Индивидуальное задание №2

Проектирование и реализация машины Тьюринга

Патычкина Елизавета Вадимовна, КТбо1-10

Вариант 1.21

1. **Постановка задачи**

На входной ленте машины Тьюринга заданы два целых неотрицательных числа в системе счисления с основанием 3, разделенных символом #. Первое число не меньше второго

Поставить слева от первого числа знак = и записать разность чисел в системе счисления с основанием 3.

1. **Словесное описание алгоритма решения задачи на машине Тьюринга**

Состояния машины Тьюринга и их смысл:

q1 – начальное состояние, выставление знака «=» слева от первого числа

q2 – бежит по числу до последней просматриваемой цифры первого числа

q3 – запоминает последнюю не просматриваемую цифру первого числа

q4 – бежит до конца первого числа, в начало второго, сохраняя в первом числе символ «0»

q5 – бежит до конца первого числа, в начало второго, сохраняя в первом числе символ «1»

q6 – бежит до конца первого числа, в начало второго, сохраняя в первом числе символ «2»

q7 – бежит до последней просматриваемой цифры второго числа (цифра первого числа «0»)

q8 – сохраняет последний не просматриваемый символ второго числа (сохраненный символ первого числа «0»)

q9 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «0» и «0» первого и второго числа соответственно

q10 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «0» и «1» первого и второго числа соответственно

q11 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «0» и «2» первого и второго числа соответственно

q12 – бежит до последней просматриваемой цифры второго числа (цифра первого числа «1»)

q13 – сохраняет последний не просматриваемый символ второго числа (сохраненный символ первого числа «1»)

q14 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «1» и «0» первого и второго числа соответственно

q15 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «1» и «1» первого и второго числа соответственно

q16 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «1» и «2» первого и второго числа соответственно

q17 – бежит до последней просматриваемой цифры второго числа (цифра первого числа «2»)

q18 – сохраняет последний не просматриваемый символ второго числа (сохраненный символ первого числа «2»)

q19 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «2» и «0» первого и второго числа соответственно

q20 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «2» и «1» первого и второго числа соответственно

q21 – бежит влево до первой пустой ячейки ленты и записывает результат характерный для комбинации символов «2» и «2» первого и второго числа соответственно

q22 – занимает единицу у старшего разряда числа

q23 – бежит до конца ленты вправо

q24 – бежит до конца ленты влево, изменяя символы чисел на исходные

q25 – удаляет незначащие нули

q26 – пишет ноль в случае пустого ответа

Алгоритм работы машина Тьюринга:

Машина запрашивает у пользователя входную строчку, соответствующую требованиям. В случае, когда строка введена неверно, пользователь может повторить ввод еще раз. Если строка введена верно, машина начинает свою работу. Машина выполняет команды до тех пор, пока не окажется в состоянии q0. Поскольку комбинация текущего символа и состояния в системе команд являются уникальными, то в зависимости от их значений машина Тьюринга меняет текущее состояние и символ, на состояние и символ, указанные в команде.

Машина Тьюринга первым делом ставит символ «=» слева от первого числа. Затем бежит до конца числа (в случае, если автомат проходит первое число в первый раз) или до первой просматриваемой цифры первого числа и запоминает последнюю цифру числа, которая еще не была рассмотрена машиной. Затем головка ленты бежит до конца числа (в случае, если автомат проходит второе число в первый раз) или до первой просматриваемой цифры второго числа и запоминает последнюю цифру числа, которая еще не была рассмотрена машиной. Далее зная пару цифр одного разряда первого и второго числа, головка бежит до первой левой пустой ячейки ленты и записывает значение, соответствующее двум данным символам. После чего машина Тьюринга повторяет описываемые ранее действия, до тех пор, пока не закончится первое число. Если первое число полностью рассмотрено, головка бежит вправо до первой пустой ячейки. Затем бежит влево, заменяя символы на исходные до последней непустой ячейки. Удаляет незначащие нули и завершает работу.

1. **Используемый алфавит ленты**

A = {0, 1, 2, #, =, a, b, c, \*, λ} – алфавит машины Тьюринга (входной, выходной, вспомогательный)

A.вх. = {0, 1, 2, #, λ} – входной алфавит машины Тьюринга

A.вых. = {0, 1, 2, #, λ, =} – выходной алфавит машины Тьюринга

A.всп. = {a, b, c, \*, λ} – вспомогательный алфавит машины Тьюринга

1. **Система команд машины Тьюринга в виде диаграммы**

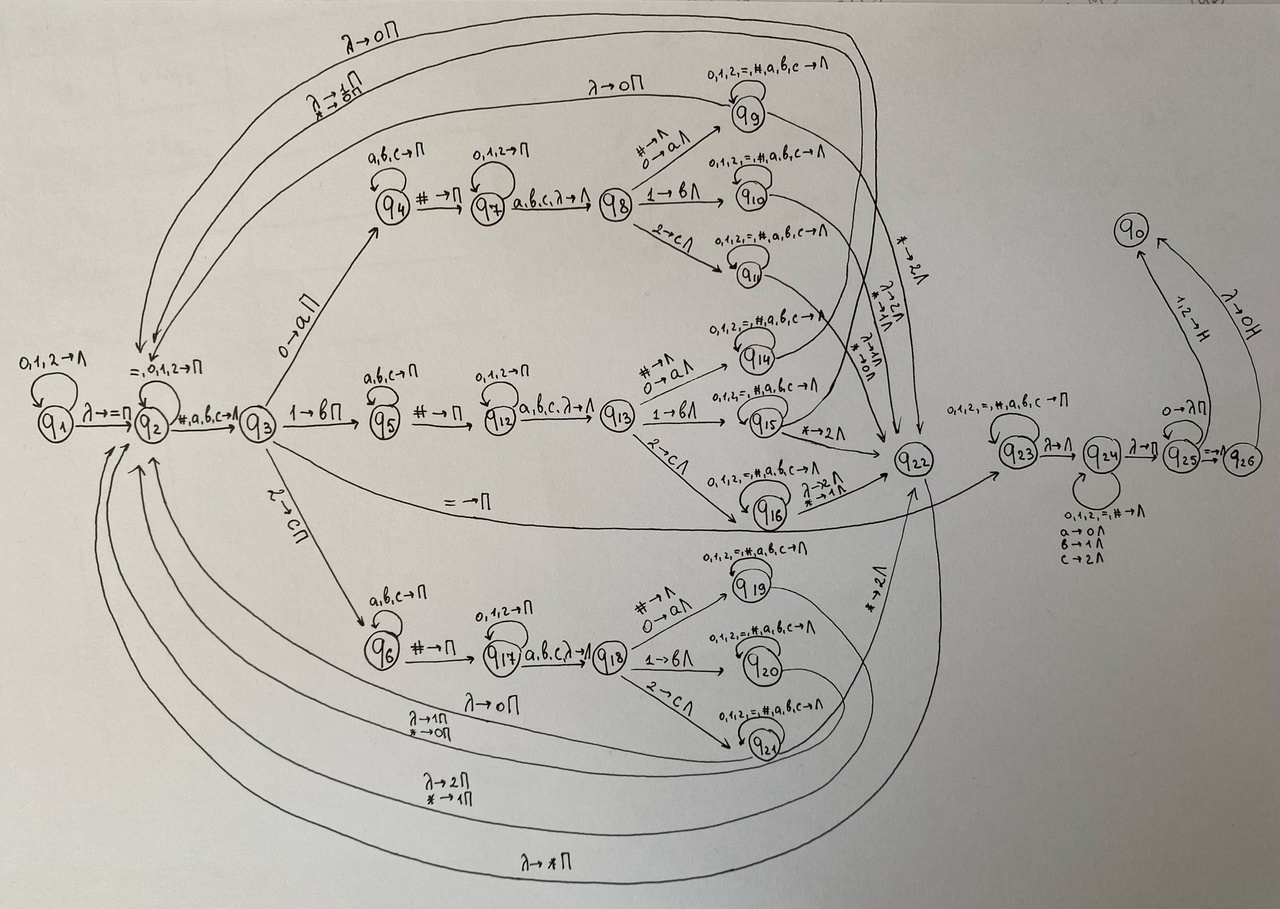


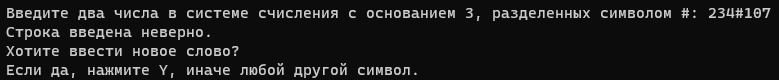
Рисунок 1 – Диаграмма машины Тьюринга

1. **Система команд машины Тьюринга в виде таблицы**

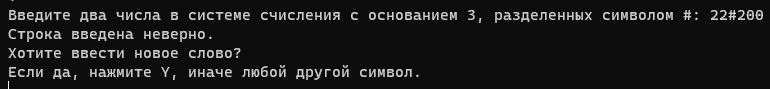
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | = | # | a | b | c | λ | \* |
| q0 |  |  |  |  |  |  |  |  |  |  |
| q1 | 0Лq1 | 1Лq1 | 2Лq1 |  |  |  |  |  | =Пq2 |  |
| q2 | 0Пq2 | 1Пq2 | 2Пq2 | =Пq2 | #Лq3 | aЛq3 | bЛq3 | cЛq3 |  |  |
| q3 | aПq4 | bП5 | cПq6 | =Пq23 |  |  |  |  |  |  |
| q4 |  |  |  |  | #Пq7 | aПq4 | bПq4 | cПq4 |  |  |
| q5 |  |  |  |  | #Пq12 | aПq5 | bПq5 | cПq5 |  |  |
| q6 |  |  |  |  | #Пq17 | aПq6 | bПq6 | cПq6 |  |  |
| q7 | 0Пq7 | 1Пq7 | 2Пq7 |  |  | aЛq8 | bЛq8 | cЛq8 | λЛq8 |  |
| q8 | aЛq9 | bЛq10 | cЛq11 |  | #Лq9 |  |  |  |  |  |
| q9 | 0Лq9 | 1Лq9 | 2Лq9 | =Лq9 | #Лq9 | aЛq9 | bЛq9 | cЛq9 | 0Пq2 | 2Лq22 |
| q10 | 0Лq10 | 1Лq10 | 2Лq10 | =Лq10 | #Лq10 | aЛq10 | bЛq10 | cЛq10 | 2Лq22 | 1Лq22 |
| q11 | 0Лq11 | 1Лq11 | 2Лq11 | =Лq11 | #Лq11 | aЛq11 | bЛq11 | cЛq11 | 1Лq22 | 0Лq22 |
| q12 | 0Пq12 | 1Пq12 | 2Пq12 |  |  | aЛq13 | bЛq13 | cЛq13 | λЛq13 |  |
| q13 | aЛq14 | bЛq15 | cЛq16 |  | #Лq14 |  |  |  |  |  |
| q14 | 0Лq14 | 1Лq14 | 2Лq14 | =Лq14 | #Лq14 | aЛq14 | bЛq14 | cЛq14 | 1Пq2 | 0Пq2 |
| q15 | 0Лq15 | 1Лq15 | 2Лq15 | =Лq15 | #Лq15 | aЛq15 | bЛq15 | cЛq15 | 0Пq2 | 2Лq22 |
| q16 | 0Лq16 | 1Лq16 | 2Лq16 | =Лq16 | #Лq16 | aЛq16 | bЛq16 | cЛq16 | 2Лq22 | 1Лq22 |
| q17 | 0Пq17 | 1Пq17 | 2Пq17 |  |  | aЛq18 | bЛq18 | cЛq18 | λЛq18 |  |
| q18 | aЛq19 | bЛq20 | cЛq21 |  | #Лq19 |  |  |  |  |  |
| q19 | 0Лq19 | 1Лq19 | 2Лq19 | =Лq19 | #Лq19 | aЛq19 | bЛq19 | cЛq19 | 2Пq2 | 1Пq2 |
| q20 | 0Лq20 | 1Лq20 | 2Лq20 | =Лq20 | #Лq20 | aЛq20 | bЛq20 | cЛq20 | 1Пq2 | 0Пq2 |
| q21 | 0Лq21 | 1Лq21 | 2Лq21 | =Лq21 | #Лq21 | aЛq21 | bЛq21 | cЛq21 | 0Пq2 | 2Лq22 |
| q22 |  |  |  |  |  |  |  |  | \*Пq2 |  |
| q23 | 0Пq23 | 1Пq23 | 2Пq23 | =Пq23 | #Пq23 | aПq23 | bПq23 | cПq23 | λЛq24 |  |
| q24 | 0Лq24 | 1Лq24 | 2Лq24 | =Лq24 | #Лq24 | 0Лq24 | 1Лq24 | 2Лq24 | λПq25 |  |
| q25 | λПq25 | 1Нq0 | 2Нq0 | =Лq26 |  |  |  |  |  |  |
| q26 |  |  |  |  |  |  |  |  | 0Нq0 |  |

Таблица 1 – Таблица системы команд

1. **Набор тестов**
2. 234#107



1. 22#200



1. 120#120

Начальная конфигурация: q1 120#120

Заключительная конфигурация: q0 0=120#120

1. 1120#202

Начальная конфигурация: q1 1120#202

Заключительная конфигурация: q0 211=1120#202

1. 1120#1011

Начальная конфигурация: q1 1120#1011

Заключительная конфигурация: q0 102=1120#1011

1. **Описание программной реализации**

Реализация ленты c использованием класса string.

Структура данных для хранения команды из системы команд

struct command

{

int state\_start;

char symbol\_start;

char position;

int state\_finish;

char symbol\_finish;

};

Целочисленная переменная state\_start хранит состояние команды, при котором выполняется данная команда. Символьная переменная symbol\_start хранит символ, при котором выполняется данная команда. Символьная переменная хранит данные о движении головки при данной команде. Целочисленная переменная state\_finish хранит состояние команды, на которое будет заменено текущее после выполнения команды. Символьная переменная symbol\_finish хранить символ, на который поменяется текущий символ после выполнения команды.

Структура данных для хранения ленты

struct M\_T

{

int position;

int state;

string ribbon;

char symbol;

};

Целочисленная переменная position хранит порядок положения головки на ленте. Целочисленная переменная state хранит значение текущего состояния машины Тьюринга. Строковая переменная ribbon хранит ленту машины. Символьная переменная symbol хранит символ, на который указывает головка в данный момент.

Алгоритм программы:

Введенная строка проверяется функцией int check(string line), и возвращает 1 или 0 в зависимости от того, введена корректная строка или нет соответственно. Инициализирует систему команд функцией void input\_sistem(vector <struct command>& cs). Программа выводит текущую ленту, указатель на текущий символ. Далее вызывает функцию void CharacterProcessing(vector<command> cs, struct M\_T& MT), которая выполняет команду в зависимости от текущего символа, на который указывает головка, и состояния. Программа выполняет функцию CharacterProcessing до тех пор, пока состояние не равно «0».

1. **Листинг программы**

//ЮФУ, ИКТИБ, МОП ЭВМ

//Программирование и основы теории алгоритмов

//Индивидуальное задание №2

//Проектирование и реализация машины Тьюринга

//КТбо1-10, Патычкина Елизавета Вадимовна

#include <iostream>

#include <vector>

#include <string>

using namespace std;

//Структура хранения команды из системы команд

struct command

{

int state\_start;//начальное состояние

char symbol\_start;//начальный символ

char position;//направление движения головки

int state\_finish;//конечное состояние(новое)

char symbol\_finish;//конечный символ(новый)

};

//Структура хранения машины Тьюринга

struct M\_T

{

int position;//коэффициент положения головки

int state;//состояние машины Тьюринга

string ribbon;//лента

char symbol;//текущий обрабатываемый символ

};

void input\_sistem(vector <struct command>& cs);

int check(string line);

void CharacterProcessing(vector<command> cs, struct M\_T& MT);

int main()

