**ЛАБОРАТОРНАЯ РАБОТА №1**

**«СОЗДАНИЕ ДИНАМИЧЕСКТХ БАЗ ДАННЫХ»**

1.1. Цель работы

Изучение технологии подготовки и выполнения Пролог-программ в интегрированной среде, исследование способов организации динамических баз данных (БД) средствами языка Пролог.

1.2. Постановка задачи

1.2.1. Изучить среду программирования Eclipse ProDT по методическим указаниям [3]. Выполнить все приведенные примеры в окне консоли среды программирования.

1.2.2. Ознакомиться по лекционному материалу или книгам [1,2] с объектами данных и сопоставлением в языке Пролог, организацией управления в прологпрограммах, предикатами обработки списков, встроенными предикатами работы с базой данных и предикатами ввода-вывода. Изучить примеры применения этих предикатов, приведенные в разделе 1.2 настоящей лабораторной работы.

1.2.2. Ознакомиться с вариантом задания и выбрать один из способов для хранения записей базы данных (см. п. 1.2.5).

1.2.3. Ознакомиться с примером кода, приведенном в приложении А, и по аналогии определить на языке Пролог для заданного варианта предикаты добавления записи в базу, удаления записи, просмотра базы, сохранения базы в файле и загрузки её из файла.

1.2.4. Создать в среде программирования Пролог-проект в соответствии с методическими указаниями [3], содержащий подготовленные определения предикатов, указанных в п. 1.4.3.

1.2.5. Выполнить частичную отладку проекта.

1.2.6. Разработать определения дополнительных предикатов выборки и корректировки записей базы данных в соответствии с вариантом. При этом выборку записей из базы выполнять путем реализации операции проекции реляционной алгебры, следуя общим рекомендациям, указанным в п. 1.2.7.

1.2.7. Разработать предикаты, реализующие примеры операций реляционной алгебры (объединение, пересечение, разность) в соответствии с п. 1.2.7.

1.2.8. Выполнить полную отладку проекта и зафиксировать результаты работы программы в виде экранных копий.

1.3. Вариант задания

Написать программу, обеспечивающую создание динамической базы данных. Структура базы данных определяется одной из таблиц в соответствии с вариантом задания. В функции программы должно входить:

− добавление записи в базу данных;

− удаление записи из базы данных;

− просмотр базы данных;

− сохранение базы данных в файле;

− загрузка базы данных из файла;

− реализация операций реляционной алгебры (на примерах).

Кроме этого, программа должна выполнять дополнительные функции, указанные в варианте задания (таблица 1.1).

Таблица 1.1. – Вариант задания

|  |  |
| --- | --- |
| Вариант | Номер таблицы и дополнительные функции |
| 19 | Таблица 1.2. Корректировка данных в базе по названию магазина; вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры; если такого магазина нет, выдать на дисплей соответствующее сообщение. |

Таблица 1.2. — Стоимость

|  |  |  |
| --- | --- | --- |
| Название товара | Название магазина | Стоимость товара, грн |

1.4. Ход работы

Был написан код программы на языке Пролог, представленный в листинге 1.

Листинг 1 – Полный код программы

:-dynamic

product/3, product\_s/3.

start:- menu. %предикат для запуска программы

%0============= отображение меню ==============================================

menu:-

repeat, nl,

**write**('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'),nl,

**write**('\* 1. Add \*'),nl,

**write**('\* 2. Delete \*'),nl,

**write**('\* 3. Watch \*'),nl,

**write**('\* 4. Load \*'),nl,

**write**('\* 5. Save \*'),nl,

**write**('\* 6. Operat. rel. alg. \*'),nl,

**write**('\* 7. Correct \*'),nl,

**write**('\* 8. Search \*'),nl,

**write**('\* 9. Exit \*'),nl,

**write**('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'), nl ,nl,

**read**(C),nl,

    proc(C),

    C=9, %Если С не равно 9, то авт. возврат к repeat

    !. %Иначе успешное завершение

%0----------------------------------------------------------------------------

%1------------------добавить новый элемент в таблицу--------------------------

proc(1):-

**write**('Name: '), nl, **read**(Prod),

**write**('Shop: '), nl, **read**(Shop),

**write**('Price: '), nl, **read**(Price),

    assertz(product(Prod, Shop, Price)),

**write**(Prod),

**write**(' added to DB.').

    get0(C).

%2------------------удалить элемент-------------------------------------------

proc(2):-

**write**('Write the name of product: '), nl, **read**(Prod),

**write**('Write the name of the shop: '), nl, **read**(Shop),

**write**('Price: '), nl, **read**(Price),

    retract(product(Prod, Shop, Price)),

**write**('It was successfully deleted.'),

    get0(C), %ожидание ввода символа

    !; %отсечение альтернативы и завершение

**write**('There is no '), %вывод сообщения о безуспешном удалении

**write**('product in DB'),nl,

**write**('Write any tab'),nl,

    get0(C). %ожидание ввода символа

%3------------------просмотр----------------------------------------------------

proc(3):-

    product(Prod, Shop, Price),

    nl,

    format('Name: ~w, Shop: ~w, Price: ~w~n', [Prod, Shop, Price]),

    %write('Название товара: '), **write**(Prod), nl,

    %write('Название магазина: '), **write**(Shop), nl,

    %write('Цена товара: '), **write**(Price), nl,

    fail;

    true.

%4========== загрузка БД из файла ===============================================

proc(4):-

    see('D:/1lab.txt'),

    retractall(product(\_,\_,\_)),

    db\_load,

    seen,

**write**('DB was loaded from file'), nl.

%загрузка термов в БД из открытого вх. потока

db\_load:-

**read**(Term),

    (Term == end\_of\_file, !;

    assertz(Term),

    db\_load).

%5========== сохранение БД в файле ============================================

proc(5):-

**tell**('D:/1lab.txt'), %открытие вых. потока

    save\_db(product(Prod, Shop, Price)), %сохранение терма

    told, %закрытие вых. потока

**write**('DB was saved in file'),nl.

%сохранение терма в открытом файле

save\_db(Term):- %сохранение терма (факта!) Term в БД

    Term, %отождествление терма с термом в БД

**write**(Term), %запись терма

**write**('.'),nl, %запись точки в конце терма

    fail; %неудача с целью поиска след. варианта

    true. %завершение, если вариантов отождествления нет

%6------------операции реляционной алгебры--------------------------------------

proc(6):-

nl,

    retractall(product\_s(\_, \_, \_)), %необходимая очисточка, чтобы не копилось после нескольких использований пункта 6

    %собираем записи, где магазин - Ашан

**write**('Formation of the r1 relationship: products of the Auchan store '), nl,

    subset\_of\_products(ashan,R1), %R1 - список сотрудников филиала 1

    list\_in\_base(R1), %добавление элементов из R1 в базу данных

    write\_list(R1),nl, %вывод списка R1 на экран

    %собираем записи, где магазин - Оксана

**write**('Formation of the r2 relationship: products of the Oksana store '), nl,

    subset\_of\_products(oksana,R2), %R2 - список сотрудников филиала 2

    list\_in\_base(R2), %добавление элементов из R2 в базу данных

    write\_list(R2),nl, %вывод списка R2 на экран

    %Объединение - вывод всех продуктов двух магазинов - Ашана и Оксаны

**write**('Combined ratio g1\_ or g2: '), nl,

    union(Rez1), %Rez1 - список продуктов магазина1 или 2

    write\_list(Rez1),nl,

    %Пересечение - вывод продуктов, которые есть и в Ашане, и в Оксане, в виде (назван\_прод., ашан, цена\_ашана, оксана, цена\_оксаны)

**write**('The intersection of the relations g1\_ and g2: '), nl,

    intersection(Rez2), %Rez2 - список сотрудников 2-х филиалов

    write\_list(Rez2),nl,

    %Разница - выводятся все продукты Ашана, за исключением тех, что есть в Оксане

**write**('The difference in the ratio r1-r2: '), nl,

    difference(Rez3), %Rez3-список сотрудников филиала 1 без фил.2

    write\_list(Rez3),nl,

    R2 = [], R1 = [], Rez1 = [], Rez2 = [], Rez3 = [],

**write**('Enter any character'),nl,

    get0(C). %Ожидание ввода символа

%--------------вспомогалка для 6 пункта----------------------------------------------------

%формирование подмножества сотрудников R заданного Филиала

%подмножество R представляется в виде списка термов "сотрудник\_ф(...)"

subset\_of\_products(Shop,R):-

    bagof(product\_s(Prod, Shop, Price),

    product(Prod, Shop, Price), R).

%правило объединения отношений - r1 или r2

union\_r1\_r2(X1,X2,X3):-

    product\_s(X1,ashan,X3),X2=ashan;

    product\_s(X1,oksana,X3),X2=oksana.

%формирование списка Rez из фактов "сотрудник\_ф1\_или\_ф2"

union(Rez):-

    bagof(product\_s1\_or\_s2(X1,X2,X3),

         union\_r1\_r2(X1,X2,X3), %условие вкл. в список

         Rez).

%один и тот же продукт в двух магазинах

intersection\_r1\_r2(X11,X12,X13,X22,X23):-

    product\_s(X11,ashan,X13),X12=ashan,

    product\_s(X11,oksana,X23),X22=oksana.

%формирование списка Rez из фактов "сотрудник\_ф1\_и\_ф2"

intersection(Rez):-

    bagof(product\_s1\_and\_s2(X11,X12,X13,X22,X23),

    intersection\_r1\_r2(X11,X12,X13,X22,X23),

    Rez).

difference\_r1\_r2(X11,X12,X13):-

    product\_s(X11,ashan,X13),X12=ashan,

    not(product\_s(X11,oksana,X23)),X22=oksana.

%построение списка Rez из фактов "сотрудник\_ф1\_и\_не\_ф2"

difference(Rez):-

    bagof(product\_s1\_and\_no\_s2(X11,X12,X13),

    difference\_r1\_r2(X11,X12,X13), %условие вкл. в список

    Rez).

%добавление термов из списка [H|T] в БД

list\_in\_base([]).

list\_in\_base([H|T]):-

    H=product\_s(Prod, Shop, Price),

    assertz(product\_s(Prod, Shop, Price)),

    list\_in\_base(T). %Рекурсивный вызов для след. терма

%вывод элементов списка [H|T] в каждой строке

write\_list([]).

write\_list([H|T]):-**write**(H),nl,write\_list(T).

%----------------Корректировка цены по названию продукта и магазу, в котором он находится

proc(7):-

**write**('Write the name of product: '), nl, **read**(Prod),

**write**('Write the name of the shop: '), nl, **read**(Shop),

**write**('Write new price: '), nl, **read**(NewPrice),

    retract(product(Prod, Shop, \_)),

    assertz(product(Prod, Shop, NewPrice)),

**write**(Prod),

**write**(': the price was changed.').

%----------------Вывод продуктов магаза с клавы

proc(8):-

**write**('Write the name of the shop: '), nl, **read**(Shop),

    findall(Prod, product(Prod, Shop, \_), Products),

    (Products = [] ->

**write**('No shop.');

**write**('Goods of the shop '), **write**(Shop), **write**(':'), nl,

        print\_products(Products)

    ).

    % Помощник для печати списка товаров

    print\_products([]).

    print\_products([Prod|Rest]) :-

**write**(Prod), nl,

        print\_products(Rest).

%-----------Выход--------------

proc(9):-**write**('Goodbye'),nl.

Было проведено тестирование разработанных функций.

Тестирование функции добавления нового элемента в таблицу.

На рисунке 1 представлена изначальная база данных.

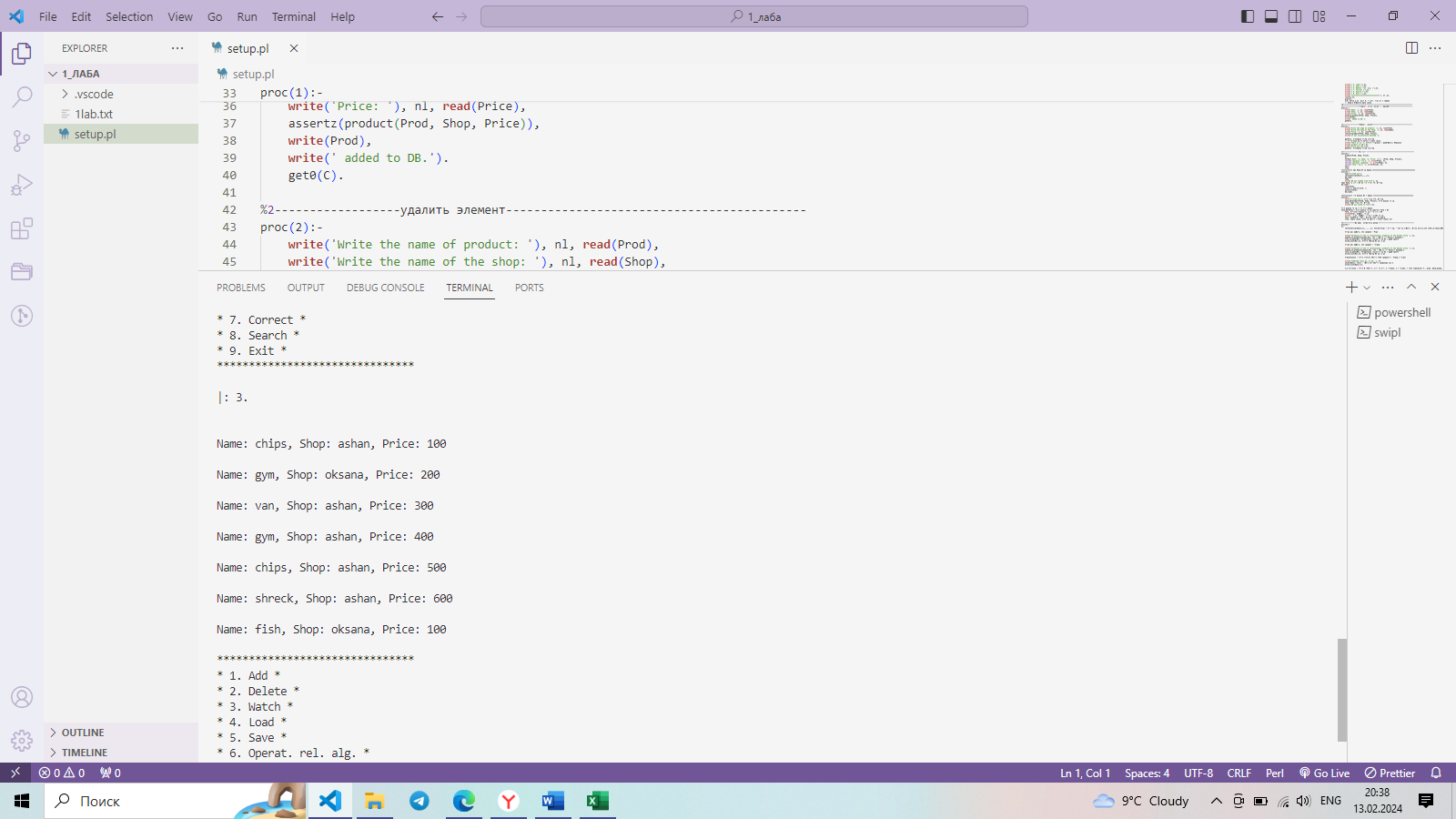


Рисунок 1 – Наполнение таблицы

Далее было проведено добавление нового элемента в таблицу, результаты представлены на рисунках 2-3.

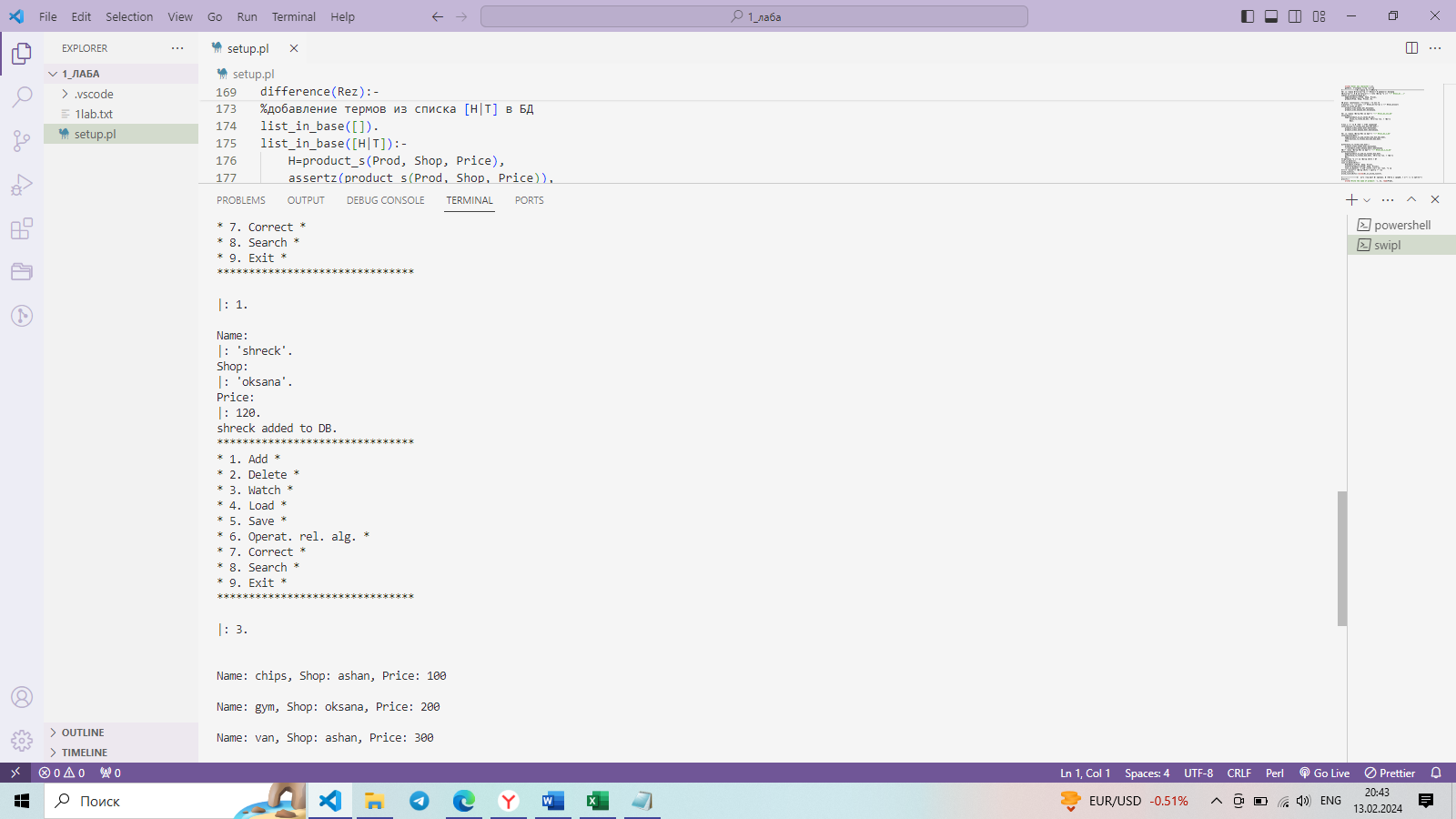


Рисунок 2 – Процедура добавления нового элемента

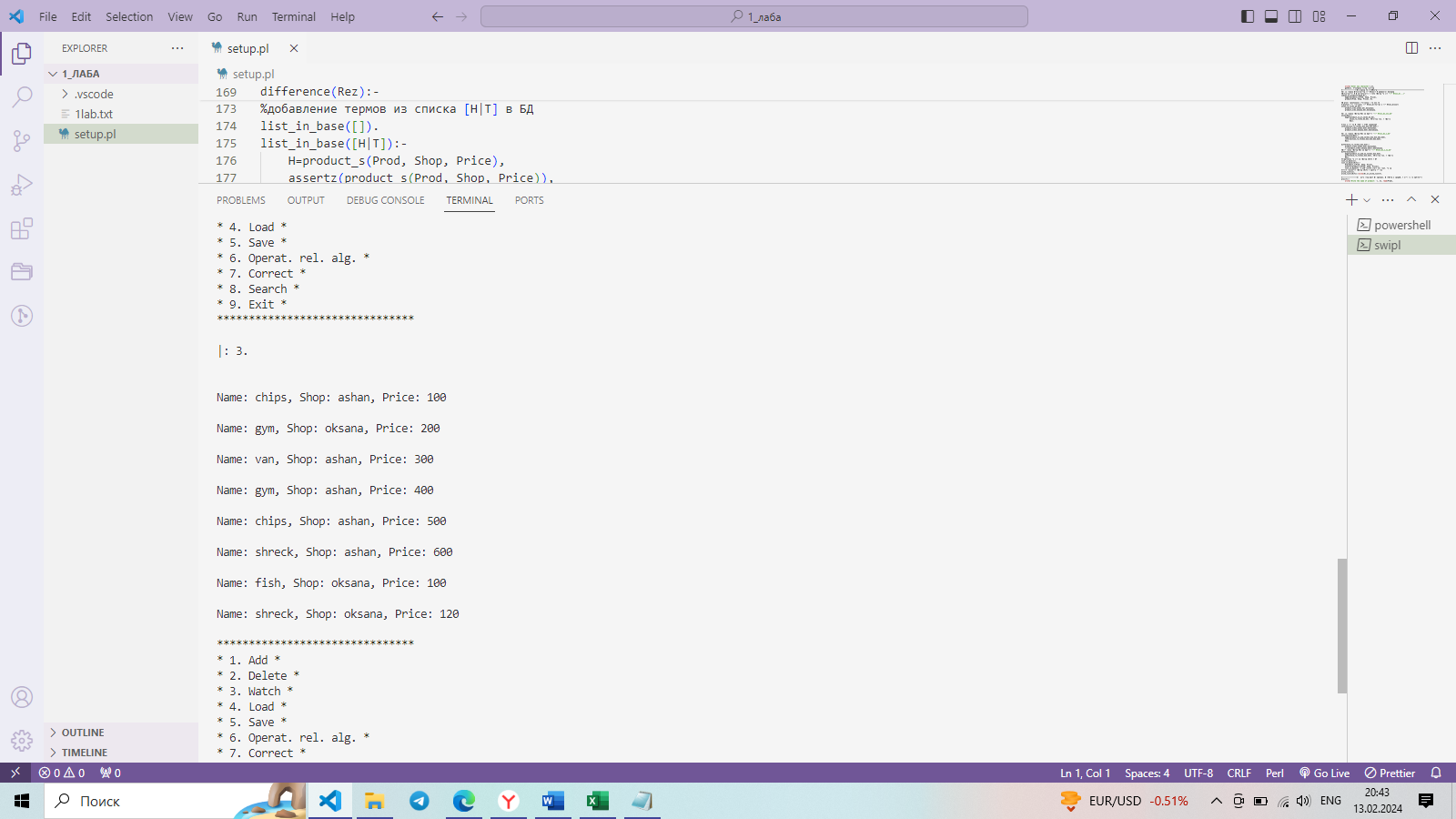


Рисунок 3 – Просмотр таблицы с новой записью

Далее было произведено тестирование функции удаления элемента, чьи поля совпадают с теми, что введет пользователь. Результаты представлены на рисунках 4-5.

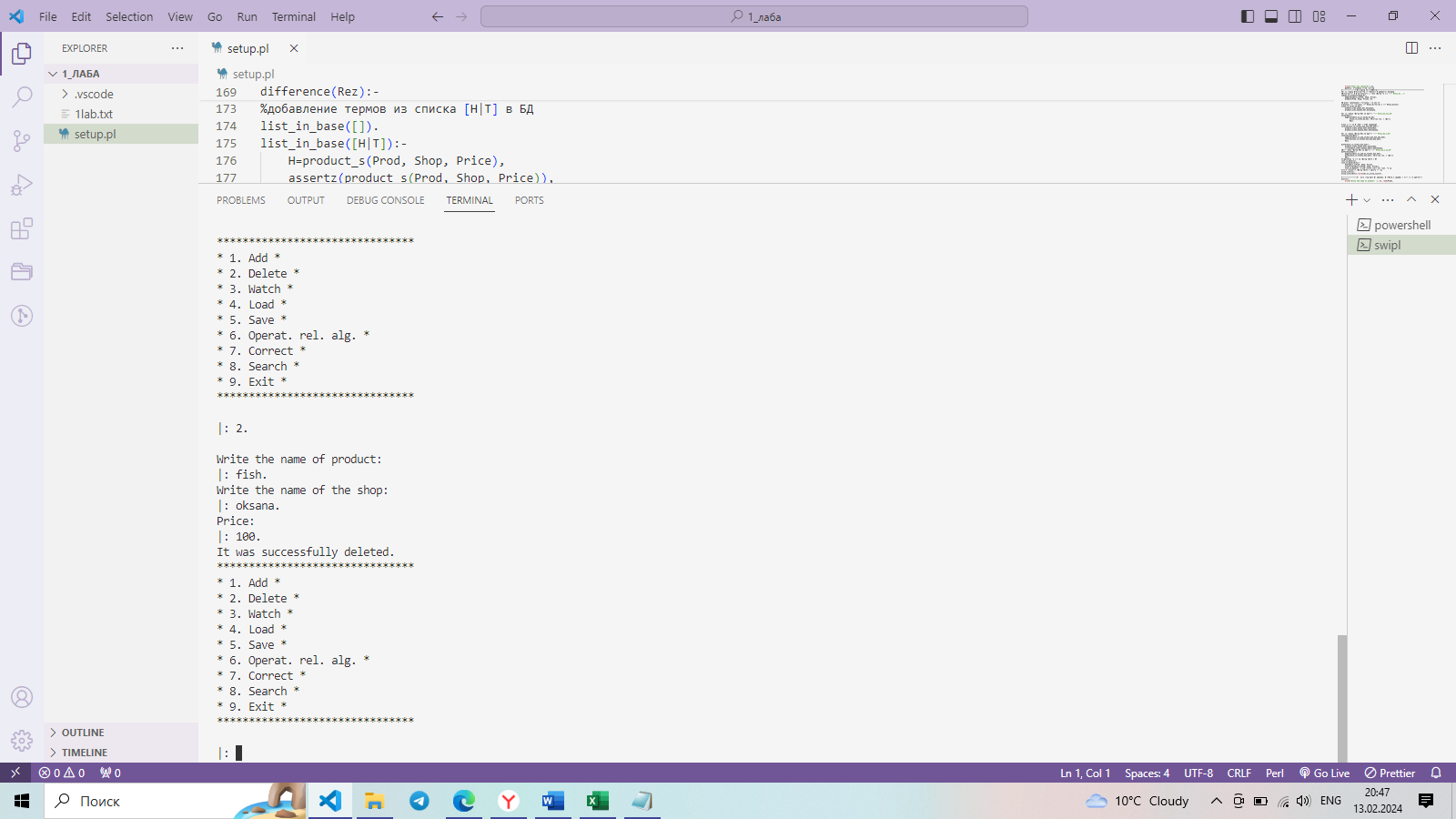


Рисунок 4 – Удаление элемента с введенными значениями полей

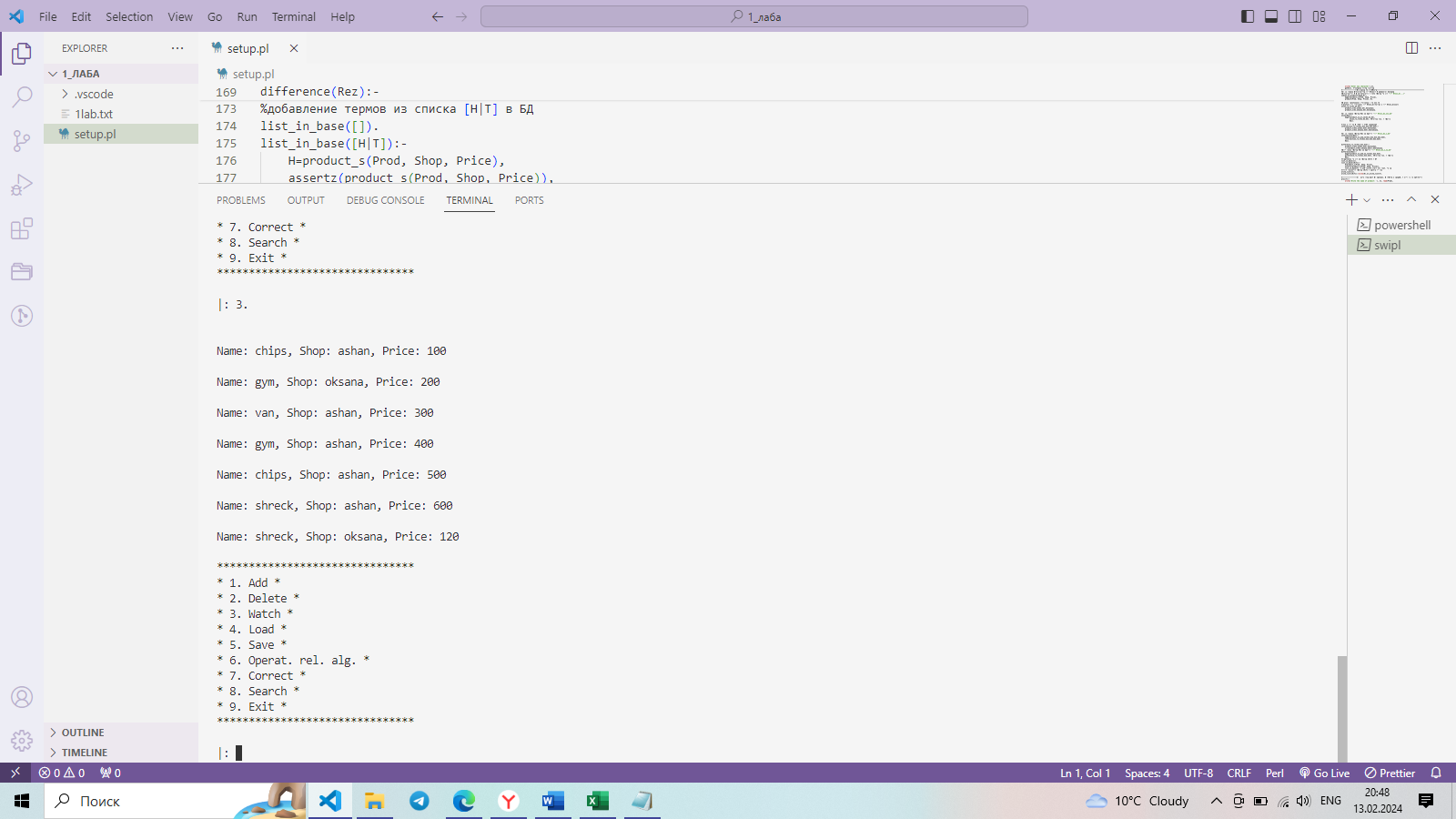
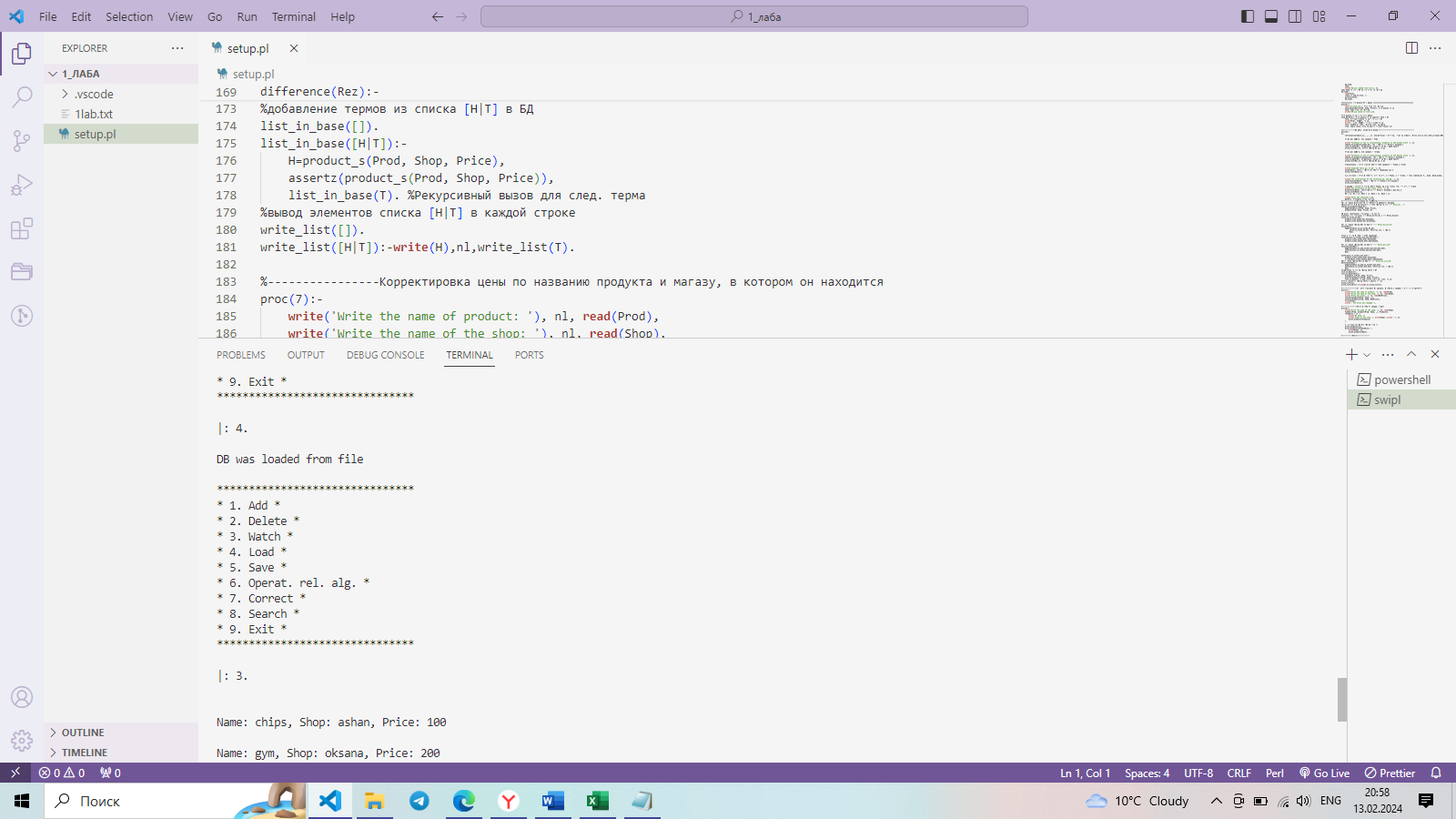


Рисунок 5 – Таблица после изменений

Были протестированы функции загрузки базы данных из текстового файла и сохранения базы в тот же файл. Результаты на рисунках 6-7.



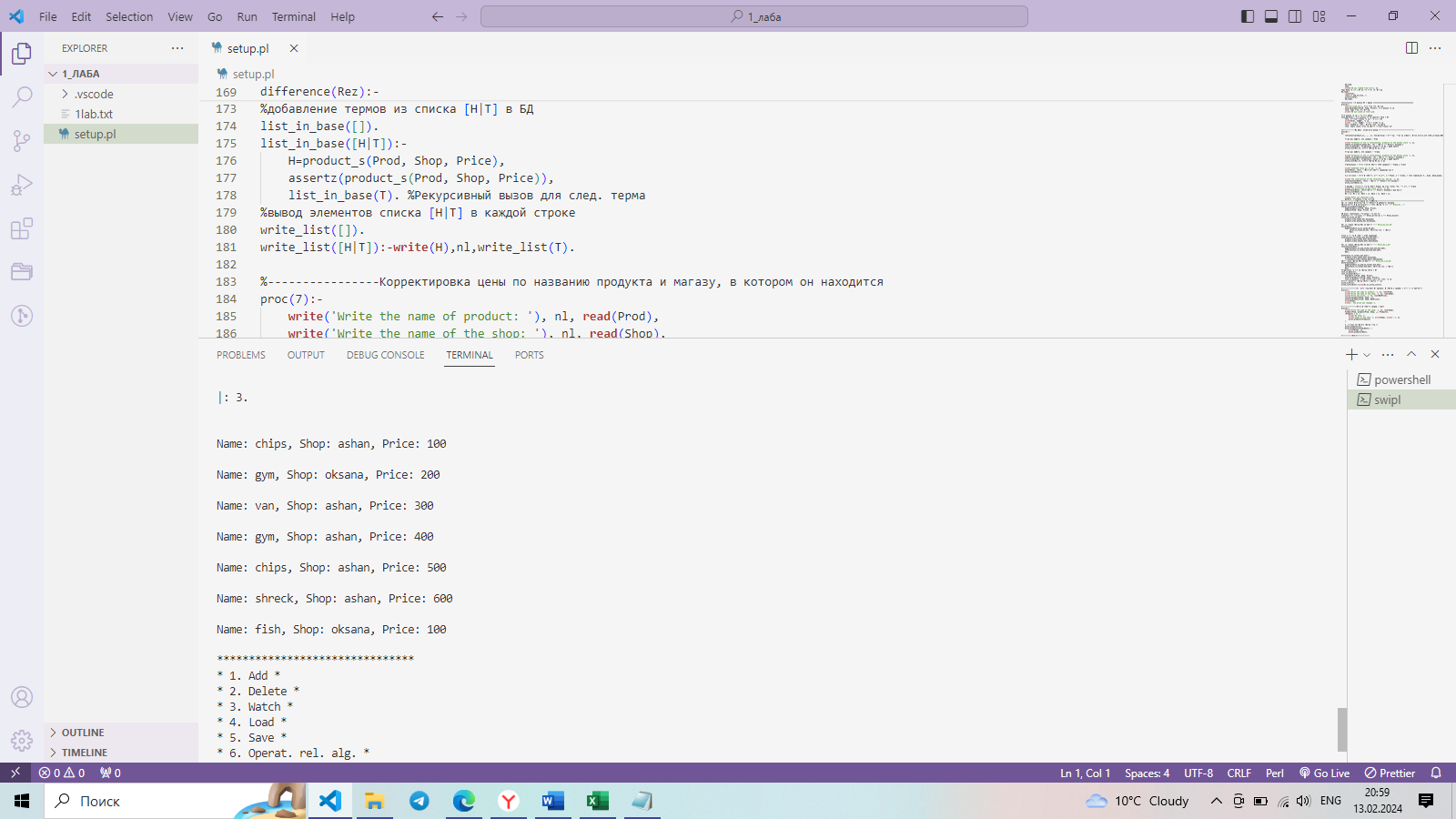


Рисунок 6 – Функция загрузки изначальной таблицы из файла

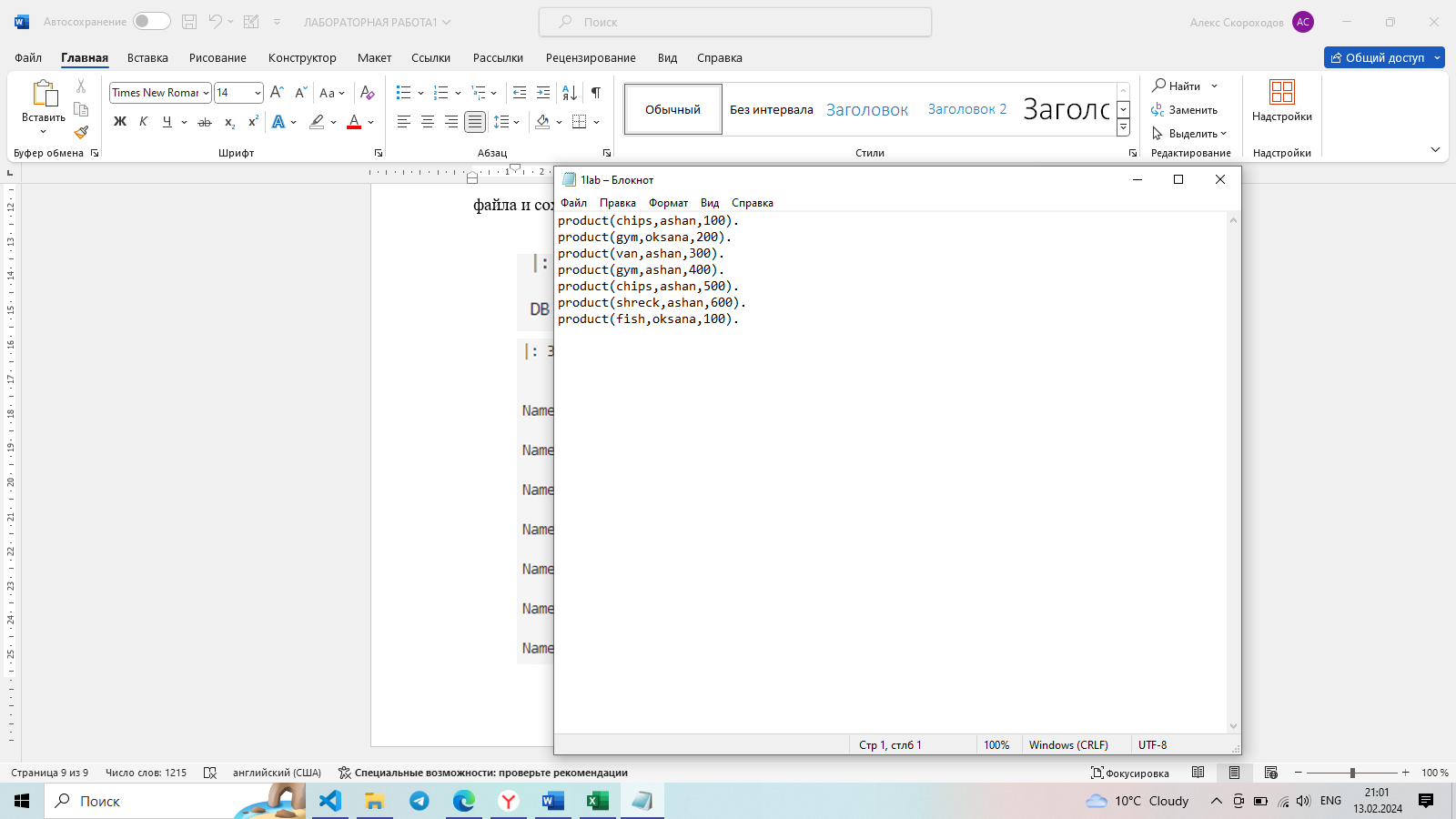


Рисунок 7 – Готовая таблица в файле

Содержимое таблицы было изменено и сохранено с помощью функции 5. Результат на рисунках 8-10.

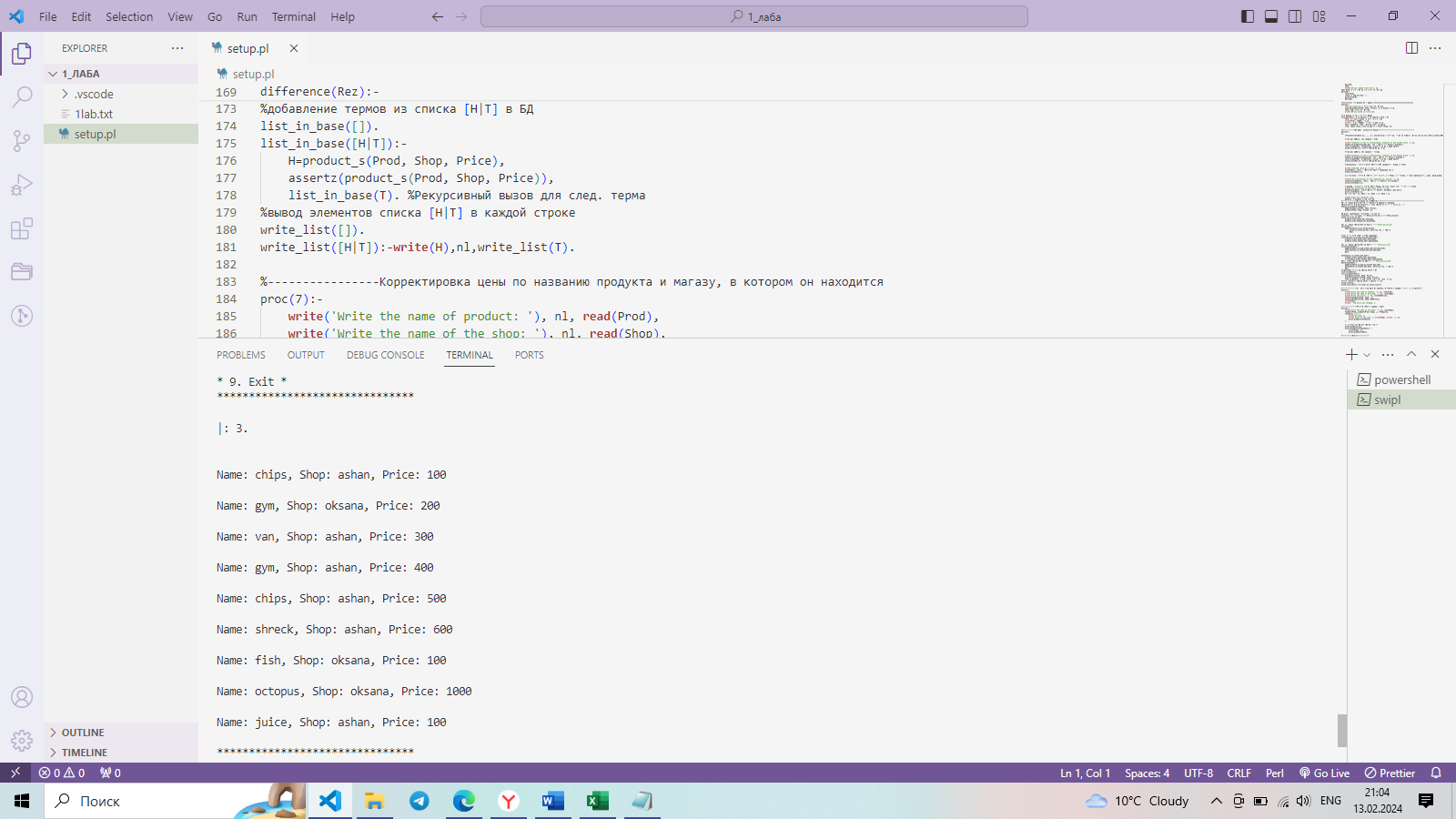


Рисунок 8 – Измененная таблица

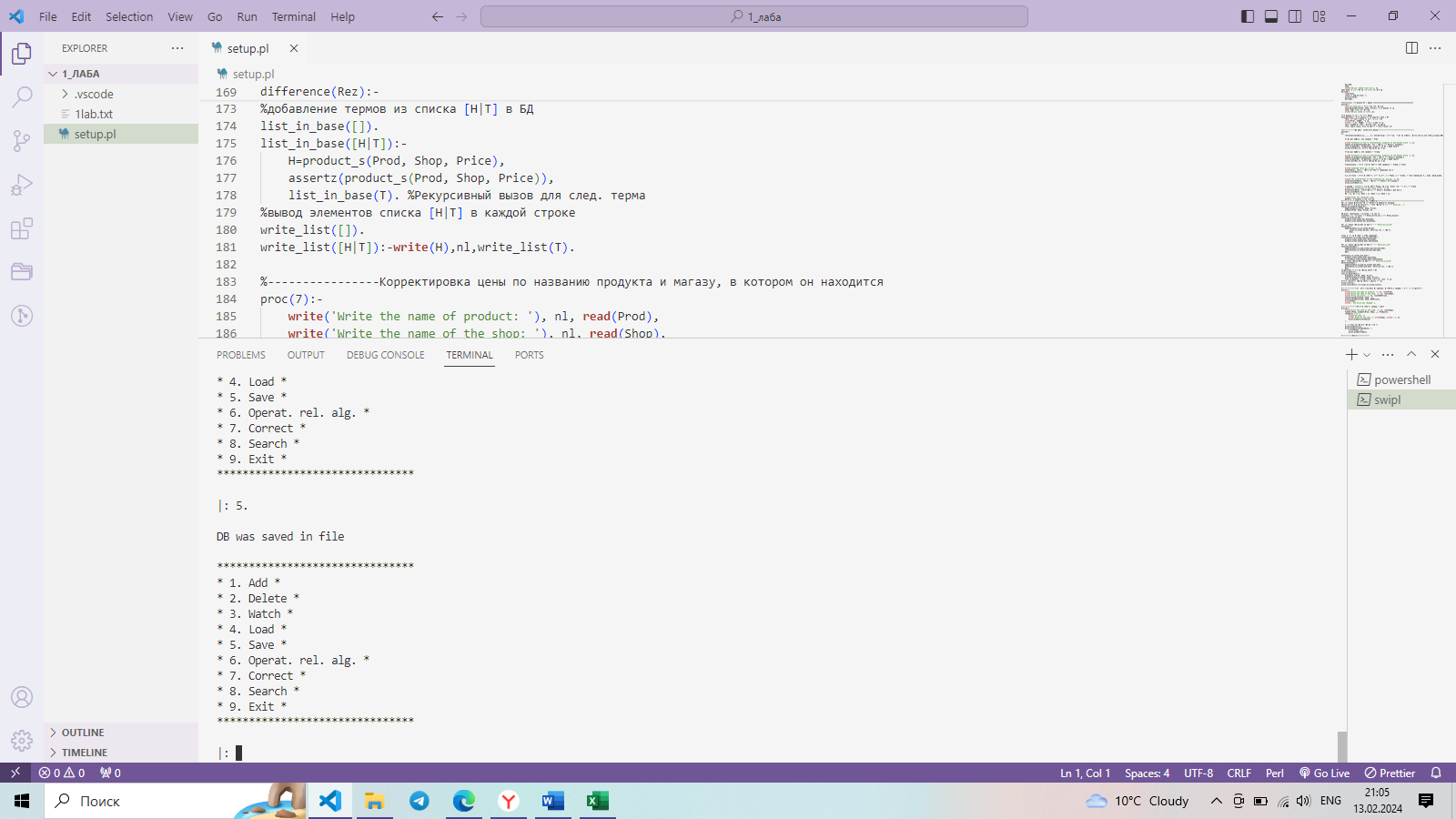


Рисунок 9 – Вызов процедуры сохранения таблицы в файл

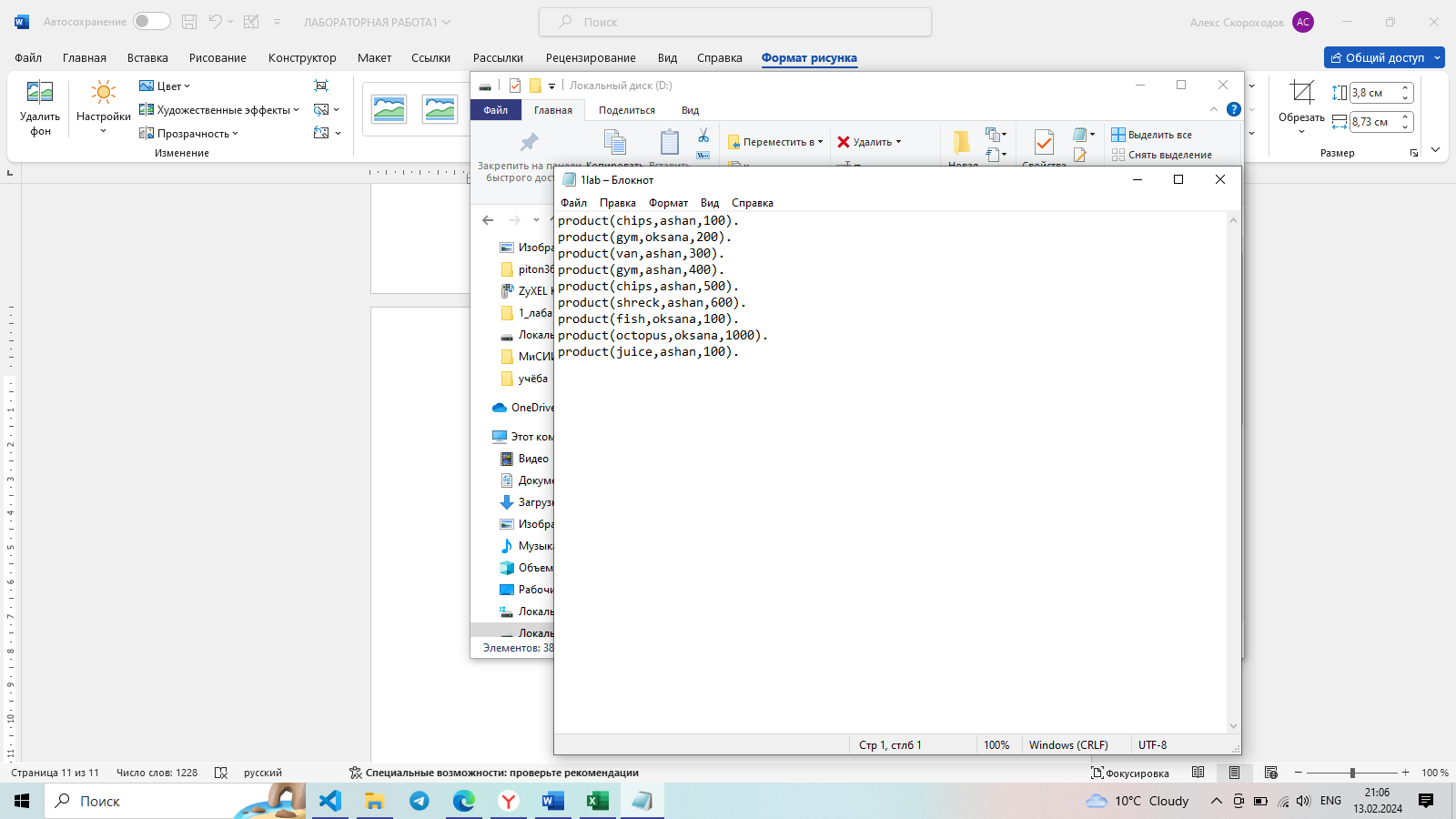


Рисунок 10 – Вид таблицы в текстовом файле после работы функции сохранения

Также были реализованы операции реляционной алгебры. Результат работы функции представлены на рисунке 11.

Были сформированы два подмножества подмножества – подмножество продуктов магазина «Ашан» и подмножество продуктов магазина «Оксана». Затем были проведены операции объединения, пересечения, разности двух множеств.

Операция объединения выводит все продукты двух магазинов.

Операция пересечения выводит только те продукты, названия которых присутствуют в двух магазинах.

Операция разности первого и второго множества выводит продукты, находящиеся в первом магазине, за исключением тех, которые также встречаются и во втором.

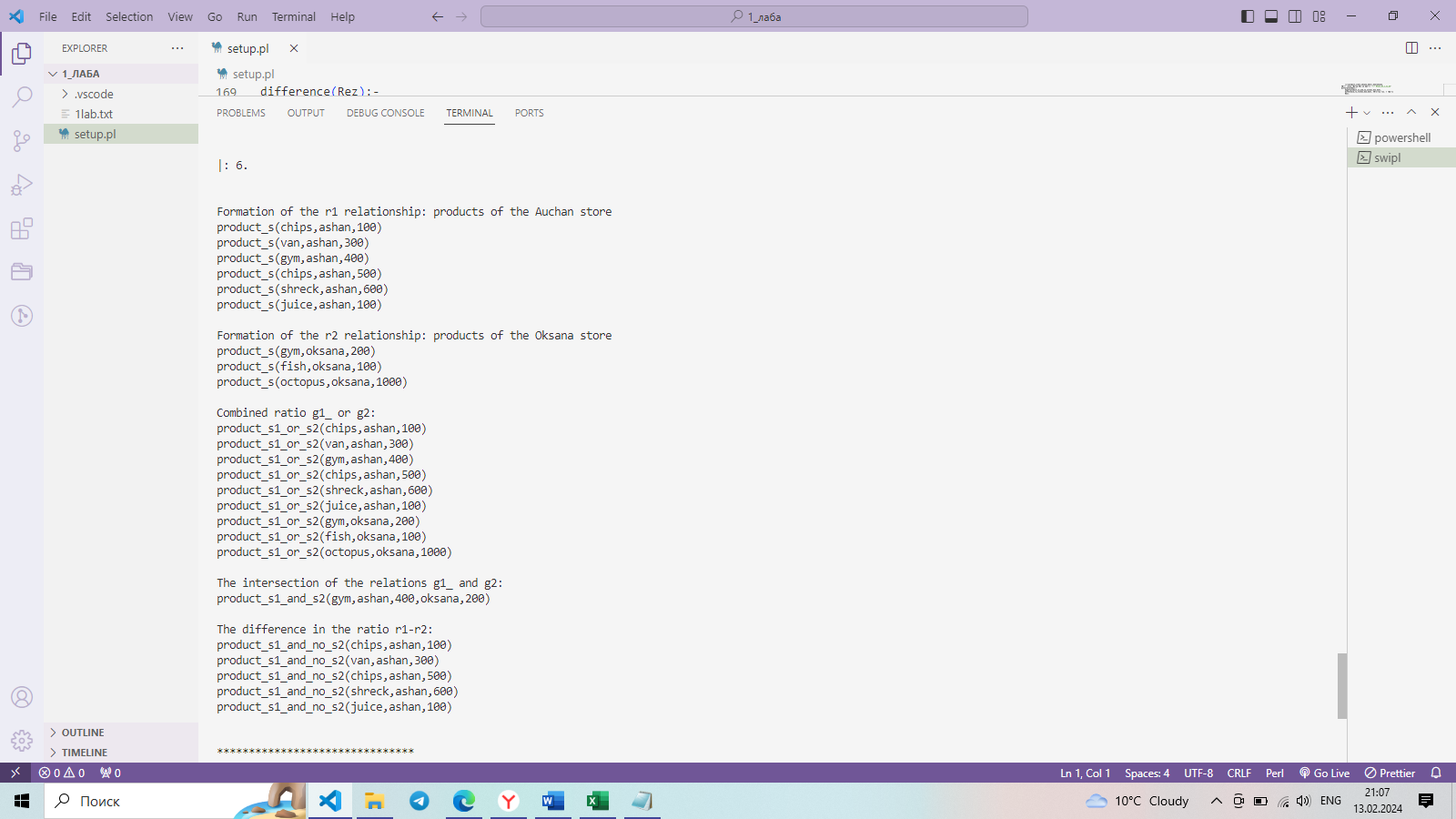


Рисунок 11 – Операции реляционной алгебры

Как видно, операции работают правильно.

Далее требовалось написать функцию корректировки записи. Результаты тестирования работы функции представлены на рисунке 12.

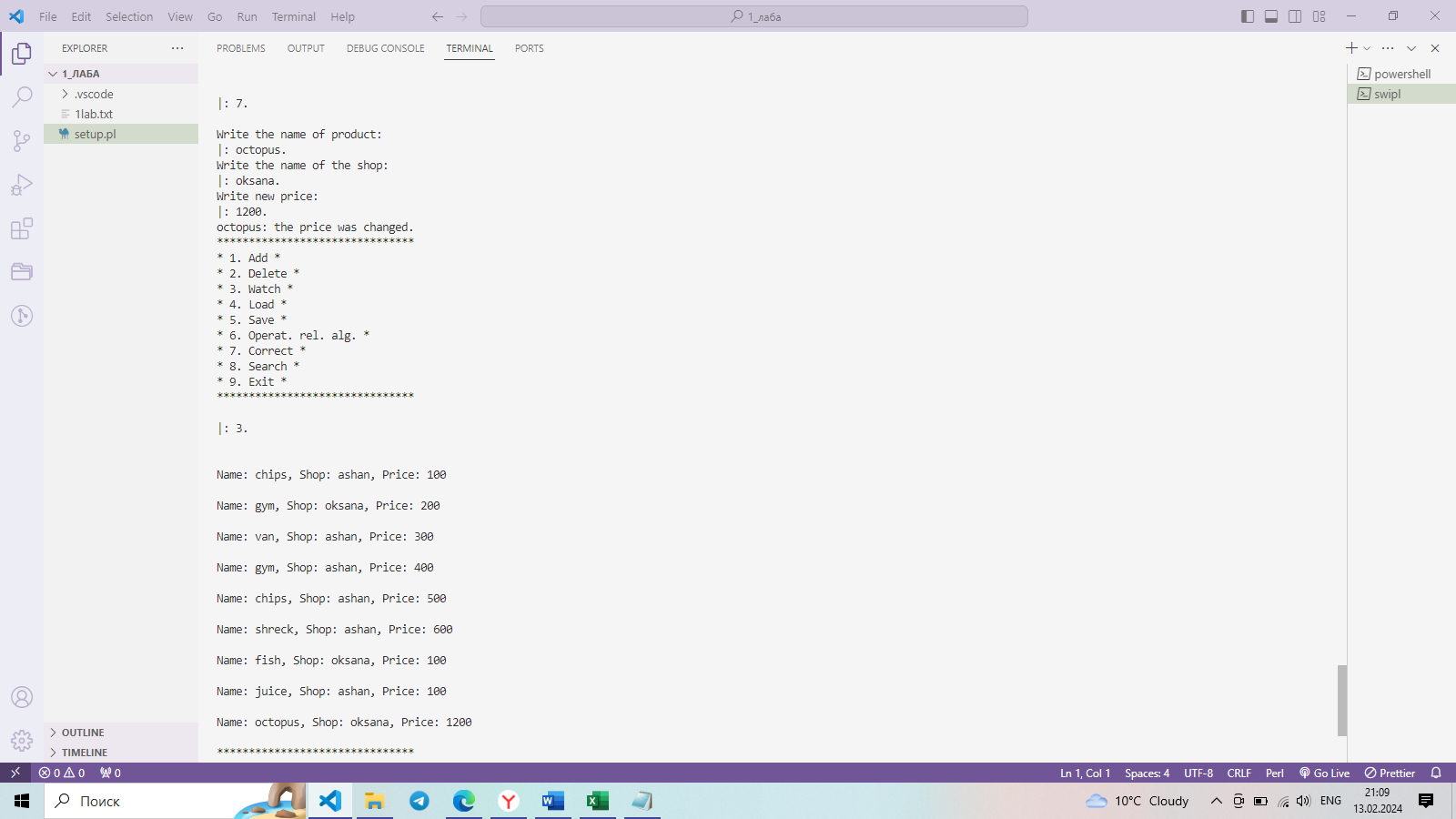


Рисунок 12 – Результат работы функции корректировки цены

Далее была проверена работа функции вывода всех товаров магазина, название которого было введено с клавиатуры. Если магазина нет в БД, требовалось вывести соответствующее сообщением.

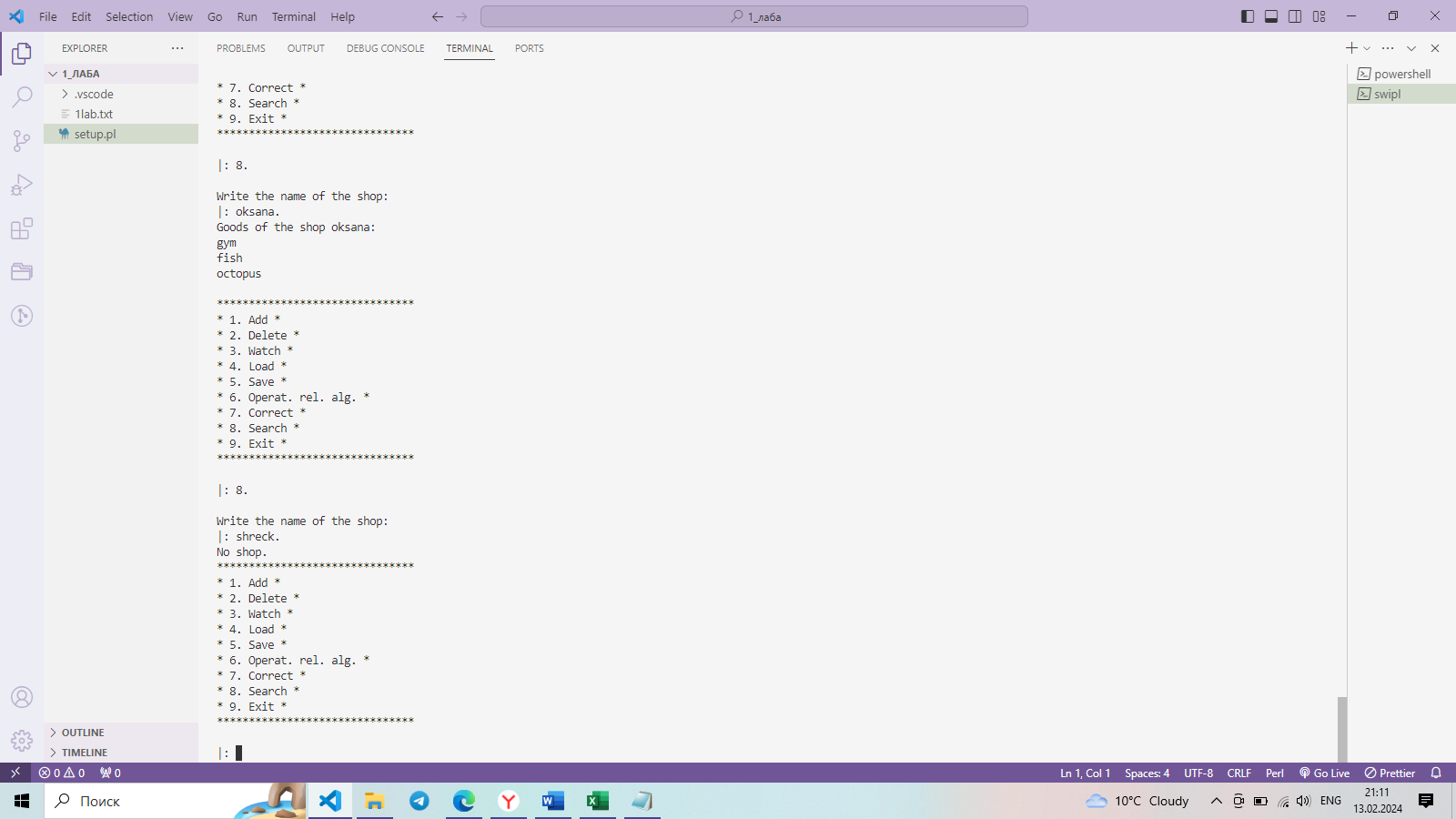


Рисунок 13 – Результат работы функции поиска товаров магазина

По результатам тестирования можно сказать, что все функции работают корректно.

