(ЗО) Самостоятельная работа № 2
Разветвляющиеся алгоритмы

Цель: приобретение навыков организации ветвлений в алгоритмах.

Задание. Разработать два алгоритма определения количества чисел, обладающих некоторым общим свойством, среди заданных целых чисел *a* и *b* или *a*, *b* и *c* (варианты представлены в таблице 1). Разработанные алгоритмы представить в виде схемы алгоритма и программы для ЭВМ на языке C#.

Выполнить тестирование программы с различными наборами исходных данных (проверить работоспособность всех ветвей алгоритма/программы и в качестве доказательства работоспособности привести экранные формы).

Задание выполнить **двумя способами**, которые описаны ниже. В выводе необходимо написать, какой способ оказался эффективнее. Если один способ оказывается эффективнее другого только при определенных условиях, то в выводе следует указать, при каких условиях так происходит.

Отчет следует начать с титульного листа.

Далее привести цель и общее задание на всю работу.

Затем привести индивидуальное задание (по вариантам) и решение индивидуального задания, состоящее из схемы алгоритма, листинга программного кода (текста программы) и экранных форм (т. е. скриншотов работы программы).

В конце отчета поместить вывод.

Таблица 1 – Варианты задания

Вариант	Определить количество	среди чисел
1	положительных чисел	*a*, *b*, *c*
2	отрицательных чисел	*a*, *b*
3	отрицательных чисел	*a*, *b*, *c*
4	неположительных чисел	*a*, *b*
5	неположительных чисел	*a*, *b*, *c*
6	неотрицательных чисел	*a*, *b*
7	неотрицательных чисел	*a*, *b*, *c*
8	равных нулю чисел	*a*, *b*
9	равных нулю чисел	*a*, *b*, *c*
10	не равных нулю чисел	*a*, *b*
11	не равных нулю чисел	*a*, *b*, *c*
12	чётных чисел	*a*, *b*
13	чётных чисел	*a*, *b*, *c*
14	нечётных чисел	*a*, *b*
15	нечётных чисел	*a*, *b*, *c*
16	чисел, кратных трём	*a*, *b*
17	чисел, кратных трём	*a*, *b*, *c*
18	чисел, не кратных трём	*a*, *b*
19	чисел, не кратных трём	*a*, *b*, *c*
20	чисел, больших числа *c*	*a*, *b*
21	чисел, больших числа *d*	*a*, *b*, *c*
22	чисел, меньших числа *c*	*a*, *b*
23	чисел, меньших числа *d*	*a*, *b*, *c*
24	чисел, не больших числа *c*	*a*, *b*
25	чисел, не больших числа *d*	*a*, *b*, *c*
26	чисел, не меньших числа *c*	*a*, *b*
27	чисел, не меньших числа *d*	*a*, *b*, *c*
28	чисел, равных числу *c*	*a*, *b*
29	чисел, равных числу *d*	*a*, *b*, *c*
30	чисел, не равных числу *c*	*a*, *b*

Теоретические сведения

Под ***ветвлением*** в алгоритме понимается организация выбора одного из двух альтернативных вариантов продолжения алгоритма в соответствии со значением некоторого логического выражения. Каждый из упомянутых альтернативных вариантов называется *ветвью* в алгоритме.

Организовать ветвления в алгоритме решения поставленной задачи можно по-разному.

Первый способ заключается в следующем:

– определить количество возможных вариантов значений искомой величины – n;

– составить для каждого из этих n вариантов соответствующее логическое выражение;

– организовать в алгоритме n ветвей с помощью $(n – 1)$ ветвления (т. е. с помощью $(n – 1)$ условных блоков), используя при этом $(n – 1)$ логическое выражение;

– внутри каждой ветви переменной, предназначенной для хранения искомой величины, присвоить соответствующее значение (это значение и будет результатом).

Второй способ требует:

– задания нулевого начального значения переменной, предназначенной для хранения искомой величины;

– организации $(n – 1)$ ветвлений алгоритма таким образом, чтобы эта переменная претерпевала поэтапные изменения до своего конечного значения. Получившееся значение и будет результатом.

Условный блок (блок «Решение») в схеме алгоритма

В схемах алгоритмов ветвления реализуются с помощью блока под названием «Решение» (который также называют условным блоком). Внутри данного блока записывается логическое выражение, а выходящие из блока ветви помечаются словами «Да» и «Нет», которые соответствуют истинному и ложному значениям логического выражения.

В программах на языке программирования C# ветвления организуются с помощью условных операторов **if** или **if-else**. Синтаксис оператора **if-else** имеет следующий вид:

if (<выражение>) <Оператор 1, если Истина>**;**
else <Оператор 2, если Ложь>**;**

На месте оператора 1 и (или) оператора 2 может располагаться блок операторов (или составной оператор), т. е. несколько операторов, записанных внутри фигурных скобок – **{**<Оператор 1>; <Оператор 2>; …**}**. Они используются тогда, когда по алгоритму требуется несколько операторов, а по синтаксису в данном месте должен быть один оператор.

Операторам **if-else** (полная форма) и **if** (сокращенная форма) соответствуют следующие конструкции, представленные на рисунках 1 и 2 соответственно.

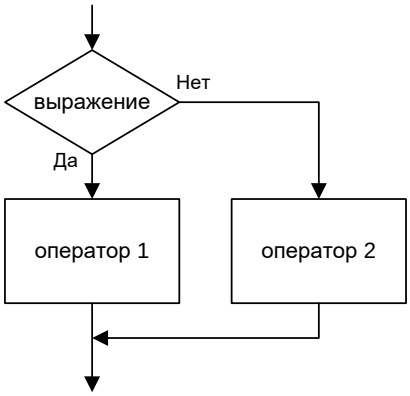

Рисунок 1 – Ветвление в схеме алгоритма (полная форма)

Рисунок 2 – Ветвление в схеме алгоритма (сокращенная форма)

Для полного ветвления: если **выражение** в заголовке условного оператора вырабатывает истинное значение (Да, true), то выполняется **оператор 1**, в противном случае – **оператор 2**. В сокращенной форме ветвь, соответствующая ложному значению (Нет, false) не заполняется другими блоками.

Операции отношений и логические операции

Для формирования логических выражений используются операции отношений и логические операции, представленные в таблице 2.

Таблица 2 – Операции отношений и логические операции

Язык	Операции отношения						Логические операции		
Математика	<	>	≤	≥	=	≠	**не**	**или**	**и**
Си	<	>	<=	>=	==	!=	!	||	&&

Установление чётности и нечётности целого числа в программах на языке C# можно осуществить с помощью операции взятия остатка от деления, обозначаемой символом %. Так, например, целое число *a* является чётным, если значение **a % 2** (остаток от деления *a* на 2) равно нулю, и нечётным – в противном случае.

В математике (а значит, и в схеме алгоритма) операция взятия остатка от деления обозначается **mod**.

Пример программы на языке C#

Задание: определить количество положительных чисел среди заданных целых чисел a, b.

Листинг программы на языке C# (первый способ).

```
using System;
class Program
{
    static void Main()
    {
        int a, b, k ;


        //Ввод исходных данных
        Console.WriteLine("Введите два целых числа");
        a = int.Parse(Console.ReadLine());
        b = int.Parse(Console.ReadLine());

        //Контрольный вывод данных
        Console.WriteLine(@"Введено: a={0}; b={1}", a, b);

        //Расчет
        if (a <= 0 && b <= 0) k = 0;
        else if (a > 0 && b > 0) k = 2;
        else k = 1;
        //Вывод результата
        Console.WriteLine(@"Количество положительных чисел при a={0}; b={1} равно {2}", a, b, k);

        Console.ReadLine();
    }
}
```

Листинг программы на языке C# (второй способ).

```
using System;
class Program
{
    static void Main()
    {
        int a, b, k;


        //Ввод исходных данных
        Console.WriteLine("Введите два целых числа");
        a = int.Parse(Console.ReadLine());
        b = int.Parse(Console.ReadLine());

        //Контрольный вывод данных
        Console.WriteLine(@"Введено: a={0}; b={1}", a, b);

        //Расчет Вариант 2
        k = 0;
        if (a > 0) k++; // тоже, что k = k+1;
        if (b > 0) k++;
        //Вывод результата
        Console.WriteLine(@"Количество положительных чисел при a={0}; b={1} равно {2}", a, b, k);

        Console.ReadLine();
    }
}
```