|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Язык программирования Python»  Вариант 1 | | |
|  | | Работу выполнила  студентка группы ИТ-6 1 курса  Булгакова Ю.О  «25» июня 2025 г. |
| Работу проверил  Рубцова М.Б  «\_\_\_» июня 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc201788130)

[Алгоритм решения 4](#_Toc201788131)

[1. Разработка алгоритма 4](#_Toc201788132)

[1.1 Поддержка вложенных выражений 4](#_Toc201788133)

[1.2 Проверка корректности ввода 4](#_Toc201788134)

[1.3 Интерактивный интерфейс 4](#_Toc201788135)

[2. Выбранные структуры данных 4](#_Toc201788136)

[2.1 Класс Stack (стек) 5](#_Toc201788137)

[2.2 Класс ExpressionCalculator 5](#_Toc201788138)

[3. Логика работы 5](#_Toc201788139)

[Тестирование 7](#_Toc201788140)

[Код программы 10](#_Toc201788141)

[Инструкция по применению стилей и оформлению работы 11](#_Toc201788142)

# Постановка задачи

Используя структуру стека, вычислить выражение, содержащее две операции: поиск минимума (обозначается m(,)) и поиск максимума (обозначается М(,)). Операции могут быть вложенными, например, M(15, m(16,8)) (в данном выражении ищем минимум из 16 и 8, а потом ищем максимум от результата m(16,8) и 15, ответ 15). В качестве аргументов могут использоваться только целые положительные числа. Строка, вводимая пользователем, корректна и не содержит пробелов.

# Алгоритм решения

1. Разработка алгоритма  
Требовалось разработать программу-калькулятор, способную вычислять математические выражения с операциями нахождения минимума (min) и максимума (max). Программа должна удовлетворять следующим требованиям:

## 1.1 Поддержка вложенных выражений

Обработка сложных конструкций, например: M(15, m(16, 8)) → 15.

Корректная интерпретация приоритета операций.

## 1.2 Проверка корректности ввода

Обработка ошибок:

Неправильное количество аргументов (например, m(5,)).

Недопустимые символы (например, k(1, 2)).

Несбалансированные скобки (например, m(5,10).

## 1.3 Интерактивный интерфейс

Возможность ввода выражений вручную.

Вывод результата или сообщения об ошибке.

Команда для выхода из программы (выход)

# **2.** Выбранные структуры данных

Для реализации калькулятора использованы:

## 2.1 Класс Stack (стек)

Реализован через односвязный список

Структура:

StackNode – класс узла (хранит значений и ссылку на следующий элемент)  
 Stack – основной класс стек

Операции:

push(item) – добавление элемента в вершину списка.

pop() – удаление и возврат элемента из вершины.

peek() – просмотр верхнего элемента без удаления.

is\_empty() – проверка на пустоту.

get\_size() – метод для получения количества элементов.

Обоснование выбора:

Односвязный список оптимален для реализации стека, так как все операции выполняются с вершиной

Экономия памяти по сравнению с двусвязными списками  
 Динамическое расширения без ограничений размера  
 Все операции выполняются за О(1)

## 2.2 Класс ExpressionCalculator

Содержит словарь операций: {'m': min, 'M': max}.

Методы:

apply\_operation(stack) – извлекает аргументы из стека, проверяет их количество и применяет операцию.

calculate(expression) – парсит выражение, используя стек, и возвращает результат.

# 3. Логика работы

Особенности реализации стека:

1. Вершина стека всегда указывает на последний добавленный элемент

2. При добавлении нового элемента создается узел, который становится новой вершиной

1. При извлечении элемента вершина перемещается на следующий узел
2. Счетчик размера поддерживается для быстрого доступа к количеству элементов

Разбор выражения:

1. Посимвольный обход строки.
2. Игнорирование пробелов и запятых.
3. Обработка чисел (последовательность цифр → целое число).
4. Фиксация операций (m или M) и скобок.

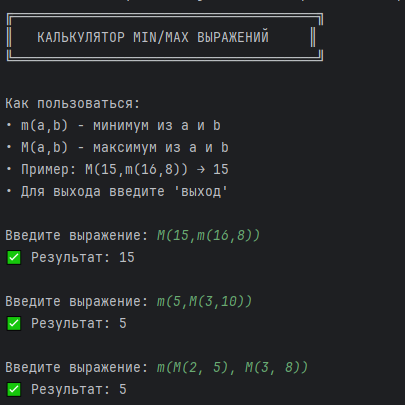
Вычисление вложенных операций:

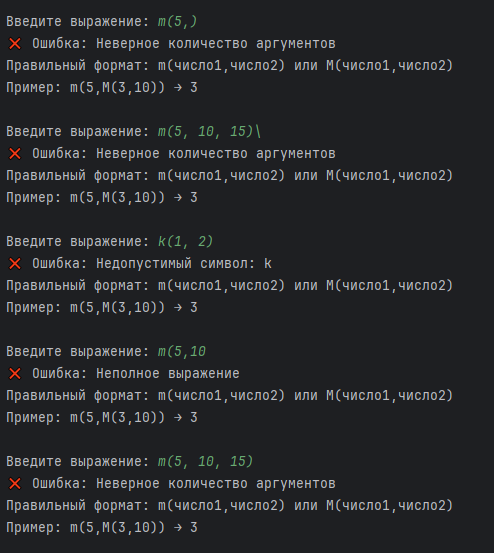
1. При обнаружении «)» извлекаются аргументы до ближайшей «(».
2. Проверяется, что аргументов ровно 2.
3. Применяется соответствующая операция (min или max).

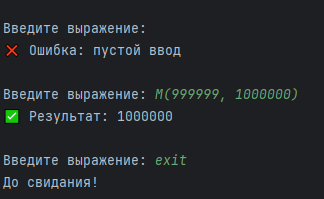
Обработка ошибок:

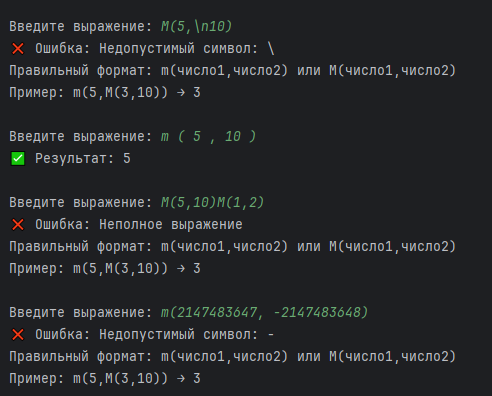
1. Неправильное количество аргументов → ValueError.
2. Неизвестная операция → ValueError.
3. Недопустимые символы → ValueError.
4. Несбалансированные скобки → ValueError.

# Тестирование

Корректные выражения:  
 

Ошибочные выражения:  
 

Граничные случаи:  
 

Особые случаи:  
 

# Код программы

class StackNode:  
 def \_\_init\_\_(self, value):  
 self.value = value  
 self.next = None  
  
  
class Stack:  
 def \_\_init\_\_(self):  
 self.top = None  
 self.size = 0  
  
 def push(self, item):  
 """Добавляет элемент на вершину стека"""  
 new\_node = StackNode(item)  
 new\_node.next = self.top  
 self.top = new\_node  
 self.size += 1  
  
 def pop(self):  
 """Извлекает элемент с вершины стека"""  
 if self.is\_empty():  
 raise IndexError("Попытка извлечения из пустого стека")  
 value = self.top.value  
 self.top = self.top.next  
 self.size -= 1  
 return value  
  
 def peek(self):  
 """Возвращает элемент с вершины стека без удаления"""  
 if self.is\_empty():  
 raise IndexError("Попытка просмотра пустого стека")  
 return self.top.value  
  
 def is\_empty(self):  
 """Проверяет наличие в стеке данных"""  
 return self.top is None  
  
 def get\_size(self):  
 """Возвращает размер стека"""  
 return self.size  
  
  
class ExpressionCalculator:  
 """Калькулятор min и max"""  
  
 def \_\_init\_\_(self):  
 self.operations = {'m': min, 'M': max}  
  
 def apply\_operation(self, stack):  
 """Применяет операцию к аргументам из стека"""  
 # Извлекаем второй аргумент  
 if stack.is\_empty():  
 raise ValueError("Недостаточно аргументов")  
 arg2 = stack.pop()  
  
 # Проверяем наличие первого аргумента  
 if stack.is\_empty() or stack.peek() == '(':  
 raise ValueError("Недостаточно аргументов")  
 arg1 = stack.pop()  
  
 # Проверяем открывающую скобку  
 if stack.is\_empty() or stack.pop() != '(':  
 raise ValueError("Несбалансированные скобки")  
  
 # Проверяем наличие операции  
 if stack.is\_empty():  
 raise ValueError("Не найдена операция")  
  
 operation\_char = stack.pop()  
 if operation\_char not in self.operations:  
 raise ValueError(f"Неизвестная операция: '{operation\_char}'")  
  
 return self.operations[operation\_char](arg1, arg2)  
  
 def calculate(self, expression):  
 """Вычисляет значение выражения"""  
 expression = expression.replace(' ', '')  
  
 if not expression:  
 raise ValueError("Пустое выражение")  
  
 stack = Stack()  
 i = 0  
 n = len(expression)  
  
 while i < n:  
 char = expression[i]  
  
 if char in self.operations:  
 stack.push(char)  
 i += 1  
 elif char == '(':  
 stack.push(char)  
 i += 1  
 elif char == ')':  
 result = self.apply\_operation(stack)  
 stack.push(result)  
 i += 1  
 elif char.isdigit():  
 num\_str = []  
 while i < n and expression[i].isdigit():  
 num\_str.append(expression[i])  
 i += 1  
 stack.push(int(''.join(num\_str)))  
 elif char == ',':  
 i += 1  
 else:  
 raise ValueError(f"Недопустимый символ: '{char}'")  
  
 if stack.get\_size() != 1:  
 raise ValueError("Неполное выражение")  
  
 return stack.pop()  
  
  
def main():  
 calculator = ExpressionCalculator()  
  
 print("КАЛЬКУЛЯТОР MIN/MAX ВЫРАЖЕНИЙ")  
 print("\nИнструкция:")  
 print("• m(a,b) - минимум из a и b")  
 print("• M(a,b) - максимум из a и b")  
 print("• Примеры: m(5,10) → 5, M(15,m(16,8)) → 15")  
 print("\nВведите 'выход' для завершения")  
  
 while True:  
 user\_input = input("\n> Введите выражение: ").strip()  
  
 if user\_input.lower() == 'выход':  
 print("До свидания!")  
 break  
  
 try:  
 result = calculator.calculate(user\_input)  
 print(f"Результат: {result}")  
 except Exception as e:  
 print(f"Ошибка: {str(e)}")  
 print("Формат: m(число1,число2) или M(число1,число2)")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()