

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Тульский государственный университет»  
Кафедра информационной безопасности

УТВЕРЖДАЮ

Зав. кафедрой ИБ

\_\_\_\_\_ А.А. Сычугов

« \_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе по дисциплине  
**«ДИСКРЕТНАЯ МАТЕМАТИКА»**  
на тему

«Машины Тьюринга и вычислимые функции»

Автор работы \_\_\_\_\_ студент гр. 230711 Павлова В.С.  
(дата, подпись) (фамилия и инициалы)

Руководитель работы \_\_\_\_\_  
(дата, подпись) (должность) (фамилия и инициалы)

Работа защищена \_\_\_\_\_ с оценкой \_\_\_\_\_  
(дата)

Члены комиссии \_\_\_\_\_  
(дата, подпись) (должность) (фамилия и инициалы)

\_\_\_\_\_ (дата, подпись) (должность) (фамилия и инициалы)

\_\_\_\_\_ (дата, подпись) (должность) (фамилия и инициалы)

Тула 20 23

УТВЕРЖДАЮ  
Зав. кафедрой ИБ

\_\_\_\_\_ А.А. Сычугов  
«\_\_\_» \_\_\_\_\_ 20\_\_ г.

## ЗАДАНИЕ

на курсовую работу по дисциплине  
«ДИСКРЕТНАЯ МАТЕМАТИКА»

студенту гр. 230711 Павловой Виктории Сергеевне  
(фамилия, имя, отчество)

Тема работы  
«Машины Тьюринга и вычислимые функции»

### Входные данные

1. Яблонский С. В. Введение в дискретную математику: учебное пособие для вузов / С. В. Яблонский. – 5-е изд. – М.: Высшая школа, 2008. – 384 с.

2. Гаврилов Г. П., Сапоженко А. А. Задачи и упражнения по дискретной математике: – 3-е изд., перераб. – М.: ФИЗМАТЛИТ, 2005. – 416 с.

Задание получил \_\_\_\_\_ 21.02.2023 г.  
(подпись) (дата)

### График выполнения работы

21.02-28.02 – Получение и ознакомление с заданием

01.03-22.03 – Изучение литературы и других исходных материалов

23.03-03.05 – Изучение теории, раскрывающей тему курсовой работы

04.05-17.05 – Разработка программной реализации курсовой работы

18.05-24.05 – Анализ результатов

25.05-07.06 – Оформление пояснительной записки и сдача на проверку

03.07.2023 – Защита курсовой работы

### Замечания консультанта

К защите. Консультант работы \_\_\_\_\_  
(подпись) (дата)

## Содержание

Введение .....	4
I. Теоретическая часть .....	5
1.1 Вычислимые и рекурсивные функции .....	5
1.2 Машина Тьюринга .....	7
II. Практическая часть .....	14
2.1 Решение задач, связанных с вычислимыми функциями .....	14
2.2 Вычисление функций с помощью машины Тьюринга .....	16
Заключение .....	22
Список использованных источников .....	23
Приложение А .....	24
Приложение Б .....	27

## Введение

Машина Тьюринга, предложенная английским математиком Аланом Тьюрингом в 1936 году, является фундаментальной моделью в теории вычислений. Она играет важную роль в исследовании сложности алгоритмов и компьютерных систем. Основным принципом машины Тьюринга состоит в использовании рекурсивных функций для описания и моделирования алгоритмических процессов.

Актуальность изучения машины Тьюринга и вычислимых рекурсивных функций заключается в их связи с разработкой эффективных компьютерных систем и теорией алгоритмов. Понимание, какие функции могут быть вычислены с помощью машины Тьюринга, является ключевым для определения возможностей и ограничений вычислительных процессов.

Вычислимые рекурсивные функции играют важную роль в описании и анализе различных алгоритмических задач. Они используются для обработки данных, оптимизации кода, создания компиляторов и интерпретаторов, а также для решения различных практических задач, включая поиск, сортировку, обработку изображений и звука, математические вычисления и многое другое. В математике и логике вычислимые рекурсивные функции применяются для формализации и доказательства теорем и законов. Они позволяют создавать точные и формальные модели для изучения различных математических структур и свойств. Вычислимые рекурсивные функции находят применение в математике и логике, где они используются для формализации и доказательства различных теорем и законов.

Целью данной курсовой работы является рассмотрение таких понятий, как машина Тьюринга, вычислимость функций и операций над ними. Работа посвящена анализу свойств машин Тьюринга, рекурсивных и вычислимых функций, а также решению практических задач, с ними связанных.

## I. Теоретическая часть

### 1.1 Вычислимые и рекурсивные функции

**Вычислимые функции** – это функции, которые можно вычислить с помощью алгоритма, то есть с помощью некоторого конечного набора инструкций или правил. Они также известны как алгоритмические функции или функции, вычислимые в смысле Тьюринга. В теории вычислимости существуют различные классы вычислимых функций, которые обладают определенными свойствами. Один из таких классов – это класс рекурсивных функций.

**Рекурсивные функции** – это подкласс вычислимых функций, который может быть определен с помощью простых базовых функций и операций. Класс вычислимых функций (или тотал-рекурсивных функций) включает в себя все функции, которые могут быть вычислены алгоритмически, используя конечное число шагов. Это означает, что для любой входной последовательности алгоритм, вычисляющий функцию, должен завершиться за конечное число шагов, выдавая соответствующий выходной результат. [1] Все рекурсивные функции также являются вычислимыми функциями, но не все вычислимые функции являются рекурсивными.

Для определения рекурсивных функций могут быть использованы следующие операции [2]:

1. *Суперпозиция;*
2. *Примитивная рекурсия;*
3. *Минимизация.*

Функция  $F(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n))$  называется **суперпозицией** функций  $f$  и  $g_1 \dots g_m$  и обозначается как  $S(f^{(m)}: g_1^{(n)}, \dots, g_m^{(n)})$ , причём она определена на наборе  $\tilde{\alpha}^n$  тогда и только тогда, когда каждая функция  $g_i$  определена на данном наборе и, помимо этого,  $f$  определена на наборе  $(g_1(\tilde{\alpha}^n), \dots, g_m(\tilde{\alpha}^n))$ . В этом случае  $F(\tilde{\alpha}^n) = f(g_1(\tilde{\alpha}^n), \dots, g_m(\tilde{\alpha}^n))$  [6].

Иначе говоря, операция композиции позволяет объединить две вычислимые функции в одну. Если  $f$  и  $g$  – вычислимые функции, то функция  $h(x) = f(g(x))$ , полученная композицией данных функций, также будет является вычислимой.

