Лабораторная работа № 2.
Программирование разветвляющихся алгоритмов

Цель работы: Формирование умений и навыков решения задач на составление разветвляющихся алгоритмов и программ на языке C#.

Теоретическая часть

Операции отношения и логические операции

Операции отношения используются для сравнения значений переменных или выражений между собой.

Логические операции используются для составления логических выражений на основе выражений, которые используют операции отношения.

Результат выполнения операций отношения и логических операций представляется в виде логического значения типа `bool`: `true` (истина) или `false` (ложь). Перечень операций приведён в таблицах 2.1 и 2.2.

Таблица 2.1. Операции отношения

Операция	Назначение
==	Равно
!=	Не равно
<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно

Таблица 2.2. Логические операции

Операция	Назначение
&	И
|	ИЛИ
^	Исключающее ИЛИ
&&	Сокращенное И
||	Сокращенное ИЛИ
!	НЕ

Логические операции &, &&, |, ||, ^, !, выполняются в соответствии со следующей таблицей истинности:

Таблица 2.3. Таблица истинности

a	*b*	*a* & *b*, *a* && *b*	*a* | *b*, *a* || *b*	*a* ^ *b*	! *a*
false	false	false	false	false	true
false	true	false	true	true	true
true	false	false	true	true	false
true	true	true	true	false	false

Примеры логических выражений, которые являются истинными когда:

1. хотя бы одно из чисел *x*, *y*, *z* является положительным:
   - `x > 0 || y > 0 || z > 0`
   - `x > 0 | y > 0 | z > 0`
2. каждое из чисел *a* и *b* является четным:
   - `a % 2 == 0 && b % 2 == 0`
   - `a % 2 == 0 & b % 2 == 0`
3. только одно из чисел *x* или *y* кратно пяти:
   - `(x % 5 == 0 && y % 5 != 0) || (x % 5 != 0 && y % 5 == 0)`
   - `x % 5 == 0 ^ y % 5 == 0`
4. числа *a*, *b* и *c* является сторонами треугольника:
   - `a + b > c && a + c > b && b + c > a`
5. натуральное число *x* – двузначное:
   - `x >= 10 && x <= 99`
   - `!(x < 10 || x > 99)`

Операции `&&` и `||`, когда возможно, *сокращают* вычисления. Если первый операнд операции `&&` имеет значение `false`, то ее результат будет иметь значение `false` независимо от значения второго операнда. Если же первый операнд операции `||` имеет значение `true`, то ее результат будет иметь значение `true` независимо от значения второго операнда. Благодаря этому экономится время и повышается эффективность кода.

Операции `&` и `|` *не поддерживают сокращение вычислений*. По этой причине эти операции используются редко в качестве логических операций. Операции `&` и `|` выполняются как поразрядные (побитовые) только в случае применения к числам.

Пример. Программа, в которой используется операция `&&`, чтобы не допустить ошибки деления на нуль.

```
using System;
class Program
{
    static void Main()
    {
        int a, d;
        a = 10;
        d = 2;
        if (d != 0 && a % d == 0)
```

```
            Console.WriteLine("{0} делится нацело на {1}", a, d);

        d =  0; // установим d равным нулю
        // так как d = 0, второй операнд не вычисляется
        if (d != 0 && a % d == 0)
            Console.WriteLine("{0} делится нацело на {1}", a, d);
        Console.ReadLine();
    }
}
```

Результат работы программы:

```
10 делится нацело на 2
```

Попытка проделать то же самое без сокращенной операции И приведет к ошибке (деление на нуль):

```
if (d != 0 & a % d == 0)
    Console.WriteLine("{0} делится нацело на {1}", a, d);
```

Условный оператор if

Условный оператор `if` используется для разветвления процесса вычислений на два направления. На рисунке 2.1. приведена блок-схема оператора:

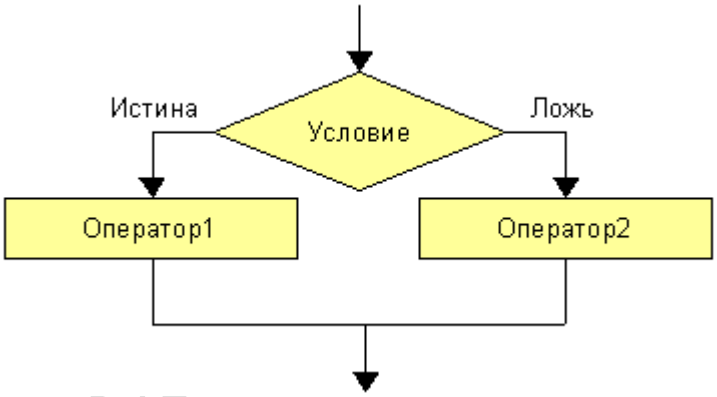

Рис. 2.1. Блок-схема оператора if

Общий вид условного оператора `if` (квадратными скобками отмечены необязательные конструкции оператора):

```
if (логическое_выражение)
    оператор1;
[else
    оператор2;]
```

Сначала вычисляется `логическое_выражение`. Если оно имеет значение `true`, выполняется `оператор1`, иначе выполняется `оператор2`. После этого управление передается на оператор, следующий за условным.