1.5. Выводы

В ходе выполнения лабораторной работы были изучены технологии подготовки и выполнения Пролог-программ в интегрированной среде VS Code, произведено исследование способов организации динамических баз данных (БД) средствами языка Пролог путем создания собственной БД согласно выбранному варианту. Были написаны функции, позволяющие работать с БД, а именно добавлять, удалять, изменять элементы таблицы и производить операции реляционной алгебры и поиск по таблице.

1.6. Контрольные вопросы

**1.6.1. Назовите составные части пролог-программы и приведите их определения.**

Программа на языке Пролог состоит из фактов, правил и целевых утверждений. **Факт** представляет собой истинное утверждение. **Правило** представляет собой утверждение, которое истинно при определенных условиях. Совокупность фактов и правил образует базу данных Пролога. Когда база данных загружена в память Пролог-системы, к ней можно обращаться с вопросами, формулируемыми в виде **целевых утверждений**.

**1.6.2. Сформулируйте определения следующих понятий языка Пролог: терм, константа, атом, переменная, анонимная переменная, структура, список.**

Все объекты данных языка Пролог представляют собой **термы**. Терм может быть константой, переменной или структурой (составным термом). **Константы** подразделяются на числа и атомы. **Атомы** — это символьные константы, которые начинаются строчной буквой или заключаются в одинарные кавычки, или состоят из специальных символов. Примеры атомов: сергей, 'Сергей'. ===>.

Имена **переменных** начинаются с заглавных букв или символа подчеркивания “\_”. Областью действия переменной является утверждение (факт или правило). Переменные, которым присвоены значения, называются *конкретизированными*. Существуют **анонимные переменные** — переменные без имени. Они обозначаются символом подчеркивания. Каждая анонимная переменная уникальна, т.е. отличается от всех других анонимных переменных в утверждении.

**Структура (составной терм)** является объектом, состоящим из нескольких компонент. Она состоит из атома, который называется функтором, и последовательности термов. Например, f(T1,T2,T3) - структура, состоящая из функтора f и трёх компонент T1,T2,T3 (термов). Число компонент структуры называется арностью.

**Список** в языке Пролог представляет собой структуру данных, состоящую из элементов, упорядоченных в определенном порядке. Он может содержать любые элементы, включая другие списки. Списки в Прологе имеют рекурсивную структуру, где каждый список состоит из головы (первого элемента) и хвоста (оставшейся части списка).

**1.6.3. Что понимают под сопоставлением (унификацией) в языке Пролог?**

**Сопоставление** — это процесс, на вход которого подаются два терма, а он проверяет, соответствуют ли эти термы друг другу. Если термы не сопоставимы, то процесс терпит неудачу. Если термы сопоставимы, то процесс находит конкретизацию переменных, делающую эти термы тождественными, и завершается успешно.

**1.6.4. Сформулируйте правила сопоставления термов в Прологе.**

В общем, два терма сопоставляются по следующим правилам:

а) если термы X и Y — константы, то они сопоставимы, только когда одинаковы;

б) если терм Х представлен константой или структурой, а терм Y — не конкретизированной переменной, то термы Х и Y сопоставимы, и в Y подставляется значение Х;

в) если термы Х и Y — структуры, то они сопоставимы, когда у них совпадают главные функторы и арность, а также сопоставимы соответствующие компоненты структуры.

**1.6.5. В чем заключаются декларативный и процедурный смыслы пролог-программы?**

**Декларативный** смысл пролог-программы связан с отношениями, объявленными (декларируемыми) в программе, он определяет, достижимо ли целевое утверждение, и при каких значениях переменных оно будет верным. **Процедурный** смысл определяет, как пролог-система обрабатывает отношения пролог-программы, каким образом пролог-система отвечает на вопросы.

**1.6.6. Объясните процедурный смысл простейших правил с конъюнкцией и дизъюнкцией условий.**

Рассмотрим правило P:- Q, R. Запятая между условиями Q и R обозначает **конъюнкцию**. Декларативная интерпретация правила может быть следующей: P истинно, если условия Q и R истинны. А процедурный вариант можно сформулировать так: чтобы решить задачу P, сначала реши подзадачу Q , а затем — подзадачу R . Таким образом, процедурная интерпретация фиксирует порядок, в котором обрабатываются подцели Q и R.

**Дизъюнкция** условий обозначается точкой с запятой. Например: P:- Q ; R. В этом случае пролог система должна оценить верность условия Q или R. Чтобы доказать P пролог-система сначала предпримет попытку доказательства Q. Если доказательство Q завершится неудачей (fail), то пролог-система вернется к точке выбора вариантов, удалит все сделанные подстановки и перейдет к доказательству альтернативного утверждения R. Однако она перейдет к доказательству R и в том случае, когда Q верно. Так как в процессе доказательства выполняются подстановки в переменные, то доказательство альтернативного утверждения обеспечивает нахождение дополнительных вариантов подстановок, которые могут интересовать пользователя.

**1.6.7. Объясните назначение предиката отсечения. Приведите пример.**

Если проверку альтернативного условия требуется исключить, то применяют встроенный предикат отсечения, который обозначается знаком “ ! ”. Этот предикат всегда выполняется успешно и стирает все альтернативные ветви в пределах утверждения, в котором он введен. Например: P:- Q, ! ; R. В этом случае при успешной попытке доказательства Q предикат отсечения сотрёт точку выбора альтернативного условия R и оно не будет проверяться.

**1.6.8. С какой целью применяется встроенный предикат fail ?**

Для управления процессом обработки условий также широко применяется встроенный предикат fail , который обеспечивает создание состояния искусственной неудачи. Состояние искусственной неудачи заставляет пролог-систему возвращаться к имеющимся точкам выбора и искать другие варианты доказательства утверждений. Точки выбора могут формироваться предикатами, допускающими альтернативные подстановки (такие предикаты называют недетерминированными), или создаваться искусственно с помощью специальных предикатов.

**1.6.9. Приведите определение цикла repeat-fail . Объясните порядок его выполнения**.

Таким специальным предикатом является предикат repeat [1,2]. Предикат определяется рекурсивно: repeat:-true; repeat. Благодаря этому он создаёт бесконечное число точек выбора. Его часто применяют совместно с fail для организации циклов, которые называются циклами repeat-fail. Общая схема организации таких циклов следующая:

**цикл:- repeat,**

**(<проверка условия выхода из цикла>, !;**

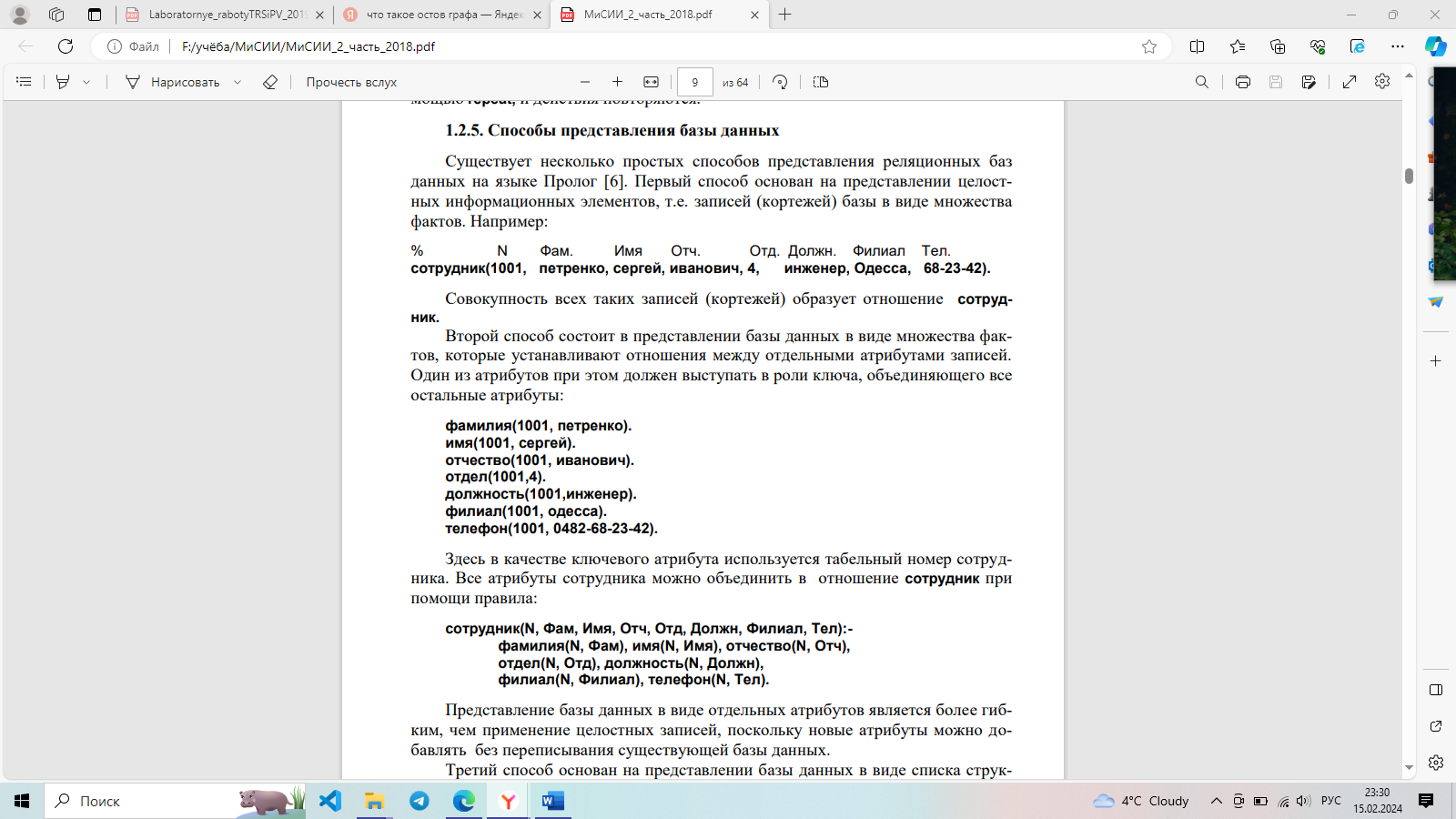
**<тело цикла>, fail).**

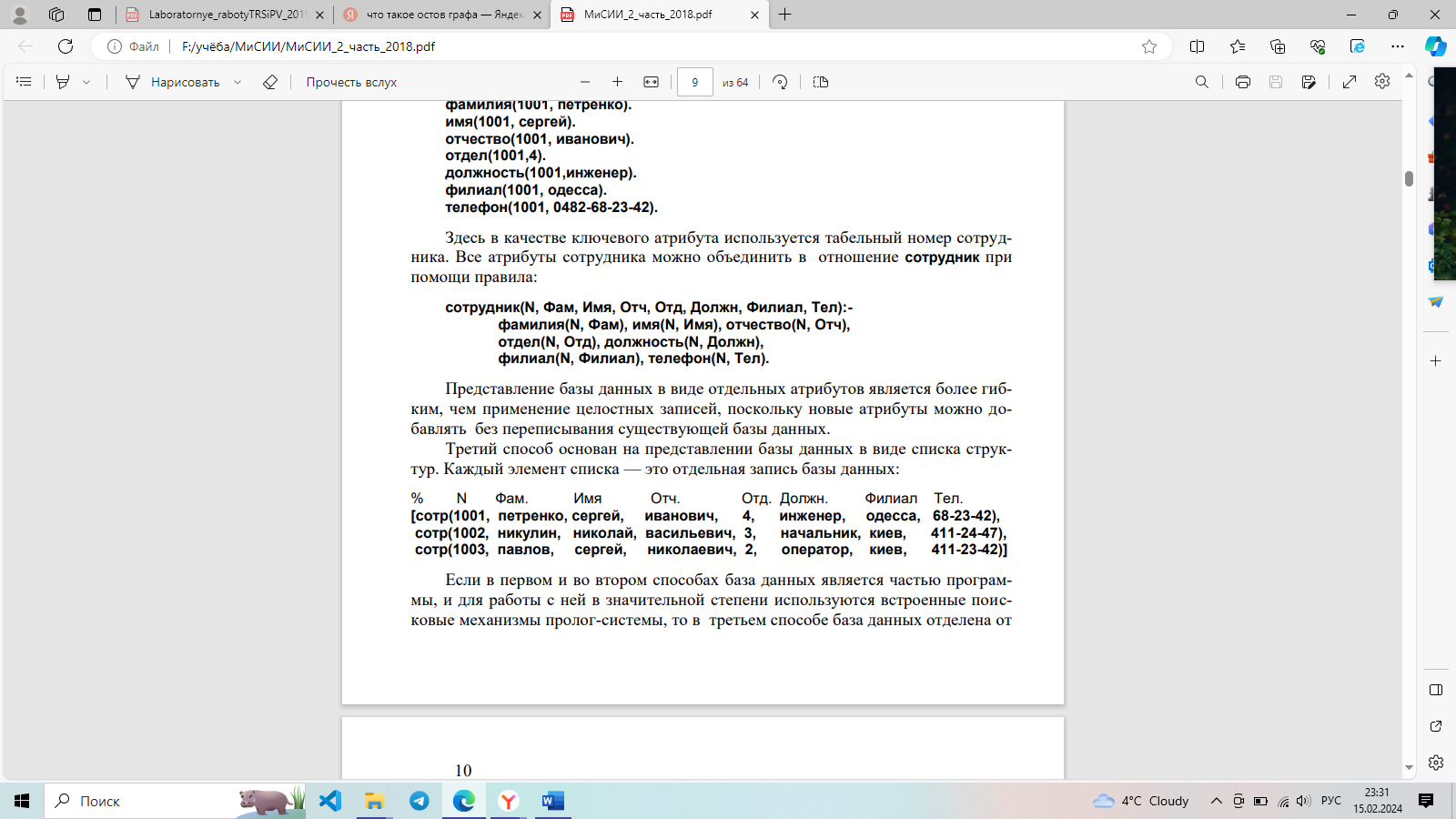
Здесь скобки образуют *группу* и тем самым ограничивают область действия дизъюнкции. Если условие завершения цикла выполняется успешно, то предикат отсечения стирает все точки выбора и цикл завершается. Иначе выполняется тело цикла, а предикат fail обеспечивает возврат к точкам выбора, создаваемым с помощью repeat, и действия повторяются.

**1.6.10. Объясните понятие группа в языке Пролог.**

Скобки образуют группу и тем самым ограничивают область действия.

**1.6.11. Назовите и объясните простейшие способы представления базы данных с помощью языка Пролог.**





**1.6.12. Определите следующие предикаты для работы со списками L: добавить(Х,L), принадлежит(X,L), объединить(L1,L2,L3).**

1. **добавить (X, L, [X|L]).**
2. **принадлежит(Х, [Х| Хвост ]).**

**принадлежит(X, [Голова|Хвост]): – принадлежит(X,[Хвост]).**

1. **объединить([], L, L).**

**объединить([X|L1], L2, [X|L3]) :- объединить(L1, L2, L3).**

**1.6.13. Определите операцию “проекция” в виде правила Пролога. Покажите, как с помощью этой операции реализуются отношения-селекторы.**

Операция "проекция" может быть реализована с помощью правила, которое фильтрует список и возвращает только те элементы, которые соответствуют определенному условию. Это позволяет реализовать отношения-селекторы, которые выбирают определенные элементы из списка на основе заданных критериев.

Вот пример реализации операции "проекция" в виде правила Пролога:

**проекция([], \_, []).**

**проекция([X|Tail], Condition, [X|Result]) :-**

**call(Condition, X),**

**проекция(Tail, Condition, Result).**

**проекция([\_|Tail], Condition, Result) :-**

**проекция(Tail, Condition, Result).**

**1.6.14. Объясните встроенный предикат bagof(X, C, L).**

В том случае, когда формируемые выборки необходимо собрать в список для дальнейшей обработки удобно использовать встроенный предикат bagof.

Цель bagof(X, C, L) формирует список L всех объектов Х, удовлетворяющих условию С. Например, требуется сформировать список L, представляющий подмножество сотрудников одного филиала:

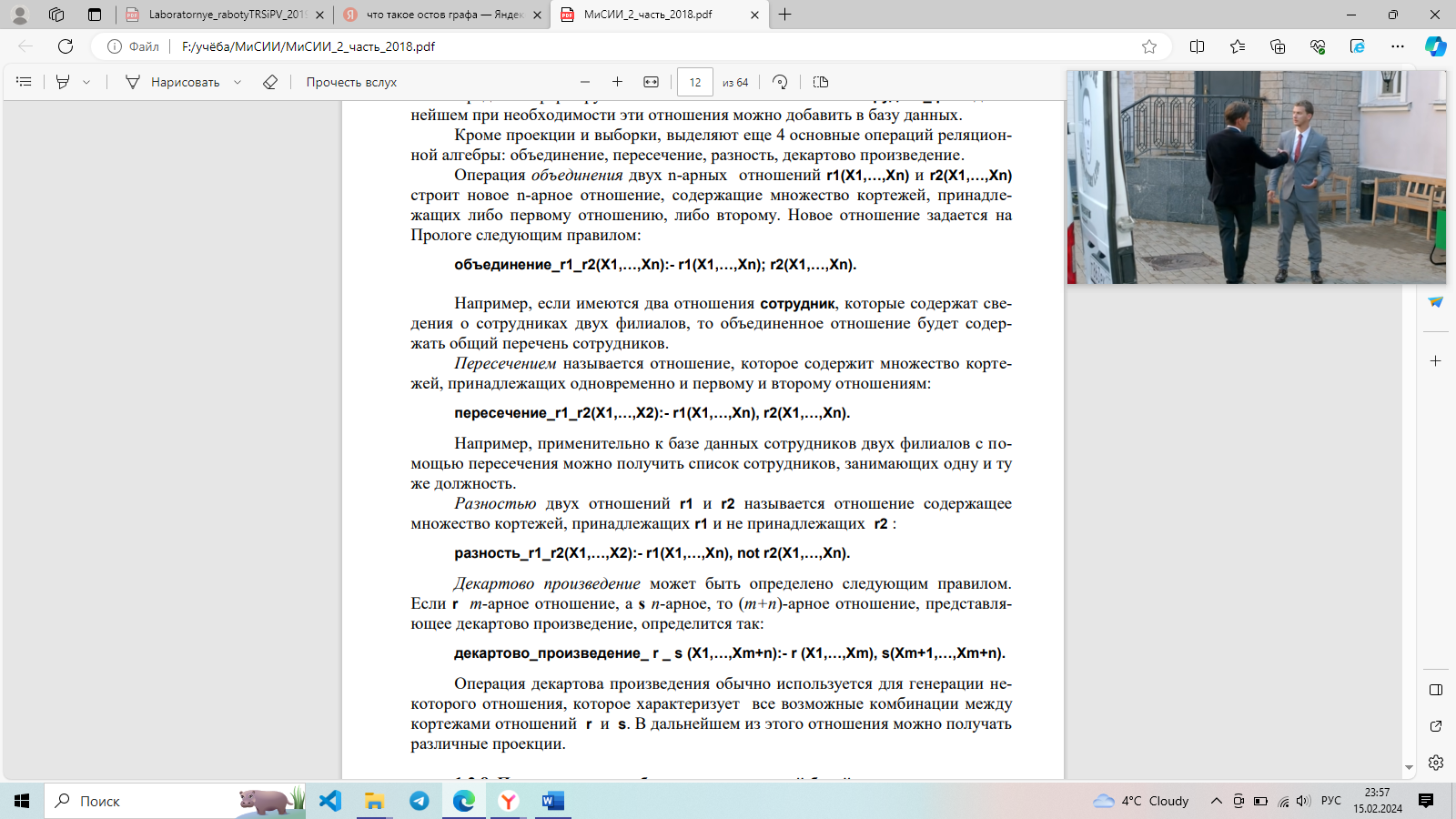
**подмножество\_сотрудников(Филиал,L):-**

**bagof(сотрудник\_ф(N,Фам,Имя,Отч,Отд,Должн,Филиал,Тел),**

**сотрудник(N,Фам,Имя,Отч,Отд,Должн,Филиал,Тел), L).**

Предикат сформирует список L из новых отношений сотрудник\_ф . В дальнейшем при необходимости эти отношения можно добавить в базу данных.

**1.6.15. Определите следующие операции реляционной алгебры в виде правил Пролога: объединение, пересечение, разность, декартово произведение.**



**1.6.16. Перечислите и объясните встроенные предикаты добавления утверждений в базу данных Пролога.**

Предикат asserta(X) добавляет утверждение Х в начало базы данных, предикат assertz(X) - в конец базы данных.

**1.6.17. Назовите и объясните встроенные предикаты для удаления утверждений из базы данных Пролога.**

Предикат retract(X) выполняет поиск утверждения Х в базе данных и удаляет первое найденное утверждение. Встроенный предикат retractall(X) удаляет из базы данных все утверждения, функтор и арность которых сопоставимы с Х.

**1.6.18. Как просмотреть утверждения базы данных Пролога?**

Для просмотра утверждений, входящих в базу данных, можно воспользоваться предикатами: listing выводит утверждения, содержащиеся в базе данных, в выходной поток; listing(X) - печатаются утверждения, сопоставимые с Х.

1.6.19. Что понимают под потоками ввода-вывода? Как открыть поток?

1.6.20. Какие предикаты используют для записи термов в поток?

1.6.21. Какие предикаты используют для чтения термов из потока?

1.6.22. Какие предикаты применяются для ввода-вывода символов?

1.6.23. Какое значение получает считываемый терм при достижении метки конец файла?