{

vector <struct command> command\_system;

input\_sistem(command\_system);

setlocale(LC\_ALL, "Russian");

string ribbon, answer;

struct M\_T MT;

do

{

cout << "Введите два числа в системе счисления с основанием 3, разделенных символом #: ";

cin >> MT.ribbon;

if (check(MT.ribbon) == 0)

{

cout << "Строка введена неверно." << endl;

}

else

{

MT.ribbon.push\_back(' ');

MT.ribbon = ' ' + MT.ribbon;

MT.position = 1;

MT.symbol = MT.ribbon[MT.position];

MT.state = 1;

int fl = 1;

while (MT.state)

{

if (fl)

{

cout << MT.ribbon << endl;

for (int i = 0; i < MT.position; i++)

{

cout << ' ';

}

cout << '^' << endl;

cout << "Текущее состояние машины Тьюринга: " << 'q' << MT.state << endl;

fl = 0;

}

else

{

MT.symbol = MT.ribbon[MT.position];

CharacterProcessing(command\_system, MT);

cout << MT.ribbon << endl;

for (int i = 0; i < MT.position; i++)

{

cout << ' ';

}

cout << '^' << endl;

cout << "Текущее состояние машины Тьюринга: " << 'q' << MT.state << endl;

}

}

}

cout << "Хотите ввести новое слово?" << '**\n**' << "Если да, нажмите Y, иначе любой другой символ." << endl;

cin >> answer;

} while (answer == "Y");

}

//Входные параметры: vector <struct command>& cs - система команд машины Тьюринга

//Функция: заполняет систему команд статическими значениями

//Выходные параметры: функция ничего не возвращает

void input\_sistem(vector <struct command>& cs)

{

cs.push\_back({ 1,'0', 'L', 1, '0' });

cs.push\_back({ 1,'1', 'L', 1, '1' });

cs.push\_back({ 1,'2', 'L', 1, '2' });

cs.push\_back({ 1,' ', 'R', 2, '=' });

cs.push\_back({ 2,'0', 'R', 2, '0' });

cs.push\_back({ 2,'1', 'R', 2, '1' });

cs.push\_back({ 2,'2', 'R', 2, '2' });

cs.push\_back({ 2,'=', 'R', 2, '=' });

cs.push\_back({ 2,'#', 'L', 3, '#' });

cs.push\_back({ 2,'a', 'L', 3, 'a' });

cs.push\_back({ 2,'b', 'L', 3, 'b' });

cs.push\_back({ 2,'c', 'L', 3, 'c' });

cs.push\_back({ 3,'0', 'R', 4, 'a' });

cs.push\_back({ 3,'1', 'R', 5, 'b' });

cs.push\_back({ 3,'2', 'R', 6, 'c' });

cs.push\_back({ 3,'=', 'R', 23, '=' });

cs.push\_back({ 4,'#', 'R', 7, '#' });

cs.push\_back({ 4,'a', 'R', 4, 'a' });

cs.push\_back({ 4,'b', 'R', 4, 'b' });

cs.push\_back({ 4,'c', 'R', 4, 'c' });

cs.push\_back({ 5,'#', 'R', 12, '#' });

cs.push\_back({ 5,'a', 'R', 5, 'a' });

cs.push\_back({ 5,'b', 'R', 5, 'b' });

cs.push\_back({ 5,'c', 'R', 5, 'c' });

cs.push\_back({ 6,'#', 'R', 17, '#' });

cs.push\_back({ 6,'a', 'R', 6, 'a' });

cs.push\_back({ 6,'b', 'R', 6, 'b' });

cs.push\_back({ 6,'c', 'R', 6, 'c' });

cs.push\_back({ 7,'0', 'R', 7, '0' });

cs.push\_back({ 7,'1', 'R', 7, '1' });

cs.push\_back({ 7,'2', 'R', 7, '2' });

cs.push\_back({ 7,'a', 'L', 8, 'a' });

cs.push\_back({ 7,'b', 'L', 8, 'b' });

cs.push\_back({ 7,'c', 'L', 8, 'c' });

cs.push\_back({ 7,' ', 'L', 8, ' ' });

cs.push\_back({ 8,'0', 'L', 9, 'a' });

cs.push\_back({ 8,'1', 'L', 10, 'b' });

cs.push\_back({ 8,'2', 'L', 11, 'c' });

cs.push\_back({ 8,'#', 'L', 9, '#' });

cs.push\_back({ 9,'0', 'L', 9, '0' });

cs.push\_back({ 9,'1', 'L', 9, '1' });

cs.push\_back({ 9,'2', 'L', 9, '2' });

cs.push\_back({ 9,'=', 'L', 9, '=' });

cs.push\_back({ 9,'#', 'L', 9, '#' });

cs.push\_back({ 9,'a', 'L', 9, 'a' });

cs.push\_back({ 9,'b', 'L', 9, 'b' });

cs.push\_back({ 9,'c', 'L', 9, 'c' });

cs.push\_back({ 9,' ', 'R', 2, '0' });

cs.push\_back({ 9,'\*', 'L', 22, '2' });

cs.push\_back({ 10,'0', 'L', 10, '0' });

cs.push\_back({ 10,'1', 'L', 10, '1' });

cs.push\_back({ 10,'2', 'L', 10, '2' });

cs.push\_back({ 10,'=', 'L', 10, '=' });

cs.push\_back({ 10,'#', 'L', 10, '#' });

cs.push\_back({ 10,'a', 'L', 10, 'a' });

cs.push\_back({ 10,'b', 'L', 10, 'b' });

cs.push\_back({ 10,'c', 'L', 10, 'c' });

cs.push\_back({ 10,' ', 'L', 22, '2' });

cs.push\_back({ 10,'\*', 'L', 22, '1' });

cs.push\_back({ 11,'0', 'L', 11, '0' });

cs.push\_back({ 11,'1', 'L', 11, '1' });

cs.push\_back({ 11,'2', 'L', 11, '2' });

cs.push\_back({ 11,'=', 'L', 11, '=' });

cs.push\_back({ 11,'#', 'L', 11, '#' });

cs.push\_back({ 11,'a', 'L', 11, 'a' });

cs.push\_back({ 11,'b', 'L', 11, 'b' });

cs.push\_back({ 11,'c', 'L', 11, 'c' });

cs.push\_back({ 11,' ', 'L', 22, '1' });

cs.push\_back({ 11,'\*', 'L', 22, '0' });

cs.push\_back({ 12,'0', 'R', 12, '0' });

cs.push\_back({ 12,'1', 'R', 12, '1' });

cs.push\_back({ 12,'2', 'R', 12, '2' });

cs.push\_back({ 12,'a', 'L', 13, 'a' });

cs.push\_back({ 12,'b', 'L', 13, 'b' });

cs.push\_back({ 12,'c', 'L', 13, 'c' });

cs.push\_back({ 12,' ', 'L', 13, ' ' });

cs.push\_back({ 13,'0', 'L', 14, 'a' });

cs.push\_back({ 13,'1', 'L', 15, 'b' });

cs.push\_back({ 13,'2', 'L', 16, 'c' });

cs.push\_back({ 13,'#', 'L', 14, '#' });

cs.push\_back({ 14,'0', 'L', 14, '0' });

cs.push\_back({ 14,'1', 'L', 14, '1' });

cs.push\_back({ 14,'2', 'L', 14, '2' });

cs.push\_back({ 14,'=', 'L', 14, '=' });

cs.push\_back({ 14,'#', 'L', 14, '#' });

cs.push\_back({ 14,'a', 'L', 14, 'a' });

cs.push\_back({ 14,'b', 'L', 14, 'b' });

cs.push\_back({ 14,'c', 'L', 14, 'c' });

cs.push\_back({ 14,' ', 'R', 2, '1' });

cs.push\_back({ 14,'\*', 'R', 2, '0' });

cs.push\_back({ 15,'0', 'L', 15, '0' });

cs.push\_back({ 15,'1', 'L', 15, '1' });

cs.push\_back({ 15,'2', 'L', 15, '2' });

cs.push\_back({ 15,'=', 'L', 15, '=' });

cs.push\_back({ 15,'#', 'L', 15, '#' });

cs.push\_back({ 15,'a', 'L', 15, 'a' });

cs.push\_back({ 15,'b', 'L', 15, 'b' });

cs.push\_back({ 15,'c', 'L', 15, 'c' });

cs.push\_back({ 15,' ', 'R', 2, '0' });

cs.push\_back({ 15,'\*', 'L', 22, '2' });

cs.push\_back({ 16,'0', 'L', 16, '0' });

cs.push\_back({ 16,'1', 'L', 16, '1' });

cs.push\_back({ 16,'2', 'L', 16, '2' });

cs.push\_back({ 16,'=', 'L', 16, '=' });

cs.push\_back({ 16,'#', 'L', 16, '#' });

cs.push\_back({ 16,'a', 'L', 16, 'a' });

cs.push\_back({ 16,'b', 'L', 16, 'b' });

cs.push\_back({ 16,'c', 'L', 16, 'c' });

cs.push\_back({ 16,' ', 'L', 22, '2' });

cs.push\_back({ 16,'\*', 'L', 22, '1' });

cs.push\_back({ 17,'0', 'R', 17, '0' });

cs.push\_back({ 17,'1', 'R', 17, '1' });

cs.push\_back({ 17,'2', 'R', 17, '2' });

cs.push\_back({ 17,'a', 'L', 18, 'a' });

cs.push\_back({ 17,'b', 'L', 18, 'b' });

cs.push\_back({ 17,'c', 'L', 18, 'c' });

cs.push\_back({ 17,' ', 'L', 18, ' ' });

cs.push\_back({ 18,'0', 'L', 19, 'a' });

cs.push\_back({ 18,'1', 'L', 20, 'b' });

cs.push\_back({ 18,'2', 'L', 21, 'c' });

cs.push\_back({ 18,'#', 'L', 19, '#' });

cs.push\_back({ 19,'0', 'L', 19, '0' });

cs.push\_back({ 19,'1', 'L', 19, '1' });

cs.push\_back({ 19,'2', 'L', 19, '2' });

cs.push\_back({ 19,'=', 'L', 19, '=' });

cs.push\_back({ 19,'#', 'L', 19, '#' });

cs.push\_back({ 19,'a', 'L', 19, 'a' });

cs.push\_back({ 19,'b', 'L', 19, 'b' });

cs.push\_back({ 19,'c', 'L', 19, 'c' });

cs.push\_back({ 19,' ', 'R', 2, '2' });

cs.push\_back({ 19,'\*', 'R', 2, '1' });

cs.push\_back({ 20,'0', 'L', 20, '0' });

cs.push\_back({ 20,'1', 'L', 20, '1' });

cs.push\_back({ 20,'2', 'L', 20, '2' });

cs.push\_back({ 20,'=', 'L', 20, '=' });

cs.push\_back({ 20,'#', 'L', 20, '#' });

cs.push\_back({ 20,'a', 'L', 20, 'a' });

cs.push\_back({ 20,'b', 'L', 20, 'b' });

cs.push\_back({ 20,'c', 'L', 20, 'c' });

cs.push\_back({ 20,' ', 'R', 2, '1' });

cs.push\_back({ 20,'\*', 'R', 2, '0' });

cs.push\_back({ 21,'0', 'L', 21, '0' });

cs.push\_back({ 21,'1', 'L', 21, '1' });

cs.push\_back({ 21,'2', 'L', 21, '2' });

cs.push\_back({ 21,'=', 'L', 21, '=' });

cs.push\_back({ 21,'#', 'L', 21, '#' });

cs.push\_back({ 21,'a', 'L', 21, 'a' });

cs.push\_back({ 21,'b', 'L', 21, 'b' });

cs.push\_back({ 21,'c', 'L', 21, 'c' });

cs.push\_back({ 21,' ', 'R', 2, '0' });

cs.push\_back({ 21,'\*', 'L', 22, '2' });

cs.push\_back({ 22,' ', 'R', 2, '\*' });

cs.push\_back({ 23,'0', 'R', 23, '0' });

cs.push\_back({ 23,'1', 'R', 23, '1' });

cs.push\_back({ 23,'2', 'R', 23, '2' });

cs.push\_back({ 23,'=', 'R', 23, '=' });

cs.push\_back({ 23,'#', 'R', 23, '#' });

cs.push\_back({ 23,'a', 'R', 23, 'a' });

cs.push\_back({ 23,'b', 'R', 23, 'b' });

cs.push\_back({ 23,'c', 'R', 23, 'c' });

cs.push\_back({ 23,' ', 'L', 24, ' ' });

cs.push\_back({ 24,'0', 'L', 24, '0' });

cs.push\_back({ 24,'1', 'L', 24, '1' });

cs.push\_back({ 24,'2', 'L', 24, '2' });

cs.push\_back({ 24,'=', 'L', 24, '=' });

cs.push\_back({ 24,'#', 'L', 24, '#' });

cs.push\_back({ 24,'a', 'L', 24, '0' });

cs.push\_back({ 24,'b', 'L', 24, '1' });

cs.push\_back({ 24,'c', 'L', 24, '2' });

cs.push\_back({ 24,' ', 'R', 25, ' ' });

cs.push\_back({ 25,'0', 'R', 25, ' ' });

cs.push\_back({ 25,'1', 'N', 0, '1' });

cs.push\_back({ 25,'2', 'N', 0, '2' });

cs.push\_back({ 25,' ', 'R', 25, ' ' });

cs.push\_back({ 25,'=', 'L', 26, '=' });

cs.push\_back({ 26,' ', 'N', 0, '0' });

}

//Входные параметры: string line - входная строка, введенная пользователем

//Функция: проверяет корректность введенной строки согласно требованиям в условии

//Выходные параметры: функция возвращает 1, если введенная строка корректна

// 0, если строка введена с ошибками

int check(string line)

{

int flag1 = 1;

int flag2 = 1;

int count = 0, count1 = 0, count2 = 0;

for (int c = 0; (c < line.size()) && (flag2); c++)

{

if (flag1 == 1)

{

if ((line[c] == '0') || (line[c] == '1') || (line[c] == '2') || (line[c] == '#'))

{

if ((line[c] == '0') || (line[c] == '1') || (line[c] == '2'))

count1 = count1 \* 10 + (line[c] - 48);

if (line[c] == '#')

flag1 = 0;

}

else

{

flag2 = 0;

}

}

else

{

if ((line[c] == '0') || (line[c] == '1') || (line[c] == '2'))

{

count2 = count2 \* 10 + (line[c] - 48);

}

else

{

flag2 = 0;

}

}

}

if (count2 > count1)

flag2 = 0;

return flag2;

}

//Входные параметры: vector <struct command>& cs - систома команд машины Тьюринга

// struct M\_T& MT - структура машины Тьюринга, содержащая строку, коэффициент положения головки, текущий символ и состояние

//Функция: обрабатывает символ, меняет состояние и совершает движение вправо/влево, в зависимости от системы команд машины Тьюринга

//Выходные параметры: функция ничего не возвращает

void CharacterProcessing(vector<struct command> cs, struct M\_T& MT)

{

int fl = 1;

for (auto i = cs.begin(); (i != cs.end()) && (fl); i++)

{

if (MT.state == i->state\_start && MT.symbol == i->symbol\_start)

{

MT.state = i->state\_finish;

MT.ribbon[MT.position] = i->symbol\_finish;

MT.symbol = i->symbol\_finish;

if (i->position == 'R')

{

MT.position++;

if (MT.position == MT.ribbon.size())

{

MT.ribbon = MT.ribbon + ' ';

}

}

if (i->position == 'L')

{

if (MT.position == 0)

{

MT.ribbon = ' ' + MT.ribbon;

}

else

{

MT.position--;

}

}

fl = 0;

}

}

}