Ветвь `else` не является обязательной и может отсутствовать.

Примеры:

```
if (a > b)              // Полная условная конструкция
    max = a;
else
    max = b;

if (a < 0)              // Сокращенная условная конструкция
    a = -a;
```

Логическое выражение помещается в скобки обязательно.

Если в какой-либо ветви требуется выполнить несколько операторов, их необходимо заключить в *блок* с помощью фигурных скобок {...}. Блок может содержать любые операторы, в том числе описания и другие условные операторы.

Пример.

```
if (a > 0)
{
    x = a - b;
    y = b / a;
}
else
{
    x = a + b;
    y = a * b;
}
```

Если требуется проверить несколько условий, их объединяют знаками логических операций.

Пример.

```
if (t < -15 || t > 35) //если температура меньше -15 или больше 35
    Console.WriteLine("Температура неблагоприятна для прогулки!");
else
    Console.WriteLine("Рекомендуем погулять");
```

Часто необходимо осуществлять последовательную проверку нескольких условий. Для решения этой задачи можно использовать вложенные условные операторы `if`. Приведем один из наиболее распространенных способов выбора по значению из нескольких вариантов:

```
if (логическое_выражение1)
    оператор1;
else if (логическое_выражение2)
    оператор2;
else if (логическое_выражение3)
    оператор3;
...  ...  ...
[else
    операторN;]
```

Логические выражения просматриваются последовательно в порядке их написания. Как только какое-то выражение становится истинным, выполняется следующий за ним оператор (или блок операторов), с завершением которого завершается и оператор `if`. Последняя `else`-часть может отсутствовать.

Пример.

```
if (a > b && a > c)
    max = a;
else if (b > c)
    max = b;
else
    max = c;
```

Оператор switch

Оператор* switch *(переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Структурная схема оператора приведена на рисунке 2.2.

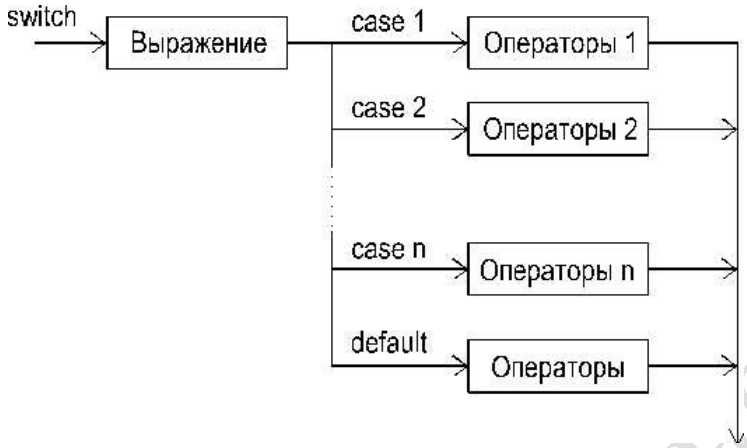

Рис. 2.2. Структурная схема оператора switch

Формат оператора:

```
switch (выражение)
{
    case константное_выражение_1:
        [ список_операторов_1; ]
    case константное_выражение_2:
        [ список_операторов_2; ]
    ...
    case константное_выражение_n:
        [ список_операторов_n; ]
    [ default: операторы; ]
}
```

Выполнение оператора начинается с вычисления выражения, следующего за ключевым словом `switch`. Это выражение должно быть целочисленным (включая `char`) или строковым. Результат вычислений сравнивается с константными выражениями блоков `case`. В случае совпадения значения управление передается оператору соответствующего блока `case`. Если совпадений не обнаружено, выполняются операторы, расположенные после слова `default`, а при его отсутствии управление передается следующему за `switch` оператору.

Каждая ветвь переключателя должна заканчиваться явным *оператором перехода*, а именно одним из операторов `break`, `goto` или `return`.

Операторы, расположенные в блоках `case` и `default`, необязательно заключать в фигурные скобки.

