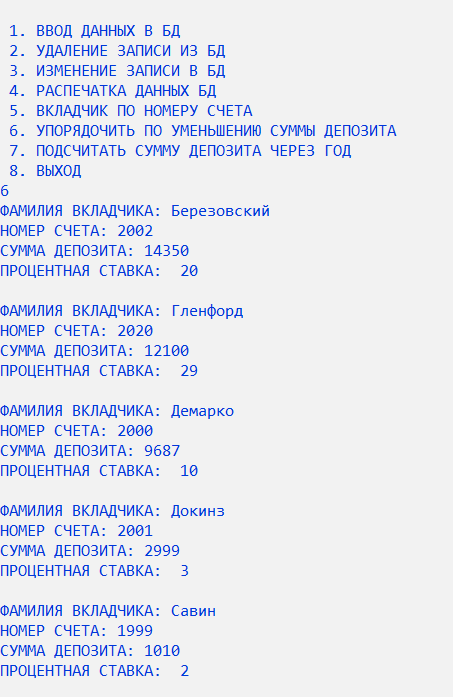


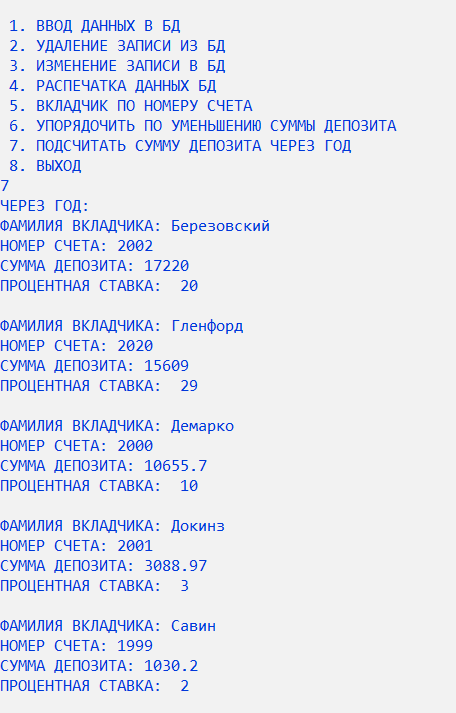
Змінити запис в БД відповідно з вибором меню:

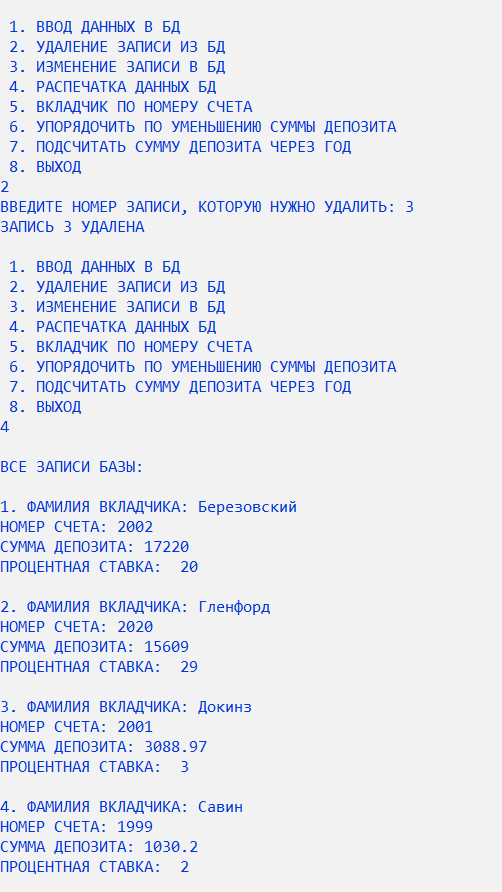
Висновок на екран всіх записів з БД: 

Знайдемо вкладника за номером рахунку:



Впорядкувати по зменшенню суму депозиту: 

Підрахувати суму депозиту через рік: 

Видалити певну запис з БД і вивести на екран: 

Вихід з програми:

