# Элементы теории алгоритмов

### Практические работы

## Машина Тьюринга

1. Опишите алгоритм работы программы для машины Тьюринга:

|  |  |  |
| --- | --- | --- |
|  | *q1* | *q2* |
| а | *q2* | 🞎 ← |
| б | *q2* | 🞎 ← |
| 🞎 | ← | *q0* |

*Ответ*:

*Алгоритм удаляет слово, находящееся справа от слова.*

При каком начальном состоянии ленты и положении каретки эта программа зацикливается?

*Ответ*:

*Если слово бесконечно, или слов нет вовсе.*

1. Составьте программу для машины Тьюринга, которая заменяет в двоичном числе все 0 на 1 и все 1 на 0 (из числа 10101100 получается 01010011). Каретка находится слева от числа.

|  |  |  |
| --- | --- | --- |
|  | *q1* | *q2* |
| 0 | Q2 | 1-> |
| 1 | Q2 | 0-> |
| 🞎 | -> | Q0 |

*Описание состояний*:

*q1 –Если требуется то, алгоритм доходит до слова.*

*q2 –меняет 1 на 0 и 0 на 1, а если доходит до пробела, то останов.*

1. Составьте программу для машины Тьюринга, которая увеличивает троичное число на 1. Каретка находится справа от числа.

|  |  |  |
| --- | --- | --- |
|  | *q1* | *q2* |
| 0 | Q2 | 1•Q0 |
| 1 | Q2 | 2•Q0 |
| 2 | Q2 | 0**←** |
| 🞎 | **←** | 1•Q0 |

*Описание состояний*:

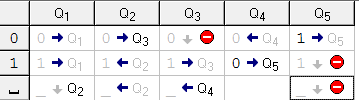
*q1 – доходит до начала слова*

*q2 –если 0, то его увеличивает и останов, аналогично для 1, для 2 записываем 0 и каретку влево, для пробела добавляет еще один разряд.*

При каком начальном положении каретки эта программа зацикливается?

*Ответ*:

*Если каретка находится слева от слова*

1. Составьте программу для машины Тьюринга, которая уменьшает двоичное число на 1. Каретка находится над числом.

*Описание состояний*:

*q1 –поиск конца слова*

*q2 –поиск первого нуля*

*q3 –поиск единицы*

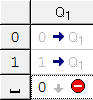
*q4 – запись нуля (при случае: 1001 → 1000)*

*q5–замена после нуля единицами (при случае: 1000 → 0111)*

При каком начальном положении каретки эта программа зацикливается?

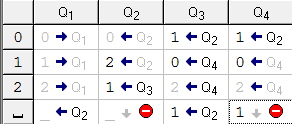
*Ответ:*

Если каретка находится перед числом, если после числа, то микрокод работает неправильно.

1. **Составьте программу для машины Тьюринга, которая умножает двоичное число на 2. Каретка находится над числом.

*Описание состояний*:

*q1 –поиск конца числа и добавление ему нуля.*

1. Составьте программу для машины Тьюринга, которая умножает троичное число на 2. Каретка находится над числом.

*Описание состояний*:

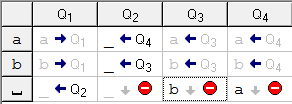
*q1 –поиск конца слова.*

*q2 –основной цикл*

*q3 –умножение при (2\*23=113)*

*q4 –до счёт числа, и если нужно добавление еще одного разряда.*

1. Дана строка, состоящая только из символов «а» и «б». Составьте программу для машины Тьюринга, которая переставляет последний символ в начало строки. Каретка находится над первым символом строки.



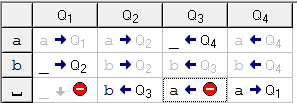
*Описание состояний*:

*q1 –конец слова.*

*q2 –замена и ветвление микрокода в следующие циклы.*

*q3 –цикл переставляющий b.*

*q4 –цикл переставляющий a.*

1. **\*Дана строка, состоящая только из символов «а» и «б». Составьте программу для машины Тьюринга, которая сортирует символы, то есть переставляет все буквы «а» в начало строки. Каретка находится над первым символом строки. Используйте состояния, которые перечислены род таблицей.

*Описание состояний*:

*q1 –* каретка идёт вправо, если встретила b, то ставит пробел и переходит к q2.

*q2 –* каретка идёт вправо и вставляет b, вместо пробела.

*q3 –* каретка идет влево, и ищет a, заменяет ее пробелом.

*q4 –* вставляет a в следующий свободный пробел.

1. \*Составьте программу для машины Тьюринга, которая складывает два числа в двоичной системе, разделенные на ленте знаком «+».
2. \*Составьте программы для машины Тьюринга, которые увеличивают и уменьшают на единицу число, записанное в десятичной системе счисления.
3. Составьте программы для машины Тьюринга, которые выполняют сложение и вычитание двух чисел в десятичной системе счисления.

## Машина Поста

1. Что делает следующая программа для машины Поста?

|  |  |
| --- | --- |
| 1 | ← |
| 2 | **? 3,4** |
| 3 | **1 1** |
| 4 | **стоп** |

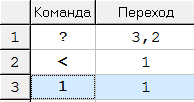
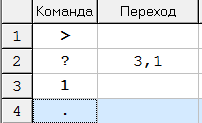
*Ответ*:

*Идет влево, и заполняет точками, пока не встретит точку.*

Как она будет работать при различных начальных состояниях ленты?

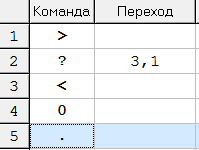
*Ответ*:

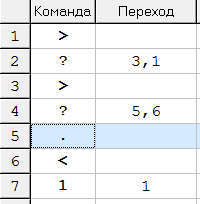
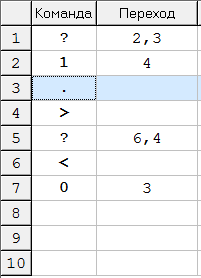
*Если точка слева, то как задумано, если справа, то не выйдет из цикла никогда, если стоит на точке, то также зациклится.*

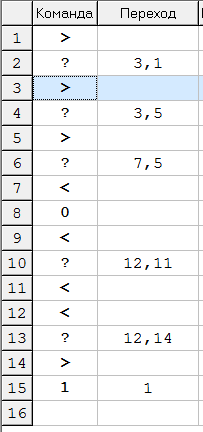
1. Напишите программу для машины Поста, которая неприменима (то есть зацикливается) при любом начальном состоянии ленты.
2. Напишите программу для машины Поста, которая увеличивает на 1 число, записанной в унарной системе счисления. Каретка стоит над первой (самой левой) отметкой.
3. Решите задачу 2 при условии, что в начале работы каретка расположена где-то справа от записи числа.

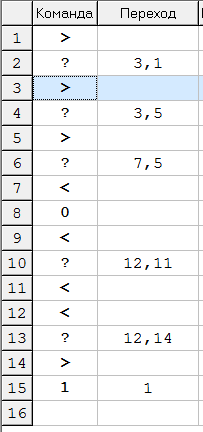


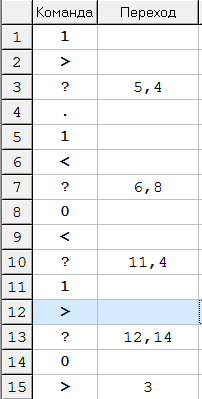
1. Напишите программу для машины Поста, которая уменьшает на 1 число, записанной в унарной системе счисления. Каретка стоит над первой (самой левой) отметкой.



1. На ленте расставлены метки, между которыми могут быть пропуски длиной в одну ячейку. Заполнить все пропуски метками. Каретка стоит над самой левой меткой.
2. \*Напишите программу для машины Поста, которая удваивает число, записанное в единичной системе счисления. Каретка расположена над первой отметкой числа.
3. \*Напишите программу для машины Поста, которая складывает два числа в единичной системе счисления. Каретка расположена над пробелом, разделяющим эти числа на ленте.
4. \*Напишите программу для машины Поста, которая складывает два числа, записанных в унарной системе. Числа расположены на неизвестном расстоянии друг от друга. Каретка находится над левой границей первого (левого) числа.



1. \*\*Напишите программу для машины Поста, которая складывает несколько натуральных чисел. Каждое число кодируется как последовательность расположенных рядом отметок (в унарной системе счисления). Числа отделены друг от друга пробелами. Каретка находится справа от первого числа.
2. \*\*Написать программу для машины Поста, которая находит единственную метку на ленте, которая расположена неизвестно где. Каретка должна остановиться на метке, все другие (временные) метки должны быть стерты. В начале работы каретка расположена над пустой ячейкой.



## Нормальные алгорифмы Маркова (НАМ)

1. Что делает следующий НАМ, если применить его к символьной цепочке, состоящей из нулей и единиц:

|  |
| --- |
| **\*0** → **0\***  **\*1** → **1\***  **\*** → **=.**  → **\*** |

*Ответ*:

*Добавляет и двигает звездочку в конец строки, в конце заменяя ее на знак “=”*

Как будет работать этот алгоритм при различных начальных состояниях ленты?

*Ответ*:

*Если лента состоит из 0 и 1, то работает как задумано, если из других символов или вообще пустая лента, то добавит звёздочку и заменит её на знак равно.*

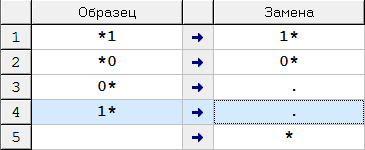
1. Напишите НАМ, который «сортирует» цифры двоичного числа так, чтобы сначала стояли все нули, а потом – все единицы.

