МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет по лабораторной работе №2**

**Вариант 12**

По дисциплине: «Скриптовые языки программирования»

**«Обработка исключений. Функции»**

Выполнил:

студент группы

10702423

Татур Е. Ю.

Проверил**:**

ст.преподаватель

Козловская Ю.Б.

Минск 2024

**Цель работы**

Выполнить подробным образом задания из разных уровней, согласно варианта.

Номера задач взять из таблицы:

|  |  |  |  |
| --- | --- | --- | --- |
| **Вариант** | **Номера заданий** | **Проекты** | |
| 12. | 4 8 13 19 23 29 35 37 44 47 | 78 | 112 |

**Задания**

4. Напишите функцию `divide(a, b)`, которая возвращает результат деления `a` на `b`.

8. Реализуйте функцию `get\_positive\_integer()`, которая запрашивает у пользователя ввод положительного числа.

13. Напишите функцию `safe\_append(lst, item)`, которая добавляет элемент в список и обрабатывает исключение, если `lst` не является списком.

19. Напишите функцию `file\_info(filename)`, которая возвращает информацию о файле, но обрабатывает ошибки, если файл не существует или не является файлом.

23. Реализуйте функцию `calculate\_square\_root(number)`, которая обрабатывает исключение при попытке вычислить квадратный корень отрицательного числа.

29. Реализуйте функцию `write\_list\_to\_file(filename, lst)`, которая записывает список в файл и обрабатывает ошибки.

35. Реализуйте функцию `find\_item\_in\_list(lst, item)`, которая ищет элемент в списке и обрабатывает исключения.

37. Напишите функцию `get\_valid\_grade()`, которая запрашивает оценку (от 1 до 5) и обрабатывает исключения.

44. Реализуйте функцию `sum\_dict\_values(data)`, которая суммирует значения словаря и обрабатывает исключения.

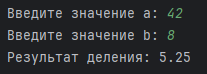
47. Реализуйте функцию `validate\_data(data)`, которая проверяет данные на соответствие определённым критериям и выбрасывает исключения.

78. Напишите программу, которая управляет списком книг с добавлением, удалением и поиском книг с обработкой исключений.

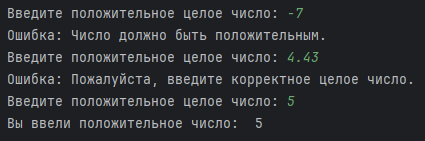
112. Создайте программу для работы с данными о студентах, позволяющую добавлять и редактировать информацию с обработкой ошибок.

**Ход выполнения работы**

**Результат работы**



*Рисунок 1 – Результат выполненного задания 4.*



*Рисунок 2 – Результат выполненного задания 8.*



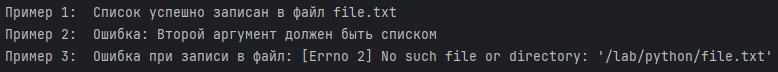
*Рисунок 3 – Результат выполненного задания 13.*



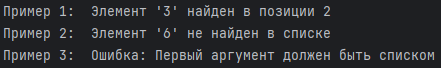
*Рисунок 4 – Результат выполненного задания 19.*



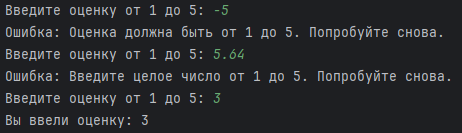
*Рисунок 5 – Результат выполненного задания 23.*



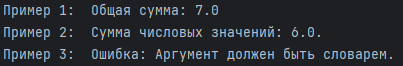
*Рисунок 6 – Результат выполненного задания 29.*



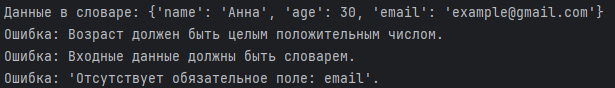
*Рисунок 7 – Результат выполненного задания 35.*



