Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

**КУРСОВА РОБОТА**

з дисципліни «Алгоритмізація та програмування»

Тема «Програмування динамічної структури даних – депозитний відділ банку.

Студента (ки) \_\_1\_\_ курсу АІ-222 групи

Спеціальності 122 – «Комп’ютерні науки»

\_\_\_\_\_\_\_\_\_Лясковського.Артем.А.\_\_

(прізвище та ініціали)

Керівник доцент, к.т.н. Бабілунга О. Ю.

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

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

м. Одеса – 2023 рік

Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

**ЗАВДАННЯ**

НА КУРСОВУ РОБОТУ

Студенту Лясковському Артему Андрійовичу група АІ-222

1. Тема роботи

«Програмування динамічної структури даних – депозитний відділ банку.

2. Термін здачі студентом закінченої роботи 16.06.2023

3. Початкові дані до проекту (роботи)

Варіант 9

Структура:

Лясковський; стаж роботи 1рік ;чоловіча; рік народження 15.07.05; адреса м.Рені вул.28червня.

Програма повинна виконувати: додавання елемента; видалення елемента; можливість коригування даних; виведення всіх даних;

Завдання видано 20.03.23 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Завдання прийнято до виконання 20.03.23 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

КОД ПРОГРАМИ

#include <iostream>  
#include <fstream>  
#include <string>  
#include <sstream>  
  
using namespace std;  
  
struct Deposit {  
 string type;  
 string client;  
 string start\_date;  
 string end\_date;  
 int term;  
 float interest\_rate;  
};  
  
class DepositNode {  
public:  
 DepositNode(Deposit\* deposit, DepositNode\* next = nullptr) {  
 this->deposit = deposit;  
 this->next = next;  
 }  
 Deposit\* deposit;  
 DepositNode\* next;  
};  
  
class DepositStack {  
public:  
 DepositStack() {  
 top = nullptr;  
 }  
 ~DepositStack() {  
 DepositNode\* temp;  
 while (top != nullptr) {  
 temp = top;  
 top = top->next;  
 delete temp->deposit;  
 delete temp;  
 }  
 }  
 bool isEmpty() {  
 return top == nullptr;  
 }  
 void push(Deposit\* deposit) {  
 DepositNode\* newNode = new DepositNode(deposit, top);  
 top = newNode;  
 }  
 Deposit\* pop() {  
 if (isEmpty()) {  
 return nullptr;  
 }  
 Deposit\* deposit = top->deposit;  
 DepositNode\* temp = top;  
 top = top->next;  
 delete temp;  
 return deposit;  
 }  
 void printAllDeposits() {  
 if (isEmpty()) {  
 cout << "Stack is empty." << endl;  
 } else {  
 DepositNode\* current = top;  
 while (current != nullptr) {  
 Deposit\* deposit = current->deposit;  
 cout << "Deposit type: " << deposit->type << endl;  
 cout << "Client name: " << deposit->client << endl;  
 cout << "Start date: " << deposit->start\_date << endl;  
 cout << "End date: " << deposit->end\_date << endl;  
 cout << "Term: " << deposit->term << endl;  
 cout << "Interest rate: " << deposit->interest\_rate << endl;  
 current = current->next;  
 }  
 }  
 }  
 bool updateDeposit() {  
 int position;  
 cout << "Enter the position of the deposit you want to update: ";  
 cin >> position;  
 DepositNode\* current = top;  
 int currentPosition = 1;  
 while (current != nullptr && currentPosition < position) {  
 current = current->next;  
 currentPosition++;  
 }  
 if (current == nullptr) {  
 cout << "Invalid position." << endl;  
 return false;  
 }  
 Deposit\* deposit = current->deposit;  
 cout << "Enter new deposit type: ";  
 cin >> deposit->type;  
 cout << "Enter new client name: ";  
 cin >> deposit->client;  
 cout << "Enter new start date: ";  
 cin >> deposit->start\_date;  
 cout << "Enter new end date: ";  
 cin >> deposit->end\_date;  
 cout << "Enter new term: ";  
 cin >> deposit->term;  
 cout << "Enter new interest rate: ";  
 cin >> deposit->interest\_rate;  
 return true;  
 }  
  
 DepositNode\* top;  
};  
  
void saveDepositsToFile(DepositStack& stack) {  
 ofstream outputFile("deposits.txt");  
 if (!outputFile) {  
 cout << "Error: could not open file." << endl;  
 return;  
 }  
 DepositNode \*current = stack.top;  
 while (current != nullptr) {  
 Deposit \*deposit = current->deposit;  
 outputFile << deposit->type << "," << deposit->client << "," << deposit->start\_date << "," << deposit->end\_date  
 << "," << deposit->term << "," << deposit->interest\_rate << endl;  
 current = current->next;  
 }  
 outputFile.close();  
}  
void loadDepositsFromFile(DepositStack& stack) {  
 ifstream inputFile("deposits.txt");  
 if (!inputFile) {  
 cout << "Error: could not open file." << endl;  
 return;  
 }  
 string line;  
 while (getline(inputFile, line)) {  
 Deposit\* deposit = new Deposit();  
 stringstream ss(line);  
 getline(ss, deposit->type, ',');  
 getline(ss, deposit->client, ',');  
 getline(ss, deposit->start\_date, ',');  
 getline(ss, deposit->end\_date, ',');  
 string termString, interestRateString;  
 getline(ss, termString, ',');  
 getline(ss, interestRateString, ',');  
 deposit->term = stoi(termString);  
 deposit->interest\_rate = stof(interestRateString);  
 stack.push(deposit);  
 }  
 inputFile.close();  
}  
int main() {  
 DepositStack stack;  
  
 loadDepositsFromFile(stack);  
  
 while (true) {  
  
 cout << endl << "===== Deposit Management System =====" << endl;  
 cout << "1. Add a new deposit" << endl;  
 cout << "2. Update an existing deposit" << endl;  
 cout << "3. Print all deposits" << endl;  
 cout << "4. Save deposits to file" << endl;  
 cout << "5. Quit" << endl;  
 cout << "Enter your choice (1-5): ";  
  
 int choice;  
 cin >> choice;  
  
 switch (choice) {  
 case 1: {  
  
 Deposit\* deposit = new Deposit();  
 cout << "Enter deposit type: ";  
 cin >> deposit->type;  
 cout << "Enter client name: ";  
 cin >> deposit->client;  
 cout << "Enter start date: ";  
 cin >> deposit->start\_date;  
 cout << "Enter end date: ";  
 cin >> deposit->end\_date;  
 cout << "Enter term: ";  
 cin >> deposit->term;  
 cout << "Enter interest rate: ";  
 cin >> deposit->interest\_rate;  
 stack.push(deposit);  
 break;  
 }  
 case 2: {  
  
 bool success = stack.updateDeposit();  
 if (!success) {  
 cout << "Failed to update deposit." << endl;  
 }  
 break;  
 }  
 case 3: {  
  
 stack.printAllDeposits();  
 break;  
 }  
 case 4: {  
  
 saveDepositsToFile(stack);  
 break;  
 }  
 case 5: {  
  
 cout << "Exiting program." << endl;  
 return 0;  
 }  
 default: {  
 cout << "Invalid choice." << endl;  
 break;  
 }  
 }  
 }  
 return 0;  
}

СКРИНШОТИ РЕЗУЛЬТАТІВ РОБОТИ

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание