Отчет по лабораторной работе №1 по курсу 1
Студент группы М80-111БВ-24, № по списку 15
Контакты e-mail: specapa@yandex.ru
Работа выполнена: «02» октября 2024 г.
Преподаватель: каф. 806 Бучкин Т. А.
Входной контроль знаний с оценкой _ _ _
Отчет сдан «04» октября 2024 г., итоговая оценка _ _ _
Подпись преподавателя _ _ _

1. Тема: "Отчет по заданию курсового проекта №1"

2. Цель работы: составить алгоритм двоичного арифметического сдвига влево второго числа на число разрядов, равное первому числу, для машины Тьюринга.

3. Задание: составить нормированный алгоритм двоичного арифметического сдвига влево второго числа на число разрядов, равное первому числу, для машины Тьюринга.

4. Оборудование ПЭВМ студента, если использовалось: 1,3 GHz 12-ядерный процессор Intel Core Ultra 5. Монитор: Универсальный монитор PnP.

5. Программное обеспечение ЭВМ студента, если использовалось: Операционная система семейства: Windows, наименование: Windows 11.
   Система программирования: нет.
   Редактор текстов: Notepad++.

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями):
   Идея:
   Копировать первое и второе число, вести подсчёт разрядов числа. При арифметическом сдвиге влево к сдвигаемому числу прибавляются нули. Если мы используем двоичное число, как показатель количества разрядов, на которые необходимо произвести сдвиг, то для единицы первого числа в позиции n, кол-во нулей, прибавляемых к второму числу составит: (n-1) ** 2.

Алгоритм:

a) При помощи машины копирования дублируем первое и второе число в соответствующем порядке справа от обоих оригиналов;
b) Второе число перемещаем вправо на одну ячейку;
c) Проводим арифметический сдвиг влево:
   i) Анализ числа 1:
      (1) Если в числе 1 обнаружен 0, то записываем его в счетчик разрядов;
      (2) Если в числе 1 обнаружена 1, то записываем 1 в счетчик разрядов и переходим к пункту “ii”;
      (3) Если в числе 1 не обнаружено ничего, переходим к пункту “d”;
   ii) Подсчитываем количество нулей, которые нужно добавить к числу 2, пока в счетчике разрядов есть 1:
      (1) Если в счетчике разрядов найдена 1, заменяем на 0 и:
         (a) Если в записи нулей пусто, записываем 1;
         (b) Иначе увеличиваем количество нулей в 2 раза;
      (2) Если единиц нет, то присоединяем нули к числу результату и заменяем 0 в счетчике разрядов на 1. Переходим к пункту “i”;
d) Удаляем счетчик разрядов и производим нормирование.

Оценка сложности алгоритма:

Копирование и сдвиг элементов – O(n);

Арифметический сдвиг (с учетом удвоителя нулей, счетчика разрядов) – O(n**2);

Общая сложность – O(n**2). (*Приложении 1)

7. ~~Сценарий выполнения работы [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].~~
8. Окончательное решение и тесты: “Приложение 1”
9. ~~Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.~~
10. Замечания автора по существу работы: Нет.
11. Выводы: в результате работы я улучшил свои навыки: создание алгоритмов, работа с машиной Тьюринга, нормирование алгоритмов для машины Тьюринга.

Недочёты при выполнении задания могут быть устранены следующим образом: Недочётов нет.