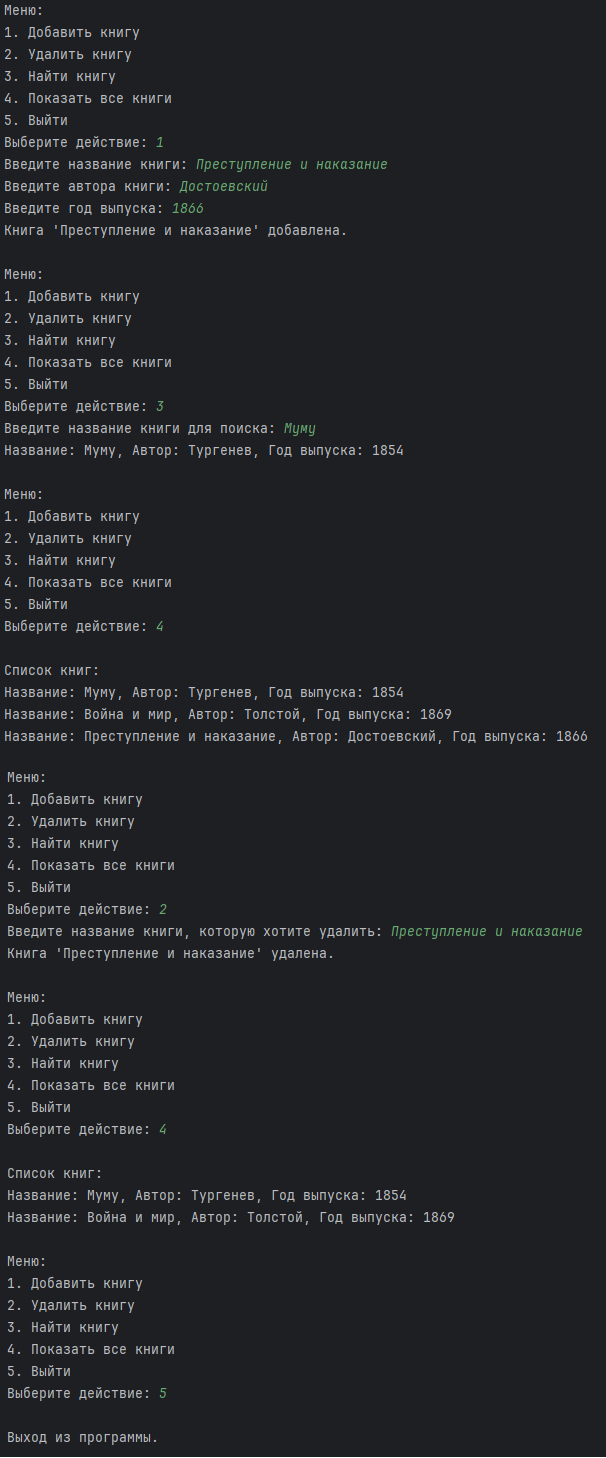
*Рисунок 8 – Результат выполненного задания 37.*

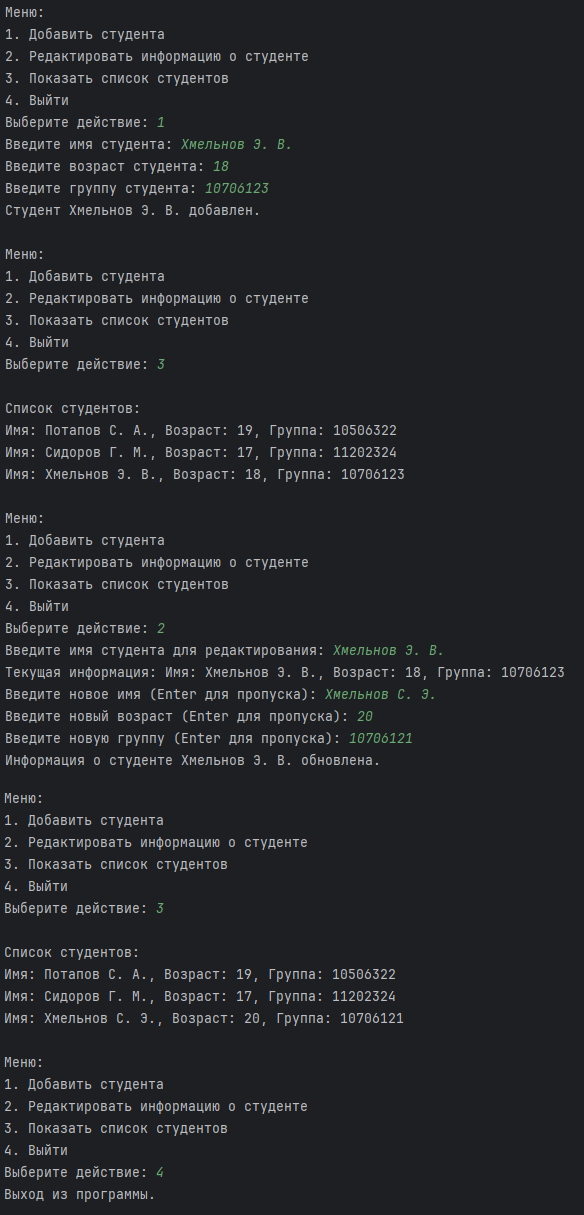


*Рисунок 9 – Результат выполненного задания 44.*



*Рисунок 10 – Результат выполненного задания 47.*

*Рисунок 11 – Результат выполненного задания 78.*

*Рисунок 12 – Результат выполненного задания 112.*

**Листинг**

**ex4.py:**

# Запрос значений у пользователя  
input\_a = float(input("Введите значение a: "))  
input\_b = float(input("Введите значение b: "))  
  
def divide(a, b):  
 try:  
 result = a / b  
 return result  
 except ZeroDivisionError:  
 return "Ошибка: Деление на ноль невозможно."  
  
# Примеры использования  
print(f"Результат деления: {divide(input\_a, input\_b)}")

**ex8.py:**

def get\_positive\_integer():  
 while True:  
 user\_input = input("Введите положительное целое число: ")  
 try:  
 value = int(user\_input)  
 if value > 0:  
 return value  
 else:  
 print("Ошибка: Число должно быть положительным.")  
 except ValueError:  
 print("Ошибка: Пожалуйста, введите корректное целое число.")  
  
# Пример использования  
print("Вы ввели положительное число: ", get\_positive\_integer())

**ex13.py:**

def safe\_append(lst, item):  
 if not isinstance(lst, list):  
 return "Ошибка: Первый аргумент должен быть списком."  
  
 lst.append(item)  
 return lst  
  
# Примеры использования  
result1 = safe\_append([1, 2, 3], 4)  
print("Ваш список: ", result1  
  
result2 = safe\_append("не список", 4)  
print("Ваш список: ", result2)

**ex19.py:**

import os  
  
def file\_info(filename):  
 try:  
 # Проверка, существует ли файл  
 if not os.path.exists(filename):  
 return f"Ошибка: файл '{filename}' не существует."  
  
 # Проверка, является ли это файлом  
 if not os.path.isfile(filename):  
 return f"Ошибка: '{filename}' не является файлом."  
  
 # Получение информации о файле  
 file\_name = os.path.basename(filename)  
 file\_size = os.path.getsize(filename)  
  
 return {  
 "Имя файла": file\_name,  
 "Размер файла (в байтах)": file\_size,  
 }  
  
 except Exception as e:  
 return f"Произошла ошибка: {str(e)}"  
  
  
# Примеры использования  
info1 = file\_info("example.txt")  
print(info1)  
  
info2 = file\_info("not\_example.txt")  
print(info2)

**ex23.py:**

import math  
  
  
def calculate\_square\_root(number):  
 if number < 0:  
 return "Ошибка: Невозможно вычислить квадратный корень отрицательного числа."  
  
  
 return math.sqrt(number)  
  
# Примеры использования  
result = float(input("Введите число: "))  
print(f"Корень: {calculate\_square\_root(result)}")

**ex29.py:**

def write\_list\_to\_file(filename, lst):  
 try:  
 if not isinstance(lst, list):  
 raise TypeError("Второй аргумент должен быть списком")  
  
 with open(filename, 'w') as file:  
 for item in lst:  
 file.write(str(item) + '\n')  
 return f"Список успешно записан в файл {filename}"  
 except IOError as e:  
 return f"Ошибка при записи в файл: {str(e)}"  
 except TypeError as e:  
 return f"Ошибка: {str(e)}"  
 except Exception as e:  
 return f"Произошла неожиданная ошибка: {str(e)}"  
  
  
# Примеры использования  
result1 = write\_list\_to\_file("file.txt", [1, 2, 3, 4, 5])  
print("Пример 1: ", result1)  
  
result2 = write\_list\_to\_file("file.txt", "не список")  
print("Пример 2: ", result2)  
  
result3 = write\_list\_to\_file("/lab/python/file.txt", [1, 2, 3, 4, 5])  
print("Пример 2: ", result3)

**ex35.py:**

def find\_item\_in\_list(lst, item):  
 try:  
 if not isinstance(lst, list):  
 raise TypeError("Первый аргумент должен быть списком")  
  
 index = lst.index(item)  
 return f"Элемент '{item}' найден в позиции {index}"  
 except ValueError:  
 return f"Элемент '{item}' не найден в списке"  
 except TypeError as e:  
 return f"Ошибка: {str(e)}"  
 except Exception as e:  
 return f"Произошла неожиданная ошибка: {str(e)}"  
  
  
# Примеры использования  
result1 = find\_item\_in\_list([1, 2, 3, 4, 5], 3)  
print("Пример 1: ", result1)  
  
result2 = find\_item\_in\_list([1, 2, 3, 4, 5], 6)  
print("Пример 2: ", result2)  
  
result3 = find\_item\_in\_list("не список", 3)  
print("Пример 3: ", result3)

**ex37.py:**

def get\_valid\_grade():  
 while True:  
 try:  
 grade = int(input("Введите оценку от 1 до 5: "))  
  
 if 1 <= grade <= 5:  
 return grade  
 else:  
 raise ValueError("Оценка должна быть от 1 до 5.")  
  
 except ValueError as e:  
 if str(e) == "Оценка должна быть от 1 до 5.":  
 print(f"Ошибка: {e} Попробуйте снова.")  
 else:  
 print("Ошибка: Введите целое число от 1 до 5. Попробуйте снова.")  
  
 except Exception as e:  
 print(f"Произошла неожиданная ошибка: {e}. Попробуйте снова.")  
  
  
# Пример использования  
user\_grade = get\_valid\_grade()  
print(f"Вы ввели оценку: {user\_grade}")

**ex44.py:**

def sum\_dict\_values(data):  
 if not isinstance(data, dict):  
 return "Ошибка: Аргумент должен быть словарем."  
  
 sum\_dict = 0  
 non\_numeric = []  
  
 for key, value in data.items():  
 try:  
 sum\_dict += float(value)  
 except (ValueError, TypeError):  
 non\_numeric.append(key)  
  
 if non\_numeric:  
 return f"Сумма числовых значений: {sum\_dict}."  
 else:  
 return f"Общая сумма: {sum\_dict}"  
  
  
# Примеры использования  
result1 = sum\_dict\_values({'a': 1.6, 'b': 2.4, 'c': 3})  
print("Пример 1: ", result1)  
  
result2 = sum\_dict\_values({'a': 1, 'b': '2', 'c': 3, 'd': 'четыре'})  
print("Пример 2: ", result2)  
  
result3 = sum\_dict\_values([1, 2, 3])  
print("Пример 3: ", result3)

**ex47.py:**

class InvalidDataError(Exception):  
 pass  
  
  
def validate\_data(data):  
 try:  
 if not isinstance(data, dict):  
 raise TypeError("Входные данные должны быть словарем")  
  
 required\_keys = ['name', 'age', 'email']  
 for key in required\_keys:  
 if key not in data:  
 raise KeyError(f"Отсутствует обязательное поле: {key}")  
  
 # Проверка возраста  
 if not isinstance(data['age'], int) or data['age'] < 0:  
 raise InvalidDataError("Возраст должен быть целым положительным числом")  
  
 # Проверка email  
 if not isinstance(data['email'], str) or '@' not in data['email']:  
 raise InvalidDataError("Некорректный формат email")  
  
 print("Данные в словаре:", data)  
 except (InvalidDataError, TypeError, KeyError) as e:  
 print(f"Ошибка: {e}.")  
  
  
# Примеры использования  
valid\_data1 = {'name': 'Анна', 'age': 30, 'email': 'example@gmail.com'}  
validate\_data(valid\_data1)  
  
  
# Примеры использования  
valid\_data2 = {'name': 'Илья', 'age': -5, 'email': 'example@yandex.com'}  
validate\_data(valid\_data2)  
  
  
not\_a\_dict2 = [1, 2, 3]  
validate\_data(not\_a\_dict2)  
  
  
missing\_key = {'name': 'Петр', 'age': 25}  
validate\_data(missing\_key)

**ex78.py:**

import os  
  
# Определяем путь к файлу для хранения списка книг  
BOOKS\_FILE = 'books.txt'  
  
  
# Функция для загрузки списка книг из файла  
def load\_books():  
 if not os.path.exists(BOOKS\_FILE):  
 return []  
  
 books = []  
 try:  
 with open(BOOKS\_FILE, 'r', encoding='utf-8') as file:  
 for line in file:  
 # Каждая строка книги в формате: Название | Автор | Год  
 title, author, year = line.strip().split(', ')  
 books.append({  
 "title": title,  
 "author": author,  
 "year": int(year)  
 })  
 except Exception as e:  
 print(f"Ошибка при чтении файла: {e}")  
  
 return books  
  
  
# Функция для сохранения списка книг в файл  
def save\_books(books):  
 try:  
 with open(BOOKS\_FILE, 'w', encoding='utf-8') as file:  
 for book in books:  
 file.write(f"{book['title']}, {book['author']}, {book['year']}\n")  
 except Exception as e:  
 print(f"Ошибка при сохранении данных в файл: {e}")  
  
