ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Дожность		Рогачев С.А
старший преподаватель	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

Машина Тьюринга

по дисциплине: Теория вычислительных процессов

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4236		Л. Мвале
	подпись, дата	инициалы, фамилия

Санкт-Петербург
2025

1. Цель работы: Изучить устройство и принципы функционирования машины Тьюринга. Научиться составлять совокупность команд (программу) для МТ, реализующую вычисление арифметической функции. На примере функции умножения двух натуральных чисел закрепить навыки построения вычислительных алгоритмов в абстрактной машине.

2. Основные сведения из теории

Машина Тьюринга — абстрактный вычислительный автомат, предложенный А. Тьюрингом, служащий для формализации понятия алгоритма. Она состоит из:

1. Управляющего устройства (конечного автомата с внутренними состояниями).
2. Бесконечной ленты, разделённой на ячейки, каждая из которых может хранить символ из конечного алфавита.
3. Считывающей/записывающей головки, которая может перемещаться по ленте и изменять символы.

Алгоритм МТ задаётся совокупностью команд (правил переходов), которые определяют действия машины в зависимости от текущего состояния и обозреваемого символа. Работа машины всегда начинается со стандартной начальной конфигурации и заканчивается в стандартной конечной конфигурации.

Для кодирования чисел часто используется унарная запись: число n представляется последовательностью из n единиц. Операция умножения двух чисел a и b сводится к многократному копированию второго множителя b столько раз, сколько единиц содержится в первом множителе a.

3. Постановка задачи

Необходимо построить машину Тьюринга, которая выполняет умножение двух натуральных чисел.

- Входные данные: два числа в унарной записи, разделённые символом *.
Пример: 11*111 (число 2 умножается на 3).

- Результат: унарное представление произведения.
Пример: 111111 (число 6).

- Начальное состояние: q0.

- Конечное состояние: qz.

- Машина должна работать в соответствии с формальным описанием (совокупность команд P) и завершать работу в стандартной конечной конфигурации.

4. Совокупность команд для машины Тьюринга

Ниже приведён пример множества команд P (обобщённый вариант для умножения a * b). Здесь используются маркеры X (для пометки уже обработанных единиц из первого числа) и Y (для временной метки в правом блоке).

```
# ИСПРАВЛЕННЫЕ ПРАВИЛА ПЕРЕХОДОВ
    transitions = {
      # Начальное состояние - найти первую 1 для пометки
      ('q0', '1'): ('X', 'R', 'q1'),
      ('q0', '*'): ('*', 'R', 'q9'),
```

```
('q9', '1'): ('y', 'R', 'q9'),

('q9', '='): ('=', 'S', 'halt'),

# После пометки левой 1, перейти к правой части

('q1', '1'): ('1', 'R', 'q1'),

('q1', '*'): ('*', 'R', 'q2'),

# Найти первую 1 в правой части для копирования

('q2', '1'): ('Y', 'R', 'q3'),

('q2', 'Y'): ('Y', 'R', 'q2'),

('q2', '='): ('=', 'L', 'q7'),

# Перейти к концу ленты чтобы добавить новую 1

('q3', '1'): ('1', 'R', 'q3'),

('q3', '='): ('=', 'R', 'q4'),

('q4', '_'): ('1', 'L', 'q5'),

('q4', '1'): ('1', 'R', 'q4'),

# Вернуться к правой части

('q5', '1'): ('1', 'L', 'q5'),

('q5', '='): ('=', 'L', 'q6'),

('q6', '1'): ('1', 'L', 'q6'),

('q6', 'Y'): ('Y', 'L', 'q6'),

('q6', '*'): ('*', 'R', 'q2'),


# Все правые 1 обработаны, вернуться к левой части - СБРОСИТЬ Y обратно в 1

('q7', 'Y'): ('1', 'L', 'q7'),  # Сбросить Y в 1
```

```
            ('q7', '1'): ('1', 'L', 'q7'),
            ('q7', '*'): ('*', 'L', 'q8'),
            # Очистка - все левые 1 обработаны
            ('q8', '1'): ('1', 'L', 'q8'),
            ('q8', 'X'): ('X', 'R', 'q0'),  # ДОБАВЛЕНО: Очистить оставшиеся Y
        }
```

