Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин, систем и сетей

Дисциплина: Основы алгоритмизации и программирования

Отчет

к учебной практике(ознакомительной)

на тему

«Услуги банка»

БГУИР КР 1-40 02 01 111 ПЗ

Студент А. Н. Климович

Руководитель А. М. Ковальчук

МИНСК 2022

**СОДЕРЖАНИЕ**

Введение………………………………………………………………………..3

1. Входные и выходные данные……………………………………………...4
2. Разработка программных модулей………………………………………...5

2.1. Разработка схемы алгоритма main функции…………………………5

2.2. Алгоритм по шагам для функции add\_person()………………………6

2.3. Алгоритм по шагам для функции del\_person()………………...…….7

3. Код программы……………………………………………………………...8

4. Результаты работы…………………………………………………………21

5. Системные и программные требования…………………………..…..…..27

6. Заключение…………………………………………………………………28

7. Список литературы………………………………………………………...29

8. Приложение А. Блок-схема функции main..……………………………...30

**ВВЕДЕНИЕ**

Данная работа посвящена разработке программы по оказанию некоторых услуг, которые может предоставить банк (просмотр остатка денежных средств на счете, получение денежных средств, пополнение счета, перевод денежных средств со счета на счет и др.).

Необходимо построить схему алгоритма головной функции, а также разработать алгоритмы по шагам для двух произвольных функций.

Предложенную программу было решено писать на языке программирования высокого уровня – языке Си.

Сегодня язык Си стал одним из наиболее важных и популярных языков программирования. Упомянем лишь некоторые достоинства этого языка.

Согласно данным рейтингов наиболее авторитетных исследовательских платформ Tiobe, GitHub и IEEE можно сказать, что на 2022 год язык Си занимает 3-е место. Стоит упомянуть некоторые достоинства этого языка:

* Эффективность. Его структура позволяет наилучшим образом использовать возможности современных ЭВМ. На языке Си программы обычно отличаются компактностью и быстротой исполнения;
* Удобство. Этот язык достаточно структурирован, чтобы поддерживать хороший стиль программирования, и вместе с тем не связывать вас смирительной рубашкой ограничений;
* Мощность и гибкость. Язык имеет широкую известность и наличие компиляторов для очень большого числа платформ, а также код, написанный на языке Си легко переписывается на другие языки;
* Современность. Программы, написанные на Си, используются для решения физических и технических проблем и даже для производства мультипликационных фильмов.

Кроме уже упомянутых достоинств язык Си имеет и некоторые другие, а также, безусловно, ряд недостатков, в которые не будем углубляться.

**ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ**

#define len 15 //длина вводимых слов

struct currency //структура, хранящая валюту человека

{

int dollar; //доллары

int euro; //евро

int RUB; //российские рубли

int BYN; //белорусские рубли

};

struct info //структура, хранящая информацию о человеке

{

char name[len + 1]; //имя

char surname[len + 1]; //фамилия

int PIN; //PIN-код пользователя

currency cash[4]; //структура, хранящая валюту человека

info\* next; //указатель на следующую структуру

};

Входные данные:

\*head, \*tail – указатели на голову и хвост очереди соответственно, в которой хранятся структуры информацией о людях типа info; \*f – указатель на рабочий файл с информацией о людях; filename[len+1] – имя рабочего файла.

Промежуточные данные:

с – переменная для выбора режима работы программы.

Выходные данные:

\*head, \*tail – указатели на голову и хвост очереди соответственно, в которой хранятся структуры информацией о людях типа info; \*f – указатель на рабочий файл с информацией о людях.

**РАЗРАБОТКА СХЕМЫ ГОЛОВНОЙ ФУНКЦИИ**

Блок-схема функции main приведена в **приложении А**.

**АЛГОРИТМ ПО ШАГАМ ДЛЯ ФУНКЦИИ add\_person()**

Данная функция добавляет человека в очередь людей и записывает его в рабочий файл.

1. Начало.
2. Объявление данных.

Входные: tail, head – указатели на хвост и голову соответственно очереди типа info; fileName – имя рабочего файла; f – указатель на рабочий файл.

Промежуточные: pointer – указатель на добавляемый элемент в очередь и файл.

1. Бесконечный цикл, выход из которого осуществляется, если функция offer() при своем вызове вернула нулевое значение.
2. Вывод приглашения для ввода имени человека: “Введите имя”.
3. Вызов функции input\_str() для ввода имени человека.
4. Вывод приглашения для ввода фамилии человека: “Введите фамилию”.
5. Вызов функции input\_str() для ввода фамилии человека.
6. Начальный PIN-код pointer->PIN = 1234;
7. Если tail = nullptr или head = nullptr, то tail = head = pointer, а иначе перейти на шаг 10.
8. pointer->next = tail;

tail = pointer;

1. Вывод приглашения для продолжения добавления людей: “Продолжить?(y\n)”.
2. Вызов функции offer(), чтобы продолжить добавлять людей или завершить операцию добавления.
3. Если функция offer() вернула нулевое значение, то вызвать функцию struct\_in\_file(), чтобы добавить новых людей в рабочий файл, и перейти на шаг 15, а иначе перейти на шаг 14.
4. Конец бесконечного цикла.
5. Конец.

**АЛГОРИТМ ПО ШАГАМ ДЛЯ ФУНКЦИИ del\_person()**

Данная функция удаляет информационную структуру о человеке из очереди структур типа info.

1. Начало.
2. Объявление данных.

Входные: tail, head – указатели на хвост и голову соответственно очереди типа info; element – указатель на удаляемую структуру из очереди типа info;

Промежуточные: pointer = tail, prev\_pointer = tail – указатели на элемент очереди типа info и на элемент предшествующий данному.

1. Если element ≠ nullptr, то перейти на шаг 4, а иначе перейти на шаг 14.
2. Если tail = nullptr или head = nullptr, то перейти на шаг 14, а иначе перейти на шаг 5.
3. Если tail = head, то перейти на шаг 6, а иначе перейти на шаг 7.
4. free(tail) – освобождение памяти от удаляемого элемента;

head = tail = nullptr. Перейти на шаг 14.

1. Цикл по очереди типа info, пока pointer ≠ nullptr и pointer ≠ element.
2. prev\_pointer = pointer;

pointer = pointer->next;

1. Конец цикла по очереди типа info.
2. Если pointer ≠ nullptr, то перейти на шаг 11, а иначе перейти на шаг 14.
3. Если tail = pointer, то tail = pointer->next, а иначе перейти на шаг 12;
4. Если head = pointer, то head = prev\_pointer, а иначе перейти на шаг 13;
5. prev\_pointer = pointer->next;

free(pointer) – освобождение памяти от удаляемого элемента.

1. Конец.

**КОД ПРОГРАММЫ**

#include <locale.h> //подключение библиотек

#include <stdio.h>

#include <stdlib.h>

#include <Windows.h>

#include <conio.h>

void instructions(); //функция выводит инструкции по работе программы

int input\_int(); //функция ввода целого числа

char\* input\_str(); //функция ввода строки

info\* input\_PIN(info\* tail); //функция ввода PIN-кода

int currency\_choice(); //функция выбора валюты

int sum\_choice(); //функция выбора суммы

int offer(); //функция продолжения выполнения программы

void clean(info\*\* tail); //функция очистки очереди

//функция пополнения\снятия со счета денежных средств

void get\_or\_push\_money(info\*, info\*, char\*, FILE\* &f, int);

int get\_push\_fun(int, int sum, info\*&, int k); //функция изменения суммы денег на счете

void show\_balance(info\*, info\*, FILE\* f, char\*); //функция просмотра баланса

void add\_person(info\*\*, info\*\*, char\*, FILE\* f); //функция добавления человека в систему

void del\_person(info\*\*, info\*\*, info\* &element); //функция удаления человека из системы

int del\_in\_file(FILE \*&f, info \*&, info\* &, char\*); //функция удаления человека из файла

void changePIN(info\*, info\*, char\*, FILE\* f); //функция смены PIN-кода

//функция перевода денежных средств с одного счета на другой

void transfer\_money(info\*, info\*, char\*, FILE\* f);

//функция изменения сумм на счете отправителя и получателя

int transfer\_fun(int sum, info\*&, info\*&, int k);

void struct\_in\_file(char\*, FILE \*&f, info\*); //функция записи структуры в текстовый файл

int file\_in\_str(info\*&, info\*&, char\*, FILE \*f); //функция переписи информации из файла

void show\_file(FILE \*f, char\*, info\*, info\*); //функция просмотра файла

void replace\_file(info\*&tail, char\*, FILE\* &f); //функция замены файла

int main()

{

SetConsoleCP(1251); //русификация консоли

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Rus");

info\* head = NULL, \*tail = NULL;

FILE\* f = NULL; char c, fileName[len+1];

printf("%c[%d;%dmВведите название файла:%c[%dm\n", 27, 1, 34, 27, 0);

strcpy\_s(fileName, input\_str()); //вызов функции ввода имени файла

while (1)

{

instructions(); //вызов функции вывода иструкций по работе программы

c = \_getch(); //ввод значения

system("cls"); //очистка консоли

switch (c)

{

case '1': //вызов функции добавления человека из файла и очереди

add\_person(&head, &tail, fileName, f);

break;

case '2': //вызов функции удаления человека из файла и очереди

del\_in\_file(f, tail, head, fileName);

break;

case '3': //вызов функции просмотра остатка денежных средств на счете

show\_balance(tail, head, f, fileName);

break;

case '4': //вызов функции просмотра информации файла

show\_file(f, fileName, head, tail);

break;

case '5': //вызов функции получения наличных денежных средств

get\_or\_push\_money(tail, head, fileName, f, 0);

break;

case '6': //вызов функции пополнения счета наличными денежными средствами

get\_or\_push\_money(tail, head, fileName, f, 1);

break;

case '7': //вызов функции перевода денежных средств со счета на счет

transfer\_money(head, tail, fileName, f);

break;

case '8': //вызов функции смены ПИН - кода дебетовой карточки

changePIN(head, tail, fileName, f);

break;

case '9':

printf("Введите название файла:\n");

strcpy\_s(fileName, input\_str()); //вызов функции смены файл

break;

case '0':

clean(&tail); //вызов функции очистки очереди

return 0; //завершение работы программы

default:

printf("Ошибка выбора операции! Пожалуйста, повторите снова\n");

continue;

}

}

}

//функция добавления человека в очередь и файл

void add\_person(info\*\* tail, info\*\* head, char\* fileName, FILE\* f)

{

while (1)

{

//указатель на добавляемую структуру и выделение под нее памяти

info\* pointer = (info\*)calloc(1, sizeof(info));

if (!pointer)

{

printf("Память не выделена!\n");

return;

}

printf("\nВведите имя:\n");

strcpy\_s(pointer->name, input\_str()); //ввод имени

printf("\nВведите фамилию:\n");

strcpy\_s(pointer->surname, input\_str()); //ввод фамилии

pointer->cash[0].dollar = pointer->cash[1].euro = pointer->cash[2].RUB = pointer->cash[3].BYN = 0; //обнуление всех валют

pointer->PIN = 1234; //начальный пин-код

printf("\nВаш PIN-код: %d", pointer->PIN);

if (!\*tail || !\*head)

\*tail = \*head = pointer; //если очередь не создана

else

{

pointer->next = \*tail; //добавление элемента в хвост очереди

\*tail = pointer; //корректировка хвоста

}

printf("\nПродолжить?(y\\n)\n");

if (!offer()) //предложение о продолжении добавления людей

{

//вызов фукнции записи структуры в текстовый файл

struct\_in\_file(fileName, f, \*tail);

return;

}

}

}

//функция смены пин-кода

void changePIN(info\* head, info\* tail, char\* fileName, FILE\* f)

{

info\* pointer; //указатель для прохода по очереди

//вызов функции записи информации из файла в очередь

file\_in\_str(tail, head, fileName, f);

if (pointer = input\_PIN(tail)) //вызов функции входа в учетную запись

{

printf("Введите новый PIN-код\n");

pointer->PIN = input\_int(); //вызов функции ввода нового пин-кода

replace\_file(tail, fileName, f); //вызов функции замены файла

printf("PIN-код успешно изменен!\n");

}

return;

}

//функция меню по удалению человека из файла и очереди

int del\_in\_file(FILE \*&f, info \*&tail, info\* &head, char\* fileName)

{

info\* element = NULL; //указатель на удаляемый элемент

//вызов функции записи информации из файла в структуры

if (file\_in\_str(tail, head, fileName, f))

{

element = input\_PIN(tail); //вызов функции входа в учетную запись

//вызов функции удаления человека из очереди

del\_person(&tail, &head, element);

replace\_file(tail, fileName, f); //вызов функции замены файла

}

else printf("Файл пуст\n");

return 0;

}

//функции удаления человека из очереди

void del\_person(info\*\* tail, info\*\* head, info\* &element)

{

info \*pointer = \*tail, //указатель для обхода очереди

//указатель на предыдущий элемент по отношению к pointer

\*prev\_pointer = \*tail;

if (element)

{

if (!\*tail || !\*head) //очередь не создана

{

puts("База данных пуста!");

return;

}

if (\*tail && \*tail == \*head) //в очереди один элемент

{

free(\*tail); //освободить память

\*head = \*tail = NULL; //зануление указателей

return;

}

//цикл по очереди, пока не нашли требуемый элемент

while (pointer && pointer != element)

{

prev\_pointer = pointer; //prev-pointer на предыдущий элемент

pointer = pointer->next; //pointer на следующий элемент

}

if (pointer) //если нашелся требуемый элемент

{

if (\*tail == pointer) //если удаляемый элемент - это хвост очереди

\*tail = pointer->next;

if (\*head == pointer) //если удаляемый элемент - это голова очереди

\*head = prev\_pointer;

else

{

prev\_pointer = pointer->next; //обход элемента

free(pointer); //удаление элемента

}

}

}

}

//функция переписи информации из файла в очередь

int file\_in\_str(info\* &tail, info\* &head, char\* name, FILE \*f)

{

errno\_t ff = fopen\_s(&f, name, "r+t"); //открытие файла для чтения

if (ff) //проверка на открытие файла

{

printf("Файл не может быть открыт!\n");

return 0;

}

int size = 0, //количество строк в файле

length = \_filelength(\_fileno(f)); //длина файла

size = length / sizeof(info); //вычисление количества структур в файле

if (size) //файл не пустой

{

//цикл по переписи информации из файла

for (int i = 0; i < size && !feof(f); i++)

{

info\* pointer = (info\*)calloc(1, sizeof(info)); //выделение памяти

if (!pointer) //если память не выделена

{

printf("Память не выделена!\n");

exit(1);

}

char c = 0; //временная переменная для хранения символа

for (int i = 0; i < len; i++) //цикл по переписи имени

{

fscanf\_s(f, "%c", &c); //считывание символа

if (c != ' ' && c != '\0') //запись символа в строку

pointer->name[i] = c;

}

for (int i = 0; i < len; i++) //цикл по переписи фамилии

{

fscanf\_s(f, "%c", &c); //считывание символа

if (c != ' ' && c != '\0') //запись символа в строку

pointer->surname[i] = c;

}

fscanf\_s(f, "%d%d%d%d%d\n", &pointer->PIN, &pointer->cash[0].dollar,

&pointer->cash[1].euro, &pointer->cash[2].RUB, &pointer->cash[3].BYN);

if (!tail || !head) //если очередь не создана

tail = head = pointer;

else

{

pointer->next = tail; //добавление элемента в очередь

tail = pointer; //корректировка хвоста

}

}

}

else puts("Файл пуст");

fclose(f); //закрытие файла

return size; //вернуть количество структур

}

//функция пополнения/снятия денег

void get\_or\_push\_money(info\* tail, info\* head, char\* fileName, FILE\* &f, int choice)

{

info\* pointer = tail; //указатель на искомый элемент

//вызов функции переписи информации из файла в очередь

file\_in\_str(tail, head, fileName, f);

if (pointer = input\_PIN(tail)) //вызов функции входа в учетную запись

{

int temp = 0, //временная переменная

k = currency\_choice(); //вызов функции выбора валюты

//вызов функции изменения суммы денег на счете

temp = get\_push\_fun(choice, sum\_choice(), pointer, k);

if (!temp && !choice) //если недостаточно денежных средств на счете

printf("Недостаточно денежных средств на балансе!\n");

else replace\_file(tail, fileName, f); //вызов фукнции замены файла

return;

}

else return;

}

//функция изменения суммы денег на счете

int get\_push\_fun(int choice, int sum, info\* &pointer, int k)

{

if (choice) //пополнение счета

pointer->cash[k].dollar += sum; //изменение суммы на счете

else //снятие денег со счета

{

if (pointer->cash[k].dollar -= sum) //денег достаточно

return 1;

else //денег недостаточно

pointer->cash[k].dollar += sum;

}

return 0;

}

//функция записи структуры в текстовый файл

void struct\_in\_file(char\* name, FILE \*&f, info\* pointer)

{

errno\_t ff = fopen\_s(&f, name, "a+t"); //открытие файла для записи информации

if (ff) //проверка на открытие файла

{

printf("%c[%d;%dm\nФайл не может быть открыт!%c[%dm\n", 27, 1, 31, 27, 0);

return;

}

while (pointer) //цикл по записыванию информации в файл

{

fprintf(f, "%-15s%-15s", pointer->name, pointer->surname);

fprintf(f, "%-10d%-10d%-10d%-10d%-10d\n",

pointer->PIN, pointer->cash[0].dollar,

pointer->cash[1].euro, pointer->cash[2].RUB, pointer->cash[3].BYN);

pointer = pointer->next; //переход к следующей структуре очереди

}

fclose(f); //закрытие файла

}

//функция просмотра баланса человека

void show\_balance(info\* tail, info\* head, FILE\* f, char\* fileName)

{

info\* pointer = tail; //указатель для прохода по очереди

//вызов функции переписи информации из файла в очередь

if (!file\_in\_str(tail, head, fileName, f))

{

puts("Файл пуст");

return;

}

if (pointer = input\_PIN(tail)) //вызов функции входа в учетную запись

{

int k = 0; //счётчик для рисовки таблицы

printf(" ");

while (k++ < 75) printf("\_"); //рисовка верхней границы таблицы

k = 71;

//вывод "шапки" таблицы

printf("\n|Имя\t\t|Фамилия\t|Доллары |");

printf(("Евро |Рос. рубли|Бел. рубли|\n|")

//цикл по подрисовке верхней границы таблицы

while (k-- > 1) {

if (k == 10 || k == 20 || k == 30 || k == 40 || k == 55) printf("|");

printf("\_");

}

printf("|");

//заполнение таблицы информацией

printf("\n|%-15s|%-15s|%-10d|%-10d|%-10d|%-10d|\n|",

pointer->name, pointer->surname, pointer->cash[0].dollar,

pointer->cash[1].euro, pointer->cash[2].RUB, pointer->cash[3].BYN);

//рисовка нижней границы таблицы

while (k++ < 70)

{

if (k == 16 || k == 31 || k == 41 || k == 51 || k == 61) printf("|");

printf("\_");

}

puts("|");

}

else return;

}

//функция очистки очереди

void clean(info\*\* tail)

{

info\* pointer; //временный указатель на структуру типа info

while (\*tail) //очистка очереди

{

pointer = \*tail; //указатель на хвост

\*tail = (\*tail)->next; //переместить хвост

free(pointer); //очистить память от элемента

}

}

//функция просмотра файла

void show\_file(FILE \*f, char\* name, info\* head, info\* tail)

{

errno\_t ff = fopen\_s(&f, name, "r+t"); //открытие текстового файла для чтения

if (ff) //ошибка открытия файла

{

printf("\nФайл не может быть открыт!\n");

return;

}

//длина файла и количество структур соответственно

int length = \_filelength(\_fileno(f)), size = length / sizeof(info);

rewind(f); //сбрасываем УТПФ

if (size)

{

int k = 0; //счетчик

printf(" ");

while (k++ < 86) printf("\_"); //рисовка верхней границы таблицы

printf("\n|Имя\t\t|Фамилия |ПИН-код |Доллары |Евро |Рос.

рубли|Бел. рубли|\n|"); //вывод "шапки"

k = 81;

while (k-- > 1) //подрисовка верхней границы таблицы

{

if (!(k % 10) || k == 65) printf("|");

printf("\_");

}

info\* pointer = NULL; //указатель на структуру

for (;!feof(f);)

{ //выделение памяти под структуру

if (!(pointer = (info\*)calloc(1, sizeof(info))))

{

printf("\nПамять не выделена!\n");

exit(1);

}

fgets(pointer->name, len+1, f); //считывание имени

fgets(pointer->surname, len+1, f); //считывание фамилии

fscanf\_s(f, "%d%d%d%d%d\n", &pointer->PIN, &pointer->cash[0].dollar,

&pointer->cash[1].euro, &pointer->cash[2].RUB, &pointer->cash[3].BYN);

printf("|\n|%-15s|%-15s|%-10d|%-10d|%-10d|%-10d|%-10d",

pointer->name, pointer->surname, pointer->PIN, pointer->cash[0].dollar,

pointer->cash[1].euro, pointer->cash[2].RUB, pointer->cash[3].BYN);

free(pointer);

}

printf("|\n|");

while (k++ < 80) //рисовка нижней границы таблицы

{

if (k == 16 || (!((k-1) % 10) && k > 21)) printf("|");

printf("\_");

}

puts("|");

}

else puts("Файл пуст");

fclose(f); //закрытие файла

}

//функция перевода денег со счета на счет

void transfer\_money(info\* head, info\* tail, char\* fileName, FILE\* f)

{

char name2[len], surname2[len]; //строки для имени и фамилии получателя

//вызов функции переписи информации из файла в очередь

file\_in\_str(tail, head, fileName, f);

info\* pointer1, \*pointer2 = tail; //указатель на отправителя и получателя

printf("\nВведите данные отправителя перевода");

if (pointer1 = input\_PIN(tail)) //вызов функции входа в учетную запись

{

while (1)

{

printf("\nВведите данные получателя перевода");

printf("\nВведите имя:\n");

strcpy\_s(name2, input\_str()); //вызов функции ввода имени получателя

printf("\nВведите фамилию:\n");

strcpy\_s(surname2, input\_str()); //вызов функции ввода фамилии

while (pointer2) //цикл по очереди, пока не структура получателя

{

if (!strcmp(pointer2->name, name2) //требуемый элемент нашелся

&& !strcmp(pointer2->surname, surname2)) break;

pointer2 = pointer2->next; //переход к следующему элементу

}

if (pointer2) //если требуемый элемент нашелся

{

int temp = 0, //временная переменная

k = currency\_choice(); //вызов функции выбора валюты

//вызов функции изменения сумм на счете отправителя и получателя

temp = transfer\_fun(sum\_choice(), pointer1, pointer2, k);

if (!temp) //на счете отправителя недостаточно денег

{

printf("Недостаточно денежных средств на балансе!\n");

return;

}

else //вызов функции замены файла

replace\_file(tail, fileName, f);

return;

}

printf("\nНе удалось найти такого человека!\nПовторите ввод\n");

}

}

else return;

}

//функция изменения сумм на счете отправителя и получателя

int transfer\_fun(int sum, info\* &pointer1, info\* &pointer2, int k)

{

if ((pointer1->cash[k].BYN -= sum)>=0) //на счете отправителя достаточно денег

{

pointer2->cash[k].BYN += sum; //изменение суммы счета получателя

return 1;

}

else pointer1->cash[k].BYN += sum; //возвращение к начальной сумме

return 0;

}

//функция ввода целого числа

int input\_int()

{

int k, n, f = 0; char symbol; //промежуточные переменные для проверки и ввода

do

{

if (f++) printf("Ошибка ввода!\n");

rewind(stdin); //чистка буфера

k = scanf\_s("%d%c", &n, &symbol); //ввод значения

if (k > 1 && n>0 && symbol == '\n')

break; //если соблюдены все условия, то прервать цикл

} while (1);

return n; //вернуть вводимое число

}

//функция ввода строки

char\* input\_str()

{

int j = 0; //счетчик по строке

static char sl[len+1] = {}; //массив для записи слова

char c; //промежуточная переменная

do

{

rewind(stdin); //чистка буфера

c = \_getche(); //ввод символа

sl[j] = c; //если вводим имя файла

if (c == '\b') //нажата клавиша Backspace

{

if (j > 0) { printf(" \b \b"); j--; } //аналог клавиши Backspace

else printf(" ");

j--;

}

else if (c == '\r') break;//если нажат Enter

} while (j++ < len - 1);

sl[j] = '\0';

return sl; //вернуть строку

}

//функция продолжения выполнения программы

int offer()

{

char s; //объявление промежуточной переменной

for (;;)

{

rewind(stdin); //чистка буфера

s = \_getch(); //ввод y или n(выбор для продолжения или нет)

if (s == 'y') return 1; //выбрано продолжение программы

if (s == 'n') return 0; //выбрано завершение программы

else

printf("\nОшибка!\nХотите продолжить?(y\\n)\n"); //текст об ошибке

}

}

//функция входа в учетную запись

info\* input\_PIN(info\* tail)

{

info\* pointer = tail; //указатель для прохода по очереди

int attempt = 3; //попытки входа

char name[len+1], surname[len+1]; //имя, фамилия человека

while (1)

{

printf("Введите имя:\n");

strcpy\_s(name, input\_str()); //вызов функции ввода имени

printf("\nВведите фамилию:\n");

strcpy\_s(surname, input\_str()); //вызов функции ввода фамилии

//цикл по очереди, пока не нашелся требуемый элемент

while (pointer)

{

if (!strcmp(pointer->name, name) && //требуемый элемент нашелся

!strcmp(pointer->surname, surname)) break;

pointer = pointer->next; //переход к следующему элементу

}

if (pointer) break;

else

printf("\nТакой человек не зарегистрирован!\n");

pointer = tail; //переходим к началу очереди

return 0;

}

while (attempt) //цикл по вводу пин-кода, пока есть попытки

{

printf("\nВведите PIN-код:\n");

int pin = input\_int(); //вызов функции ввода пин-кода

if (pointer->PIN == pin) //если введен правильный пин-код

{

printf("\nДобро пожаловать, %s %s!\n",

pointer->name, pointer->surname, 27, 0);

return pointer;

}

else

{

--attempt; //уменьшение количетства попыток ввода

printf("\nНеверный PIN-код!\nПопробуйте ещё раз\n");

printf("Осталось %d попыток\n", attempt);

}

if (!attempt) //если попыток не осталось

return 0;

}

return 0;

}

//функция выбора валюты

int currency\_choice()

{

while (1) //бесконечный цикл, пока не выбрана валюта

{

printf("\nВыберите валюту:");

puts("\n1 - Американский доллар"

"\n2 - Евро"

"\n3 - Российский рубль"

"\n4 - Белорусский рубль");

switch (char c = \_getch())

{

case '1': return 0; //выбран доллар

case '2': return 1; //выбран евро

case '3': return 2; //выбран российский рубль

case '4': return 3; //выбран белорусский рубль

default: printf("\nОшибка выбора валюты! Пожалуйста, повторите снова\n");

continue; //продолжить цикл

} break; //прервать цикл

}

}

//функция выбора суммы

int sum\_choice()

{

while (1) //бесконечный цикл, пока не выбрана сумма

{

printf("\nВыберите сумму:");

puts("\n1 - 5"

"\n2 - 10"

"\n3 - 20"

"\n4 - 50"

"\n5 - другая сумма");

switch (char c = \_getch())

{

case '1': return 5; //выбрана сумма в 5 номиналов

case '2': return 10; //выбрана сумма в 10 номиналов

case '3': return 20; //выбрана сумма в 20 номиналов

case '4': return 50; //выбрана сумма в 50 номиналов

case '5': return input\_int(); //вызов функции ввода другой суммы

default: printf("\nОшибка выбора суммы! Пожалуйста, повторите снова\n");

continue; //продолжить цикл

} break; //прервать цикл

}

}

//функция замены файла

void replace\_file(info\*&tail, char\* fileName, FILE\* &f)

{

info \*p = NULL; //указатель для прохода по очереди

char fileCopyName[] = "tempfile"; //временное имя нового файла

struct\_in\_file(fileCopyName, f, tail); //вызов функции занесения очереди в файл

while (tail) { p = tail; tail = tail->next; free(p); } //удаление очереди

if (remove(fileName))

{

printf("Ошибка удаления старого файла!\n");

exit(1);

}

if (rename(fileCopyName, fileName)) //переименование нового файла

{

printf("Ошибка переименования файла!\n");

exit(1);

}

printf("Операция успешно проведена!\n");

return;

}

//функция вывода инструкций по работе программы

void instructions()

{

printf("\nВыберите услугу:");

puts("\n1 - добавить человека в базу данных"

"\n2 - удалить человека из базы данных"

"\n3 - просмотр остатка денежных средств на счете пользователя"

"\n4 - просмотр данных всех зарегистрированных пользователей"

"\n5 - получение денежных средств"

"\n6 - пополнение счета денежными средствами"

"\n7 - перевод денежных средств со счета на счет"

"\n8 - смена ПИН-кода дебетовой карточки"

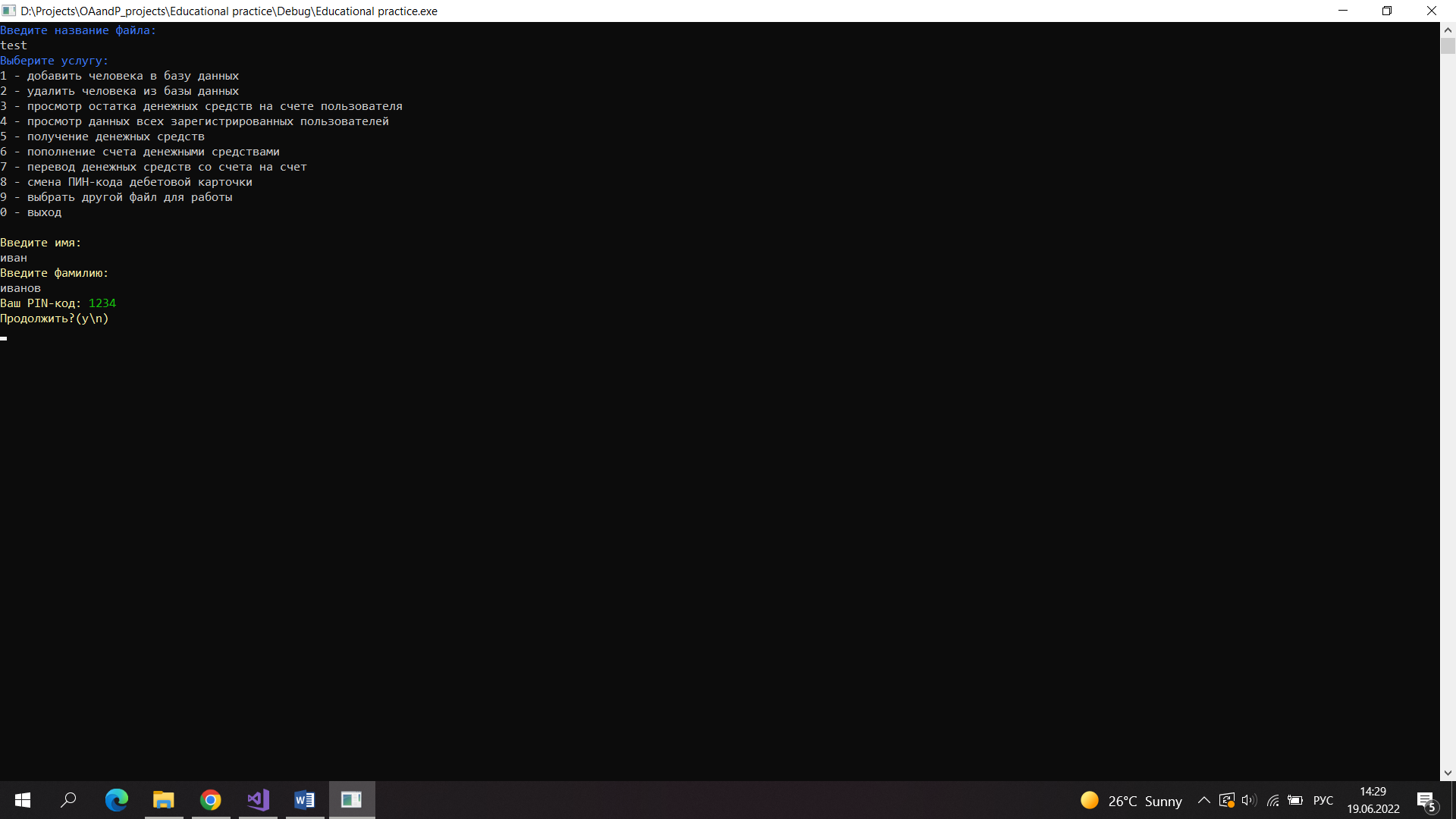
"\n9 - выбрать другой файл для работы"

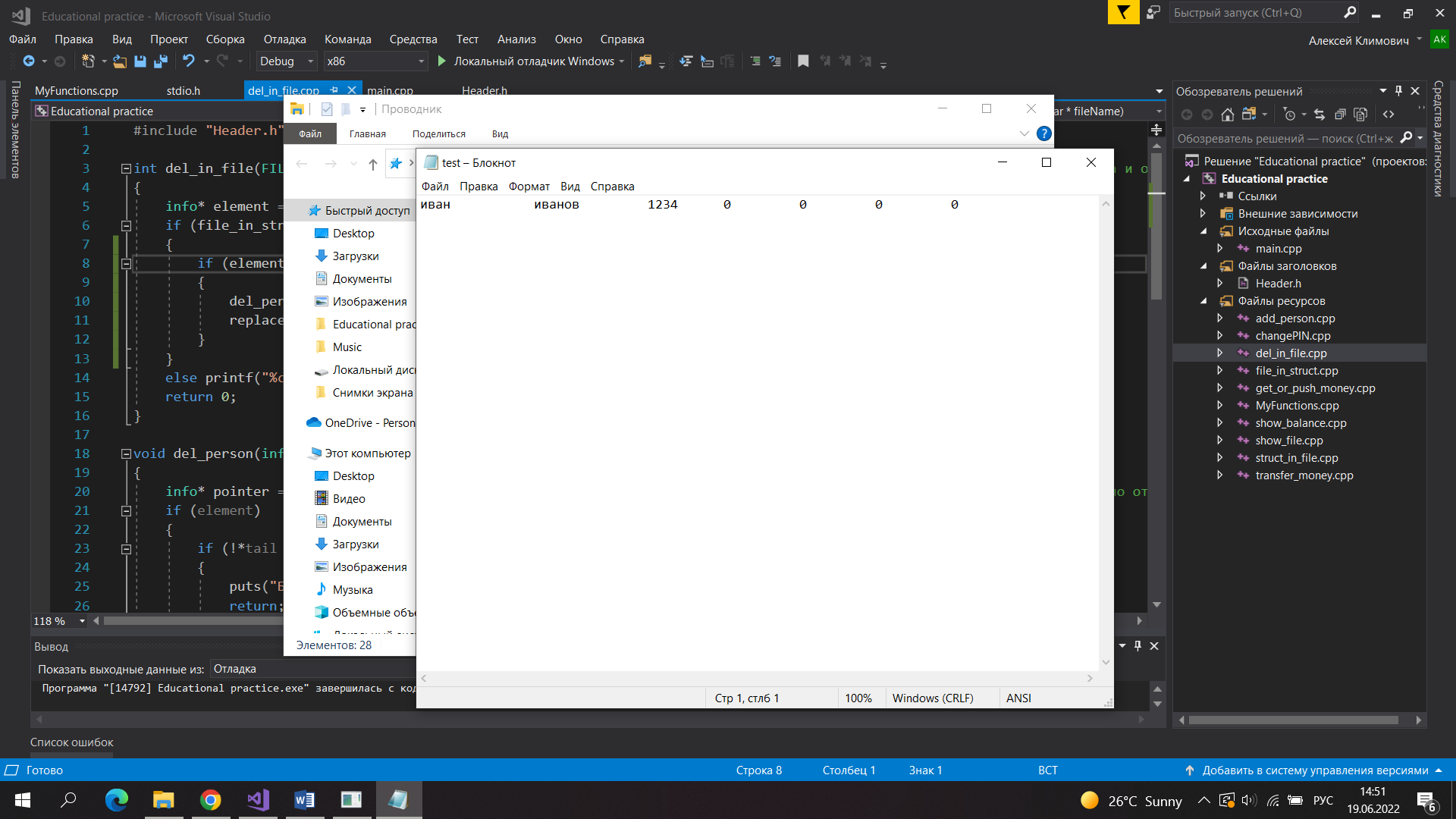
"\n0 - выход");

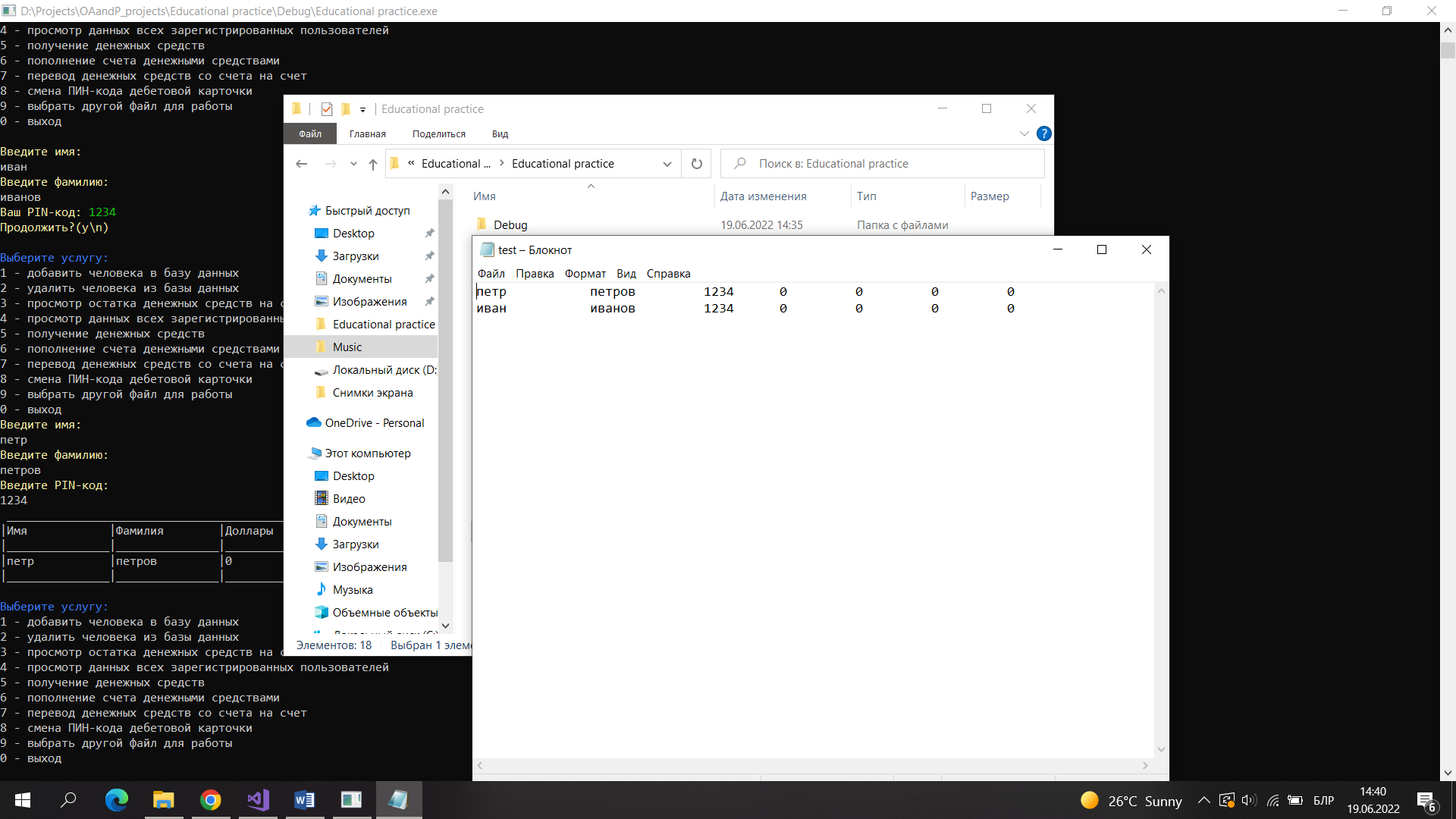
}

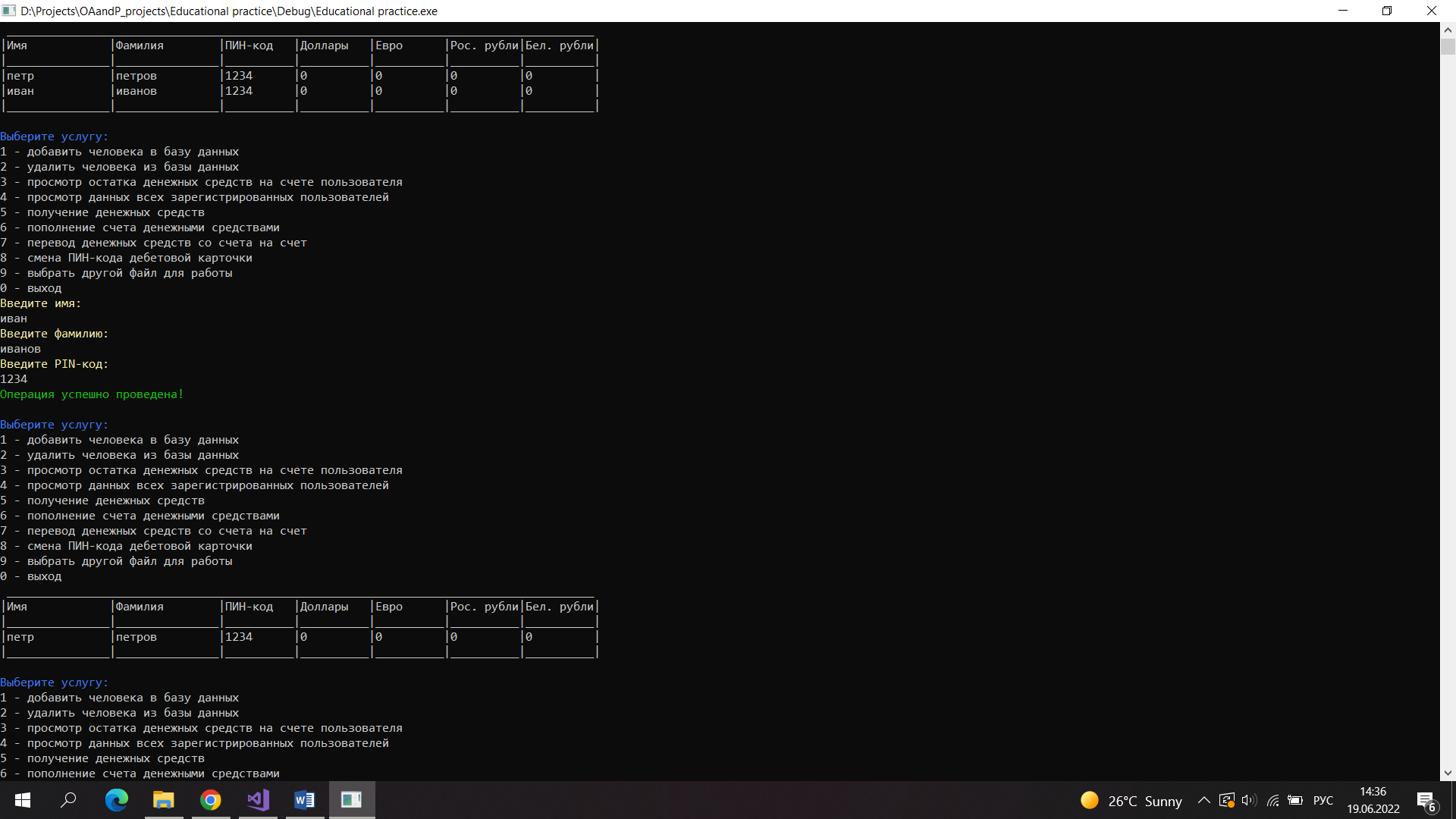
**РЕЗУЛЬТАТЫ РАБОТЫ**

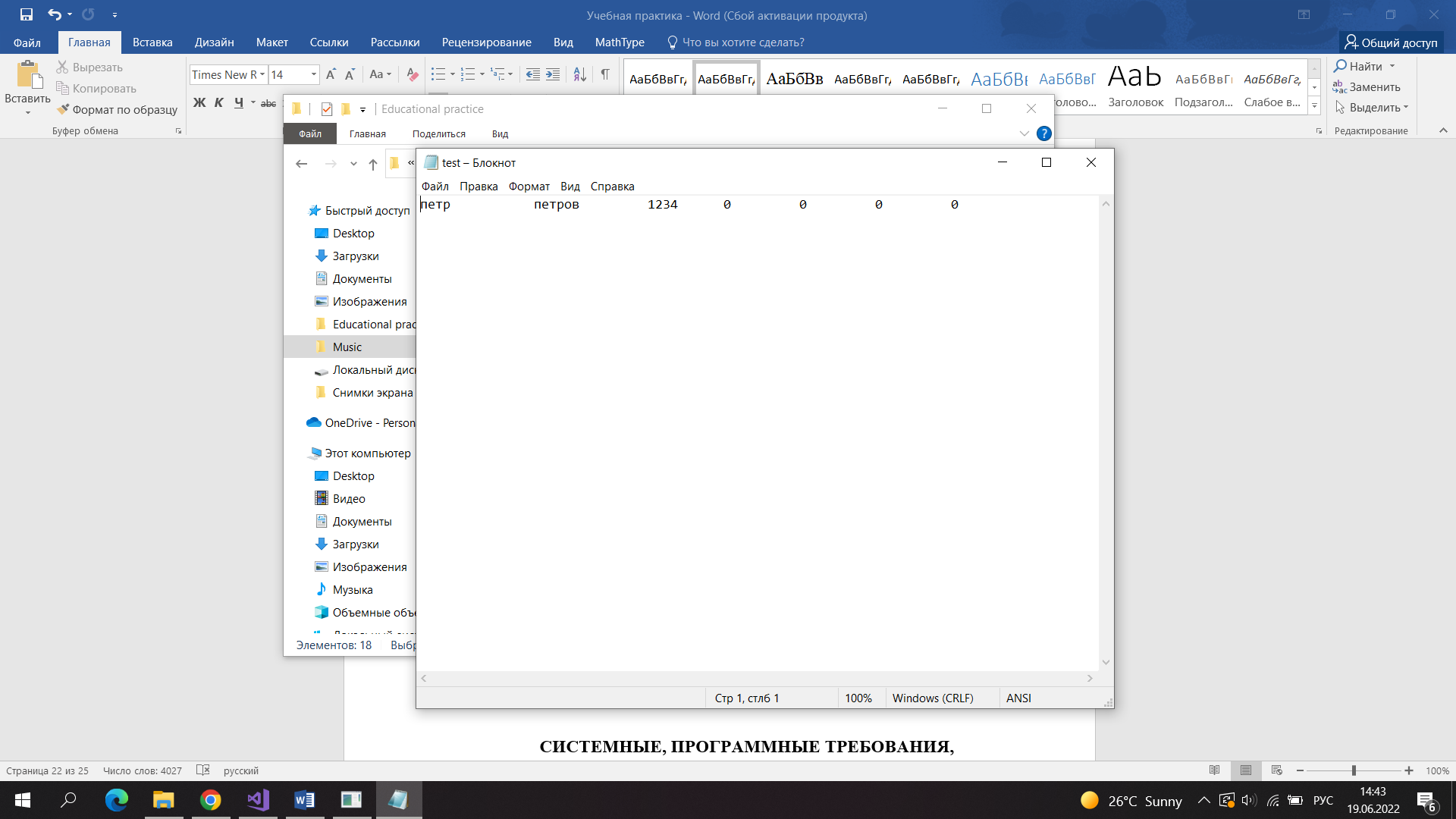
Рисунок 1.1. Пример работы функции добавления человека в базу данных.

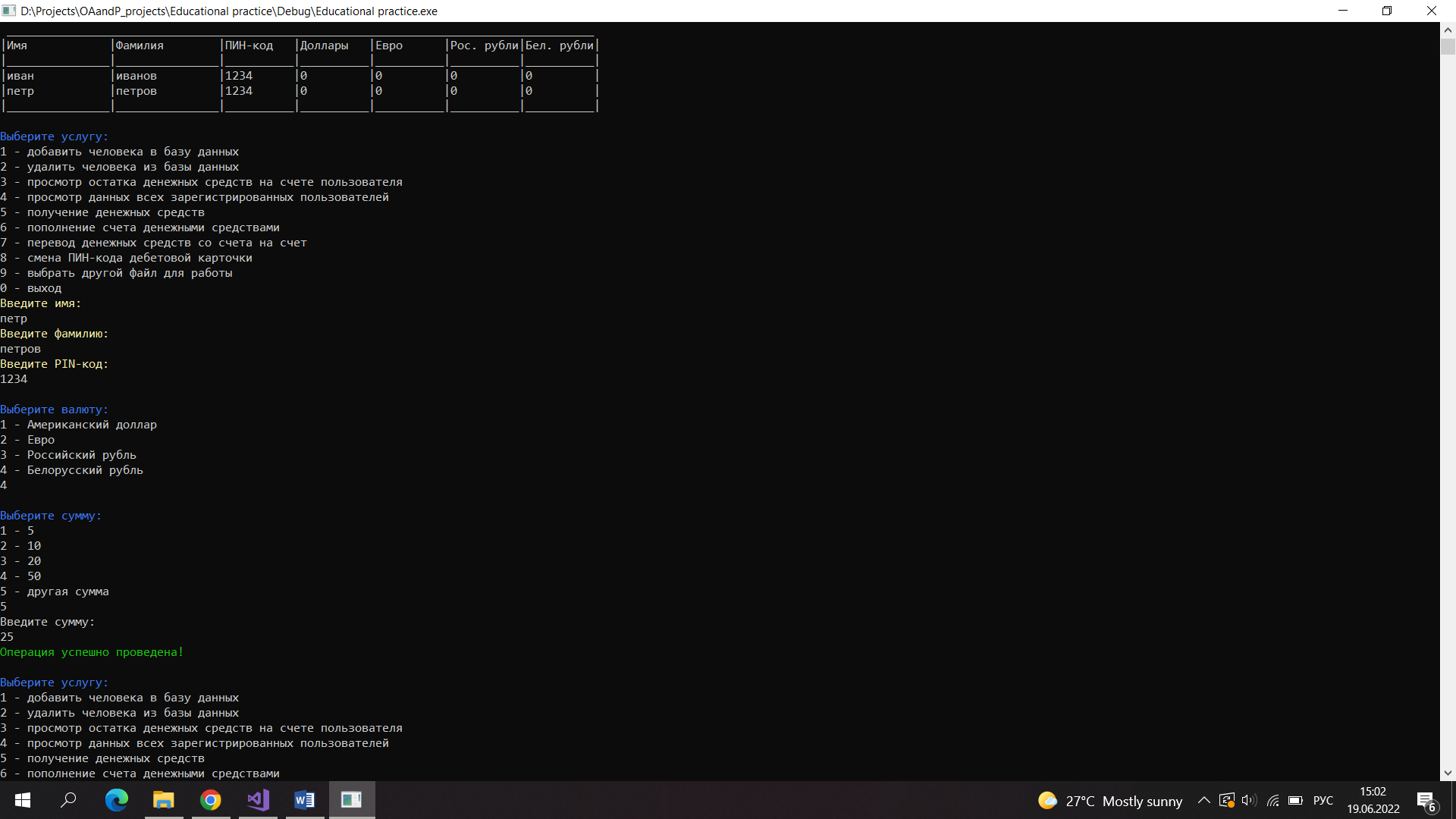


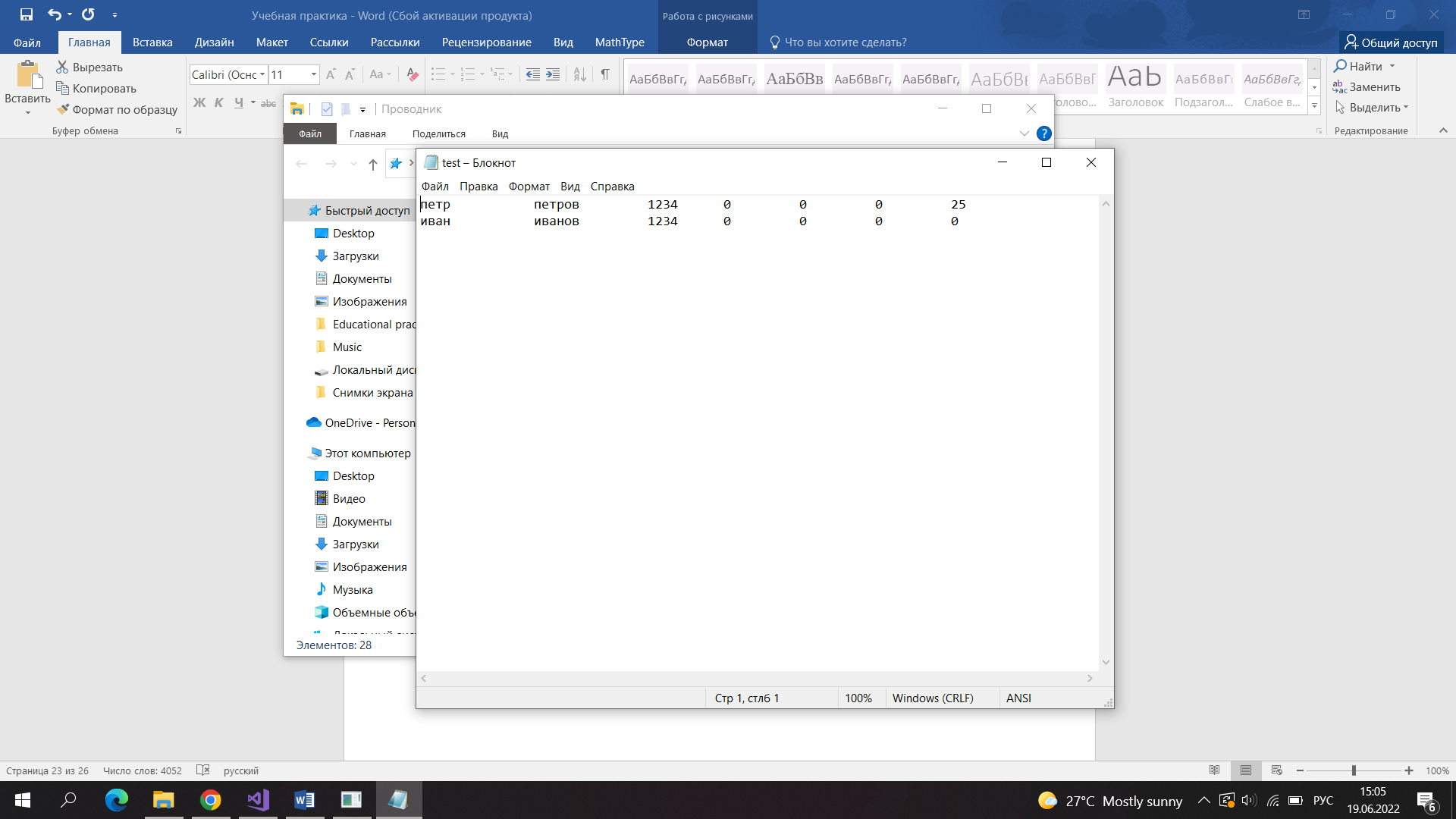
Рисунок 1.2. Рабочий файл после выполнения операции добавления.

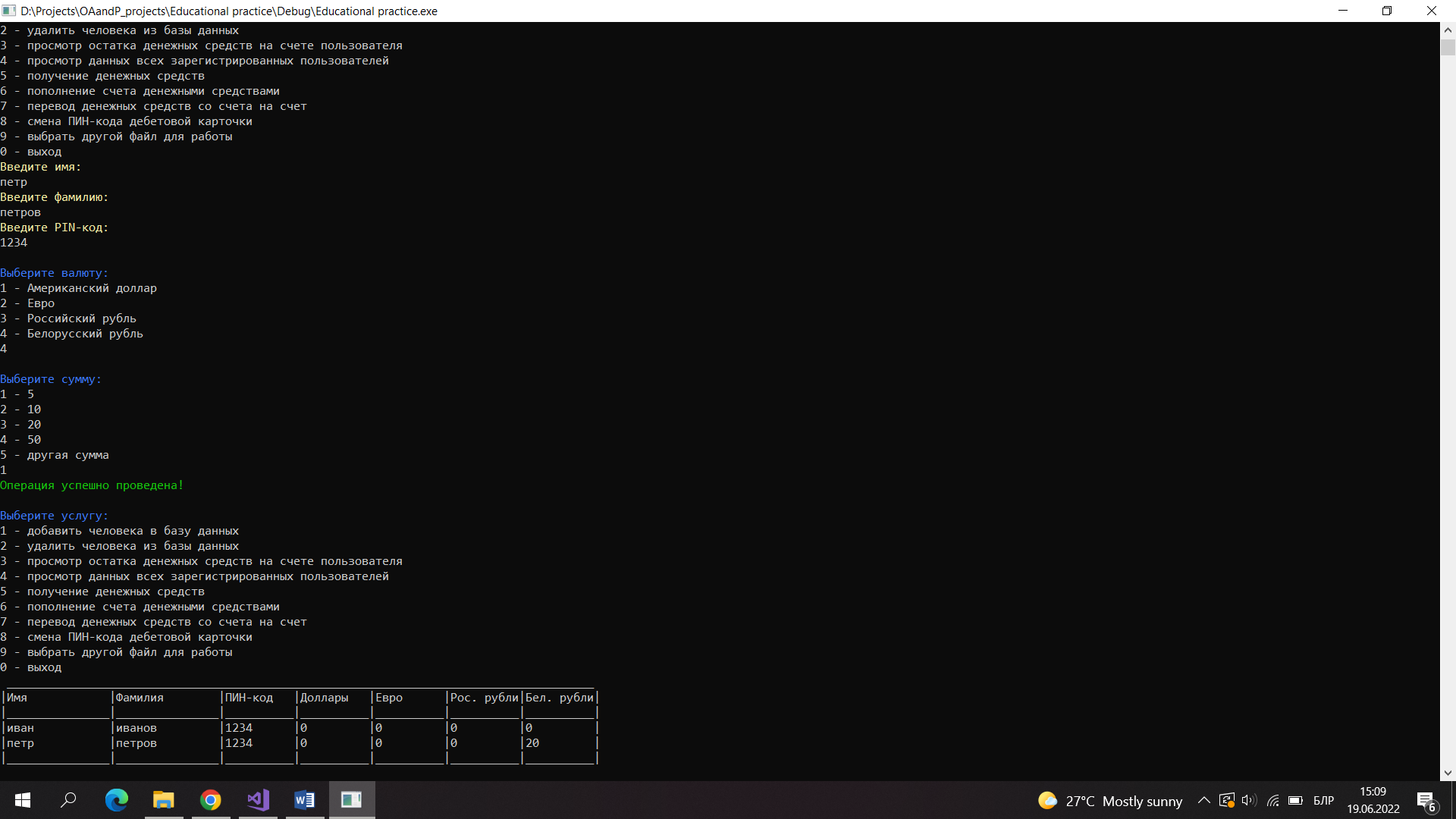
Рисунок 2.1. Рабочий файл до выполнения операции удаления.

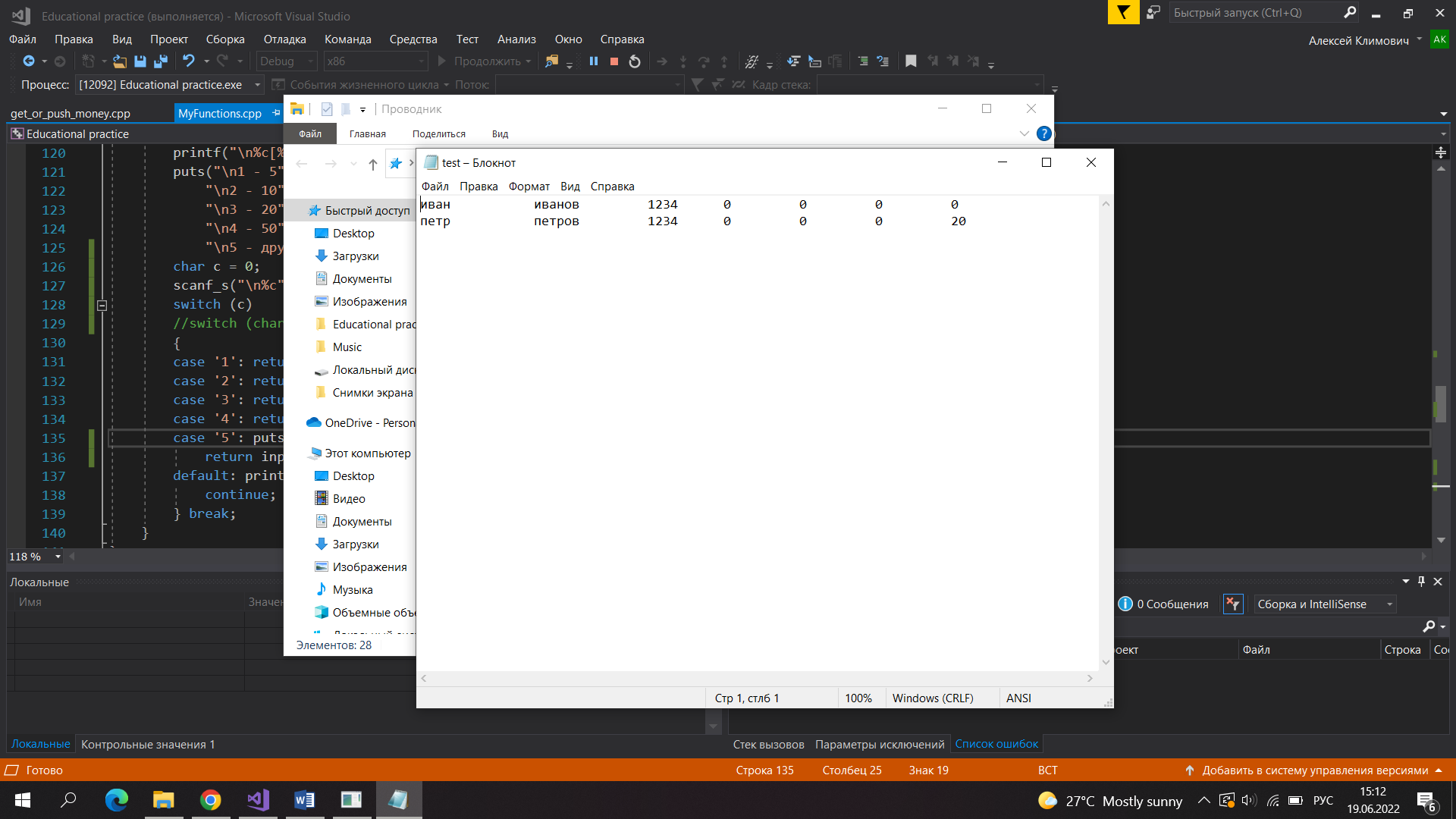
Рисунок 2.2. Пример работы функции удаления человека из базы данных.

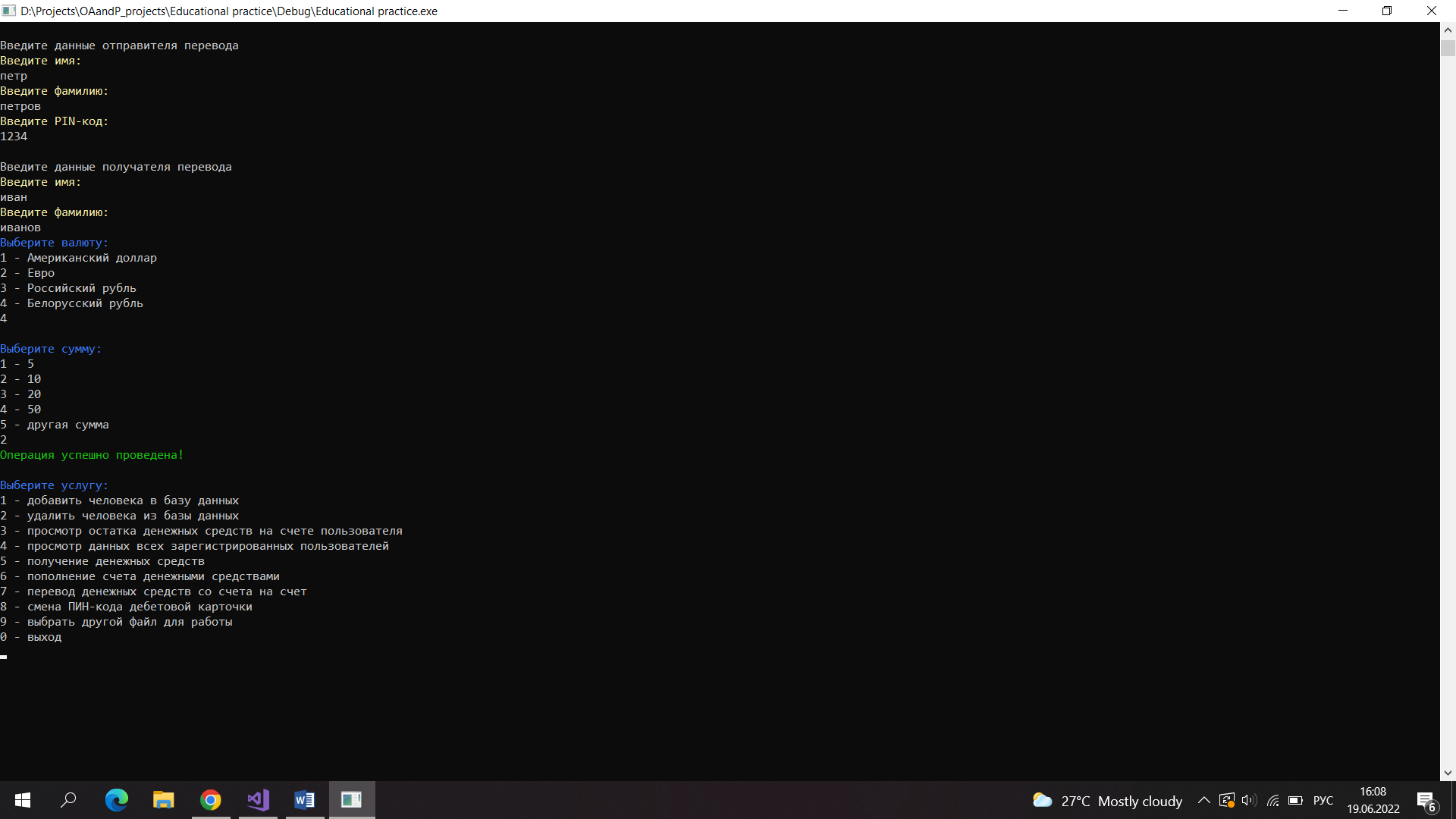
Рисунок 2.3. Рабочий файл после выполнения операции удаления.

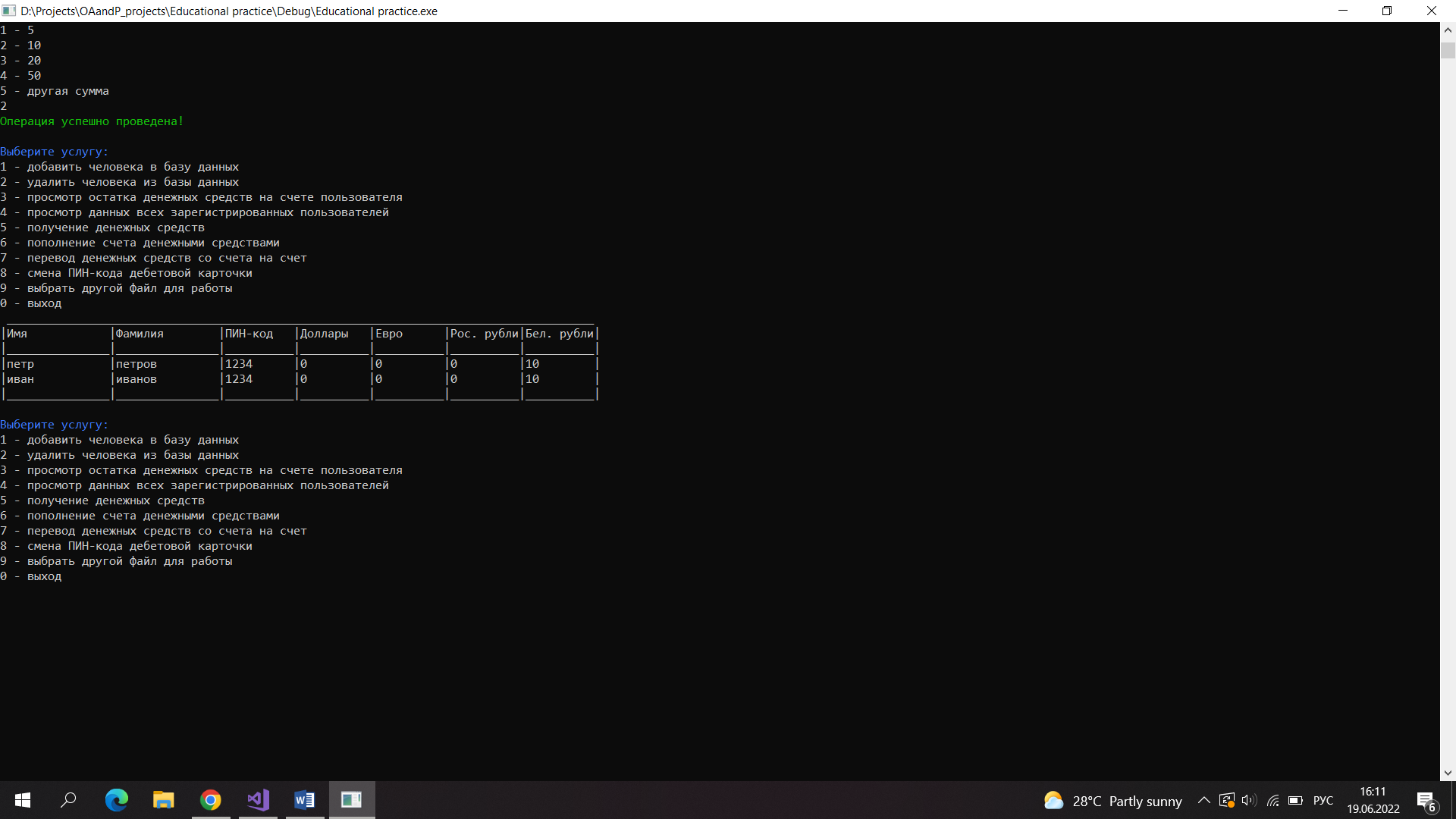
Рисунок 3.1. Пример работы операции пополнения счета.

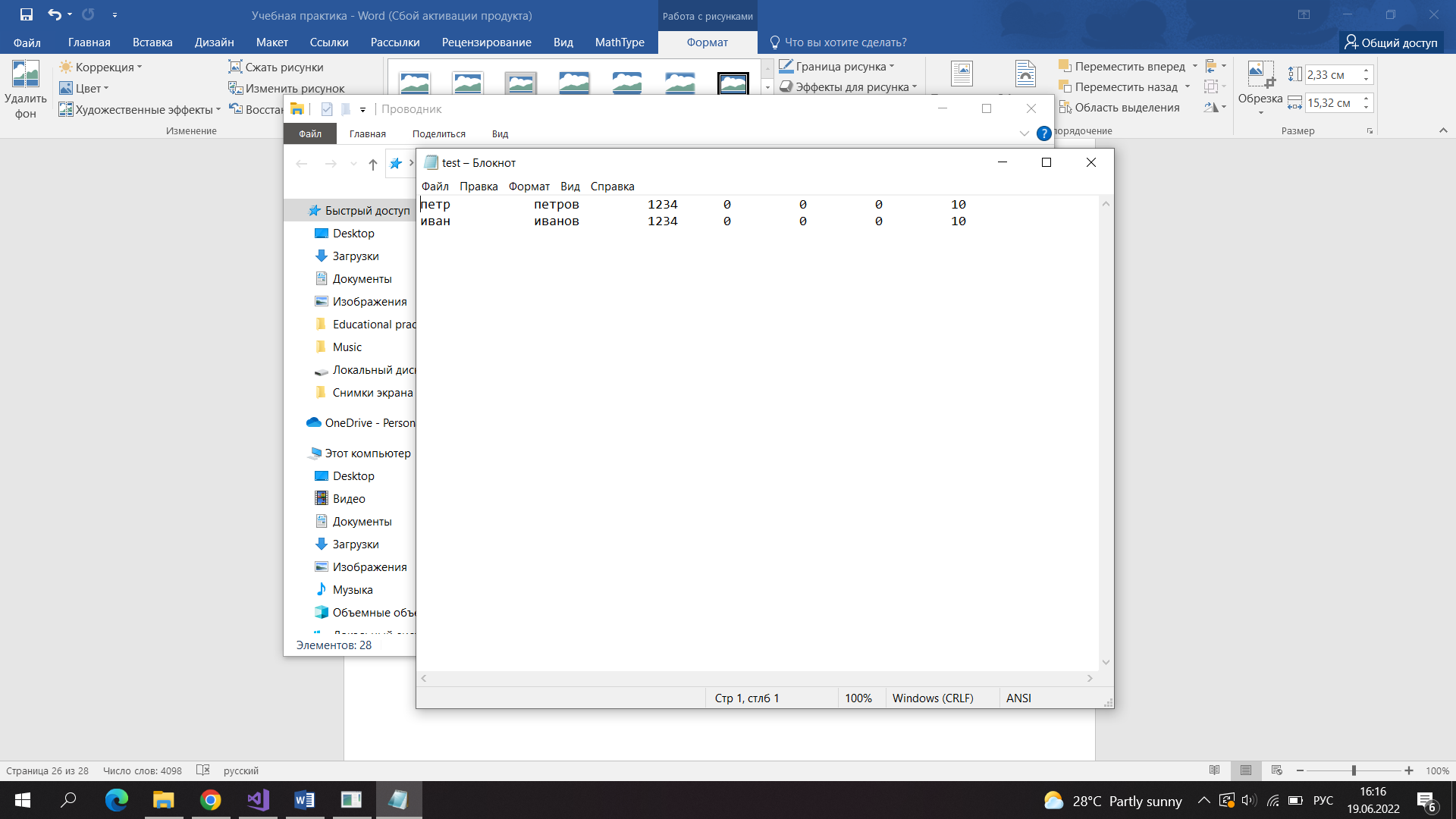
Рисунок 3.2. Рабочий файл после операции пополнения счета.

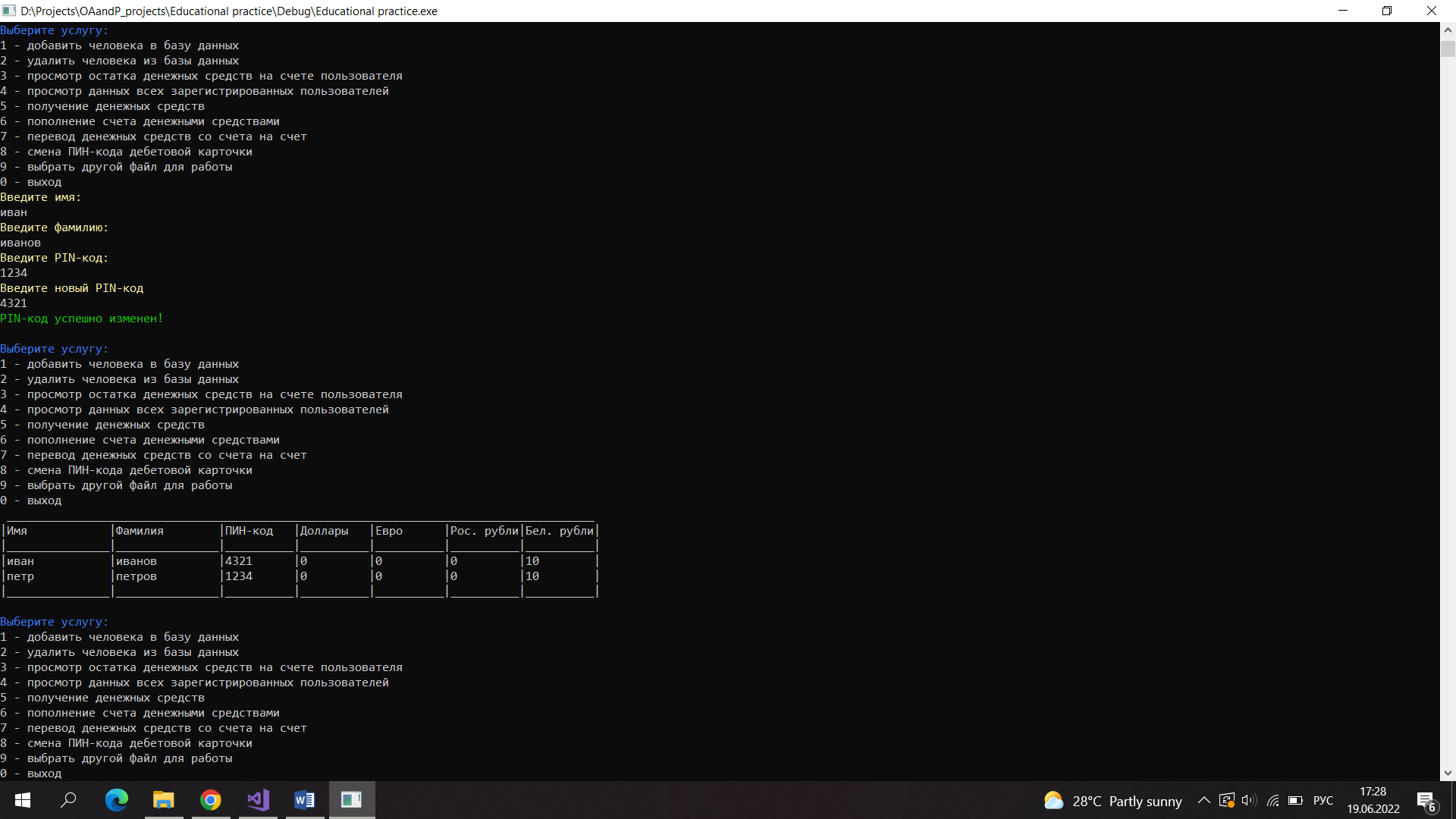
Рисунок 4.1. Пример выполнения операции снятия денежных средств.

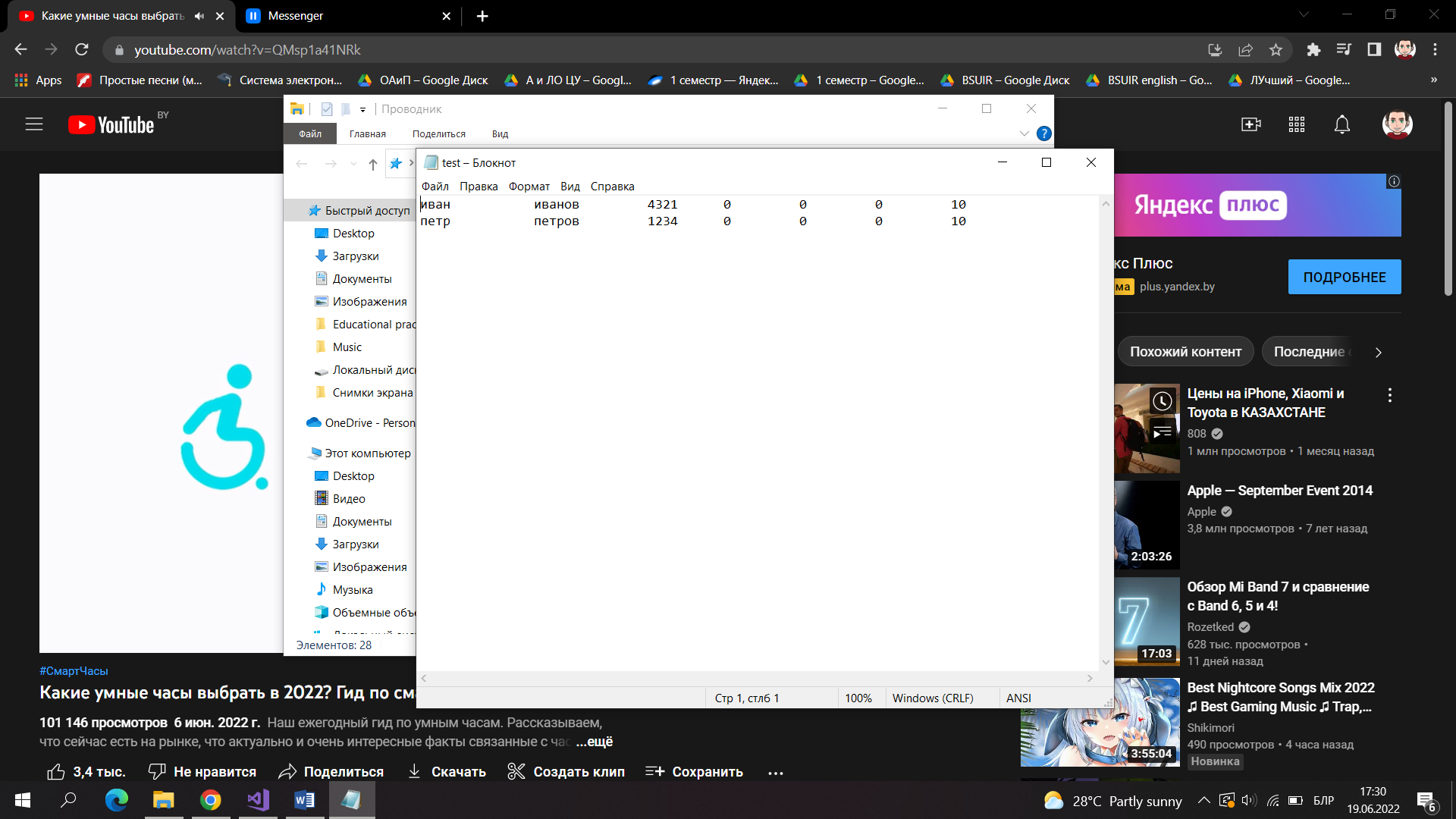
Рисунок 4.2. Рабочий файл после операции снятия денежных средств.

Рисунок 5.1. Пример выполнения операции перевода денежных средств со счета на счет.

Рисунок 5.2. Результат функции перевода денег вызовом функции просмотра информации в файле программно.

Рисунок 5.3. Рабочий файл после выполнения функции перевода денежных средств.

Рисунок 6.1. Пример работы функции смены PIN-кода.

Рисунок 6.2. Рабочий файл после выполнения функции смены PIN-кода.

**СИСТЕМНЫЕ, ПРОГРАММНЫЕ ТРЕБОВАНИЯ,  
ОГРАНИЧЕНИЯ, РЕКОМЕНДАЦИИ**

Программные ограничения:

1. Нельзя вводить одно и то же имя и фамилию для двух разных людей.

2. Нельзя вводить числа, которые не соответствуют диапазону вводимых чисел для типа “int”.

3. Нельзя вводить слова (имя человека, фамилия, имя файла), которые содержат больше len символов.

4. Нельзя переводить деньги с одной валюты в другую.

Системные требования и рекомендации:

1. Для компиляции кода рекомендуется использовать Visual Studio 2017 IDE или Clion IDE и компиляторы MSVC, MinGW, Clang, GCC и др.

2. Операционная система: Windows 7/8/10/11/XP.

3. Процессор мощностью 1,6 ГГц или выше.

4. 1 ГБ ОЗУ (1,5 ГБ для виртуальной машины).

**ЗАКЛЮЧЕНИЕ**

В процессе выполнения работы была разработана программа, предоставляющая некоторые услуги банка, а также схема алгоритма головной функции и алгоритмы по шагам некоторых функций.

В программе реализованы следующие функции:

* добавить человека в базу данных;
* удаление человека из базы данных;
* просмотр остатка денежных средств на счете пользователя;
* просмотр данных всех зарегистрированных пользователей;
* получение денежных средств;
* пополнение счета денежными средствами;
* перевод денежных средств со счета на счет;
* смена PIN-кода дебетовой карточки;
* выбор другого файла для дальнейшей работы.

В наше время информационных технологий, когда все операции с денежными средствами можно проводить онлайн, разработанная программа будет весьма полезной. Подобные программы сейчас активно используются не одним банком. Работу такой реальной программы можно увидеть, пользуясь банкоматом или интернет-банкингом.

В дальнейшем, как одно из направлений по развитию процессов функционирования банкомата можно предложить инсталляцию в банкоматы биометрических сканеров, для более эффективной защиты данных; создание универсальных карт, позволяющих совместить в себе несколько функций: банковская карта, права, удостоверение личности и прочие документы. Эти нововведения, на мой взгляд должны облегчить процесс получения банковских услуг для населения, а также сократить кол-во переносимых с собой документов.

Данный проект может служить прототипом, чтобы в будущем использоваться во вполне реальных банковских системах.

**СПИСОК ЛИТЕРАТУРЫ**

Луцик Ю.А., Ковальчук А.М., Сасин Е.А. – Учебное-методическое пособие “Основы алгоритмизации и программирования: язык Си”. – Минск: БГУИР, 2015. – 170 с.

Демидович Ю.А. – Пособие для студентов БГУИР “Основы алгоритмизации и программирования. Язык СИ”. – Мн.: Бестпринт, 2003. – 384 с. ­

Керниган Б., Ритчи Д. – “Язык программирования Си”. – СПб.: “Невский Диалект”, 2001. – 352 с.

Беларусбанк [Электронный ресурс]: обслуживание в банкоматах банка. – URL: <https://belarusbank.by/ru/fizicheskim_licam/cards/uslugi>

(дата обращения: 20.05.2022).

**ПРИЛОЖЕНИЕ А**

Блок-схема функции main.