*Ответ*:

*10 → 01*

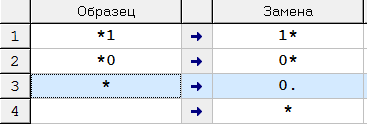
1. Напишите НАМ, который удаляет последний символ строки, состоящей из цифр 0 и 1. Какую операцию он выполняет, если рассматривать строку как двоичную запись числа?

*Ответ*:

**

1. Напишите НАМ, который умножает двоичное число на 2, добавляя 0 в конец записи числа.

*Ответ*:

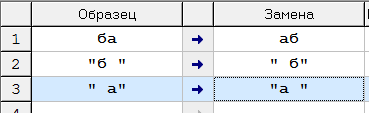
**

1. Напишите НАМ, который переводит число из двоичной системы счисления в единичную (унарную).

*Ответ*:

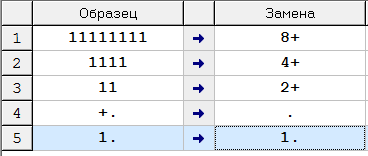
1. Дано слово, состоящее из букв «а», «б» и пробелов. Постройте нормальный алгоритм Маркова, который символы «а» переносит влево, символы «б» – вправо, а пробелы оставляет посередине.

*Ответ*:

**

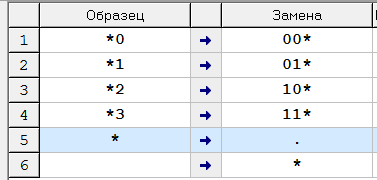
1. Дано число в унарной системе счисления (от 1 до 15), сразу после числа стоит точка. Напишите НАМ, который представляет это число в виде суммы степеней двойки (например, для числа 15 нужно получить «8+4+2+1»), и удаляет точку в конце.

*Ответ*:

**

1. \*Напишите НАМ, который переводит число из четверичной системы счисления в двоичную. Используйте специальный знак (например, \*), который отделяет обработанную часть числа от необработанной.

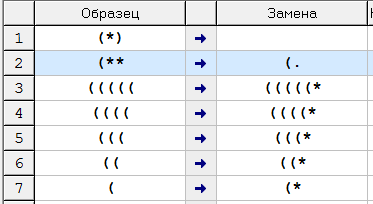
*Ответ*:

**

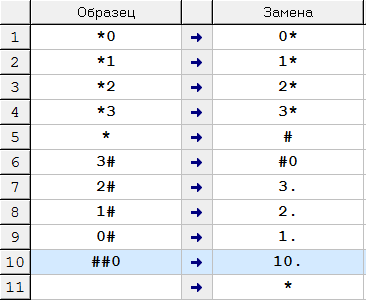
1. \*\*Дана последовательность скобок. Напишите НАМ, который проверяет правильность скобочной структуры (парность и вложенность скобок). Например, выражение ()()(())(()())(())() – правильное, а выражения

()()(())(()())(()))( и ()()(())(()())(())) – неправильные.

Если все правильно, лента должна быть пустой, если выражение неверное, на ленте должны остаться «неправильные» скобки.



1. \*\*Напишите НАМ, который увеличивает на единицу число, записанное в троичной системе счисления. Используйте специальные символы (например, # и \*) для обозначения начала числа и того разряда, который нужно увеличивать первым.

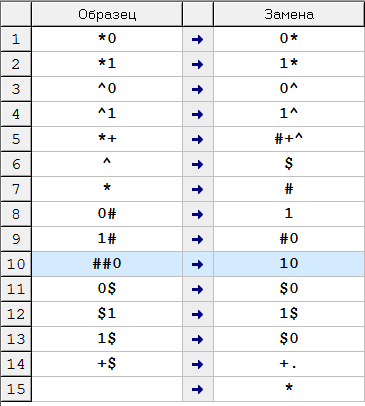


1. \*\*Напишите НАМ, который уменьшает на единицу число, записанное в троичной системе счисления. Используйте специальные символы (например, # и \*) для обозначения начала числа и того разряда, который нужно уменьшать первым.



1. \*\*Напишите НАМ, который складывает два числа в двоичной системе счисления. Числа разделены знаком «+», например, из строки «1011+1101» нужно получить «11000». Используйте алгоритм, в котором последовательно первое число увеличивается на 1, а второе – уменьшается на 1, пока второе не станет равно 0.

(не рабочий, частично, проблема с добавлением разряда, когда происходит сумма)



## Вычислимые функции

1. Целое число *n* записано на ленте в унарной системе счисления (как последовательность из *n* меток). Напишите программу для машины Поста, которая вычисляет функцию



если – чётное

если – нечётное

и таким образом доказывает разрешимость этой задачи. В начальный момент каретка стоит над первой меткой числа.

*Ответ*:

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |

1. Используя любого универсального исполнителя (машину Тьюринга, машину Поста или нормальный алгорифм Маркова), докажите вычислимость функции



если – делится на 3

если – не делится на 3

Целое число *n* записано в унарной системе счисления.

*Ответ:*

## Инвариант цикла

1. Определите инвариант цикла для следующего алгоритма двоичного поиска (предполагается, что элементы массива A отсортированы по неубыванию):

**L:= 1; R:= n + 1**

**нц пока L < R - 1**

**c:= div(L + R, 2)**

**если X < A[c] то**

**R:= c**

**иначе**

**L:= c**

**все**

**кц**

*Ответ*:

Используя найденный инвариант, определите, какой именно элемент массива будет найден, если в массиве есть несколько элементов, равных X.

*Ответ*:

Как нужно изменить инвариант (и цикл!), чтобы найти *первый* элемент, равный X?

*Ответ*:

1. Определите инвариант для следующего цикла.

**k := 0; b := 1**

**нц пока k < n**

**k := k + 1**

**b := b \* a**

**кц**

*Ответ*:

Что будет вычислено в переменной b?

*Ответ*:

1. Определите инвариант для следующего цикла.

**k := n; b := 1**

**нц пока k > 0**

**k := k - 1**

**b := b \* a**

**кц**

*Ответ*:

Что будет вычислено в переменной b?

*Ответ*:

1. Запишите предусловие и постусловие для алгоритма вычисления сумму всех делителей числа.

*Ответ*:

1. Запишите предусловие и постусловие для алгоритма проверки числа на простоту.

*Ответ*:

1. Запишите предусловие и постусловие для алгоритма определения количества слов в символьной строке.

*Ответ*:

1. Запишите предусловие и постусловие для алгоритма двоичного поиска в отсортированном массиве.

*Ответ*:

1. Запишите предусловие и постусловие для алгоритма перестановки элементов массива в обратном порядке.

*Ответ*:

1. Запишите предусловие и постусловие для алгоритма преобразования числа из символьной записи в значение целого типа.

*Ответ*:

1. Предложите другие начальные значения переменных ,  и  в алгоритме быстрого возведения в степень (см. Пример 4 в §37 учебника). Инвариант цикла должен сохраниться.

*Ответ*:

1. Оцените сложность алгоритма быстрого возведения в степень при , где *m* – натуральное число.

*Ответ*:

## Практическая работа № 40а.

## Сложность вычислений

1. Запишите алгоритм, который находит все делители натурального числа.

*Ответ*:

Какие элементарные операции в нём можно выделить?

*Ответ*:

Оцените количество этих операций.

*Ответ*:

Определите асимптотическую сложность алгоритма.

*Ответ*:

1. Запишите алгоритм, который определяет, является ли заданное натуральное число простым.

*Ответ*:

Какие элементарные операции в нём можно выделить?

*Ответ*:

Оцените количество этих операций.

*Ответ*:

Определите асимптотическую сложность алгоритма.

*Ответ*:

1. Запишите алгоритм, который находит количество положительных элементов массива.

*Ответ*:

Какие элементарные операции в нём можно выделить?

*Ответ*:

Оцените количество этих операций.

*Ответ*:

Определите асимптотическую сложность алгоритма.

*Ответ*:

1. Запишите алгоритм, который находит три минимальных элемента массива.

*Ответ*:

Какие элементарные операции в нём можно выделить?

*Ответ*:

Оцените количество этих операций.

*Ответ*:

Определите асимптотическую сложность алгоритма.

*Ответ*:

1. Запишите алгоритм, который находит количество элементов массива, равных минимальному элементу.

*Ответ*:

Какие элементарные операции в нём можно выделить?

*Ответ*:

Оцените количество этих операций.

*Ответ*:

Определите асимптотическую сложность алгоритма.

*Ответ*:

1. Запишите алгоритм, который находит (и выводит на экран) символы, которые встречаются в символьной строке более одного раза.

*Ответ*:

Какие элементарные операции в нём можно выделить?

*Ответ*:

Оцените количество этих операций.

*Ответ*:

Определите асимптотическую сложность алгоритма.

*Ответ*:

1. Алфавит языка племени «тумба-юмба» содержит *k* символов. Запишите алгоритм построения всех возможных слов этого языка, имеющих длину *n* символов.

*Ответ*:

Какие элементарные операции в нём можно выделить?

*Ответ*:

Оцените количество этих операций.

*Ответ*:

Определите асимптотическую сложность алгоритма.

*Ответ*: