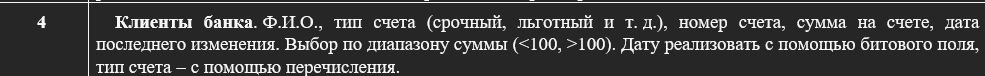
Лабораторная работа 5

По дисциплине «Основы алгоритмизации и программирования»

На тему «Объединения, перечисления, битовые поля»

***Задание#1***



Macros.h

#include <iostream>

#include <vector>

#include <ctime>

#include <cmath>

#include <iomanip>

#include <stack>

#include <map>

#include <string>

#include <chrono>

#include <fstream>

#include <Windows.h>

#define vi cout <<

#define vv cin >>

#define nl cout << "\n";

#define nw cout << "\t";

#define SCOCP1251 SetConsoleOutputCP(1251);

#define SCCP1251 SetConsoleCP(1251);

#define ret return

#define pause system("pause");

using namespace :: std;

.cpp

#include "Macros.h"

enum AccountType {

SAVINGS, // Сберегательный

PREFERENTIAL, // Льготный

};

struct Date {

unsigned day : 5; // 0-31

unsigned month : 4; // 0-15

unsigned year : 12; // 0-4095

};

struct BankClient {

string fullName; // Фамилия, имя клиента банка

AccountType accountType; // Тип аккаунта

int accountNumber = 0; // Номер счета

long long accountBalance = 0; // Баланс

Date lastModifiedDate; // Битовое поле для даты последнего изменения информации

};

const int MAX\_CLIENTS = 15; // максимальное количество клиентов

BankClient clients[MAX\_CLIENTS]; // массив для хранения клиентов

int currentClientsCount = 0; //счетчик клиентов

// Ввод информации о клиенте

void inputClientsData();

// Вывод информации обо всех клиентах

void displayAllClientsData();

// Удаление данных о клиенте

void deleteClient(int accountNumberToDelete);

// Поиск клиента

void searchClient();

int main() {

SCOCP1251

SCCP1251

int choice;

do { // вывод меню

nl

vi "Банковская система управления клиентами"; nl

vi "1. Ввод данных о клиентах"; nl

vi "2. Вывести данные клиента"; nl

vi "3. Удалить клиента"; nl

vi "4. Поиск клиента"; nl

vi "0. Выход"; nl

vi "Введите свой выбор: "; nl

vv choice; cin.ignore(); // Очищаем буфер от знака новой строки после ввода числа

switch (choice) {

case 1:

inputClientsData();

break;

case 2:

displayAllClientsData();

break;

case 3: {

int accountNumberToDelete;

vi "Введите номер счета клиента, которого необходимо удалить: ";

vv accountNumberToDelete;

deleteClient(accountNumberToDelete);

break;

}

case 4:

searchClient();

break;

case 0:

vi "Выход из программы...."; nl

break;

default:

vi "Некорректный выбор"; nl

}

} while (choice != 0);

pause

ret 0;

}

void inputClientsData() {

if (currentClientsCount >= MAX\_CLIENTS) { // Проверяем, что еще есть место в массиве для клиентов

cerr << "Максимальное количество клиентов уже достигнуто"; nl

ret;

}

// Создаем новый объект структуры для представления текущего клиента

BankClient newClient;

vi "Введите Фамилия, имя: ";

getline(cin, newClient.fullName); // используем getline для получения строки и записи в поле структуры

vi "Введите тип аккаунта (0 - Сберегательный, 1 - Льготный): ";

int accountTypeInput; // Поскольку сразу считать в типе перечисления нельзя

vv accountTypeInput;

newClient.accountType = static\_cast<AccountType>(accountTypeInput); // Конвертируем из int в тип данных перечисления AccountType, используя static\_cast и записываем в структуру

vi "Введите номер счета: ";

vv newClient.accountNumber;

vi "Введите баланс: ";

vv newClient.accountBalance;

vi "Введите дату последнего изменения (день месяц год, через пробел): ";

unsigned day, month, year; // также нельзя записать в битовое поле сразу

vv day >> month >> year; // сначала сохраняем в переменные

newClient.lastModifiedDate.day = day; // затем присваиваем их значения соответствующим полям

newClient.lastModifiedDate.month = month;

newClient.lastModifiedDate.year = year;

clients[currentClientsCount++] = newClient; // сохраняем клиента

}

void displayClientInfo(const BankClient& client) {

// принимаем объект структуры, который представляет клиента и выводим содержимое всех его полей

vi "ФИО: " << client.fullName; nl

switch (client.accountType)

