Задание

на лабораторную работу № 4

по дисциплине «Теория алгоритмов и вычислительных процессов»

**Проектирование алгоритмов Маркова,**

**реализующих арифметические вычисления.**

**Время:** 2 часа (90 минут).

**Учебные цели:**

1. Выработать практические умения и навыки в построении НАМ, в том числе с помощью симуляторов.
2. Формировать способности: применять компьютерные /суперкомпьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности (ОПК-2); применять в профессиональной деятельности современные языки программирования и методы параллельной обработки данных, операционные системы, электронные библиотеки и пакеты программ, сетевые технологии (ПК-5)

Программный симулятор доступен по ссылке:

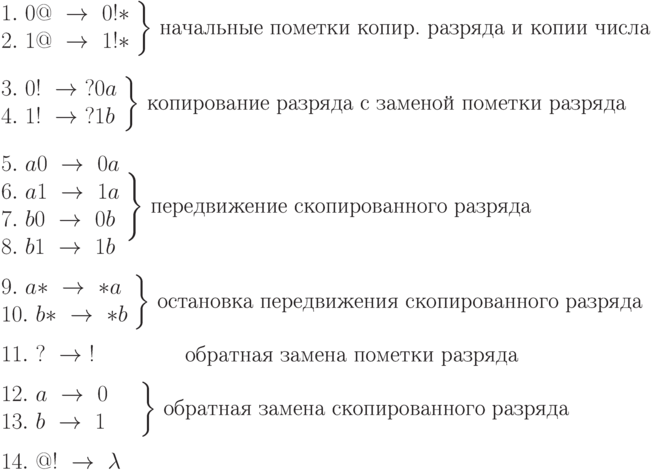
<https://kpolyakov.spb.ru/prog/nma.htm>

Пароль к архиву – kpolyakov.spb.ru

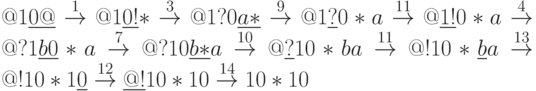
**Возможности нормальных алгоритмов и тезис Маркова**

Рассмотрим возможности реализации *арифметических операций* с помощью *нормальных алгоритмов Маркова*. Сначала обратим внимание на одно обстоятельство, связанное с работой любого *НАМ*: нужно вводить дополнительное *правило остановки* работы *нормального алгоритма* (иначе в примере увеличения числа на 1 *алгоритм* продолжит работу и снова будет увеличивать полученный результат еще на 1 и т.д. неограниченное число раз).

*Пример 1*. Рассмотрим, как довольно типичный пример, используемый часто в других алгоритмах, *алгоритм* копирования двоичного числа. В этом случае прежде всего исходное и скопированное числа разделим символом "\*". В разрабатываемом алгоритме мы будем копировать разряды числа по очереди, начиная с младшего, но нужно решить 2 проблемы: как запоминать *значение* символа, который мы копируем, и как запоминать *место* копируемого символа. Для решения второй проблемы используем символ "!", которым мы будем определять еще не скопированный разряд числа, после которого и стоит этот символ. Для запоминания значения копируемого разряда мы будем образовывать для значения 0 символ "a", а для значения 1 - символ "b". Меняя путем *подстановок* эти символы "a" или "b" с последующими, мы будем передвигать разряды "a" или "b" в начало копируемого числа (после "\*"), но для того, чтобы пока не происходило *копирование* следующего разряда справа, мы перед передвижением разряда временно символ "!" заменим на символ "?", а после передвижения сделаем обратную замену. После того как все число окажется скопированным в виде символов "a" и "b", мы заменим эти символы на 0 и 1 соответственно. В результате *нормальный алгоритм* копирования двоичного числа можно определить следующей последовательностью *подстановок*:



Продемонстрируем работу алгоритма для числа 10:



Для построения алгоритма сложения двух чисел можно использовать идею уменьшения одного числа на 1 с последующим увеличением другого числа на 1 и повторением этого до тех пор, пока уменьшаемое число не исчезнет после того, как станет равным 0. Но можно использовать и более сложную идею поразрядного сложения с переносом 1 в разряд слева.

Приведенные примеры показывают также возможности аппарата *нормальных алгоритмов Маркова* по организации *ветвления* и цикличных процессов вычисления. Это показывает, что **всякий алгоритм может быть нормализован,** т. е. задан нормальным алгоритмом Маркова. В этом и состоит *тезис Маркова*, который следует понимать как *определение алгоритма*.

Вместе с тем построение алгоритма в последнем приведенном примере подсказывает следующую *методику разработки НАМ*:

1. Произвести *декомпозицию* (разбиение на части) строящегося алгоритма. В примере это следующие части:
   1. разделение исходного числа и копии;
   2. копирование разряда;
   3. повторение предыдущей части до полного копирования всех разрядов.
2. Решение проблем реализации каждой части. В примере это следующие проблемы:
   1. запоминание копируемого разряда - разряд 1 запоминается как символ "a", а разряд 0 - как символ "b";
   2. запоминание места копируемого разряда - пометка еще не скопированного символа дополнительным символом "!" с заменой его на символ "?" при передвижении копируемого разряда и обратной заменой после передвижения.
3. Если часть для реализации является сложной, то она также подвергается декомпозиции.
4. Сборка реализации в единый алгоритм.

***Вариант №1***

**Задача №1.**

Определите *нормальный алгоритм*, который уменьшает число на единицу.

**Задача №2.**

Определите *нормальный алгоритм* сложения двух двоичных чисел методом уменьшения одного числа на 1 и увеличением другого числа на 1 до тех пор, пока уменьшаемое число не станет равным 0.

**Задача №3.**

Определите *нормальный алгоритм* логического сложения двух двоичных чисел.

**Задача №4.**

Определите *нормальный алгоритм* логического умножения двух двоичных чисел.

**Задача №5.**

Определите *нормальный алгоритм* сложения по модулю 2 двух двоичных чисел.

**Задача №6.**

Определите *нормальный алгоритм* вычисления наименьшего общего кратного двух двоичных чисел

***Вариант №2***

**Задача №1.**

Определите *нормальный алгоритм*, который увеличивает число на единицу.

**Задача №2.**

Определите *нормальный алгоритм* поразрядного сложения двух двоичных чисел.

**Задача №3.**

Определите *нормальный алгоритм* *вычитания* двоичных чисел.

**Задача №4.**

Определите *нормальный алгоритм* умножения двух двоичных чисел столбиком. **Задача №5.**

Определите *нормальный алгоритм* деления двух двоичных чисел с определением частного и остатка.

**Задача №6.**

Определите *нормальный алгоритм* вычисления наибольшего общего делителя двух двоичных чисел.