  
# Функция для добавления книги  
def add\_book(title, author, year):  
 try:  
 age = int(year) # Проверяем, что возраст — это число  
 except ValueError:  
 print("Ошибка: введите верно год.")  
 return  
  
 books = load\_books()  
 books.append({  
 "title": title,  
 "author": author,  
 "year": year  
 })  
 save\_books(books)  
 print(f"Книга '{title}' добавлена.")  
  
  
# Функция для удаления книги по названию  
def remove\_book(title):  
 books = load\_books()  
 filtered\_books = [book for book in books if book['title'].lower() != title.lower()]  
  
 if len(books) == len(filtered\_books):  
 print(f"Книга '{title}' не найдена.")  
 else:  
 save\_books(filtered\_books)  
 print(f"Книга '{title}' удалена.")  
  
  
# Функция для поиска книги по названию  
def find\_book(title):  
 books = load\_books()  
 found\_books = [book for book in books if title.lower() in book['title'].lower()]  
  
 if found\_books:  
 for book in found\_books:  
 print(f"Название: {book['title']}, Автор: {book['author']}, Год выпуска: {book['year']}")  
 else:  
 print(f"Книга '{title}' не найдена.")  
  
  
# Функция для вывода всех книг  
def list\_books():  
 books = load\_books()  
 if not books:  
 print("Список книг пуст.")  
 else:  
 print("\nСписок книг:")  
 for book in books:  
 print(f"Название: {book['title']}, Автор: {book['author']}, Год выпуска: {book['year']}")  
  
  
# Основное меню программы  
def menu():  
 while True:  
 print("\nМеню:")  
 print("1. Добавить книгу")  
 print("2. Удалить книгу")  
 print("3. Найти книгу")  
 print("4. Показать все книги")  
 print("5. Выйти")  
 choice = input("Выберите действие: ")  
  
 if choice == '1':  
 title = input("Введите название книги: ")  
 author = input("Введите автора книги: ")  
 year = input("Введите год выпуска: ")  
 add\_book(title, author, year)  
 elif choice == '2':  
 title = input("Введите название книги, которую хотите удалить: ")  
 remove\_book(title)  
 elif choice == '3':  
 title = input("Введите название книги для поиска: ")  
 find\_book(title)  
 elif choice == '4':  
 list\_books()  
 elif choice == '5':  
 print("Выход из программы.")  
 break  
 else:  
 print("Некорректный выбор. Попробуйте снова.")  
  
  
# Запуск программы  
menu()

**ex112.py:**

import os  
  
# Определяем путь к файлу для хранения данных студентов  
STUDENTS\_FILE = 'students.txt'  
  
  
# Функция для загрузки данных студентов из файла  
def load\_students():  
 if not os.path.exists(STUDENTS\_FILE):  
 return []  
  
 students = []  
 try:  
 with open(STUDENTS\_FILE, 'r', encoding='utf-8') as file:  
 for line in file:  
 # Каждая строка студента в формате: Имя | Возраст | Группа  
 name, age, group = line.strip().split(', ')  
 students.append({  
 "name": name,  
 "age": int(age),  
 "group": group  
 })  
 except Exception as e:  
 print(f"Ошибка при чтении файла: {e}")  
  
 return students  
  
  
# Функция для сохранения данных студентов в файл  
def save\_students(students):  
 try:  
 with open(STUDENTS\_FILE, 'w', encoding='utf-8') as file:  
 for student in students:  
 file.write(f"{student['name']} | {student['age']} | {student['group']}\n")  
 except Exception as e:  
 print(f"Ошибка при сохранении данных в файл: {e}")  
  
  
# Функция для добавления нового студента  
def add\_student(name, age, group):  
 try:  
 age = int(age) # Проверяем, что возраст — это число  
 except ValueError:  
 print("Ошибка: введите верно возраст.")  
 return  
  
 students = load\_students()  
 students.append({  
 "name": name,  
 "age": age,  
 "group": group  
 })  
 save\_students(students)  
 print(f"Студент {name} добавлен.")  
  
  
# Функция для редактирования информации о студенте  
def edit\_student(name):  
 students = load\_students()  
 for student in students:  
 if student['name'].lower() == name.lower():  
 print(f"Текущая информация: Имя: {student['name']}, Возраст: {student['age']}, Группа: {student['group']}")  
 new\_name = input("Введите новое имя (Enter для пропуска): ")  
 new\_age = input("Введите новый возраст (Enter для пропуска): ")  
 new\_group = input("Введите новую группу (Enter для пропуска): ")  
  
 if new\_name:  
 student['name'] = new\_name  
 if new\_age:  
 try:  
 student['age'] = int(new\_age) # Проверяем, что возраст — это число  
 except ValueError:  
 print("Ошибка: возраст должен быть числом.")  
 return  
 if new\_group:  
 student['group'] = new\_group  
  
 save\_students(students)  
 print(f"Информация о студенте {name} обновлена.")  
 return  
  
 print(f"Студент с именем {name} не найден.")  
  
  
# Функция для вывода всех студентов  
def list\_students():  
 students = load\_students()  
 if not students:  
 print("Список студентов пуст.")  
 else:  
 print("\nСписок студентов:")  
 for student in students:  
 print(f"Имя: {student['name']}, Возраст: {student['age']}, Группа: {student['group']}")  
  
  
# Основное меню программы  
def menu():  
 while True:  
 print("\nМеню:")  
 print("1. Добавить студента")  
 print("2. Редактировать информацию о студенте")  
 print("3. Показать список студентов")  
 print("4. Выйти")  
 choice = input("Выберите действие: ")  
  
 if choice == '1':  
 name = input("Введите имя студента: ")  
 age = input("Введите возраст студента: ")  
 group = input("Введите группу студента: ")  
 add\_student(name, age, group)  
 elif choice == '2':  
 name = input("Введите имя студента для редактирования: ")  
 edit\_student(name)  
 elif choice == '3':  
 list\_students()  
 elif choice == '4':  
 print("Выход из программы.")  
 break  
 else:  
 print("Некорректный выбор. Попробуйте снова.")  
  
  
# Запуск программы  
menu()

**Контрольные вопросы**

1. В чем заключается обработка исключений?

Обработка исключений в Python позволяет управлять ошибками, которые могут возникнуть во время выполнения программы, предотвращая её аварийное завершение. Вместо того чтобы программа завершалась при возникновении ошибки, мы можем перехватить это исключение, выполнить необходимые действия (например, выдать предупреждение или попытаться повторить операцию) и продолжить работу программы. Основные компоненты обработки исключений включают:

1. **try**: В этом блоке размещается код, который может вызвать исключение. Если в процессе выполнения этого кода возникает ошибка, выполнение переходит к блоку except.
2. **except**: Этот блок обрабатывает исключения, возникающие в блоке try. Вы можете указать конкретный тип исключения для обработки или использовать общий обработчик для всех типов ошибок.
3. **else**: Этот блок выполняется, если в блоке try не возникло никаких исключений. Он позволяет выполнять код, который должен выполняться только при отсутствии ошибок.
4. **finally**: Этот блок выполняется всегда, независимо от того, произошло ли исключение или нет. Он часто используется для освобождения ресурсов, таких как закрытие файлов или соединений.
5. Какие функции вы использовали в ходе выполнения лабораторной работы?

В процессе выполнения задачи были использованы следующие функции:

1. Функция `divide(a, b)` возвращает результат деления `a` на `b`.
2. Функция `get\_positive\_integer()` запрашивает у пользователя ввод положительного числа.
3. Функция `calculate\_square\_root(number)` обрабатывает исключение при попытке вычислить квадратный корень отрицательного числа.
4. Функция `get\_valid\_grade()` запрашивает оценку (от 1 до 5) и обрабатывает исключения.
5. Функция `sum\_dict\_values(data)` суммирует значения словаря и обрабатывает исключения.
6. И др.
7. Можно ли было реализовать выполнение предложенных заданий другими инструментами? Какими?
8. Вместо использования механизма обработки исключений можно было бы реализовать цикл с явной проверкой возвращаемых значений, например, возвращать коды состояния (успех или ошибка) и перезапускать функцию на основе этих кодов.
9. Использование сторонних библиотек: Вместо создания собственной функции retry, можно было бы использовать готовые библиотеки, которые предлагают более гибкие средства для повторного выполнения функций при ошибках.