МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Отчёт по лабораторной работе №3
по дисциплине
«Информатика»
«Реализация базовых алгоритмов в системах счисления.»

Выполнил студент гр. ИВТб-1301-05-00 _____/Черкасов А. А./
Проверил доцент кафедры ЭВМ _____/Коржавина А.С./

Киров
2024

Цель работы

Закрепить на практике лекционный материал по теме «Системы счисления», реализовав несколько базовых алгоритмов работы в системах счисления с произвольными основаниями.

Задания

1. Определить количество нулей в двоичной записи числа.

   Формат ввода.

   Целое неотрицательное число в десятичной системе счисления.

   Формат вывода.

   Количество нулей в двоичной записи числа.

Ввод	Вывод
16	4
7	0

2. Определить, какая цифра, 0 или 1, стоит в разряде N в двоичной записи числа.

   Формат ввода.

   Через пробел: целое неотрицательное число в десятичной системе счисления и номер разряда в двоичной записи числа.

   Формат вывода.

   Двоичная цифра в разряде номер N.

Ввод	Вывод
9 1	0
11 0	1

3. Перевести вещественное число X из системы счисления с основанием K. Перевести число в систему счисления с основанием M.

   Формат ввода.

   В одну строку через пробел 3 числа: вещественное число X, Целое число K из диапазона 2 .. 10, целое число M из диапазона 2 .. 10.

   Формат вывода.

   Вещественное число в системе счисления с основанием M. Количество знаков дробной части определять исходя из количества знаков исходного числа.

Ввод	**Вывод**
9.5 10 2	1001.1
12.1 3 5	10.1

4. Вывести результат выполнения операции (a+b) в системе остаточных классов с N основаниями p1, p2, ..., pN. В случае, если результат выходит за границы диапазона представления чисел, вывести -1, иначе вывести результат в десятичной системе счисления.

   Формат ввода.

   В строке через пробел число модулей N, модули, цифры числа a, цифры числа b.

   Формат вывода.

   Результат в десятичной системе счисления либо «-1», если результат выходит за границы диапазона.

Ввод	**Вывод**
3 2 3 5 1 1 1 0 2 2	3
3 2 3 5 0 2 0 0 2 0	-1

Задание 1.

Схема алгоритма Задания 1 представлена на Рисунке 1.

