Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Теория автоматов**

(название дисциплины)

**Лабораторная (практическая) работа №3**

**Элементы языка сценариев. Арифметические выражения**

(название (тема) работы)

**Вариант №8**

Выполнил

Студент группы ИВТАСбд-21

Зюзин Г.А.

(Фамилия И.О.)

Проверил(а):

ст. преподаватель кафедры «ВТ»\_\_\_\_

(должность)

\_\_\_\_\_\_\_\_\_Лылова А.В.\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)

Ульяновск

2023

# Постановка задачи

Выполнить программную реализацию автомата Мура в виде класса, содержащего два метода. Первый – проверяет выражение на корректность и возвращает True или False соответственно, второй - выполняет вычисления в обратной польской записи. Программа работает с числами с плавающей точкой от -32768 до 32767.

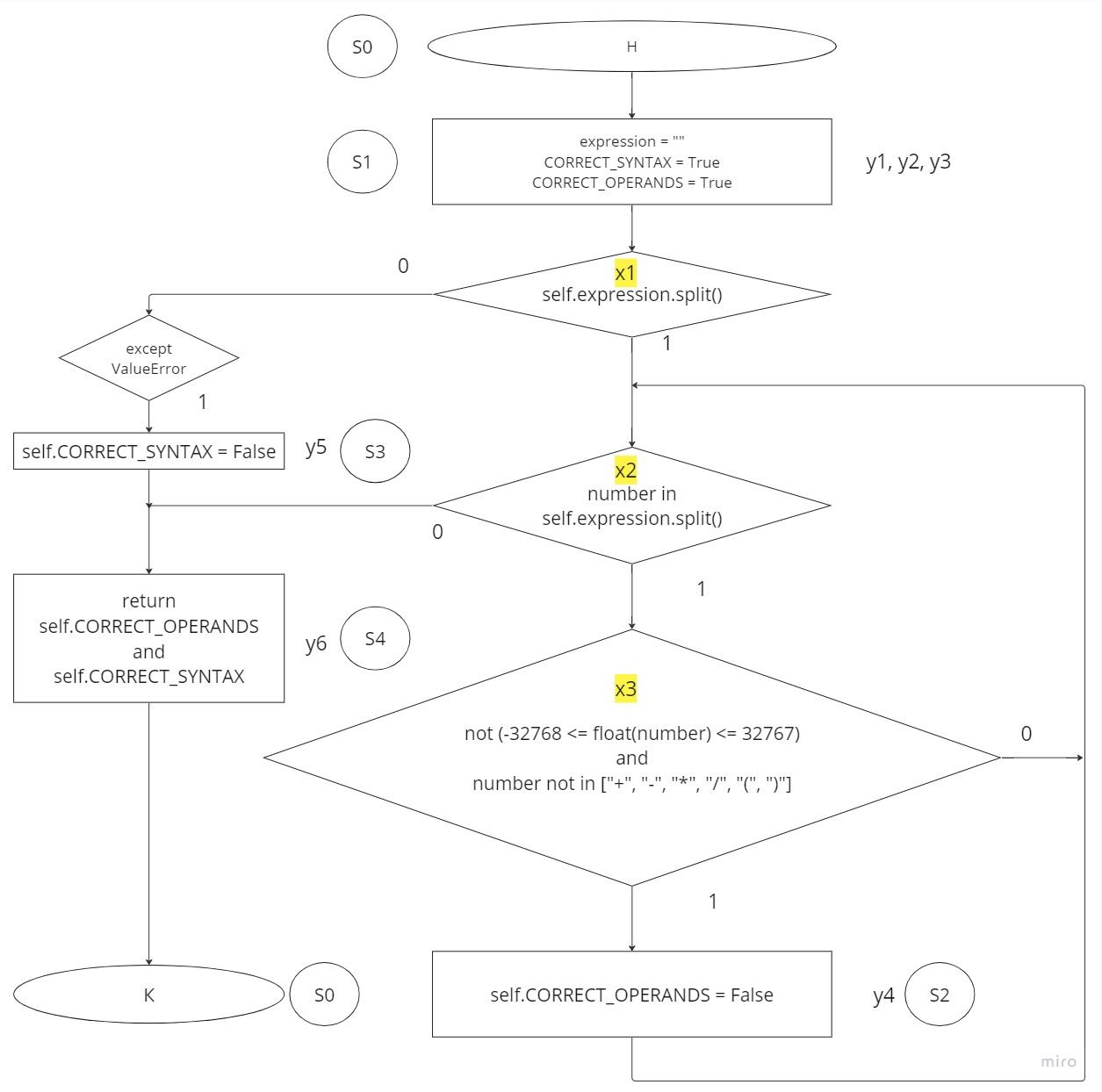
# Реализация

Класс содержит три метода. Первый метод проверяет корректность, второй - вычисляет результат по ОПС, а третий составляет ОПС.

Метод check\_correct(self) - метод класса Expression, проверяющий корректность введённого выражения и возвращающий булевское значение.

def check\_correct(self):  
 try:  
 for number in self.expression.split():  
 if number not in ["+", "-", "\*", "/", "(", ")"] \  
 and not (-32768 <= float(number) <= 32767):  
 self.CORRECT\_OPERANDS = False  
 except ValueError:  
 self.CORRECT\_SYNTAX = False  
  
 return self.CORRECT\_OPERANDS and self.CORRECT\_SYNTAX

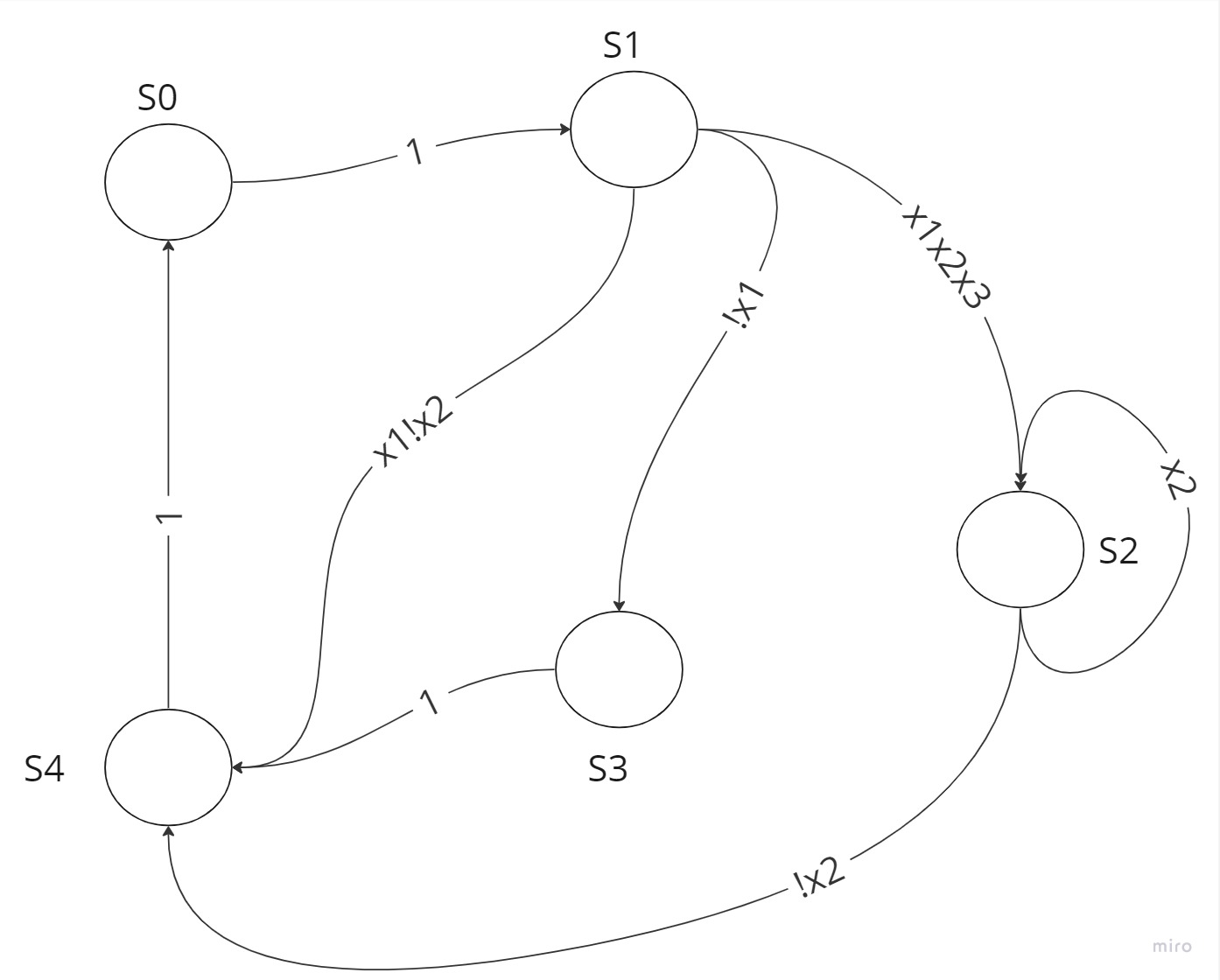
# Блок-схема алгоритма.