*Пример:* пусть  $f(x) = x^2$ ,  $g(x) = x + 1$ . Тогда функция  $h(x) = f(g(x)) = (x + 1)^2$  тоже является вычислимой.

Рассмотрим теперь функции  $g(x_1, \dots, x_{n-1})$  и  $h(x_1, \dots, x_{n-1}, x_n, x_{n+1})$ , для которых  $n \geq 2$ . Для них можно определить  $f(x_1, \dots, x_{n-1}, x_n, x_{n+1})$  по следующей схеме, называемой **примитивной рекурсией** [1]:

$$\begin{aligned} f(x_1, \dots, x_{n-1}, 0) &= g(x_1, \dots, x_{n-1}), \\ f(x_1, \dots, x_{n-1}, y + 1) &= h(x_1, \dots, x_{n-1}, y, f(x_1, \dots, x_{n-1}, y)) \end{aligned}$$

Таким образом, операция примитивной рекурсии позволяет определить новую функцию из двух заданных функций. Если  $f$  и  $g$  – вычислимые функции, то с помощью примитивной рекурсии можно определить новую функцию  $h(x)$ , такую, что  $h(0) = f$ , а  $h(n + 1) = g(h(n), n)$  для всех  $n$ .

*Пример:* пусть  $f(x) = x$ ,  $g(x, y) = x + y$ . Тогда функция  $h(x) = g(h(x - 1), x - 1)$  со значением  $h(0) = f(0) = 0$  будет вычислимой и будет вычислять сумму всех чисел от 0 до  $x$ .

Пусть  $f(x_1, \dots, x_{n-1}, x_n)$ , где  $n \geq 1$  – некоторая частично числовая функция. Определим  $g(x_1, \dots, x_{n-1}, x_n)$  следующим образом, учтя, что  $\tilde{\alpha}^n = (\alpha_1, \dots, \alpha_{n-1}, \alpha_n)$  – произвольный набор целых неотрицательных чисел, а также

$$f(\alpha_1, \dots, \alpha_{n-1}, y) = \alpha_n:$$

Если данное уравнение имеет решение  $y_0 \in N$  и при всех неотрицательных  $y \in N$  таких, что  $y \leq y_0$ , а функция определена и её значения отличны от  $\alpha_n$ , то полагаем  $g(\tilde{\alpha}^n) = y_0$ . Функцию  $g(\tilde{\alpha}^n)$  называют полученной с помощью **минимизации** по  $x_n$  [1]. Это можно обозначить так:  $g = Mf$ . Операция минимизации позволяет найти минимальное значение аргумента, для которого заданная функция принимает значение 0. Если  $f$  – вычислимая функция, то

функция  $g(x)$ , которая возвращает минимальный неотрицательный аргумент  $u$ , при котором  $f(y) = 0$ , также является вычислимой.

Теория вычислимости помогает доказывать, что существуют некоторые задачи, которые не могут быть решены компьютером независимо от его мощности. Это имеет практическое значение, поскольку позволяет оценить сложность решения некоторых задач и определить, какие задачи являются вычислимыми, а какие нет. Также понятия класса вычислимых и рекурсивных функций используются в теории формальных языков и автоматов, где они помогают определить, какие языки можно распознать с помощью различных типов автоматов.

## ***1.2 Машина Тьюринга***

***Абстрактная вычислительная машина Тьюринга***, названная в честь математика Алана Тьюринга, является математической моделью вычислительной системы. Она представляет собой устройство с бесконечной лентой, разделенной на ячейки, в каждой из которых может находиться символ из заданного алфавита.

Машина Тьюринга может перемещаться влево и вправо по ленте, считывая символы и выполняя определенные действия в зависимости от текущего состояния и символа, который она видит. Она имеет конечное множество состояний, входных символов и правил перехода, которые определяют ее поведение. Формально машина Тьюринга может быть определена следующей пятёркой объектов  $T = (A, Q, \delta, \vartheta, \mu)$ , где [3]:

$A$  – конечное множество входных символов (алфавит);

$Q$  – конечное множество состояний;

$\delta: A \times Q \rightarrow Q$  – функция перехода;

$\vartheta: A \times Q \rightarrow A$  – функция выхода;

$\mu: A \times Q \rightarrow \{L, R, N\}$  – функция управления.

Машина Тьюринга и вычислимые функции тесно связаны друг с другом: машина Тьюринга является абстрактной вычислительной моделью, предложенной Аланом Тьюрингом, которая описывает устройство, способное выполнить вычисления на основе набора простых инструкций; вычислимые функции, с другой стороны, являются математическими функциями, которые могут быть вычислены при помощи алгоритма или процесса. Они описывают вычислительные задачи, которые можно выполнить с помощью компьютера или другого универсального вычислительного устройства.

Связь между машинами Тьюринга и вычислимыми функциями заключается в том, что машина Тьюринга может использоваться для моделирования и вычисления вычислимых функций. Машина Тьюринга предоставляет формальную систему, в рамках которой можно описать и реализовать алгоритмы для вычисления различных вычислимых функций. Машина Тьюринга позволяет представить вычислимые функции в виде алгоритмов, которые можно выполнить на такой машине. Таким образом, МТ является теоретической моделью, которая иллюстрирует, какие вычислимые функции могут быть реализованы с помощью алгоритма и вычислительного процесса.

**Командой машины Тьюринга** называется запись вида  $a_k q_i \rightarrow a_p D q_j$ , где  $a_p, D$ , и  $q_j$  – значения на наборе функций  $\delta, \vartheta$  и  $\mu$  соответственно. Программой машины Тьюринга называется набор всех её команд. Работа машины Тьюринга связана с бесконечной лентой, разбитой на ячейки, причём в каждой ячейке может быть записан один символ некоторого алфавита, где  $\lambda$  является символом пустой ячейки. Работа над словом  $\alpha$ , записанным на ленте, проходит следующим образом:

1. Машина Тьюринга начинает свою работу всегда в состоянии  $q_0$ , а её считывающее устройство (каретка) расположено над первым слева символом слова, записанного на ленте;



2. Считав символ в ячейке, обозреваемой считывающим устройством машины Тьюринга, она печатает в эту ячейку символ, найденный с помощью функции выхода  $\vartheta$ , движется вдоль ленты вправо, влево или остаётся на месте в случае, если функция  $\mu$  принимает значения R, L или N соответственно и переходит в состояние, определяемое с помощью функции перехода  $\delta$ ;
3. При переходе машины Тьюринга в конечное состояние  $q_n$  считают, что она закончила работу над словом и говорят, что машина Тьюринга применима к слову  $\alpha$ . Если машина Тьюринга при работе над словом не переходит в конечное состояние, то говорят, что она *не применима* к слову  $\alpha$ .

Таким образом, машина Тьюринга начинает свою работу с определенной исходной конфигурации ленты и состояния. Она последовательно применяет функции перехода, выхода и управления, осуществляя чтение символов с ленты, запись новых символов, изменение состояний и перемещение по ленте. Вычисление считается завершённым, когда машина достигает состояния остановки. Машина Тьюринга может быть использована для моделирования работы любого алгоритма или вычислительной системы. Она является универсальной вычислительной моделью и используется в теории алгоритмов для исследования вычислительной сложности задач [3].

Одно из важнейших свойств машины Тьюринга – её универсальность. Это означает, что существует такая машина Тьюринга, которая может симулировать работу любой другой машины Тьюринга. Такое свойство позволяет использовать машину Тьюринга в качестве теоретической основы для разработки компьютерных программ и алгоритмов, а также для изучения пределов вычислительной мощности.

Как показано на рисунке 1.1, данное устройство снабжено устройством управления, хранящим программу, информационной лентой, разделённой на

участки (квадраты), каждый из которых может содержать символ. Вдоль бесконечной ленты скользит считывающая головка (каретка), действия которой определяются программой. Головка считывает по одному символу за раз, умеет перемещаться вдоль ленты в обе стороны и записывать на неё любые символы алфавита.

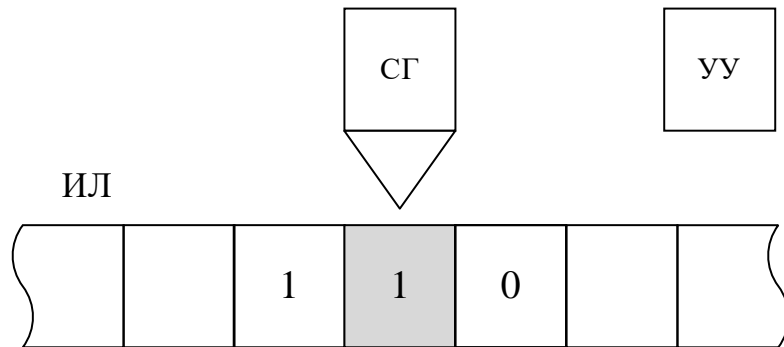


Рисунок 1.1 – Схематичное изображение машины Тьюринга; ИЛ – информационная лента, СГ – считывающая головка, УУ – устройство управления

Машина Тьюринга работает тактами (по шагам), которые выполняются один за другим. На каждом такте автомат МТ выполняет три следующих действия, причем обязательно в указанном порядке  $[S, \langle N, L, R \rangle, q_i]$ :

1) записывает некоторый символ  $S$  в текущую клетку (может быть записан тот же символ, что и был, т.е. содержимое клетки не меняется);

2) сдвигается на одну клетку влево (обозначение –  $L$ , от англ. left) или вправо (обозначение –  $R$ , от англ. right), либо остается неподвижным (обозначение –  $N$ ).

3) переходит в некоторое состояние  $q$  (в частности, может остаться в прежнем состоянии).

**Полнота по Тьюрингу** – понятие, тесно связанное с абстрактными вычислительными машинами. Набор алгоритмов полон по Тьюрингу, если любую машину Тьюринга можно смоделировать на этом наборе алгоритмов, то

есть с помощью такого набора алгоритмов можно решить любую задачу, которую может решить машина Тьюринга [5].

Таким образом, числовая функция  $f(x_1, x_2, \dots, x_k)$ , где  $x_i \in N_0, i = 1 \dots k$ , называется **вычислимой функцией** по Тьюрингу, если существует машина Тьюринга, применимая к любому слову вида  $(*)$ , переводящая его в слово  $1^{y+1}$ , где  $y = f(x_1, x_2, \dots, x_k)$ .

Машина Тьюринга М вычисляет **частичную** (то есть определённую не на всех элементах базового множества) функцию  $f(x)$ , если, начиная работу на первом символе основного кода набора  $x$  в состоянии  $q_1$ , машина: [6]

1. Если  $f(x)$ , определено, то машина через конечное число тактов останавливается, и в этот момент на ленте представлено значение  $f(x)$ .
2. Если  $f(x)$  не определено, то М либо не останавливается, либо останавливается, но на ленте не оказывается основной код числа.

Известно, что если машина вычисляет некоторую функцию, то можно изменить её так, чтобы она правильно вычисляла эту функцию. Машина Тьюринга **правильно вычисляет** частичную функцию  $f(x)$  если, начиная работу на первой единице основного кода набора  $x$  (остальные символы ленты — нули) в состоянии  $q_1$ , машина: [4]

1. Если  $f(x)$  определено, то машина через конечное число итераций останавливается, и в этот момент на ленте представлено значение  $f(x)$  в основном коде, причём головка машины находится на первом символе этого основного кода.
2. Если  $f(x)$  не определено, то М не останавливается через конечное число итераций.

В качестве примера правильно вычислимой функции можно привести функцию инверсии двоичного числа. Находясь в состоянии  $q_0$  и с кареткой в начале числа, машина Тьюринга последовательно просматривает каждый символ на ленте, начиная с левого конца. Если на ленте встречается символ «0»,

записываться должен символ «1» и наоборот. Если машина достигает конца числа, она останавливается и завершает работу. Таким образом, таблица переходов машины Тьюринга для простейшей операции – инверсии двоичного числа будет выглядеть следующим образом (таблица 1.1):

Таблица 1.1 – Программа инверсии числа для машины Тьюринга

Состояние	0	1	$\lambda$ (пробельный символ)
$q_0$	$1 R q_0$	$0 R q_0$	!