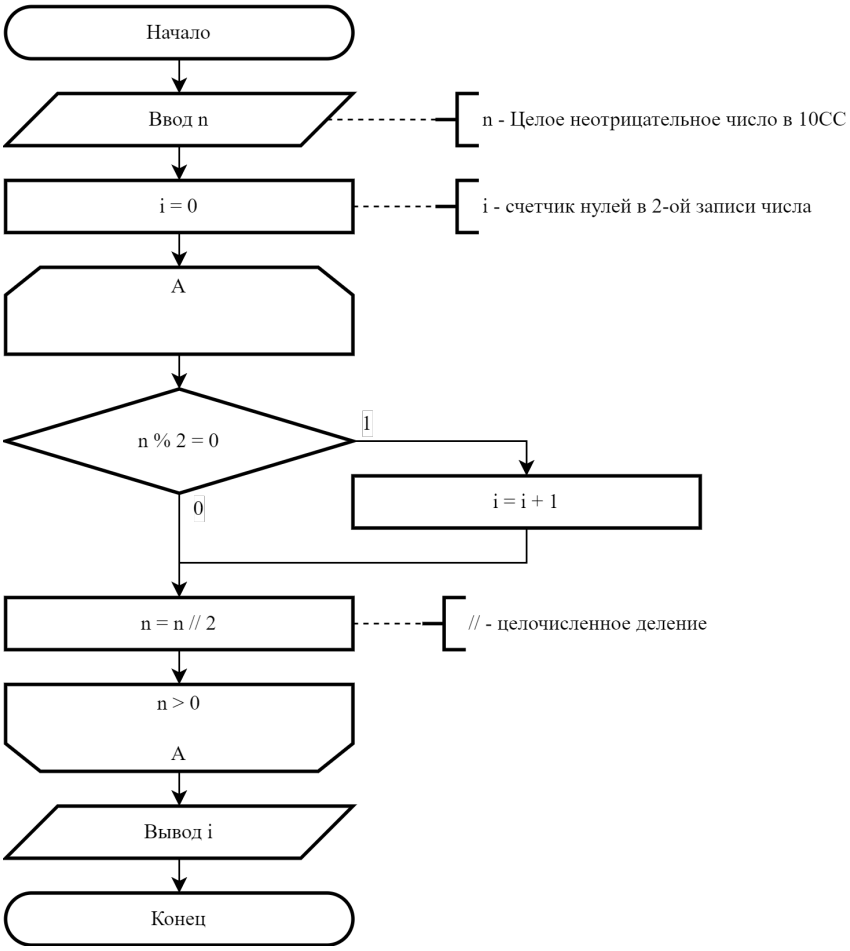

Рисунок 1 - Схема алгоритма Задания 1.

Задание 2.

Схема алгоритма Задания 2 представлена на Рисунке 2.

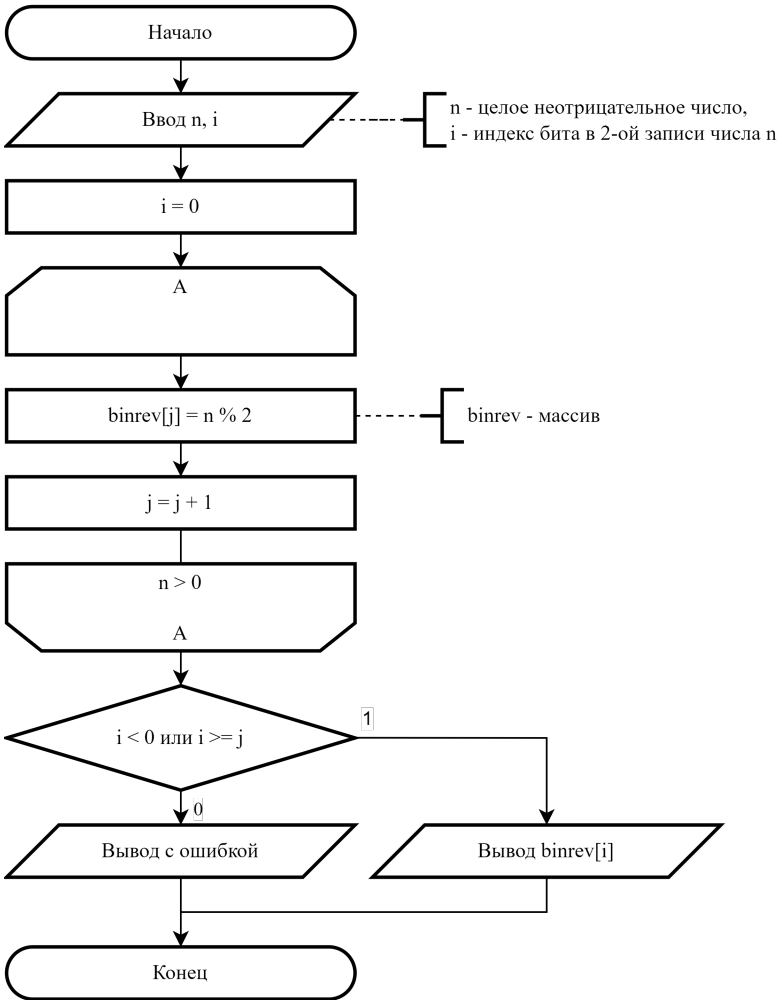

Рисунок 2 - Схема алгоритма Задания 2.

Задание 3.

Схема алгоритма Задания 3 представлены на Рисунках 3.1 и 3.2.

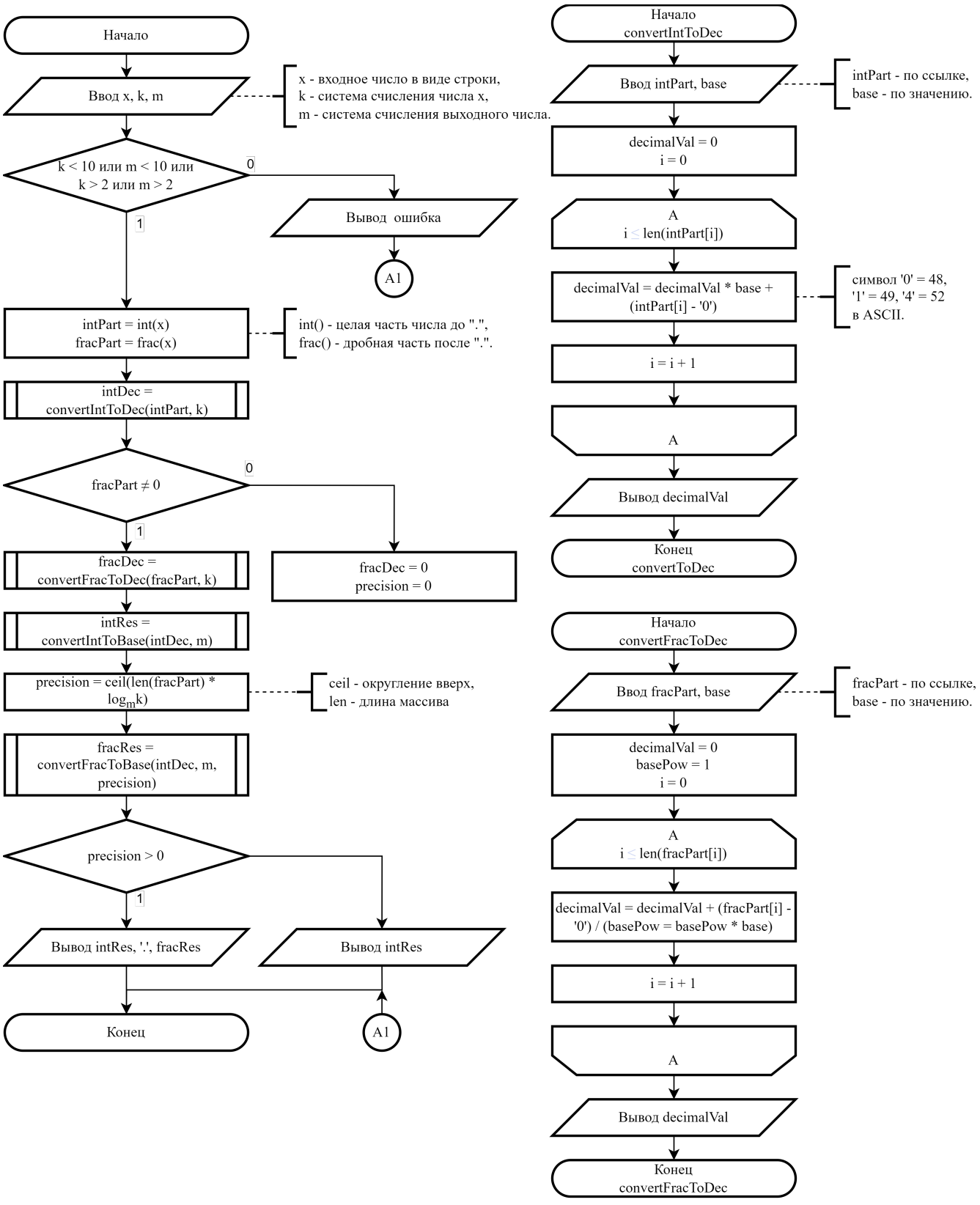

Рисунок 3.1 - Схема алгоритма Задания 3.

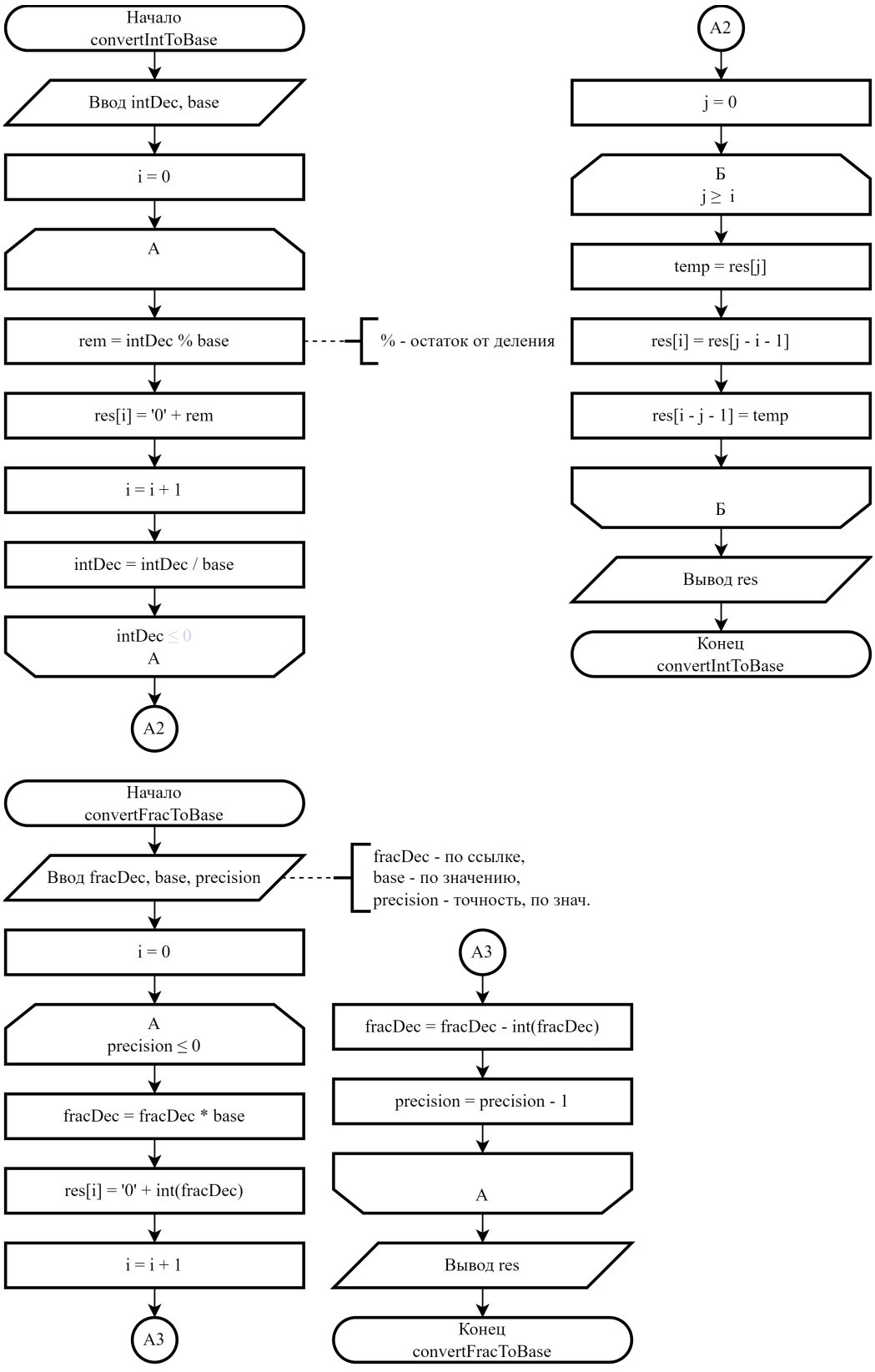

Рисунок 3.2 - Схема алгоритма Задания 3.

Задание 4.

Схема алгоритма Задания 4 представлены на Рисунках 4.1 и 4.2.