Выходные и входные сигналы были закодированы Yi и Xi соответственно. Таблица кодировки:

|  |  |
| --- | --- |
| X1 | self.expression.split() |
| X2 | number in self.expression.split() |
| X3 | not (-32768 <= float(number) <= 32767) and number not in ["+", "-", "\*", "/", "(", ")"] |
| Y1 | expression = "" |
| Y2 | CORRECT\_SYNTAX = True |
| Y3 | CORRECT\_OPERANDS = True |
| Y4 | self.CORRECT\_OPERANDS = False |
| Y5 | self.CORRECT\_SYNTAX = False |
| Y6 | return self.CORRECT\_OPERANDS and self.CORRECT\_SYNTAX |

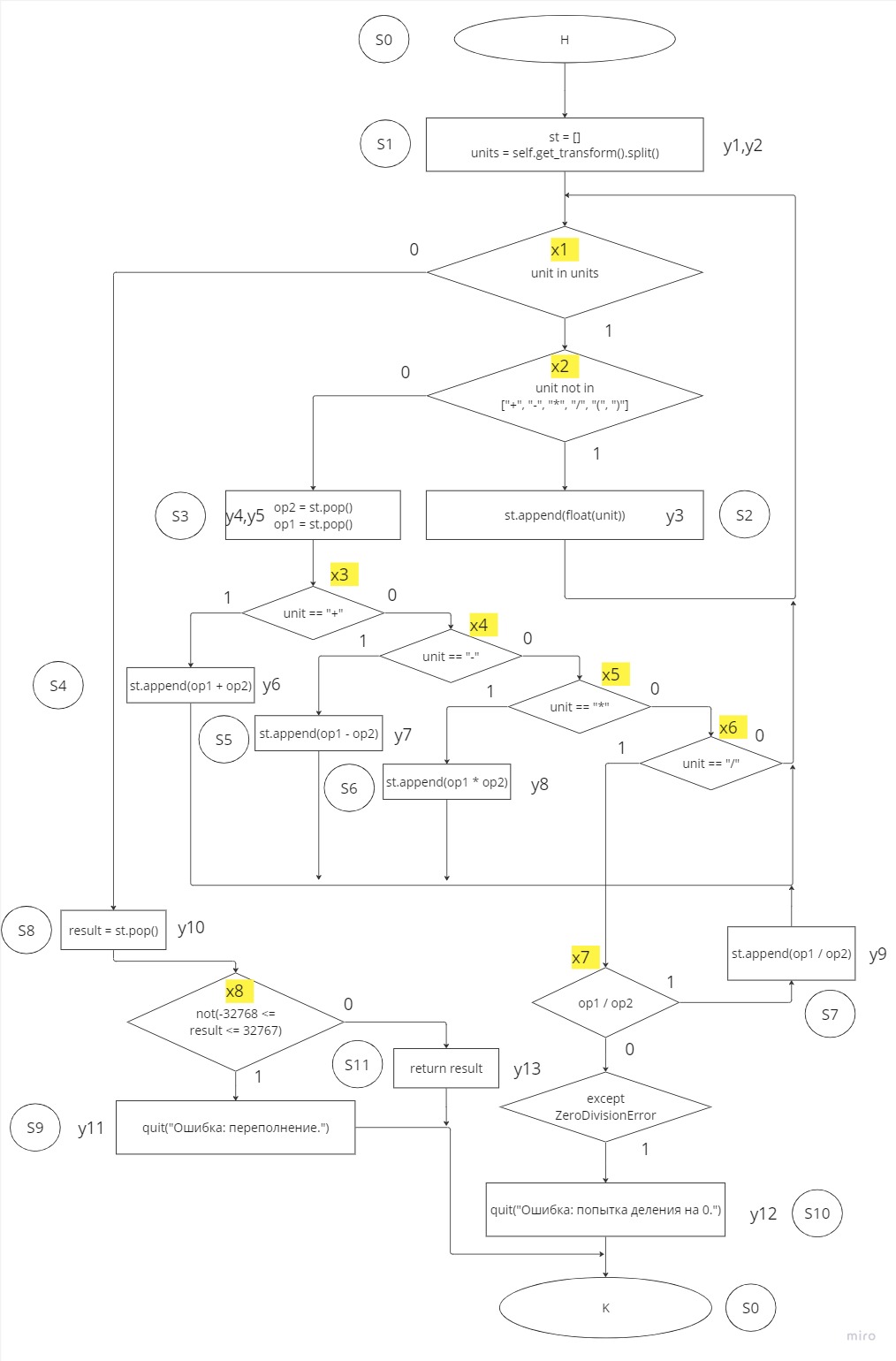
# Граф переходов состояний.



Метод get\_result(self) - метод класса Expression, вычисляющий выражение с использованием стека и возвращающий результат.

def get\_result(self):  
 st = []  
 units = self.get\_transform().split()  
  
 for unit in units:  
 if unit not in ["+", "-", "\*", "/", "(", ")"]:  
 st.append(float(unit))  
 else:  
 op2 = st.pop()  
 op1 = st.pop()  
  
 if unit == "+":  
 st.append(op1 + op2)  
 elif unit == "-":  
 st.append(op1 - op2)  
 elif unit == "\*":  
 st.append(op1 \* op2)  
 elif unit == "/":  
 try:  
 st.append(op1 / op2)  
 except ZeroDivisionError:  
 quit("Ошибка: попытка деления на 0.")  
  
 result = st.pop()  
 if not(-32768 <= result <= 32767):  
 quit("Ошибка: переполнение.")  
  
 return result

# Блок схема алгоритма.



Выходные и входные сигналы были закодированы Yi и Xi соответственно. Таблица кодировки:

|  |  |
| --- | --- |
| X1 | unit in units |
| X2 | unit not in ["+", "-", "\*", "/", "(", ")"] |
| X3 | unit == "+" |
| X4 | unit == "-" |
| X5 | unit == "\*" |
| X6 | unit == "/" |
| X7 | op1 / op2 |
| X8 | not(-32768 <= result <= 32767) |
| Y1 | st = [] |
| Y2 | units = self.get\_transform().split() |
| Y3 | st.append(float(unit)) |
| Y4 | op2 = st.pop() |
| Y5 | op1 = st.pop() |
| Y6 | st.append(op1 + op2) |
| Y7 | st.append(op1 - op2) |
| Y8 | st.append(op1 \* op2 |
| Y9 | st.append(op1 / op2) |
| Y10 | result = st.pop() |
| Y11 | quit("Ошибка: переполнение.") |
| Y12 | quit("Ошибка: попытка деления на 0.") |
| Y13 | return result |

# Граф переходов состояний.

# Untitled - Frame 4 (3)Тестирование программы.

**Тест №1**:

# Введите выражение, где элементы отделены пробелом: 44.5 \* ( 56.4 - 12.93 ) / 7 - 66.6

# 209.74499999999998

# 44.5 56.4 12.93 - \* 7 / 66.6 -

**Тест №2**:

Введите выражение, где элементы отделены пробелом: 5.65 + ( 14 - 4 ) \* 4.4

49.65

5.65 14 4 - 4.4 \* +

**Тест №3**:

Введите выражение, где элементы отделены пробелом: (6.8/) - (15 + 9)

Ошибка: выражение введено некорректно.

**Тест №4**:

Введите выражение, где элементы отделены пробелом: ( 55.5 + 3 ) \* 45000.5

Ошибка: в выражение могут быть числа с плавающей точкой, которые не меньше -32768 и не больше 32767.

**Тест №5**:

Введите выражение, где элементы отделены пробелом: ( -17.7 + 3.67 ) \* 2.2 / 0

Ошибка: попытка деления на 0.

**Тест №6**:

Введите выражение, где элементы отделены пробелом: ( 3108.1 + 505.6 ) \* 10.4

Ошибка: переполнение.

**Тест №7**:

Введите выражение, где элементы отделены пробелом: ( 310 + 505.6 ) / 10 \* 3 / 5

48.936

310 505.6 + 10 / 3 \* 5 /

# Листинг.

class Expression:  
 expression = ""  
 CORRECT\_SYNTAX = True  
 CORRECT\_OPERANDS = True  
  
 def \_\_init\_\_(self, expression):  
 self.expression = expression  
 def check\_correct(self):  
 try:  
 for number in self.expression.split():  
 if number not in ["+", "-", "\*", "/", "(", ")"] \  
 and not (-32768 <= float(number) <= 32767):  
 self.CORRECT\_OPERANDS = False  
 except ValueError:  
 self.CORRECT\_SYNTAX = False  
  
 return self.CORRECT\_OPERANDS and self.CORRECT\_SYNTAX  
  
 def get\_transform(self):  
 st = []  
 units = []  
  
 for unit in self.expression.split():  
 if unit in ["+", "-"]:  
 if len(st) == 0 or st[len(st) - 1] == "(":  
 st.append(unit)  
 else:  
 while len(st) != 0 and st[len(st) - 1] in ["\*", "/", "+", "-"]:  
 units.append(st.pop())  
 st.append(unit)  
 elif unit in ["\*", "/"]:  
 if len(st) == 0 or st[len(st) - 1] in ["+", "-", "("]:  
 st.append(unit)  
 else:  
 while len(st) != 0 and st[len(st) - 1] in ["\*", "/"]:  
 units.append(st.pop())  
 st.append(unit)  
 elif unit == "(":  
 st.append(unit)  
 elif unit == ")":  
 while len(st) != 0 and st[len(st) - 1] != "(":  
 units.append(st.pop())  
 if len(st) != 0:  
 st.pop()  
 else:  
 units.append(unit)  
  
 while len(st) > 0:  
 units.append(st.pop())  
  
 transform\_record = " ".join(units)  
  
 return transform\_record  
  
 def get\_result(self):  
 st = []  
 units = self.get\_transform().split()  
  
 for unit in units:  
 if unit not in ["+", "-", "\*", "/", "(", ")"]:  
 st.append(float(unit))  
 else:  
 op2 = st.pop()  
 op1 = st.pop()  
  
 if unit == "+":  
 st.append(op1 + op2)  
 elif unit == "-":  
 st.append(op1 - op2)  
 elif unit == "\*":  
 st.append(op1 \* op2)  
 elif unit == "/":  
 try:  
 st.append(op1 / op2)  
 except ZeroDivisionError:  
 quit("Ошибка: попытка деления на 0.")  
  
 result = st.pop()  
  
 if not(-32768 <= result <= 32767):  
 quit("Ошибка: переполнение.")  
  
 return result  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 e = Expression(input("Введите выражение, где элементы отделены пробелом: "))  
  
 if e.check\_correct():  
 print(e.get\_result())  
 print(e.get\_transform())  
 elif not(e.CORRECT\_SYNTAX):  
 quit("Ошибка: выражение введено некорректно.")  
 elif not(e.CORRECT\_OPERANDS):  
 quit("Ошибка: в выражение могут быть числа с плавающей точкой, которые не меньше -32768 и не больше 32767.")