5. Листинг программы

```
class TuringMachine:
    def __init__(self, input_tape):
        self.tape = list(input_tape)
        self.head = 0
        self.state = 'q0'
        self.steps = 0

    def _get_current_symbol(self):
        if self.head < 0 or self.head >= len(self.tape):
            return '_'
        return self.tape[self.head]
    def _set_current_symbol(self, symbol):
        if self.head < 0:
            self.tape = [symbol] + ['_'] * (-self.head - 1) + self.tape
            self.head = 0
        elif self.head >= len(self.tape):
```

```
            self.tape = self.tape + ['_'] * (self.head - len(self.tape)) + [symbol]
        else:
            self.tape[self.head] = symbol
    def _move(self, direction):
        if direction == 'R':
            self.head += 1
        elif direction == 'L':
            self.head -= 1
        elif direction != 'S':
            raise ValueError(f"Неверное направление: {direction}")
    def _print_state(self):
        tape_str = ''.join(self.tape)
        head_pos = ' ' * self.head + '^'
        print(f"Шаг {self.steps:3d}: Состояние={self.state:2s} | Лента: {tape_str}")
        print(f"           {' ' * 12} {head_pos}")
    def run(self, max_steps=500, verbose=True):
        if verbose:
            print("Начальное состояние:")
            self._print_state()
            print("\n" + "="*50)
        # ИСПРАВЛЕННЫЕ ПРАВИЛА ПЕРЕХОДОВ
        transitions = {
            # Начальное состояние - найти первую 1 для пометки
            ('q0', '1'): ('X', 'R', 'q1'),
            ('q0', '*'): ('*', 'R', 'q9'),
```

```
        ('q9', '1'): ('y', 'R', 'q9'),
        ('q9', '='): ('=', 'S', 'halt'),
        # После пометки левой 1, перейти к правой части
        ('q1', '1'): ('1', 'R', 'q1'),
        ('q1', '*'): ('*', 'R', 'q2'),
        # Найти первую 1 в правой части для копирования
        ('q2', '1'): ('Y', 'R', 'q3'),
        ('q2', 'Y'): ('Y', 'R', 'q2'),
        ('q2', '='): ('=', 'L', 'q7'),
        # Перейти к концу ленты чтобы добавить новую 1
        ('q3', '1'): ('1', 'R', 'q3'),
        ('q3', '='): ('=', 'R', 'q4'),
        ('q4', '_'): ('1', 'L', 'q5'),
        ('q4', '1'): ('1', 'R', 'q4'),
        # Вернуться к правой части
        ('q5', '1'): ('1', 'L', 'q5'),
        ('q5', '='): ('=', 'L', 'q6'),
        ('q6', '1'): ('1', 'L', 'q6'),
        ('q6', 'Y'): ('Y', 'L', 'q6'),
        ('q6', '*'): ('*', 'R', 'q2'),
        # Все правые 1 обработаны, вернуться к левой части - СБРОСИТЬ Y обратно в 1
        ('q7', 'Y'): ('1', 'L', 'q7'),  # Сбросить Y в 1
        ('q7', '1'): ('1', 'L', 'q7'),
        ('q7', '*'): ('*', 'L', 'q8'),
        # Очистка - все левые 1 обработаны
```