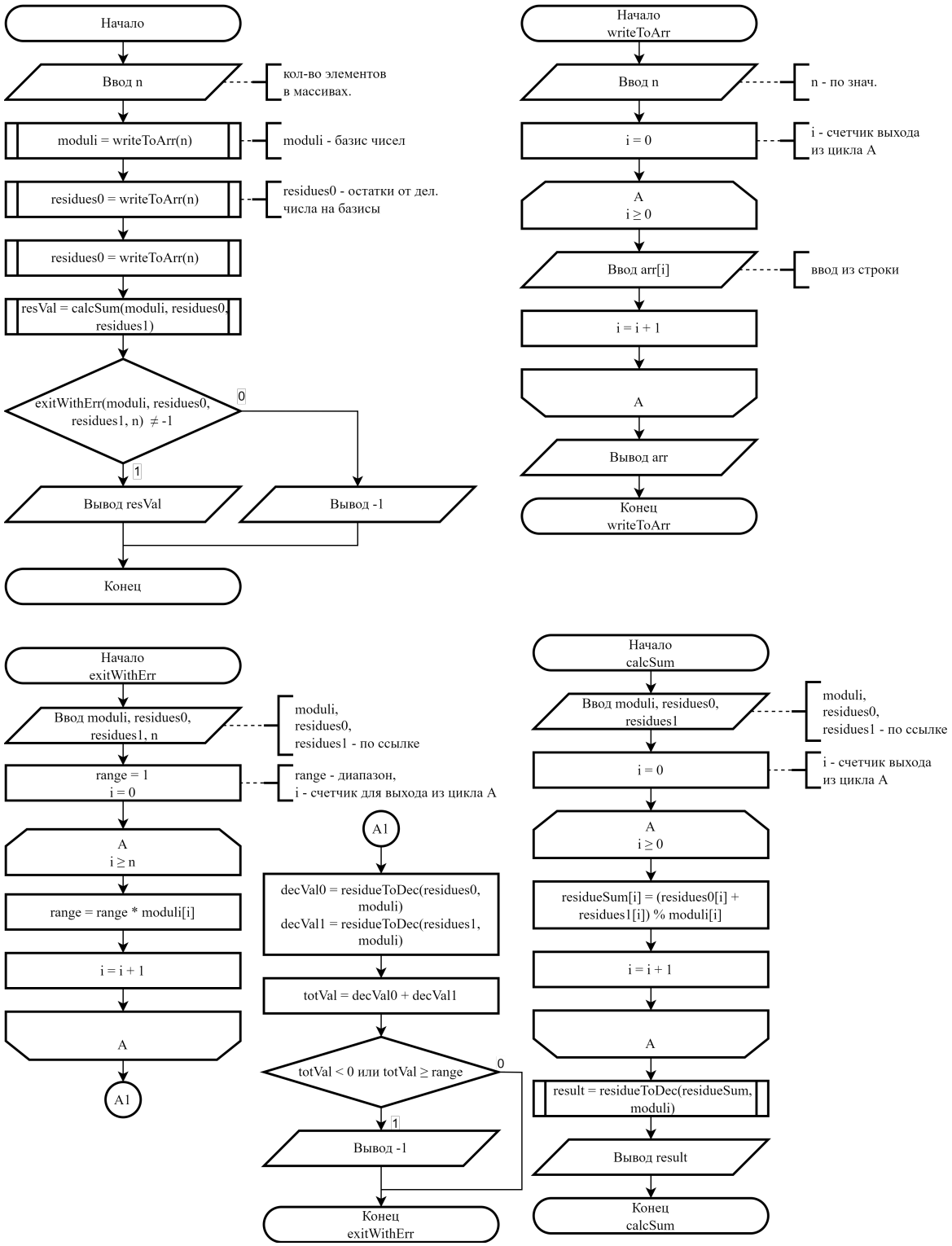

Рисунок 4.1 - Схема алгоритма Задания 4.

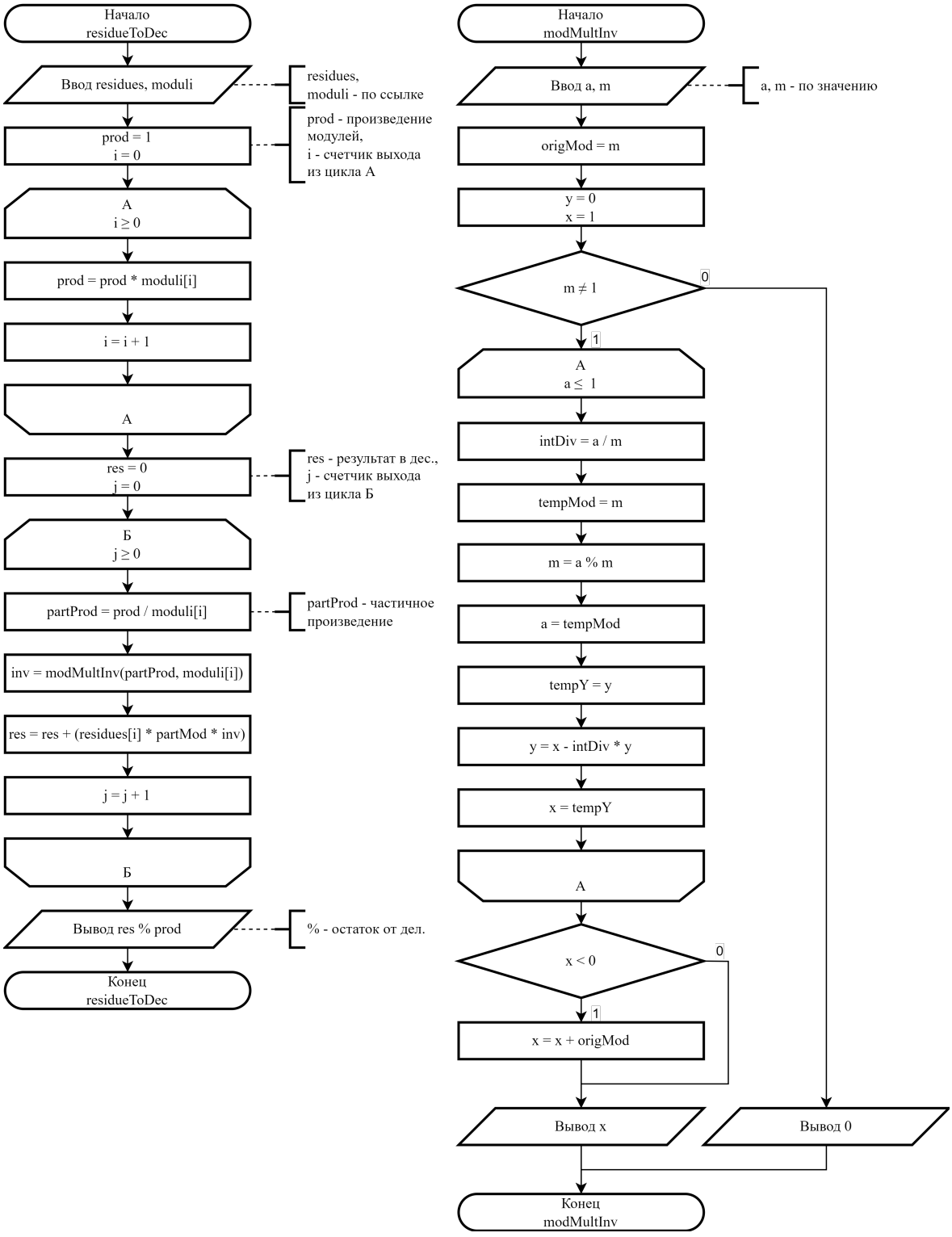

Рисунок 4.2 - Схема алгоритма Задания 4.

Приложение А1. Исходный код для Задания 1.

```
#include <stdio.h>

int cntBin(int n){
    int i = 0; //Счетчик нулей в 2-ой записи числа n
    do {
        if (n % 2 == 0){
            i++;
        }
        n /= 2;
    } while (n > 0);
    printf("%d", i); //Вывод i
}

int main(){
    int n;
    scanf("%d", &n); //Ввод числа n
    if(n < 0){
        return -1;
    } //Вывод ошибки
    cntBin(n);
    return 0;
}
```

Приложение А2. Исходный код для Задания 2.

```
#include <stdio.h>

// Функция для перевода числа в 2СС
int toBin(int n, int i) {
    int binrev[64];
    int j = 0;

    do {
        // Массив в котором представление записывается в обратном порядке
        binrev[j] = n % 2;
        n /= 2;
        j++;
    } while (n > 0);

    if (i < 0 || i >= j) {
        printf("Индекс %d выходит из диапазона представления числа в 2СС.\n", i);
        return -1;
    } // Вывод с ошибкой
    return binrev[i];
}

int main() {
    int n, i;
    printf("Введите число n и индекс i бита, который хотите узнать: ");
    scanf("%d %d", &n, &i);
    printf("Бит под индексом %d: %d\n", i, toBin(n, i));
    return 0;
}
```