В каждой ячейке таблицы указаны три значения  $[S, \langle N, L, R \rangle, q_i]$ , которые представляют действия, которые машина Тьюринга должна выполнить, когда она находится в соответствующем состоянии и видит соответствующий символ на ленте. Если в ячейке указано «!», то головка останавливается и машина Тьюринга завершает работу. Схематичное изображение данного процесса приведено на рисунке 1.2:

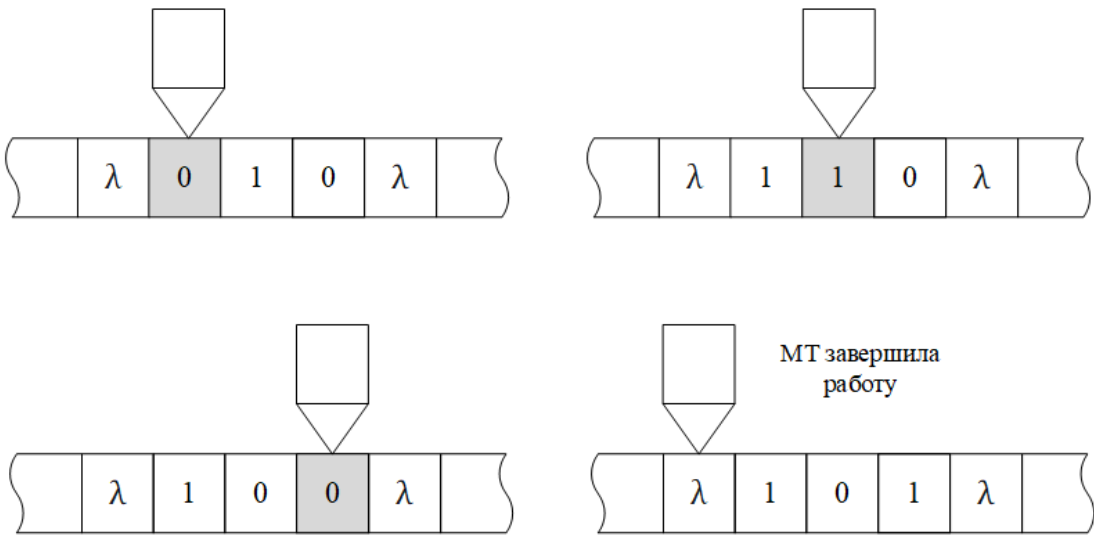


Рисунок 1.2 – Инверсия двоичного числа с помощью машины Тьюринга

Подводя итог, заключим, что основные практические применения машины Тьюринга, вычислимых и рекурсивных функций включают в себя:

- ***Теоретические исследования:*** машина Тьюринга, вычислимые и рекурсивные функции служат основой для теоретических исследований в области теории вычислений, алгоритмов и компьютерных наук.
- ***Разработка и анализ алгоритмов:*** машина Тьюринга и вычислимые функции позволяют описывать и анализировать различные алгоритмические процессы, что является основой для разработки эффективных алгоритмов в различных областях.
- ***Разработка программного обеспечения:*** вычислимые и рекурсивные функции используются для разработки программ и систем, включая обработку данных, создание компиляторов и интерпретаторов, оптимизацию кода и другие задачи разработки ПО.

## II. Практическая часть

### 2.1 Решение задач, связанных с вычислимыми функциями

Прежде, чем перейти к непосредственной работе с машиной Тьюринга, которая формально предоставляет среду, в которой можно запустить алгоритмы, реализующие вычислимые функции, рассмотрим некоторые примеры применения разрешённых операций к этим функциям.

**Задание №1.** Обосновать примитивную рекурсивность функции  $f(x, y) = x + (3 - y)$ , построив описывающие её примитивно рекурсивные схемы.

**Решение.**

1. Введём рекурсию по переменной  $x$  и запишем схему примитивной рекурсии для исходной функции  $f(x, y)$ :

$$f(0, y) = 3 - y = g(y)$$

$$f(x + 1, y) = x + 1 + (3 - y) = s(f(x, y)) = I_3^3(x, y, s(f(x, y)))$$

2. Из приведённой схемы заметим, что для описания исходной функции достаточно функций  $g(y) = 3 - y$  и  $h(x, y, z) = I_3^3(x, y, s(z))$ , причём  $h(x, y, z)$  представима в виде суперпозиции простейших функций. Примитивно рекурсивное описание  $g(y)$  будет выглядеть следующим образом [1]:

$$g(0) = 3 = s(s(0))$$

$$g(y + 1) = h_1(y, g(y)) = 3 - (y + 1) = 2 - y$$

3. Следовательно,  $h_1(y, z) = I_1^2(\varphi(y), z)$ , где  $\varphi(y) = 2 - y$ . Причём для  $\varphi(y)$  справедливо:

$$\varphi(0) = 2 = s(0),$$

$$\varphi(y + 2) = 0$$

4. Таким образом,  $f(x, y)$  строится из простейших функций с помощью операций суперпозиции и примитивной рекурсии, значит она является примитивно рекурсивной, что и требовалось доказать.

**Задание №2.** Применить операцию минимизации к функции  $f(x)$ .  
Результирующую функцию представить в т.н. аналитической форме. Дано:

$$f(x) = \begin{cases} 3x + 2, & \text{если } x \neq 2 \\ \text{не определена,} & \text{если } x = 2 \end{cases}$$

**Решение.**

1. Опираясь на теоретическое определение, для каждого  $x_0 \in N$  найдем минимальное решение уравнения  $f(y) = x_0$ . Поскольку множеством значений функции  $f(x)$  является множество  $E = \{2, 5\} \cup \{11, 14, 17, \dots, 3n + 2, \dots\}$ , то уравнение  $f(y) = x_0$  имеет решение лишь при  $x_0 = 2, 5, 11, 14, \dots$  и оно равно  $\frac{x_0 - 2}{3} [1]$ .
2. Решения, больше числа 2, необходимо исключить, поскольку они не являются допустимыми по определению, поскольку  $f(x)$  при  $x = 2$  не определена. Тогда заключаем, что  $g(x) = \mu_x f(x)$  определена только при  $x = 2$  и  $x = 5$ , причём  $g(2) = 0, g(5) = 1$ .
3. Приведём полученное решение к необходимой форме записи. Заметим, что в качестве «аналитической» формы можно использовать формулу  $\frac{x-2}{3} + \overline{sg}(6-x)$ , где под  $\overline{sg}(6-x)$  понимается функция, обратная функции знака числа, определённая для  $x \leq 6$ , причём  $\overline{sg}(6-2) = \overline{sg}(6-5) = 0$ .
4. Покажем это: поскольку сама функция  $sg(x)$  определяется следующим образом:

$$sg(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases},$$

Тогда очевидно, что  $sg(6-2) = sg(4) = 1$ . А так как знак черты сверху над функцией означает ее комплемент (дополнение), то  $\overline{sg}(x) = 1 - sg(x)$ , следовательно  $\overline{sg}(6-2) = 1 - sg(6-x) = 1 - 1 = 0$ .

## ***2.2 Вычисление функций с помощью машины Тьюринга***

Для функций, заведомо вычислимых, результат, соответственно, можно вычислить с помощью машины Тьюринга. Для этого необходимо составить соответствующий алгоритм, а также применить его на некотором конечном автомате. Важно отметить, что существуют функции, которые могут быть вычислены с помощью алгоритма или машины Тьюринга, но не могут быть описаны с использованием примитивной рекурсии. Такие функции называются *нерекурсивными вычислимыми функциями* или *вычислимыми функциями более высокого уровня* [5].