Подпись студента: _____

Приложение 1.

```
// Вычисление двоичного арифметического сдвига второго числа влево на число разрядов, равное первому

0, ,<,1 // Начало программы

// Курсор влево

1,0,<,1  1,1,<,1  1, ,<,2
2,0,<,2  2,1,<,2  2, ,>,n3

// Копирование числа 1

n3,0, ,n0_s  n3,1, ,n1_s  n3, , ,move_1 // Копирование: определяем 0 или 1

n0_s, ,>,n0_1  // Копирование для 0
n0_1,0,>,n0_1  n0_1,1,>,n0_1  n0_1, ,>,n0_2 // Проходим первое число
n0_2, ,>,n0_2  n0_2,0,>,n0_3  n0_2,1,>,n0_3 // Проходим пробелы
n0_3,0,>,n0_3  n0_3,1,>,n0_3  n0_3, ,>,n0_4 // Проходим второе число
n0_4,0,>,n0_4  n0_4,1,>,n0_4  n0_4, ,0,n0_5 // Записываем 0
n0_5, ,<,n0_5  n0_5,0,<,n0_6  n0_5,1,<,n0_6 // Движемся обратно
n0_6,0,<,n0_6  n0_6,1,<,n0_6  n0_6, ,<,n0_7 // Проходим копию числа 1
n0_7,0,<,n0_7  n0_7,1,<,n0_7  n0_7, ,<,n0_72 // Проходим число 2
          // Записываем 0 обратно в оригинал
n0_72, ,0,n0_73  n0_72,0,<,n0_72  n0_72,1,<,n0_72
n0_73,0,>,n3

n1_s, ,>,n1_1  // Копирование для 1
n1_1,0,>,n1_1  n1_1,1,>,n1_1  n1_1, ,>,n1_2 // Проходим первое число
n1_2, ,>,n1_2  n1_2,0,>,n1_3  n1_2,1,>,n1_3 // Проходим пробелы
n1_3,0,>,n1_3  n1_3,1,>,n1_3  n1_3, ,>,n1_4 // Проходим второе число
n1_4,0,>,n1_4  n1_4,1,>,n1_4  n1_4, ,1,n1_5 // Записываем 1
n1_5, ,<,n1_5  n1_5,0,<,n1_6  n1_5,1,<,n1_6 // Движемся обратно
n1_6,0,<,n1_6  n1_6,1,<,n1_6  n1_6, ,<,n1_7 // Проходим число 2
n1_7,0,<,n1_7  n1_7,1,<,n1_7  n1_7, ,<,n1_72
          // Записываем 1 обратно в оригинал
n1_72, ,1,n1_73  n1_72,0,<,n1_72  n1_72,1,<,n1_72
n1_73,1,>,n3

move_1, ,>,k3  // Переход к числу 2

// Копирование числа 2

k3,0, ,k0_s  k3,1, ,k1_s  k3, , ,move_2 // Копирование: определяем 0 или 1

k0_s, ,>,k0_1  // Копирование для 0
k0_1,0,>,k0_1  k0_1,1,>,k0_1  k0_1, ,>,k0_2 // Проходим первое число
k0_2, ,>,k0_2  k0_2,0,>,k0_3  k0_2,1,>,k0_3 // Проходим пробелы
k0_3,0,>,k0_3  k0_3,1,>,k0_3  k0_3, ,>,k0_31 // Проходим второе число
k0_31, ,>,k0_4
k0_4,0,>,k0_4  k0_4,1,>,k0_4  k0_4, ,0,k0_5 // Записываем 1
k0_5, ,<,k0_5  k0_5,0,<,k0_6  k0_5,1,<,k0_6 // Движемся обратно
k0_6,0,<,k0_6  k0_6,1,<,k0_6  k0_6, ,<,k0_71 // Проходим число 2
k0_71, ,<,k0_7
k0_7,0,<,k0_7  k0_7,1,<,k0_7  k0_7, ,<,k0_72
          // Записываем 0 обратно в оригинал
k0_72, ,0,k0_73  k0_72,0,<,k0_72  k0_72,1,<,k0_72
k0_73,0,>,k3

k1_s, ,>,k1_1  // Копирование для 1
k1_1,0,>,k1_1  k1_1,1,>,k1_1  k1_1, ,>,k1_2 // Проходим первое число
k1_2, ,>,k1_2  k1_2,0,>,k1_3  k1_2,1,>,k1_3 // Проходим пробелы
```

```
k1_3,0,>,k1_3   k1_3,1,>,k1_3   k1_3, ,>,k1_31 // Проходим второе число
k1_31, ,>,k1_4
k1_4,0,>,k1_4   k1_4,1,>,k1_4   k1_4, ,1,k1_5 // Записываем 1
k1_5, ,<,k1_5   k1_5,0,<,k1_6   k1_5,1,<,k1_6 // Движемся обратно
k1_6,0,<,k1_6   k1_6,1,<,k1_6   k1_6, ,<,k1_71 // Проходим число 2
k1_71, ,<,k1_7
k1_7,0,<,k1_7   k1_7,1,<,k1_7   k1_7, ,<,k1_72
            // Записываем 1 обратно в оригинал
k1_72, ,1,k1_73   k1_72,0,<,k1_72   k1_72,1,<,k1_72
k1_73,1,>,k3


move_2, ,>,move_3   // Переход к pre
move_3, ,>,move_4   move_3,0,>,move_3   move_3,1,>,move_3
move_4, ,<,pre      move_4,0,>,move_4   move_4,1,>,move_4

// Чтение числа 1

pre, , ,main                    // Переход в main


main, ,<,10   10,0, ,11   11, ,>,12   12, ,1,114б // Если 0 -> Добавляем пометку в счётчик разрядов
10, , ,exit      10,1, ,200  200, ,>,20             // Если 1 -> Считаем разряд

20, , ,21     21, ,1,22     22, ,>,22  // Счетчик разрядов пуст -> заполняем 1