Приложение А3. Исходный код для Задания 3.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

// Преобразование целой части из заданной системы счисления в десятичную
int convertIntToDec(const char *intPart, int base) {
  int decimalValue = 0; // Переменная для хранения десятичного значения целой части

  // Проходим по каждому символу целой части
  for (int i = 0; i < strlen(intPart); i++) {
    int digit = intPart[i] - '0';               // char to int
    decimalValue = decimalValue * base + digit; // Обновляем десятичное значение
  }
  return decimalValue; // Возвращаем десятичное значение целой части
}

// Преобразование дробной части из заданной системы счисления в десятичную
double convertFracToDec(const char *frac, int base) {
  double decValue = 0.0;  // Переменная для хранения десятичного значения дробной части
  double basePower = 1.0; // Переменная для хранения текущей степени основания

  // Проходим по каждому символу дробной части
  for (int i = 0; i < strlen(frac); i++) {
    int digit = frac[i] - '0';               // char to int
    decValue += digit / (basePower *= base); // Добавляем значение к десятичному
  }
  return decValue; // Десятичное значение дробной части
}

// Преобразования целого числа в заданную систему счисления
int convertIntToBase(int number, int base, char *result) {
```

```
  int index = 0; // Индекс для записи символов в результат

  // Преобразование числа в заданную систему счисления
  do {
    int remainder = number % base;     // Остаток от деления на основание
    result[index++] = '0' + remainder; // Записываем символ в результат
    number /= base;                    // Делим число на основание
  } while (number > 0); // Продолжаем, пока число больше нуля

  // Завершаем строку нулевым символом
  result[index] = '\0';

  // Обратим порядок символов в строке
  for (int i = 0; i < index / 2; i++) {
    char temp = result[i];             // Временная переменная для обмена
    result[i] = result[index - i - 1]; // Меняем местами символы
    result[index - i - 1] = temp;      // Меняем местами символы
  }
  return 0; // Успешное выполнение
}

double logBase(double base, double val) {
    return log(val) / log(base);
}

// Функция для преобразования дробной части в заданную систему счисления
int convertFracToBase(double frac, int base, char *result, int precision) {
  int index = 0; // Индекс для записи символов в результат

  // Преобразуем дробную часть с заданной точностью
  while (precision-- > 0) {
    frac *= base;                  // Умножаем дробную часть на основание
    int digit = (int)frac;         // Получаем целую часть
```

```
    result[index++] = '0' + digit; // Записываем символ в результат
    frac -= digit;                 // Убираем целую часть из дробной
  }
  result[index] = '\0'; // Завершаем строку нулевым символом
  return 0;             // Успешное выполнение
}

int main() {
  char x[50]; // Массив для хранения входной строки
  printf("Введите число x, основание его системы счисления k и основание системы\
счисления m в которую хотите перевести: ");
  fgets(x, sizeof(x), stdin); // Читаем входные данные

  char *token = strtok(x, " "); // Разделяем строку на 3 токена
  char *numberStr = token;      // Сохраняем строку числа
  token = strtok(NULL, " ");    // Основание k
  int k = atoi(token);          // str в int k
  token = strtok(NULL, " ");    // Основание m
  int m = atoi(token);          // str в int m

  if (k > 10 || m > 10 || k < 2 || m < 2){
    printf("Основания систем счисления k и m должны быть от 2 до 10\n");
    return -1;
  }

  // Разделяем число на целую и дробную части
  char *intPart = strtok(numberStr, "."); // Целая часть до .
  char *fracPart = strtok(NULL, ".");     // Дробная часть после .

  // Целая часть в десятичную
  int intDec = convertIntToDec(intPart, k);

  // Дробная часть в десятичную
```

```
  double fracDec = fracPart ? convertFracToDec(fracPart, k) : 0.0;

  // Целая часть в систему счисления M
  char intRes[50];
  convertIntToBase(intDec, m, intRes);

  // Дробная часть в систему счисления M
  char fracRes[50];
  // Количество знаков дробной части
  int precision = fracPart ? ceil(strlen(fracPart) * logBase(m, k)) : 0;
  convertFracToBase(fracDec, m, fracRes, precision);

  if (precision > 0) {
    printf("%s.%s\n", intRes, fracRes);
  }
  else {
    printf("%s\n", intRes);
  }
  return 0;
}
```

Приложение А4. Исходный код для Задания 4.

```
#include <stdio.h>
#include <stdlib.h>

int n;

// Функция для нахождения обратного по модулю числа a по модулю m
int modMultInverse(int a, int m) {
    int originalModulus = m; // Сохраняем оригинальное значение m
    int y = 0;               // Коэффициент для y (временная переменная)
    int x = 1;               // Коэффициент для x (временная переменная)

    // Если m равно 1, обратного элемента не существует, возвращаем 0
    if (m == 1) return 0;

    // Алгоритм Евклида для нахождения обратного элемента
    while (a > 1) {
        // div - целая часть от деления a на m
        int intDiv = a / m;
        int tempMod = m; // Временная переменная для хранения m

        // Обновляем m и a
        m = a % m;     // Остаток от деления a на m
        a = tempMod;   // Обновляем a на старое значение m
        int tempY = y; // Временная переменная для хранения y

        // Обновляем y и x
        y = x - intDiv * y; // Обновляем y
        x = tempY;          // Обновляем x
    }

    // Если x отрицательное, значение m для получения положительного результата
    if (x < 0) x += originalModulus;
```

```
    return x; // Возвращаем обратное по модулю значение
}

// Функция для преобразования числа из СОК в десятичную СС (ост, базисы)
int residueToDec(int *residues, int *moduli) {
    int prod = 1; // Произведение модулей
    int res = 0;  // Результат в десятичной системе

    // Вычисление произведения модулей
    for (int i = 0; i < n; i++) {
        prod *= moduli[i];
    }

    // Вычисление результата в десятичной системе
    for (int i = 0; i < n; i++) {
        int partProd = prod / moduli[i];               // Частичное произведение
        int inv = modMultInverse(partProd, moduli[i]); // Обратное по модулю число
        res += residues[i] * partProd * inv;
    }
    return res % prod; // Результат по модулю product
}

// Функция для сложения чисел в системе остаточных классов (базисы, ост1, ост2)
int calcSum(int *moduliArr, int *residues1Arr, int *residues2Arr) {
    int *residueSum = (int *)malloc(n * sizeof(int));

    for (int i = 0; i < n; i++) {
        residueSum[i] = (residues1Arr[i] + residues2Arr[i]) % moduliArr[i];
    }
    int result = residueToDec(residueSum, moduliArr);
    free(residueSum); // Освобождаем память
    return result;
```

```
}

// Функция для записи чисел в массивы (массив)
int* writeToArr() {
    int *arr = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    return arr;
}

int exitWithErr(int *moduli, int *residues0, int *residues1) {
    int range = 1; // находим диапазон +1
    for (int i = 0; i < n; i++) {
        range *= moduli[i];
    }

    // Преобразуем остатки в десятичное число
    int decimalValue0 = residueToDec(residues0, moduli);
    int decimalValue1 = residueToDec(residues1, moduli);

    int totValue = decimalValue0 + decimalValue1;
    if (totValue < 0 || totValue >= range) {
        return -1; // Если сумма выходит за пределы диапазона, возвращаем -1
    }
    return 0;
}

int main() {
    scanf("%d", &n);

    int *moduli = writeToArr();
    int *residues0 = writeToArr();
```

```
    int *residues1 = writeToArr();

    int resVal = calcSum(moduli, residues0, residues1);
    if (exitWithErr(moduli, residues0, residues1) == -1) {
        printf("-1\n");
    } else {
        printf("%d\n", resVal);
    }

    // Освобождаем выделенную память
    free(moduli);
    free(residues0);
    free(residues1);

    return 0;
}
```

Вывод

В ходе выполнения лабораторной работы по теме «Системы счисления» удалось закрепить лекционный материал, реализовав несколько базовых алгоритмов работы с системами счисления и их произвольными основаниями.