В качестве симуляции машины Тьюринга в данной курсовой работе используется программа, написанная на языке программирования C#, код которой представлен в приложении в листинге 1. В нем содержится описание класса, хранящего информацию о командах машины, а также описание основной программы, исполняющей переданные ей инструкции аналогично настоящей вычислительной машине. Наложим на эмулятор конечного автомата некоторые ограничения. Он будет работать корректно, предварительно считав из файла следующие параметры:

- 1) Алфавит разрешённых символов, который обязательно должен включать пробельный символ «\_»;
- 2) Количество состояний машины, номер начального и конечного состояния;
- 3) Матрица, описывающая состояния. Каждая строка матрицы содержит номер текущего состояния, символ, находящийся на текущей позиции на ленте, символ, который будет записан на эту позицию, направление сдвига (влево, вправо или без сдвига) и номер следующего состояния.

Таким образом, для программы сложения двоичных чисел на ленте автомата, формат входных данных для неё будет иметь вид, приведённый на рисунке 2.1. При запуске программы на вход в консоль необходимо также



подать машинное слово, которое может содержать только символы алфавита. Машинное слово обязательно должно оканчиваться символом пробела «\_». Программа считывает его и обрабатывает согласно указанным инструкциям.

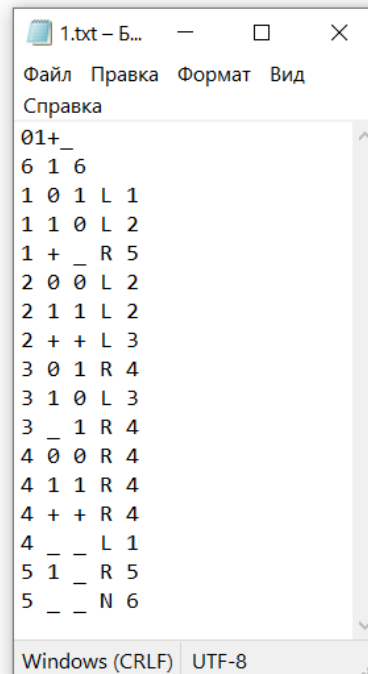


Рисунок 2.1 – Формат входных данных программной реализации машины Тьюринга

**Задание №1.** Два двоичных числа  $a$  и  $b$  записаны на ленту машины в формате « $a+b\_$ ». Реализовать вычислимую функцию сложения (дизъюнкцию, логическое «ИЛИ») двоичных чисел для машины Тьюринга. Результат вычислений должен записываться на месте первого числа.

**Решение.** Для реализации сложения двух двоичных чисел с помощью машины Тьюринга используем следующий алгоритм:

1. Вычесть из числа  $b$  единицу.
2. Перейти к младшему разряду числа  $a$  и прибавить к этому разряду единицу.
3. Вернуться к числу  $b$ , повторять шаги 1-2 до тех пор, пока оно не обнулится.

4. Стереть число  $b$  для того, чтобы на ленте остался только результат сложения.

Для удобства расположим считывающую головку (каретку) справа от числа  $b$ . Теперь опишем алгоритм программы для машины Тьюринга. Он приведён в таблице 2.1.

Таблица 2.1 – Программа сложения двух двоичных чисел для машины Тьюринга

Состояние	0	1	+	$\lambda$	Комментарий
$q_0$	1 L $q_0$	0 L $q_1$	$\lambda$ R $q_5$	$\lambda$ N $q_0$	Вычесть 1 из $b$
$q_1$	0 L $q_1$	1 L $q_1$	+ L $q_1$	$\lambda$ N $q_1$	Переход к младшему разряду $a$
$q_2$	1 R $q_4$	0 L $q_2$	+ N $q_2$	1 R $q_3$	Прибавить 1 к $a$
$q_3$	0 R $q_3$	1 R $q_3$	+ R $q_3$	$\lambda$ L $q_1$	Встать на младший символ $b$
$q_4$	0 N $q_4$	$\lambda$ R $q_4$	+ N $q_4$	!	Стереть лишние символы

**Пример №1.** Пусть  $a = 010$  и  $b = 111$ . После применения дизъюнкции получим результат вычисления искомой функции  $f(a, b) = 010 + 111 = 1001$ . Проверку полученного результата удобно осуществить, сложив числа в десятичной системе счисления:  $a = 010_2 = 0 * 2^0 + 1 * 2^1 + 0 * 2^2 = 2$ ,  $b = 111_2 = 1 * 2^0 + 1 * 2^1 + 1 * 2^2 = 7$  тогда искомая сумма – это результат сложения  $2 + 7 = 9$ , а  $9 = 1 * 2^3 + 1 * 2^0 = 1001$ , верно. Результат работы симуляции машины Тьюринга для указанных входных данных представлен на рисунке 1 приложения Б.

**Пример №2.** Пусть  $a = 11$  и  $b = 11$ . После применения дизъюнкции получим  $f(a, b) = 11 + 11 = 110$ . Проверку полученного результата удобно осуществить, сложив числа в десятичной системе счисления:  $11_2 = 1 * 2^0 + 1 * 2^1 = 3$ , тогда искомая сумма – это результат сложения  $3 + 3 = 6$ , а  $6 = 1 * 2^1 +$

$1 * 2^2 = 110$ , верно. Результат работы симуляции машины Тьюринга для указанных входных данных представлен на рисунке 2 приложения Б.

**Задание №2.** Два двоичных числа  $a$  и  $b$  записаны на ленту машины в формате «a|b\_». Вычислить функцию Шеффера (штрих Шеффера, «И-НЕ») для машины Тьюринга. Результат вычислений должен записываться на месте первого числа.

**Решение.** Прежде всего опишем алгоритм вычисления функции Шеффера. Пусть имеется два двоичных числа  $a$  и  $b$ , тогда функция Шеффера для них определяется следующим образом:  $f(a, b) = a | b = \overline{(a \& b)} = \bar{a} \vee \bar{b}$ .

Для вычисления данной функции необходимо применить инверсию к каждому из чисел, а после сложить их. Теперь опишем алгоритм программы для машины Тьюринга:

Таблица 2.2 – Программа вычисления функции Шеффера для двух двоичных чисел

Состояние	0	1		$\lambda$	Комментарий
$q_0$	1 R $q_0$	0 R $q_0$	R $q_0$	$\lambda$ N $q_1$	Проинвертировать a и b
$q_1$	1 L $q_1$	0 L $q_2$	$\lambda$ R $q_5$	$\lambda$ L $q_1$	Вычесть 1 из b
$q_2$	0 L $q_2$	1 L $q_2$	L $q_3$	$\lambda$ N $q_2$	Встать на младший разряд a
$q_3$	1 R $q_4$	0 L $q_3$	N $q_3$	1 R $q_4$	Прибавить 1 к a
$q_4$	0 R $q_4$	1 R $q_4$	R $q_4$	$\lambda$ L $q_1$	Встать на младший символ b
$q_5$	0 N $q_5$	$\lambda$ R $q_5$	N $q_5$	!	Стереть лишние символы

Согласно описанному ранее синтаксису машины, для программы вычисления функции Шеффера, формат входных данных будет иметь вид, приведённый на рисунке 2.2. При запуске программы на вход в консоль всё также необходимо также подать машинное слово, которое может содержать

только символы алфавита. Машинное слово обязательно должно оканчиваться символом пробела «\_».

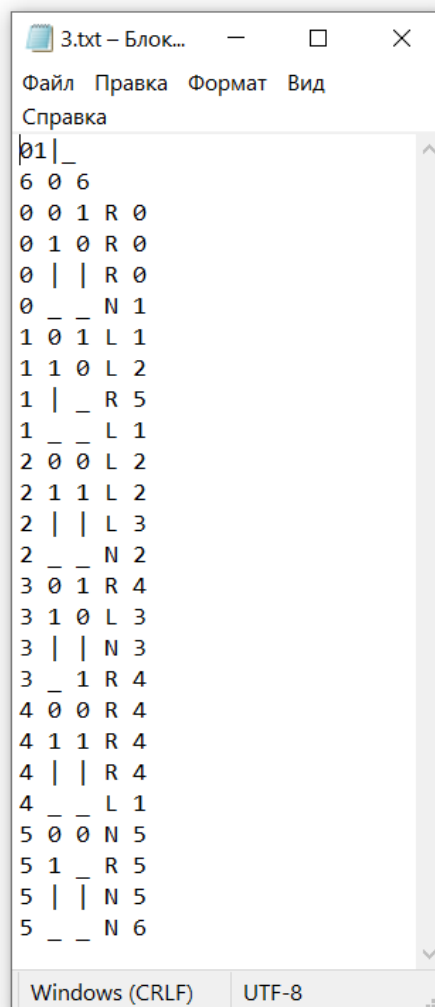


Рисунок 2.2 – Формат входных данных программы вычисления функции Шеффера

**Пример №1.** Пусть  $a = 101$ ,  $b = 110$ , применим к ним функцию Шеффера. После инверсии  $a = 010$ ,  $b = 001$ . После применения дизъюнкции  $a+b = 011$  получим результат вычисления искомой функции  $f(a, b) = 101|110 = 011$ . Результат работы симуляции машины Тьюринга для указанных входных данных представлен на рисунке 3 приложения Б. Каретка стоит слева от числа  $a$ . Проверим полученный результат по таблице истинности для функции Шеффера, приведённой ниже (таблица 2.3):

Таблица 2.3 – Таблица истинности функции Шеффера

x	y	$x   y$
0	0	1
0	1	1
1	0	1
1	1	0

**Пример №2.** Пусть  $a = 111$ ,  $b = 100$ , применим к ним функцию Шеффера. После инверсии  $a = 000$ ,  $b = 011$ . После применения дизъюнкции  $a+b = 011$  получим результат вычисления искомой функции  $f(a, b) = 111|100 = 011$ . Результат работы симуляции машины Тьюринга для указанных входных данных представлен на рисунке 4 приложения Б.

Сравнив все полученные результаты работы программы, приведённые в приложении Б, с аналитическими расчётами, можно сделать вывод, что вычисления, проведённые согласно полученным алгоритмам, корректны.

## Заключение

Изучение машины Тьюринга, вычислимых и рекурсивных функций позволяет получить фундаментальные знания о принципах вычислений и алгоритмов, а также развить навыки разработки программного обеспечения. Эти концепции имеют широкий спектр применений в различных областях компьютерных наук, математики и информатики, и являются основой для дальнейших исследований и разработок в этих областях. В ходе данной курсовой работы были рассмотрены такие понятия, как машина Тьюринга и вычислимые функции, а также решены практические задачи, связанные с ними.

Машина Тьюринга является универсальной вычислительной моделью, способной моделировать любые алгоритмы и вычислительные процессы. Она имеет важное значение в теории вычислений и служит основой для исследования сложности алгоритмов и компьютерных систем. В ходе работы были рассмотрены основные понятия, связанные с абстрактной вычислительной машиной Тьюринга, вычислимыми и рекурсивными функциями, а также их практические применения.

Вычислимые и рекурсивные функции представляют собой математические концепции, которые используются для описания и анализа различных алгоритмических процессов. Они находят широкое применение в практике разработки программного обеспечения и решения различных задач. Например, вычислимые функции используются для обработки данных, оптимизации кода, создания компиляторов и интерпретаторов, а также для решения задач поиска, сортировки, обработки изображений и звука, математических вычислений и других алгоритмических задач.

### **Список использованных источников**

1. Гаврилов Г. П., Сапоженко А. А. Задачи и упражнения по дискретной математике: Учеб. пособие. – 3-е изд., перераб. – М.: ФИЗМАТЛИТ, 2005. – 416 с.
2. Яблонский С. В. Введение в дискретную математику: учебное пособие для вузов / С. В. Яблонский .– 5-е изд., стер. – М.: Высшая школа, 2008. – 384 с.
3. Введение в дискретную математику: Учеб. пособие для вузов /Под ред. В.А. Садовниченко. – 4-е изд., стер. – М.: Высш. шк.; 2003. – 384 с.
4. Тишин В.В. Дискретная математика в примерах и задачах. – СПб.: БХВ-Петербург, 2008. – 352 с.
5. Пильщиков В.Н., Абрамов В.Г., Вылиток А.А., Горячая И.В. Машина Тьюринга и алгоритмы Маркова. Решение задач. (Учебно-методическое пособие) – М.: МГУ, 2016. – 72 с.
6. Марченков С. С. Избранные главы дискретной математики. — М.: МАКС Пресс, 2016. — 136 с