Пример. Программа, которая по введенной школьной оценке выводит ее словесное наименование.

```
using System;
class Program
{
    static void Main()
    {
        int a;
        Console.WriteLine("Введите оценку:");
        a = Convert.ToInt32(Console.ReadLine());
```

```
        switch (a)
        {
            case 2: Console.WriteLine("Неудовлетворительно"); break;
            case 3: Console.WriteLine("Удовлетворительно"); break;
            case 4: Console.WriteLine("Хорошо"); break;
            case 5: Console.WriteLine("Отлично"); break;
            default: Console.WriteLine("Ошибка"); break;
        }
        Console.ReadLine();
    }
}
```

Результат работы программы:

```
Введите оценку:
5
Отлично
```

В некоторых случаях требуется выполнить одну и ту же обработку для разных значений выражения, расположенного после ключевого слова `switch`.

Пример. Программа, которая по номеру дня недели выводит одно из сообщений: "Рабочий день" или "Выходной".

```
using System;
class Program
{
    static void Main()
    {
        int n;
        Console.WriteLine("Введите номер дня недели:");
        n = Convert.ToInt32(Console.ReadLine());
        switch (n)
        {
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
                Console.WriteLine("Рабочий день"); break;
            case 6:
            case 7:
                Console.WriteLine("Выходной"); break;
            default:
                Console.WriteLine("Неверный номер"); break;
        }
        Console.ReadLine();
    }
}
```

Результат работы программы:

```
Введите номер дня недели:
4
Рабочий день
```

Условная операция

Условная операция (чаще называемая *тернарной операцией*, поскольку принимает три операнда) используется для сокращения объема кода. Ею можно заменять простые по сложности операторы `if…else`. Формат операции:

```
логическое_выражение ? выражение1 : выражение2
```

Сначала вычисляется `логическое_выражение`. Если оно истинно, то результатом условной операции является `выражение1`, иначе – `выражение2`.

Пример использования условной операции для проверки числа на чётность:

```
using System;
class Program
{
    static void Main()
    {
       int a;
       Console.WriteLine("Введите число:");
       a = Convert.ToInt32(Console.ReadLine());
       Console.WriteLine(a % 2 == 0 ? "Число чётное" : "Число нечётное");
       Console.ReadLine();
    }
}
```

Результат работы программы:

```
Введите число:
15
Число нечётное
```

Условную операцию также можно использовать для присваивания значений.

Пример. Программа, которая находит большее число из двух вводимых.

```
using System;
class Program
{
    static void Main()
    {
        double a, b, max;
        Console.WriteLine("Введите первое число:");
        a = Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Введите второе число:");
        b = Convert.ToDouble(Console.ReadLine());
        max = a > b ? a : b;
        Console.WriteLine("Максимальное число: " + max);
        Console.ReadLine();
    }
}
```

Результат работы программы:

```
Введите первое число:
7
Введите второе число:
5
Максимальное число: 7
```

Практическая часть

Пример 1. Составить программу для решения задачи: «Даны четыре вещественных числа. Найти среднее арифметическое положительных».

```
using System;
class Program
{
    static void Main()
    {
        int k = 0;
        double a, b, c, d, S = 0;
        Console.WriteLine("Введите четыре числа: ");
        a = Convert.ToDouble(Console.ReadLine());
        b = Convert.ToDouble(Console.ReadLine());
        c = Convert.ToDouble(Console.ReadLine());
        d = Convert.ToDouble(Console.ReadLine());
        if (a > 0)
        {
            S += a; k++;
        }
        if (b > 0)
        {
            S += b; k++;
        }
        if (c > 0)
        {
            S += c; k++;
        }
        if (d > 0)
        {
            S += d; k++;
        }
        if (k == 0)
            Console.WriteLine("Положительные числа отсутствуют");
        else
            Console.WriteLine("Средн. арифметическое положит.-х чисел: " + S /
k);
        Console.ReadLine();
    }
}
```

Результат работы программы:

```
Введите четыре числа:
5
-7
-2
6
Средн. арифметическое положит.-х чисел: 5,5
```

Пример 2. Программа реализует простейший калькулятор на 4 действия.

```
using System;
class Program
{
    static void Main()
    {
        double a, b, res = 0;
```

```
            char op;
            bool flag = true;
            Console.WriteLine("Введите первый операнд:");
            a = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Введите знак операции:");
            op = Convert.ToChar(Console.ReadLine());
            Console.WriteLine("Введите второй операнд:");
            b = Convert.ToDouble(Console.ReadLine());
            switch (op)
            {
                case '+':
                    res = a + b; break;
                case '-':
                    res = a - b; break;
                case '*':
                    res = a * b; break;
                case ':':
                case '/':
                    if (b != 0)
                        res = a / b;
                    else
                    {
                        Console.WriteLine("Ошибка. Делить на нуль нельзя!");
                        flag = false;
                    }
                    break;
                default:
                    Console.WriteLine("Недопустимая операция");
                    flag = false;
                    break;
            }
            if (flag) Console.WriteLine("Результат: " + res);
            Console.ReadLine();
        }
}
```

Результат работы программы:

```
Введите первый операнд:
5
Введите знак операции:
*
Введите второй операнд:
7
Результат: 35
Введите первый операнд:
8
Введите знак операции:
/
Введите второй операнд:
0
Ошибка. Делить на нуль нельзя!
```

Задания для самостоятельной работы

1. Разработайте приложения для решения следующих задач:

1.1. Даны четыре вещественных числа. Найти сумму тех чисел, которые больше пяти.

1.2. Даны четыре целых числа. Найти сумму тех чисел, которые кратны трем.

1.3. Даны четыре вещественных числа. Найти сумму тех чисел, которые принадлежат отрезку [1; 10].

1.4. Даны четыре целых числа. Определить, сколько из них четных.

1.5. Даны четыре вещественных числа. Определить, сколько из них принадлежат интервалу (0; 15).

1.6. Даны четыре целых числа. Определить, есть ли среди них нечетные.

1.7. Даны четыре целых числа. Определить, сколько из них двузначных.

1.8. Даны четыре вещественных числа. Вывести на экран те из них, которые являются отрицательными.

1.9. Даны четыре целых числа. Вывести на экран те из них, которые кратны пяти.

1.10. Даны четыре целых числа. Вывести на экран те из них, которые являются нечетными.

1.11. Даны четыре вещественных числа. Вывести на экран те, которые принадлежат интервалу (0; 40).

1.12. Даны координаты точки *A*, не лежащей на координатных осях *OX* и *OY*. Определить номер координатной четверти, в которой находится данная точка.

1.13. На плоскости *XOY* задана своими координатами точка *A*. Указать, где она расположена (на какой оси или в каком координатном угле).

1.14. Заданы три положительных числа *a, b* и *c*. Определить, являются ли они последовательно стоящими элементами арифметической прогрессии. Если являются, то определить разность прогрессии.

1.15. Дано целое число. Вывести его строку-описание вида «отрицательное четное число», «нулевое число», «положительное нечетное число» и т. д.

1.16. Дано целое число, лежащее в диапазоне 1–999. Вывести его строку-описание вида «четное двузначное число», «нечетное трехзначное число» и т. д.

1.17. Подсчитать, хватит ли *A* руб. на поездку *S* км, если расход бензина – 1 литр на 10 км, стоимость 1 литра бензина *K* руб.

1.18. Имеется комната длиной *A* метров и шириной *B* метров. Определить, хватит ли *N* банок краски, чтобы покрасить пол, если одной банки краски хватает на 5 м^2.

1.19. В магазине продается костюмная ткань длиной *X* метров и шириной 2 м. На изготовление одного костюма уходит *Y* м^2 ткани. Определить, хватит ли ткани на *K* костюмов.

1.20. Известны результаты двух попыток прыжков в высоту двух спортсменов. Определить лучшую попытку каждого спортсмена.

1.21. Даны три числа. Определить порядковый номер меньшего из них.

1.22. Даны три числа. Найти сумму двух наибольших из них.

1.23. Даны три числа. Найти произведение двух наименьших из них.

1.24. На плоскости заданы три точки своими координатами. Определить, какая из них расположена ближе к началу координат.

1.25. Даны три числа. Найти среднее из них (то есть число, расположенное между наименьшим и наибольшим).

1.26. Даны три числа. Найти сумму большего и меньшего из них.

1.27. Даны три числа. Вывести вначале большее, затем меньшее из них.

1.28. Даны три числа. Определите, сколько среди них совпадающих.

1.29. Заданы три стороны треугольника *a, b* и *c*. Определить какая из сторон (*a, b* или *c*) является наибольшей.

1.30. На числовой оси *OX* расположены три точки: *A, B, C*. Определить, какая из двух последних точек (*B* или *C*) расположена ближе к *A*, и вывести эту точку и ее расстояние от точки *A*.

1.31. Заданы три стороны треугольника *a, b* и *c*. Определить, является ли этот треугольник прямоугольным, и какая сторона служит гипотенузой.

1.32. Даны вещественные положительные числа *a, b, c*. Если существует треугольник со сторонами *a, b, c*, то определить, является ли он остроугольным.

