**1 Постановка задачи**

Целью данной лабораторной работы является разработка программы на языке Python, которая способна вычислять наибольший общий делитель (НОД) двух положительных целых чисел. Программа должна учитывать, что входные данные могут быть различными, и обрабатывать случаи, когда числа не удовлетворяют условиям задачи (например, когда одно из чисел отрицательное или равно нулю), сообщая пользователю об ошибке.

**2 Составление спецификации требований**

Введение

а) Цели

Целью программы является предоставление пользователю возможности быстро и эффективно находить наибольший общий делитель для любой пары положительных целых чисел. Программа должна быть простой в использовании и доступной для людей без специальных технических знаний.

б) Соглашения о терминах

НОД (Наибольший Общий Делитель) – наибольшее число, на которое два или более чисел делятся без остатка.

в) Ссылки на источники

Документация и теоретические материалы по алгоритмам нахождения НОД.

Общее описание

а) Видение продукта

Программа является прикладным инструментом, предназначенным для образовательных целей и повседневного использования, когда необходимо найти НОД двух чисел, например, для решения математических задач или при программировании.

б) Функциональность продукта

Программа обеспечивает ввод двух положительных целых чисел и вычисляет их наибольший общий делитель. В случае ввода некорректных данных (отрицательные числа, ноль), программа выводит сообщение об ошибке.

в) Классы и характеристики пользователей

Программа предназначена для широкого круга пользователей: от школьников и студентов, изучающих математику, до программистов и научных сотрудников, работающих с числовыми данными.

Функциональные требования

Ввод данных: Программа должна предоставлять пользователю возможность ввести два положительных целых числа.

Обработка ошибок: Программа должна проверять введенные данные на соответствие условиям (положительность и целостность чисел) и, в случае обнаружения ошибки, выводить соответствующее сообщение.

Вычисление и вывод результатов: На основе введенных данных программа вычисляет НОД и выводит его на экран. В случае ошибки пользователю предлагается повторить ввод.

Эти элементы спецификации обеспечивают четкое и полное описание требований к программе, определяя ее основные функции и предполагаемое использование.

**3 Шаги тестирования**

Первый цикл тестирования

Шаг 1. Начало тестирования с простого и наиболее очевидного теста

Первым делом стоит запустить программу с простым набором данных, чтобы убедиться в её стабильности и работоспособности. Например, можно использовать пару чисел 25 и 5, ожидаемый результат – 5. Последовательность действий и результаты приведены в таблице 1.1.

Таблица 1.1 – Порядок действия и результаты сложения чисел 2 и 3

|  |  |  |
| --- | --- | --- |
| **№ действия** | **Ввод** | **Вывод** |
| 1 | Запускаете программу | В командной строке появилась надпись «Введите первое число» |
| 2 | Введите число 25 | В командной строке после надписи «Введите первое число» появилось число 25 |
| 3 | Нажмите «Enter» | В командной строке появилась надпись «Введите второе число» |
| 4 | Введите число 5 | В командной строке после надписи «Введите второе число» появилось число 5 |
| 5 | Нажмите «Enter» | В командной строке появилась надпись «Наибольший общий делитель: 5» |

На рисунке 1 видно, как выглядит экран по окончанию теста. Курсор указывает, где будет отображаться следующее вводимое число.

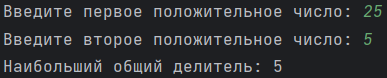


Рисунок 1 – Вид командной строки по окончанию теста

Шаг 2. Составление заметок о том, что ещё должно быть протестировано

Далее необходимо составить список дополнительных тестов, включая:

1. Проверку граничных условий (например, очень большие числа).
2. Проверку на ввод некорректных данных (отрицательные числа, нули, нечисловые значения).
3. Проверку на ввод только одного числа или пустого ввода.

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый**  **результат** | **Замечания** |
| 25 5 | 5 | Результат соответствует ожиданиям, так как 5 является наибольшим общим делителем для чисел 25 и 5. |
| 857129835798127359 283758721835721355 | 1 | Результат соответствует ожиданиям, подтверждая, что любое число является НОД самому себе. |
| 0 5 | Ошибка | Результат соответствует ожиданиям, так как НОД с числом 0 не определён для положительных чисел больше нуля. Правильно обрабатывается крайний случай. |
| -5 5 | Ошибка | Результат соответствует ожиданиям, подчёркивая, что функция корректно обрабатывает ситуацию с отрицательным числом, возвращая ошибку, поскольку оба числа должны быть положительными. |

Шаг 3. Проверка допустимых значений и наблюдение за реакцией программы

Создание и выполнение тестов для проверки реакции программы на допустимые и недопустимые входные данные. Особое внимание уделить обработке ошибок и выводу соответствующих сообщений.

Шаг 4. Тестирование в режиме «свободного полёта»

Использование программы в менее структурированном порядке, ввод различных комбинаций данных, в том числе не валидных, для выявления потенциальных недочётов и ошибок, которые не были покрыты формальными тестами.

|  |  |  |
| --- | --- | --- |
| **Тест** | **Особенности теста** | **Замечания** |
| <*Enter*> +  <*Enter*> | Проверка реакции программы на отсутствие числовых данных. | Вывелась ошибка: было введено не числовое значение. |
| 1,5 0,5 | Числа с десятичной дробью. | Вывелась ошибка: было введено не числовое значение. |
| a b | Недопустимые символы | Вывелась ошибка: было введено не числовое значение. |

Итоги первого цикла тестирования программы нахождения НОД

Стартовали с базового теста, проверяя программу на паре чисел с известным НОД. Успешное прохождение этого теста подтвердило основную работоспособность программы. Далее, были проведены формальные тесты с разнообразными валидными входными данными, которые показали корректность основных вычислений программы.