## Листинг 1. Текст программы эмулятора машины Тьюринга на языке программирования C#

```

using System;
using System.Collections.Generic;
using System.IO;

// Структура для хранения инструкций машины Тьюринга

struct Instruction
{
    public char writeSymbol;
    public char moveDirection;
    public int nextState;
}

class Program
{
    // Вывод текущего состояния ленты

    static void PrintTape(List<char> tape)
    {
        Console.Write("Current tape: ");
        foreach (char symbol in tape)
        {
            Console.Write(symbol);
        }
        Console.WriteLine();
    }

    static void Main(string[] args)
    {
        Console.Write("Enter input word: ");
        string inputWord = Console.ReadLine();

        List<char> tape = new List<char>();
        foreach (char symbol in inputWord)
        {
            tape.Add(symbol);
        }

        string alphabet;
        int numberOfStates;
        int startState;
        int haltState;

        Dictionary<(int, char), Instruction> instructions = new Dictionary<(int,
char), Instruction>();

        // Чтение входных данных из файла input.txt

        using (StreamReader inputFile =
            new StreamReader(@"D:\WORK\2 КУРС\КУРСОВАЯ ДМ 4
СЕМЕСТР\TuringMachine\input.txt"))
        {

```



## Листинг 1. Текст программы эмулятора машины Тьюринга на языке программирования C# (продолжение)

```
alphabet = inputFile.ReadLine();
string inputLine = inputFile.ReadLine();
string[] inputValues = inputLine.Split(' ');
numberOfStates = int.Parse(inputValues[0]);
startState = int.Parse(inputValues[1]);
haltState = int.Parse(inputValues[2]);

// Чтение инструкций из файла input.txt и сохранение их в таблицу
инструкций

while ((inputLine = inputFile.ReadLine()) != null)
{
    inputValues = inputLine.Split(' ');
    int state = int.Parse(inputValues[0]);
    char symbol = char.Parse(inputValues[1]);
    Instruction instr = new Instruction
    {
        writeSymbol = char.Parse(inputValues[2]),
        moveDirection = char.Parse(inputValues[3]),
        nextState = int.Parse(inputValues[4])
    };
    instructions[(state, symbol)] = instr;
}

// Инициализация начального состояния машины Тьюринга

int currentState = startState;
int headPosition = tape.Count-2;

// Цикл эмуляции работы машины Тьюринга

while (currentState != haltState)
{
    char currentSymbol = tape[headPosition];
    Instruction instr = instructions[(currentState, currentSymbol)];
    tape[headPosition] = instr.writeSymbol;

    if (instr.moveDirection == 'L')
    {
        if (headPosition == 0)
        {
            tape.Insert(0, alphabet[0]);
        }
        else
        {
            headPosition--;
        }
    }
    else if (instr.moveDirection == 'R')
    {
        headPosition++;
        if (headPosition == tape.Count)
        {
            tape.Add(alphabet[0]);
        }
    }
    PrintTape(tape);
    currentState = instr.nextState;
}
```

## Листинг 1. Текст программы эмулятора машины Тьюринга на языке программирования C# (продолжение)

```
        // Вывод результата работы машины Тьюринга на экран

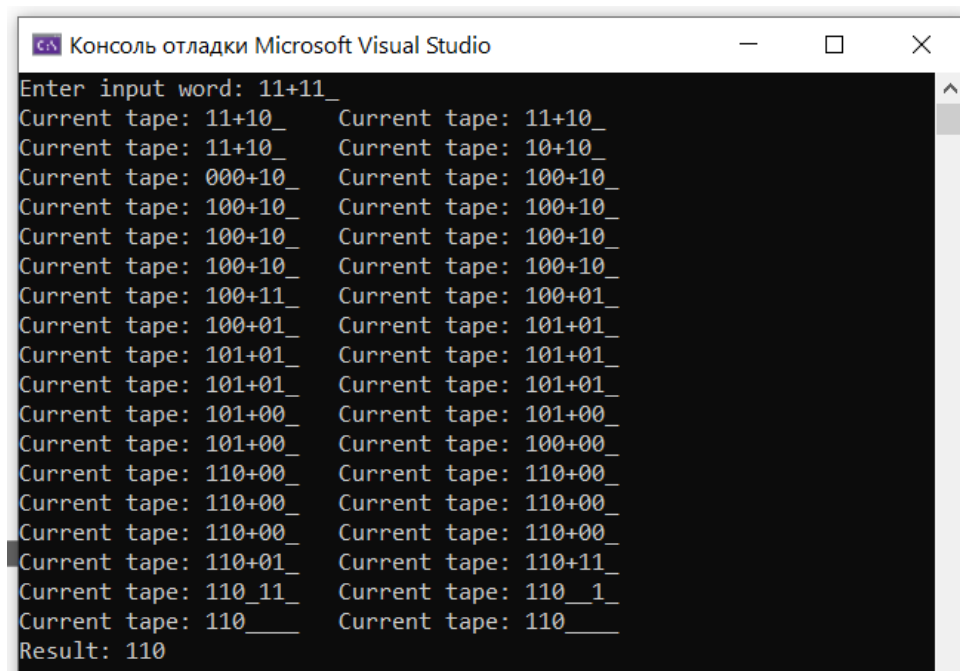
        Console.Write("Result: ");
        foreach (char symbol in tape)
        {
            if (symbol != '_') Console.Write(symbol);
        }
        Console.WriteLine();
    }
}
```

```

Консоль отладки Microsoft Visual Studio
Enter input word: 010+111_
Current tape: 010+110_ Current tape: 010+110_
Current tape: 010+110_ Current tape: 010+110_
Current tape: 011+110_ Current tape: 011+110_
Current tape: 011+110_ Current tape: 011+110_
Current tape: 011+110_ Current tape: 011+110_
Current tape: 011+111_ Current tape: 011+101_
Current tape: 011+101_ Current tape: 011+101_
Current tape: 010+101_ Current tape: 000+101_
Current tape: 100+101_ Current tape: 100+101_
Current tape: 100+101_ Current tape: 100+101_
Current tape: 100+101_ Current tape: 100+101_
Current tape: 100+101_ Current tape: 100+101_
Current tape: 100+100_ Current tape: 100+100_
Current tape: 100+100_ Current tape: 100+100_
Current tape: 101+100_ Current tape: 101+100_
Current tape: 101+100_ Current tape: 101+100_
Current tape: 101+100_ Current tape: 101+100_
Current tape: 101+101_ Current tape: 101+111_
Current tape: 101+011_ Current tape: 101+011_
Current tape: 100+011_ Current tape: 110+011_
Current tape: 110+011_ Current tape: 110+011_
Current tape: 110+011_ Current tape: 110+011_
Current tape: 110+011_ Current tape: 110+011_
Current tape: 110+010_ Current tape: 110+010_
Current tape: 110+010_ Current tape: 110+010_
Current tape: 111+010_ Current tape: 111+010_
Current tape: 111+010_ Current tape: 111+010_
Current tape: 111+010_ Current tape: 111+010_
Current tape: 111+011_ Current tape: 111+001_
Current tape: 111+001_ Current tape: 111+001_
Current tape: 110+001_ Current tape: 100+001_
Current tape: 0000+001_ Current tape: 1000+001_
Current tape: 1000+001_ Current tape: 1000+001_
Current tape: 1000+001_ Current tape: 1000+001_
Current tape: 1000+001_ Current tape: 1000+001_
Current tape: 1000+001_ Current tape: 1000+001_
Current tape: 1000+000_ Current tape: 1000+000_
Current tape: 1000+000_ Current tape: 1000+000_
Current tape: 1001+000_ Current tape: 1001+000_
Current tape: 1001+000_ Current tape: 1001+000_
Current tape: 1001+000_ Current tape: 1001+000_
Current tape: 1001+001_ Current tape: 1001+011_
Current tape: 1001+111_ Current tape: 1001_111_
Current tape: 1001_11_ Current tape: 1001__1_
Current tape: 1001_____ Current tape: 1001_____
Result: 1001

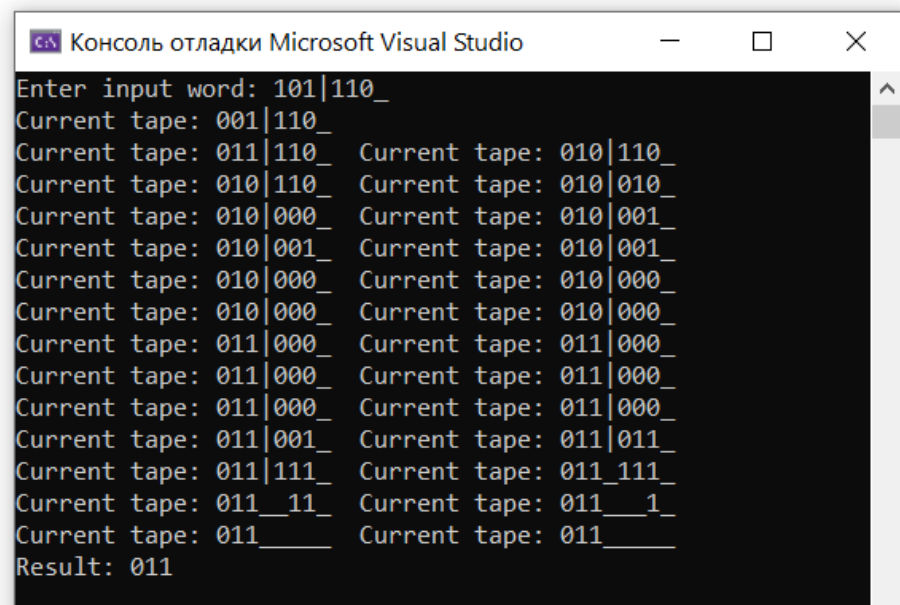
```

Рисунок 1 – Результат работы программы сложения при  $a = 010$ ,  $b = 111$



```
Enter input word: 11+11_
Current tape: 11+10_   Current tape: 11+10_
Current tape: 11+10_   Current tape: 10+10_
Current tape: 000+10_   Current tape: 100+10_
Current tape: 100+10_   Current tape: 100+10_
Current tape: 100+10_   Current tape: 100+10_
Current tape: 100+10_   Current tape: 100+10_
Current tape: 100+11_   Current tape: 100+01_
Current tape: 100+01_   Current tape: 101+01_
Current tape: 101+01_   Current tape: 101+01_
Current tape: 101+01_   Current tape: 101+01_
Current tape: 101+00_   Current tape: 101+00_
Current tape: 101+00_   Current tape: 100+00_
Current tape: 110+00_   Current tape: 110+00_
Current tape: 110+00_   Current tape: 110+00_
Current tape: 110+00_   Current tape: 110+00_
Current tape: 110+01_   Current tape: 110+11_
Current tape: 110_11_   Current tape: 110_1_
Current tape: 110_____   Current tape: 110_____
Result: 110
```

Рисунок 2 – Результат работы программы сложения при  $a = b = 11$



```
Enter input word: 101|110_
Current tape: 001|110_
Current tape: 011|110_   Current tape: 010|110_
Current tape: 010|110_   Current tape: 010|010_
Current tape: 010|000_   Current tape: 010|001_
Current tape: 010|001_   Current tape: 010|001_
Current tape: 010|000_   Current tape: 010|000_
Current tape: 010|000_   Current tape: 010|000_
Current tape: 011|000_   Current tape: 011|000_
Current tape: 011|000_   Current tape: 011|000_
Current tape: 011|000_   Current tape: 011|000_
Current tape: 011|001_   Current tape: 011|011_
Current tape: 011|111_   Current tape: 011_111_
Current tape: 011_11_   Current tape: 011_1_
Current tape: 011_____   Current tape: 011_____
Result: 011
```

Рисунок 3 – Результат работы программы вычисления функции Шеффера при  $a = 101$ ,  
 $b = 110$

```
Консоль отладки Microsoft Visual Studio
Enter input word: 111|100_
Current tape: 011|100_
Current tape: 001|100_   Current tape: 000|100_
Current tape: 000|100_   Current tape: 000|000_
Current tape: 000|010_   Current tape: 000|011_
Current tape: 000|011_   Current tape: 000|011_
Current tape: 000|010_   Current tape: 000|010_
Current tape: 000|010_   Current tape: 000|010_
Current tape: 001|010_   Current tape: 001|010_
Current tape: 001|010_   Current tape: 001|010_
Current tape: 001|010_   Current tape: 001|010_
Current tape: 001|011_   Current tape: 001|001_
Current tape: 001|001_   Current tape: 001|001_
Current tape: 000|001_   Current tape: 010|001_
Current tape: 010|001_   Current tape: 010|001_
Current tape: 010|001_   Current tape: 010|001_
Current tape: 010|001_   Current tape: 010|001_
Current tape: 010|000_   Current tape: 010|000_
Current tape: 010|000_   Current tape: 010|000_
Current tape: 011|000_   Current tape: 011|000_
Current tape: 011|000_   Current tape: 011|000_
Current tape: 011|000_   Current tape: 011|000_
Current tape: 011|001_   Current tape: 011|011_
Current tape: 011|111_   Current tape: 011_111_
Current tape: 011__11_   Current tape: 011__1_
Current tape: 011_____   Current tape: 011_____
Result: 011
```

Рисунок 4 – Результат работы программы вычисления функции Шеффера при  $a = 111$ ,  
 $b = 100$