Студент группы ИС-26 Жирков В.А

Практическое занятие № 4

Вариант 10

Тема: составление программ циклической структуры в IDE PyCharm Community.

Цель: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ циклической структуры в IDE PyCharm Community.

Постановка задачи 1

Дано целое число N (>0). Найти произведение 1.1 • 1.2 • 1.3 •... (N сомножителей).

Алгоритм

1. Ввод данных:

   • Запросить у пользователя ввод целого числа N, где N > 0.

2. Проверка условия:

   • Если N <= 0, выдать сообщение об ошибке и завершить выполнение программы.

3. Инициализация переменной:

• Создать переменную total_sum и присвоить ей значение 0 для хранения суммы квадратов.

4. Цикл для вычисления суммы квадратов:

• Использовать цикл for, чтобы перебрать все целые числа от N до 2N включительно:

• Для каждого числа i в диапазоне от N до 2N:

• Вычислить квадрат числа i (i ** 2).

• Добавить этот квадрат к переменной total_sum.

5. Возврат результата:

• Вернуть итоговое значение total_sum.

6. Вывод результата:

• Вывести на экран сумму квадратов от N до 2N.

Блок-схема

Текст программы на Python

```
def calculate_sum_of_squares(N):
    try:
        if N <= 0:
```

```
            raise ValueError("N должно быть больше 0.")

        total_sum = 0
        for i in range(N, 2 * N + 1):
            total_sum += i ** 2

        return total_sum

    except Exception as e:
        return f"Произошла ошибка: {e}"

N = int(input("Введите целое число N (> 0): "))
result_sum = calculate_sum_of_squares(N)
print("Сумма квадратов от", N, "до", 2 * N, "равна:", result_sum)
```

Протокол работы программы

1. Пользователь запускает программу.

2. Программа запрашивает ввод числа N.

3. Пользователь вводит значение N (например, 3).

4. Программа проверяет, что N > 0. Если условие не выполняется, выводит сообщение об ошибке.

5. Программа инициализирует переменную total_sum со значением 0.

6. Программа начинает цикл от N до 2N (в данном случае от 3 до 6):

- Для i = 3: total_sum = 0 + 3^2 = 9
- Для i = 4: total_sum = 9 + 4^2 = 25
- Для i = 5: total_sum = 25 + 5^2 = 50
- Для i = 6: total_sum = 50 + 6^2 = 86

7. Программа завершает цикл и возвращает значение total_sum.

8. Программа выводит на экран результат: "Сумма квадратов от 3 до 6 равна: 86".

Постановка задачи 2

Дано целое число N (> 1). Найти наименьшее целое число K, при котором выполняется неравенство 3K > N.

Алгоритм

1. Ввод данных:

- Запросить у пользователя ввод целого числа N, где N > 1.

2. Проверка условия:

• Если N <= 1, выдать сообщение об ошибке и завершить выполнение программы.

3. Инициализация переменной:

• Создать переменную K и присвоить ей значение 0 для хранения наименьшего целого числа K.

4. Цикл для нахождения K:

• Использовать цикл while, который будет продолжаться до тех пор, пока не выполнится условие 3 * (K + 1) < N:

• Увеличивать значение K на 1 в каждой итерации цикла.

5. Возврат результата:

• Вернуть итоговое значение K.

6. Обработка исключений:

• В случае возникновения ошибки (например, если пользователь вводит некорректные данные), вернуть сообщение об ошибке.

7. Вывод результата:

• Вывести на экран наименьшее целое число K, при котором выполняется неравенство 3K > N.

Блок-схема

Текст программы на Python

```
def find_max_K(N):
    try:
        if N <= 1:
            raise ValueError("N должно быть больше 1.")

        K = 0

        while 3 * (K + 1) < N:
            K += 1

        return K

    except Exception as e:
        return f"Произошла ошибка: {e}"

N = int(input("Введите целое число N (> 1): "))
```

```
result_K = find_max_K(N)

print("Наибольшее целое число K, при котором 3K < N:",
result_K)
```

Протокол работы программы

1. Пользователь запускает программу.

2. Программа запрашивает ввод числа N: "Введите целое число N (> 1): ".

3. Пользователь вводит значение N (например, 5).

4. Программа проверяет, что N > 1. Если условие не выполняется, генерируется исключение с сообщением "N должно быть больше 1.".

5. Программа инициализирует переменную K со значением 0.

6. Программа начинает цикл while, проверяя условие 3 * (K + 1) < N:

- Для K = 0: 3 * (0 + 1) = 3 < 5 → K увеличивается до 1.
- Для K = 1: 3 * (1 + 1) = 6 < 5 → цикл завершается.

7. Программа возвращает значение K (в данном случае K = 1).

8. Программа выводит на экран результат: "Наибольшее целое число K, при котором 3K < N: 1".

Вывод

В результате выполнения данной работы были успешно освоены основные принципы работы с циклическими структурами в Python. Овладение навыками обработки пользовательского ввода и исключений создало прочную основу для дальнейшего изучения языка программирования. Полученные знания будут полезны при решении более сложных задач и разработке программ различной сложности. Работа с IDE PyCharm Community также показала, как удобные инструменты могут повысить эффективность разработки и улучшить качество кода.