В процессе тестирования обнаружены недочеты в обработке некорректных входных данных, таких как отрицательные числа и нули. Это указало на необходимость усовершенствования валидации ввода и улучшения сообщений об ошибках для пользователей.

Было решено провести дополнительные неформальные тесты для исследования устойчивости программы к неожиданным или неправильным вводам данных. Эти тесты выявили необходимость дальнейших улучшений программы, включая оптимизацию интерфейса и логики обработки ошибок.

На основании этих тестов сформирован список рекомендаций для улучшения программы, который будет использован при разработке следующей версии. Первый цикл тестирования подчеркнул важность комплексного тестирования на разных этапах разработки для гарантии качества и надежности программного продукта.

Завершение первого цикла тестирования является лишь началом процесса улучшения и оптимизации программы, предвещая более глубокое тестирование и доработку в будущем.

Второй цикл тестирования программы нахождения НОД

После первого цикла тестирования и обсуждения с программистом были получены следующие резолюции по отчетам о найденных ошибках и предложениях по улучшению программы:

Ошибка проектирования: На экране нет названия программы. Резолюция: Не будет исправлена, так как фокус на функциональности и производительности.

Ошибка проектирования: На экране нет инструкций. Резолюция: Не будет исправлена. Программист считает, что дополнительные инструкции могут замедлить работу программы.

Ошибка проектирования: Невозможность остановить программу. Резолюция: Исправлена. Добавлена подсказка «Для выхода нажмите <Ctrl + C>».

Ошибка кодирования: Некорректная обработка отрицательных чисел. Резолюция: Исправлена. Программа теперь корректно обрабатывает отрицательные числа.

Ошибка кодирования: Программа интерпретирует третий введенный символ как нажатие <Enter>. Резолюция: В работе, пока не исправлена.

Ошибка кодирования: Проблемы при вводе нечисловых данных и управляющих символов. Резолюция: Остаются без изменений, с рекомендацией пользователю избегать такого ввода.

Шаг 1. Ознакомление с резолюцией программиста

Учитывая резолюции программиста, тестовый план адаптируется так, чтобы сосредоточить внимание на улучшенной обработке отрицательных чисел и проверке новой функции выхода из программы. Тесты, связанные с интерфейсом пользователя и нечисловыми данными, будут исключены из текущего цикла.

Шаг 2. Анализ комментариев и подготовка новых тестов

На основе полученных резолюций и предыдущих замечаний будет сформирована серия новых тестов, включая:

Проверка обработки отрицательных чисел.

Проверка реакции программы на комбинации клавиш, в том числе <Ctrl + C> для выхода.

Повторение успешных тестов с положительными числами для подтверждения стабильности улучшенной версии программы.

Шаг 3. Проведение тестов и наблюдение за реакцией программы

Тестирование будет включать в себя как структурированные тесты, так и свободные эксперименты для выявления неучтенных сценариев использования и потенциальных уязвимостей в новой версии программы.

Итоги второго цикла тестирования

Второй цикл тестирования должен подтвердить, что внесенные изменения улучшили функциональность программы, а также повысить уверенность в ее надежности и удобстве использования. Важно также обратить внимание на то, как программа обрабатывает граничные случаи и реагирует на ошибочный ввод, чтобы гарантировать, что пользовательский опыт остается на высоком уровне. Все выявленные в ходе этого цикла проблемы и замечания будут документированы для дальнейшего анализа и улучшения программы.

**4 Результаты работы программы**

Результаты работы программы представлены на рисунках 2, 3 и 4.

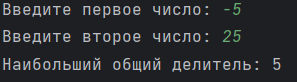


Рисунок 2 – Результат работы с отрицательным числом

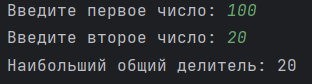


Рисунок 3 – Результат работы с обычными числами

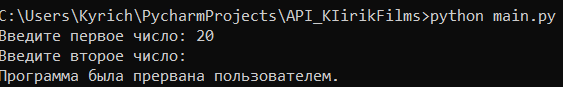


Рисунок 4 – Проверка закрытия программы при нажатии <Ctrl+C>

**5 Отчет о проблеме**



**6 Листинг программы**

Первая версия:

def gcd(x1, x2):  
 if x1 < 0 or x2 < 0:  
 return "Ошибка: оба числа должны быть положительными и больше нуля."  
 while x2:  
 x1, x2 = x2, x1 % x2  
 return x1  
  
def main():  
 try:  
 x1 = int(input("Введите первое число: "))  
 x2 = int(input("Введите второе число: "))  
 result = gcd(x1, x2)  
 print(f"Наибольший общий делитель: {result}")  
 except ValueError:  
 print("Ошибка: было введено не числовое значение.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Вторая версия:

def gcd(x1, x2):  
 if x1 == 0 or x2 == 0:  
 return "Ошибка: оба числа не должны быть равны нулю."  
 while x2:  
 x1, x2 = x2, x1 % x2  
 return x1  
  
def main():  
 try:  
 x1 = int(input("Введите первое число: "))  
 x2 = int(input("Введите второе число: "))  
 result = gcd(x1, x2)  
 print(f"Наибольший общий делитель: {result}")  
 except KeyboardInterrupt:  
 print("\nПрограмма была прервана пользователем.")  
 exit(0)  
 except ValueError:  
 print("Ошибка: было введено не числовое значение.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()