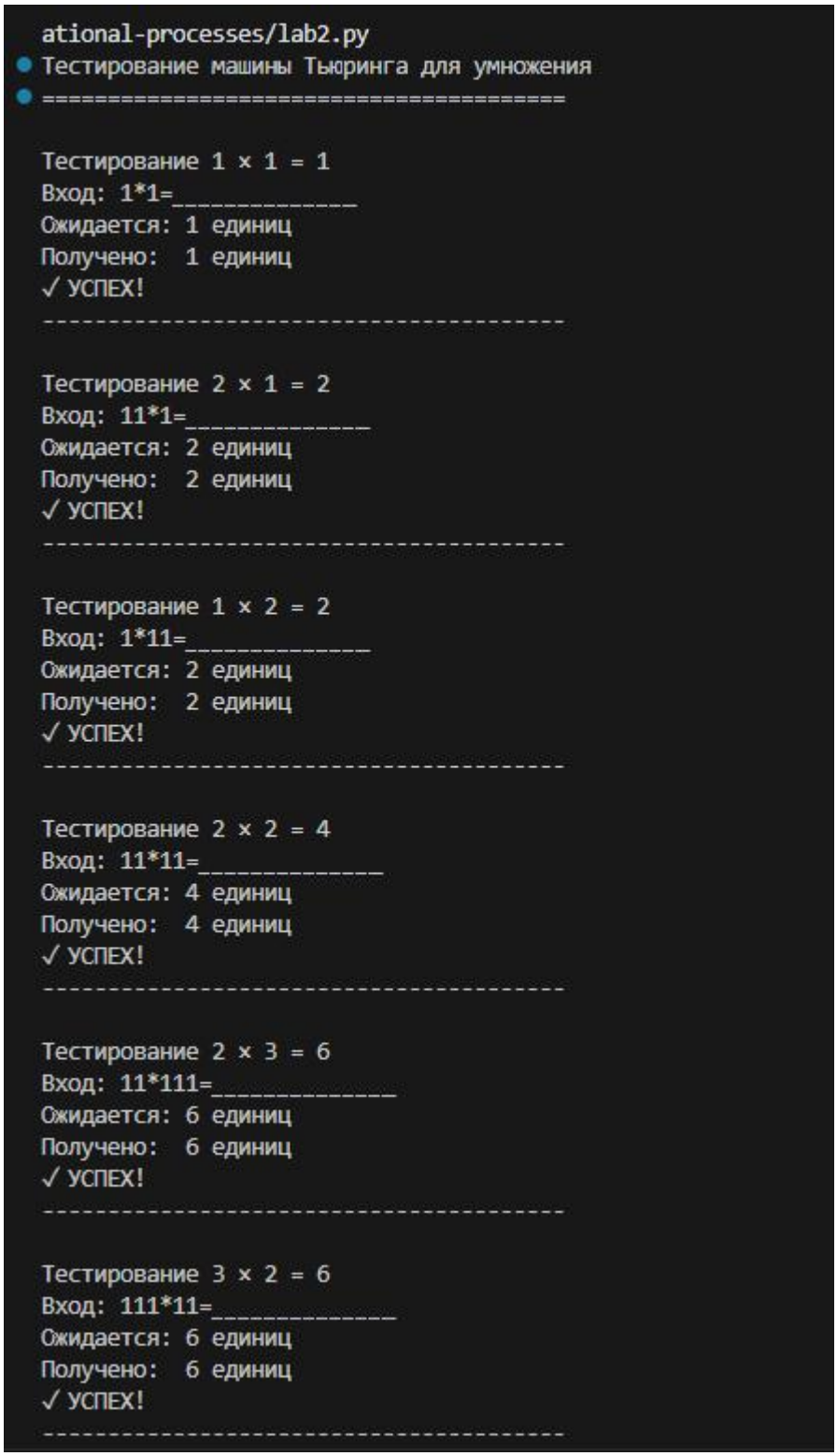
```
            ('q8', '1'): ('1', 'L', 'q8'),
            ('q8', 'X'): ('X', 'R', 'q0'),  # ДОБАВЛЕНО: Очистить оставшиеся Y
        }
        while self.state != 'halt' and self.steps < max_steps:
            current_symbol = self._get_current_symbol()
            key = (self.state, current_symbol)
            if key not in transitions:
                if verbose:
                    print(f"\nНе определен переход для (состояние={self.state},
символ={current_symbol})")
                    print("Текущая лента:", ".join(self.tape))
                break
            write_symbol, move_direction, next_state = transitions[key]
            self._set_current_symbol(write_symbol)
            self._move(move_direction)
            self.state = next_state
            self.steps += 1
            if verbose and self.steps <= 100:
                self._print_state()
        if verbose:
            print("\n" + "="*50)
            if self.state == 'halt':
                print("Вычисления завершены успешно!")
            else:
                print(f"Вычисления остановлены после {self.steps} шагов")
            result = ".join([c for c in self.tape if c == '1'])
```

```
            print(f"Финальный результат: {len(result)} единиц -> {result}")
        return self.tape, self.state
def create_input_string(a, b):
    """Создать входную строку в формате '111*111=' для заданных чисел"""
    left_ones = '1' * a
    right_ones = '1' * b
    return f"{left_ones}*{right_ones}=_____________"
def test_multiplication():