          22,1,0,30     22,0,>,400 // Счетчик разрядов не пуст 1 / 0
400, , ,300   400,0,>,400   400,1,0,30 // Поиск хотя бы одной 1


// Для счетчика разрядов, в котором нашлась 1

30,0,>,30   30,1,>,30   30, ,>,31 // Движемся по счетчику разрядов
31,0,>,31   31,1,>,31   31, ,>,32 // Движемся по числу 2
32,0,0,34   32,1,1,34   32, , ,33   33, ,1,130 // Проверка пуста ли память -> записать 1
34,0,1,34   34,1,>,34   34, , ,35   35, ,<,36  // Идем по памяти - не пусто -> удвоить память

// Для счетчика разрядов, в котором не нашлось 1

300,0,>,300   300,1,>,300   300, ,>,301 // Движемся по счетчику разрядов
301,0,>,301   301,1,>,301   301, , ,40п // Движемся по числу 2

// Машина удвоения памяти

36,1,0,37   36,0,>,40   // Меняем 1 на 0 или начинаем закрытие множителя
37,0,>,37   37,1,>,37   37, ,1,38 // Прибавляем 1 на правом конце памяти

38,1,<,38   38,0,<,39 // Движемся в начало памяти
39,0,<,39   39,1,1,36 // Находим 1 -> возвращаемся к пункту 36
          39, , ,40п // Не находим 1 -> заменяем всю память на 0

40п, ,>,40   40,1,0,41   40,0,>,40   41,0,>,500   //   идем в конец памяти   // 40

// Важный этап: проверка наличия еще одного удвоителя разряда:

500, ,<,501   501,0,<,501   501,1,<,501   501, ,<,502   // переход к числу 2
500,0,>,500   502,0,<,502   502,1,<,502   502, ,<,503   // переход к разрядам
500,1,0,500   503,0,<,503   503,1,<,503   503, ,>,22    // начинаем проверку

40, ,<,42   42,0, ,43   43, ,<,44   44,0,<,44   44, ,0,50   // переносим ноль в начало памяти
50,0,<,50   50,1,<,50   50, ,<,51 // переходим к счетчику разрядов
51,0,1,52   51,1,1,52   51, , ,main // возобновление счетчика разрядов
52,1,<,51
```

```
// Возврат в счётчик разрядов (в ходе подсчета)

113,0,<,113   113,1,<,113   113, ,<,114 // Проходим число 2
114,0,<,114   114,1,<,114   114, , ,115 // Проходим счетчик разрядов
115, , ,22

// Возврат в счётчик разрядов (после добавления 1 от нулевого разряда)

114б,0,<,114б   114б,1,<,114б   114б, , ,main

130,1,<,130   130, ,<,113

// Выход из программы

exit, ,>,e0
e0, ,>,clear
clear,0, ,step   clear,1, ,step  step, ,>,clear
clear, ,>,fix



// Финальное нормирование

fix, ,>,fix   fix,0,>,check0   fix,1,>,check0
check0, ,<,final0   check0,0,<,cmove   check0,1,<,cmove
cmove,0, ,replace0   cmove,1, ,replace1

replace0, ,<,replace0
replace1, ,<,replace1
replace0,0,>,r0m1   replace0,1,>,r0m1
replace1,0,>,r1m1   replace1,1,>,r1m1
r0m1, ,>,r0m2   r0m2, ,0,r0m3   r0m3,0,>,r0m4
r1m1, ,>,r1m2   r1m2, ,1,r1m3   r1m3,1,>,r1m4

r0m4, ,>,r0m4   r0m4,0,>,check   r0m4,1,>,check
r1m4, ,>,r1m4   r1m4,0,>,check   r1m4,1,>,check

check, ,<,final
check,0,<,lr   check,1,<,lr

lr,0, ,lm0   lr,1, ,lm1
lm0, ,<,lm0   lm0,0,>,r0m2   lm0,1,>,r0m2
lm1, ,<,lm1   lm1,0,>,r1m2   lm1,1,>,r1m2

final0,0, ,fm00   final0,1, ,fm01
fm00, ,<,fm00   fm00,0,>,fr00m2   fm00,1,>,fr00m2
fm01, ,<,fm01   fm01,0,>,fr01m2   fm01,1,>,fr01m2
fr00m2, ,>,fr00m3   fr00m3, ,0,leave
fr01m2, ,>,fr01m3   fr01m3, ,1,leave

final,0, ,fm0   final,1, ,fm1
fm0, ,<,fm0   fm0,0,>,fr0m2   fm0,1,>,fr0m2
fm1, ,<,fm1   fm1,0,>,fr1m2   fm1,1,>,fr1m2

fr0m2, ,0,leave
fr1m2, ,1,leave

leave,0,>,leave   leave,1,>,leave   leave, ,#,leave
```

Тесты:

0 0 -> 0
1 1 -> 10
10 1 -> 100
100 1 -> 10000
101 1 -> 100000
110 1 -> 1000000
110 10 -> 10000000
1111 11 -> 11000000000000000

Экспериментальная проверка сложности алгоритма: