МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
по проведению лабораторной работы № 5
по дисциплине «Теория вычислительных процессов и структур»

Проектирование алгоритмов Маркова, реализующих арифметические вычисления.

Время: 2 часа (90 минут).

Учебные цели:

1. Выработать у студентов практические умения и навыки в построении НАМ, в том числе с помощью симуляторов.
2. Формировать у студентов способность применять современный математический аппарат, связанный с проектированием, разработкой, реализацией и оценкой качества программных продуктов и программных комплексов в различных областях человеческой деятельности (ОПК-2).

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 либо 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), либо перед началом работы *нормального алгоритма* добавлять к *входной строке специальные символы*, отличные от других символов строки, которые учитываются *подстановками* алгоритма в начале его работы и которые удаляются в конце работы алгоритма. Мы будем придерживаться второго способа, как и одна из наиболее успешных реализаций *нормальных алгоритмов Маркова* в виде языка программирования *Рефал*. В качестве добавляемого символа возьмем символ "@".

Пример 1. Рассмотрим простейшую операцию увеличения десятичного числа на 1. В этом случае почти всегда необходимо увеличить последнюю цифру на 1, а последняя цифра отличается тем, что после нее идет символ "@". Поэтому первыми *подстановками* должны быть *подстановки* типа

$$\langle\text{цифра}\rangle\ @ \ \rightarrow\ \langle\text{цифра} + 1\rangle.$$

Но если это цифра 9, то ее нужно заменить 0 и увеличение на 1 перенести в предыдущий разряд. Этому отвечает *подстановка*

$$9@\ \rightarrow\ @0.$$

Наконец, если число начинается с 9 и перед этой цифрой нужно поставить 1, то этому будет отвечать *подстановка*

$$@@ \;\to\; 1,$$

а если это не так, то в конце работы алгоритма символы @ надо стереть, что выполнит *подстановка*

$$@ \;\to\; \lambda.$$

Таким образом, мы получаем следующий *НАМ* увеличения десятичного числа на 1:

1. $0@ \to 1$	9. $8@ \to 9$
2. $1@ \to 2$	10. $9@ \to @0$
3. $2@ \to 3$	11. $@@ \to 1$
....	12. $@ \to \lambda$

Приведем работу построенного алгоритма для чисел 79 и 99:

$$@7\underline{9@} \xrightarrow{10} @\underline{7@}0 \xrightarrow{8} \underline{@}80 \xrightarrow{12} 80,$$
$$@9\underline{9@} \xrightarrow{10} @\underline{9@}0 \xrightarrow{10} \underline{@@}00 \xrightarrow{11} 100.$$

Аналогичным образом разрабатывается *нормальный алгоритм Маркова* для уменьшения числа на 1.

Пример 2. 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*:

$$\left.\begin{array}{l} 1.\ 0@ \ \rightarrow\ 0!* \\ 2.\ 1@ \ \rightarrow\ 1!* \end{array}\right\}\text{начальные пометки копир. разряда и копии числа}$$

$$\left.\begin{array}{l} 3.\ 0! \ \rightarrow\ ?0a \\ 4.\ 1! \ \rightarrow\ ?1b \end{array}\right\}\text{копирование разряда с заменой пометки разряда}$$

$$\left.\begin{array}{l} 5.\ a0 \ \rightarrow\ 0a \\ 6.\ a1 \ \rightarrow\ 1a \\ 7.\ b0 \ \rightarrow\ 0b \\ 8.\ b1 \ \rightarrow\ 1b \end{array}\right\}\text{передвижение скопированного разряда}$$

$$\left.\begin{array}{l} 9.\ a* \ \rightarrow\ *a \\ 10.\ b* \ \rightarrow\ *b \end{array}\right\}\text{остановка передвижения скопированного разряда}$$

$$11.\ ? \ \rightarrow\ ! \qquad \text{обратная замена пометки разряда}$$

$$\left.\begin{array}{l} 12.\ a \ \rightarrow\ 0 \\ 13.\ b \ \rightarrow\ 1 \end{array}\right\}\text{обратная замена скопированного разряда}$$

$$14.\ @! \ \rightarrow\ \lambda$$

Продемонстрируем работу алгоритма для числа 10:

$$@1\underline{0@} \xrightarrow{1} @1\underline{0!}* \xrightarrow{3} @1?0\underline{a*} \xrightarrow{9} @1\underline{?}0*a \xrightarrow{11} @\underline{1!}0*a \xrightarrow{4}$$
$$@?1\underline{b0}*a \xrightarrow{7} @?10\underline{b*}a \xrightarrow{10} @\underline{?}10*ba \xrightarrow{11} @!10*\underline{b}a \xrightarrow{13}$$
$$@!10*1\underline{0} \xrightarrow{12} \underline{@!}10*10 \xrightarrow{14} 10*10$$

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.

<u>Вариант №1</u>

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

<u>Вариант №2</u>

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.