    """Протестировать машину Тьюринга для умножения"""
    test_cases = [
        (1, 1),  # 1 × 1 = 1
        (2, 1),  # 2 × 1 = 2
        (1, 2),  # 1 × 2 = 2
        (2, 2),  # 2 × 2 = 4
        (2, 3),  # 2 × 3 = 6
        (3, 2),  # 3 × 2 = 6
    ]
    print("Тестирование машины Тьюринга для умножения")
    print("=" * 40)
    for a, b in test_cases:
        input_str = create_input_string(a, b)
        expected = a * b
        print(f"\nТестирование {a} × {b} = {expected}")
        print(f"Вход: {input_str}")
        tm = TuringMachine(input_str)
```

```
        final_tape, final_state = tm.run(verbose=False, max_steps=500)
        result_ones = ''.join([c for c in final_tape if c == '1'])
        print(f"Ожидается: {expected} единиц")
        print(f"Получено:  {len(result_ones)} единиц")
        if len(result_ones) == expected:
            print("✓ УСПЕХ!")
        else:
            print("✗ НЕУДАЧА!")
            print(f"Финальная лента: {''.join(final_tape)}")
        print("-" * 40)
def debug_case(a, b):
    """Отладить конкретный случай с подробным выводом"""
    print(f"\nОТЛАДКА {a} × {b}:")
    print("=" * 30)

    input_str = create_input_string(a, b)
    expected = a * b
    print(f"Вход: {input_str}")
    print(f"Ожидается: {expected} единиц")
    tm = TuringMachine(input_str)
    final_tape, final_state = tm.run(verbose=True, max_steps=200)
    result_ones = ''.join([c for c in final_tape if c == '1'])
    print(f"\nОжидается: {expected} единиц, Получено: {len(result_ones)} единиц")
    if len(result_ones) == expected:
        print("✓ УСПЕХ!")
```

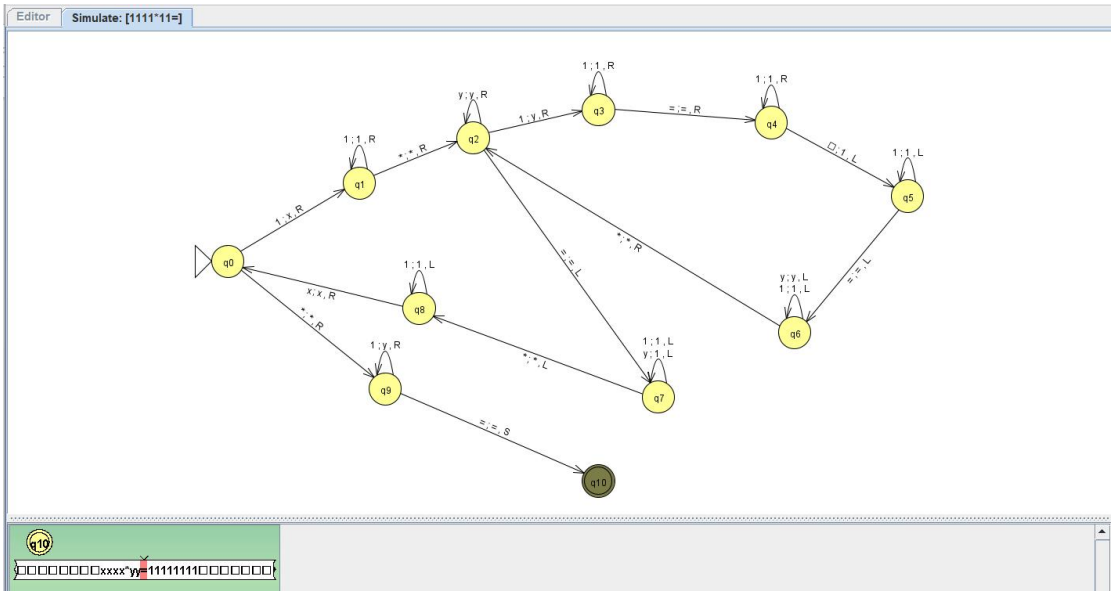
```
    else:
        print("✗ НЕУДАЧА!")
        print(f"Финальная лента: {''.join(final_tape)}")
if __name__ == "__main__":
    # Запустить комплексные тесты
    test_multiplication()
    # Отладить конкретные случаи которые не работают
    print("\n" + "="*60)
    print("ОТЛАДКА НЕУДАЧНЫХ СЛУЧАЕВ:")
    print("="*60)

    debug_case(2, 1)
    debug_case(2, 2)
```

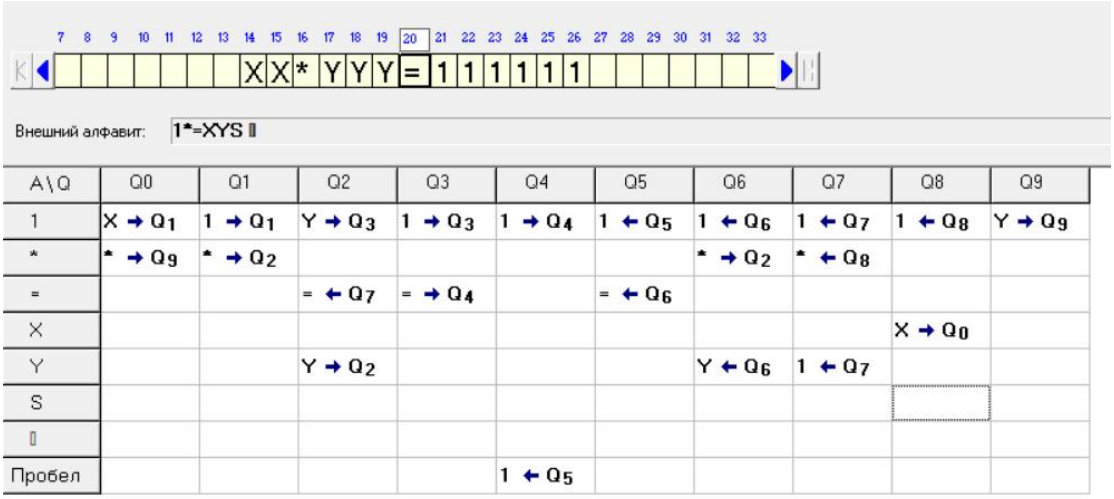
6. Пример выполнения



```
============================================================
ОТЛАДКА НЕУДАЧНЫХ СЛУЧАЕВ:
============================================================

ОТЛАДКА 2 × 1:
==============================
Вход: 11*1=_____________
Ожидается: 2 единиц
Начальное состояние:
Шаг   0: Состояние=q0 | Лента: 11*1=_____________
                        ^


================================================
Шаг   1: Состояние=q1 | Лента: X1*1=_____________
                         ^
Шаг   2: Состояние=q1 | Лента: X1*1=_____________
                          ^
Шаг   3: Состояние=q2 | Лента: X1*1=_____________
                           ^
Шаг   4: Состояние=q3 | Лента: X1*Y=_____________
                            ^
Шаг   5: Состояние=q4 | Лента: X1*Y=_____________
                             ^
Шаг   6: Состояние=q5 | Лента: X1*Y=1____________
                            ^
Шаг   7: Состояние=q6 | Лента: X1*Y=1____________
                           ^
Шаг   8: Состояние=q6 | Лента: X1*Y=1____________
                          ^
Шаг   9: Состояние=q2 | Лента: X1*Y=1____________
                           ^
Шаг  10: Состояние=q2 | Лента: X1*Y=1____________
                            ^
Шаг  11: Состояние=q7 | Лента: X1*Y=1____________
                           ^
Шаг  12: Состояние=q7 | Лента: X1*1=1____________
                            ^
Шаг  13: Состояние=q8 | Лента: X1*1=1____________
                           ^
Шаг  14: Состояние=q8 | Лента: X1*1=1____________
                          ^
Шаг  15: Состояние=q0 | Лента: X1*1=1____________
                           ^
Шаг  16: Состояние=q1 | Лента: XX*1=1____________
```

```
Шаг  19: Состояние=q4 | Лента: XX*Y=1___________
                              ^
Шаг  20: Состояние=q4 | Лента: XX*Y=1___________
                               ^
Шаг  21: Состояние=q5 | Лента: XX*Y=11__________
                              ^
Шаг  22: Состояние=q5 | Лента: XX*Y=11__________
                               ^
Шаг  23: Состояние=q6 | Лента: XX*Y=11__________
                              ^
Шаг  24: Состояние=q6 | Лента: XX*Y=11__________
                               ^
Шаг  25: Состояние=q2 | Лента: XX*Y=11__________
                              ^
Шаг  26: Состояние=q2 | Лента: XX*Y=11__________
                               ^
Шаг  27: Состояние=q7 | Лента: XX*Y=11__________
                              ^
Шаг  28: Состояние=q7 | Лента: XX*1=11__________
                               ^
Шаг  29: Состояние=q8 | Лента: XX*1=11__________
                              ^
Шаг  30: Состояние=q0 | Лента: XX*1=11__________
                               ^
Шаг  31: Состояние=q9 | Лента: XX*1=11__________
                              ^
Шаг  32: Состояние=q9 | Лента: XX*y=11__________
                               ^
Шаг  33: Состояние=halt | Лента: XX*y=11__________
                               ^


================================================
Вычисления завершены успешно!
Финальный результат: 2 единиц -> 11

Ожидается: 2 единиц, Получено: 2 единиц
✓ УСПЕХ!
```





A\Q	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
1	X → Q1	1 → Q1	Y → Q3	1 → Q3	1 → Q4	1 ← Q5	1 ← Q6	1 ← Q7	1 ← Q8	Y → Q9
*	* → Q9	* → Q2					* → Q2	* ← Q8		
=			= ← Q7	= → Q4		= ← Q6				
X									X → Q0	
Y			Y → Q2				Y ← Q6	1 ← Q7		
S										
▯										
Пробел					1 ← Q5					

7. Выводы

В ходе лабораторной работы были изучены принципы работы машины Тьюринга и реализована программа для выполнения умножения двух чисел в унарной записи. Построена совокупность команд P, которая корректно выполняет вычисления и завершает работу в стандартной конечной конфигурации. Таким образом, подтверждается универсальность машины Тьюринга как модели вычислений и её способность реализовывать любые алгоритмы, в том числе арифметические операции.