{

case SAVINGS:

vi "Тип счета: Сберегательный"; nl

break;

case PREFERENTIAL:

vi "Тип счета: Льготный"; nl

break;

default:

vi "Тип счета: Неизвестно"; nl

break;

}

vi "Номер счета: " << client.accountNumber; nl

vi "Баланс: " << client.accountBalance; nl

vi "Дата последнего изменения: " << client.lastModifiedDate.day

<< " " << client.lastModifiedDate.month

<< " " << client.lastModifiedDate.year; nl nl

}

void displayAllClientsData() {

// Перебираем всех клиентов в массиве и для каждого вызываем функцию вывода

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

displayClientInfo(clients[i]);

}

}

void searchClient() {

char comparisonOperator;

long long Balance;

vi "Введите критерий(<, >, =) и сумму сравнения(например, <100, >100, =100): ";

vv comparisonOperator; // для char считается только один символ

vv Balance; //остальные для числа

switch (comparisonOperator) {

case '<': {

for (int i = 0; i < currentClientsCount; i++) { // Перебираем всех клиентов в массиве и для каждого сравниваем баланс с введенным балансом

if (clients[i].accountBalance < Balance) { // если условие выполняется, то выводим информацию о клиенте

vi "Найден клиент с балансом меньше " << Balance << ":"; nl

displayClientInfo(clients[i]); //Выводим данные о клиенте

}

}

break;

}

case '>': {

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

if (clients[i].accountBalance > Balance) {

vi "Найден клиент с балансом больше " << Balance << ":"; nl

displayClientInfo(clients[i]);

}

}

break;

}

case '=': {

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

if (clients[i].accountBalance == Balance) {

vi "Найден клиент с точным балансом " << Balance << ":"; nl

displayClientInfo(clients[i]);

}

}

break;

}

default:

vi "Некорректный оператор сравнения"; nl

}

}

void deleteClient(int accountNumberToDelete) {

int foundIndex = -1;

// сначала находим индекс клиента для удаления в массиве по номеру счета

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

if (clients[i].accountNumber == accountNumberToDelete) {

foundIndex = i;

break;

}

}

// если нашли, то удаляем его путем сдвига всех остальных клиентов, которые стоят справа от него, влево на 1

if (foundIndex != -1) {

for (int i = foundIndex; i < currentClientsCount - 1; i++) {

clients[i] = clients[i + 1];

}

--currentClientsCount; // после уменьшения счетчика, при вводе след. клиента, удаленный будет перезаписан

vi "Клиент с номером счета " << accountNumberToDelete << " был удален."; nl

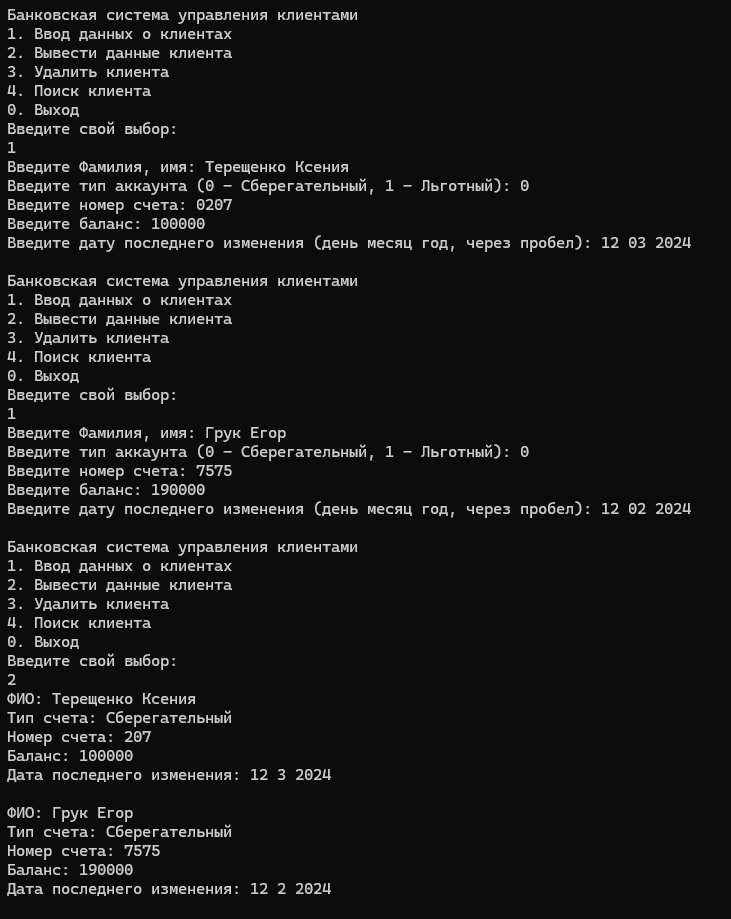
}

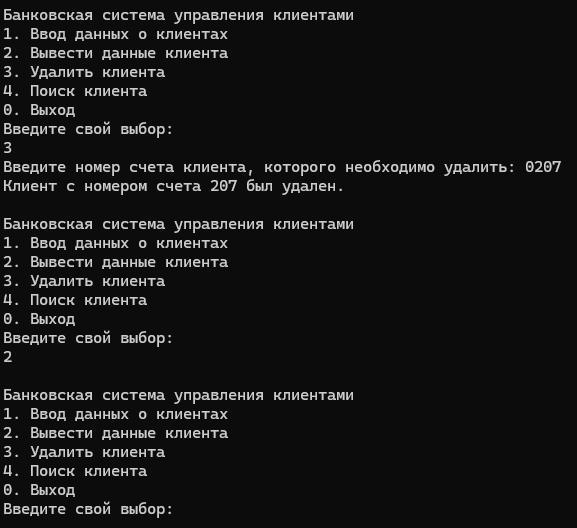
else {

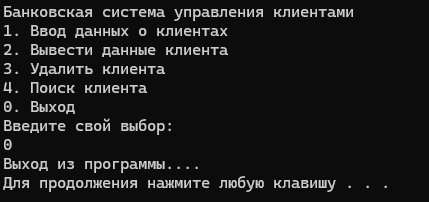
vi "Клиент с номером счета " << accountNumberToDelete << " не найден."; nl

}

}





***Задание#2***



.cpp

#include "Macros.h"

struct ClientInfo {

char fullName[50]; // Фамилия, имя клиента банка

char accountType[40]; // Тип аккаунта

char lastModifiedDate[20]; // Дата последнего изменения информации

int accountNumber = 0; // Номер счета

long long accountBalance = 0; // Баланс

};

union BankClient

{

ClientInfo clientInfo;

char allData[122]; // Сумма размеров всех полей в структуре

BankClient() : clientInfo() {}; // Конструктор по умолчанию

};

const int MAX\_CLIENTS = 15;

BankClient clients[MAX\_CLIENTS];

int currentClientsCount = 0;

// Вводим информацию о клиенте

void inputClientsData();

// Вывод информации обо всех клиентах

void displayClientsData();

// Ищем клиента

void searchClient();

// Чтение данных из файла

void readFromFile();

// Запись данных в файл

void writeToFile();

int main() {

SCOCP1251

SCCP1251

int choice;

do {

nl

vi "Банковская система управления клиентами"; nl

vi "1. Ввод данных о клиенте"; nl

vi "2. Показать данные клиента"; nl

vi "3. Поиск клиентов"; nl

vi "4. Сохранить данные в файл"; nl

vi "5. Прочитать данные из файла"; nl

vi "0. Выход"; nl

vi "Введите свой выбор: ";

vv choice;

cin.ignore(); // Удаляем символ новой строки из буфера после ввода числа, чтобы getline не получал пустую строку вместо ввода пользователя

switch (choice) {

case 1:

inputClientsData();

break;

case 2:

displayClientsData();

break;

case 3:

searchClient();

break;

case 4:

writeToFile();

break;

case 5:

readFromFile();

break;

case 0:

vi "выход из программы...."; nl

break;

default:

vi "некорректный выбор"; nl

}

} while (choice != 0);

pause

ret 0;

}

void inputClientsData() {

if (currentClientsCount >= MAX\_CLIENTS) {

cerr << "максимальное количество клиентов уже достигнуто"; nl

ret;

}

BankClient newClient; // создаем новый объект объединения

// Получаем от пользователя все сведения и сохраняем

vi "Введите Фамилию, имя: ";

cin.getline(newClient.clientInfo.fullName, sizeof(newClient.clientInfo.fullName)); // Как и в прошлом задании считываем строку в поле, но количество символов для записи определяем с помощью sizeof

vi "Введите тип аккаунта: ";

cin.getline(newClient.clientInfo.accountType, sizeof(newClient.clientInfo.accountType));

vi "Введите номер счета: ";

vv newClient.clientInfo.accountNumber;

vi "Введите баланс: ";

vv newClient.clientInfo.accountBalance;

vi "Введите дату последнего изменения: ";

cin.ignore();// Удаляем символ новой строки из буфера после ввода числа

cin.getline(newClient.clientInfo.lastModifiedDate, sizeof(newClient.clientInfo.lastModifiedDate));

// Увеличиваем количество клиентов при использовании переменной в индексе при сохранении сведений о клиенте

clients[currentClientsCount++] = newClient;

}

void searchClient() {

int accountNumberToSearch;

vi " введите номер счета для поиска: ";

vv accountNumberToSearch;

// Проход по массиву клиентов

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

// Проверка, совпадает ли номер аккаунта с искомым

if (clients[i].clientInfo.accountNumber == accountNumberToSearch) {

// Вывод информации о найденном клиенте

vi "найденный клиент: "; nl

vi "ФИО: " << clients[i].clientInfo.fullName; nl

vi "тип счета: " << clients[i].clientInfo.accountType; nl

vi "номер счета: " << clients[i].clientInfo.accountNumber; nl

vi "Баланс: " << clients[i].clientInfo.accountBalance; nl

vi "дата последней модификации: " << clients[i].clientInfo.lastModifiedDate; nl nl

ret; // клиент найден, завершаем функцию

}

}

// Если цикл завершился без возврата, клиент не найден

vi "клиент с номером счета " << accountNumberToSearch << " не найден."; nl

}

void displayClientsData() {

// Проход по массиву клиентов для вывода информации о каждом

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

// Вывод информации о клиенте

vi "\nКлиент " << i + 1 << ": "; nl

vi "ФИО: " << clients[i].clientInfo.fullName; nl

vi "тип счета: " << clients[i].clientInfo.accountType; nl

vi "номер счета: " << clients[i].clientInfo.accountNumber; nl

vi "Баланс: " << clients[i].clientInfo.accountBalance; nl

vi "дата последней модификации: " << clients[i].clientInfo.lastModifiedDate; nl nl

}

}

void readFromFile() {

ifstream inFile("bank\_clients.txt"); //открытие файла для чтения

if (!inFile.is\_open()) { // Проверка на успешность открытия файла

cerr << "Ошибка открытия файла для чтения."; nl

ret;

}

while (currentClientsCount < MAX\_CLIENTS) { // Пока в массиве клиентов есть место

BankClient newClient;

if (!inFile.getline(newClient.allData, sizeof(newClient.clientInfo.fullName), ',')) break; // Пока получается получить строку из файла

inFile.getline(newClient.allData + sizeof(newClient.clientInfo.fullName), sizeof(newClient.clientInfo.accountType), ','); //записываем строку в поле newClient.clientInfo.accountType, используя общий массив newClient.allData и добавляя к нему начальные индексы

(inFile >> newClient.clientInfo.accountNumber);

inFile.ignore(); // пропускаем символ ',' в файле

(inFile >> newClient.clientInfo.accountBalance);

inFile.ignore(); // пропускаем символ ',' в файле

inFile.getline(newClient.allData + sizeof(newClient.clientInfo.fullName) + sizeof(newClient.clientInfo.accountType), sizeof(newClient.clientInfo.lastModifiedDate));

clients[currentClientsCount++] = newClient;

}

vi "Данные успешно прочитаны из файла."; nl

inFile.close();

}

void writeToFile() {

// Открытие файла для записи

ofstream outFile("bank\_clients.txt");

// Проверка успешности открытия файла

if (!outFile) {

cerr << "Ошибка открытия файла для записи."; nl

ret;

}

// Запись данных каждого клиента в файл, разделяя поля запятой

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

outFile << clients[i].clientInfo.fullName << ","

<< clients[i].clientInfo.accountType << ","

<< clients[i].clientInfo.accountNumber << ","

<< clients[i].clientInfo.accountBalance << ","

<< clients[i].clientInfo.lastModifiedDate <<"\n";

}

// Вывод сообщения о успешной записи данных в файл

vi "Данные успешно записаны в файл."; nl

// Закрытие файла после завершения записи

outFile.close();

}