1.33. Пройдет ли кирпич со сторонами *a, b* и *c* сквозь прямоугольное отверстие со сторонами *p* и *q*? Стороны отверстия должны быть параллельны граням кирпича.

1.34. Известны длина и ширина открытки и конверта. Определить, поместится ли открытка в конверт.

1.35. Даны три угла. Проверить, могут ли они быть углами треугольника. Если да, то проверить, будет ли этот треугольник остроугольным.

1.36. Даны вещественные положительные числа *a, b, c*. Если существует треугольник со сторонами *a, b, c*, то определить, является ли он тупоугольным.

1.37. Даны три угла. Проверить, могут ли они быть углами треугольника. Если да, то проверить, будет ли этот треугольник тупоугольным.

1.38. Дано трехзначное число. Определить, верно ли, что оно содержит две одинаковые цифры. Например: 363, 844, 113.

1.39. Известны площадь круга S_1 и площадь квадрата S_2. Определить, поместится ли круг в квадрат.

1.40. Даны два трехзначных числа. Найти произведение их минимальных цифр.

1.41. Известны площадь круга S_1 и площадь квадрата S_2. Определить, поместится ли квадрат в круг.

1.42. Для натурального числа *k* вывести на экран фразу «мы нашли *k* грибов в лесу», согласовав окончание слова «гриб» с числом *k*.

1.43. Для натурального числа *k* (от 1 до 120) вывести на экран фразу «Мне *k* лет», при этом в нужных случаях слово «лет» заменяя на слово «год» или «года».

1.44. Дано трехзначное число. Определить, будут ли цифры этого числа являться членами арифметической прогрессии.

1.45. Дано четырехзначное число. Определить, верно ли, что оно содержит три одинаковые цифры. Например: 3363, 4844, 1113.

1.46. В восточном календаре принят 60-летний цикл, состоящий из 12-летних подциклов, обозначаемых названиями цвета: зеленый, красный, желтый, белый и черный. В

каждом подцикле года носят названия животных: крысы, быка, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, петуха, собаки и свиньи. По номеру года вывести на экран его название (Справка: 1984 год – год Зеленой Крысы – начало очередного цикла).

1.47. Дано четырехзначное число. Определить, различны ли все цифры этого числа. Например: 3678 – различны, 1123 – нет.

1.48. В зависимости от порядкового номера месяца (1, 2, …) вывести на экран количество дней в этом месяце. Год может быть как високосный, так и не високосный (информация об этом вводится с клавиатуры). *Указание.* В современном (григорианском) календаре каждый год, номер которого делится на 4, является високосным, за исключением тех, которые делятся на 100 и не делится на 400. Например, 1900 год – не високосный, 2000 год – високосный.

1.49. Правильная дата некоторого дня невисокосного года определяется двумя натуральными числами: d (число) и m (порядковый номер месяца). По заданным d и m вывести на экран дату следующего дня.

1.50. Правильная дата некоторого дня невисокосного года определяется двумя натуральными числами: d (число) и m (порядковый номер месяца). По заданным d и m вывести на экран дату предыдущего дня.

2. Разработайте приложения для решения задач из сборника задач по программированию согласно вашему варианту (В.И. Великодный. Задачи по программированию: Учебное пособие. Раздел «Алгоритмы с ветвлением»).

№ варианта	**Задания из лаб. работы**	**Задания из сборника задач по программированию**
1	1.1, 1.11, 1.21, 1.31, 1.41	2.1-1, 2.2-1, 2.3-1
2	1.2, 1.12, 1.22, 1.32, 1.42	2.1-2, 2.2-2, 2.3-2
3	1.3, 1.13, 1.23, 1.33, 1.43	2.1-3, 2.2-3, 2.3-3
4	1.4, 1.14, 1.24, 1.34, 1.44	2.1-4, 2.2-4, 2.3-4
5	1.5, 1.15, 1.25, 1.35, 1.45	2.1-5, 2.2-5, 2.3-5
6	1.6, 1.16, 1.26, 1.36, 1.46	2.1-6, 2.2-6, 2.3-6
7	1.7, 1.17, 1.27, 1.37, 1.47	2.1-7, 2.2-7, 2.3-7
8	1.8, 1.18, 1.28, 1.38, 1.48	2.1-8, 2.2-8, 2.3-8
9	1.9, 1.19, 1.29, 1.39, 1.49	2.1-9, 2.2-9, 2.3-9
10	1.10, 1.20, 1.30, 1.40, 1.50	2.1-0, 2.2-0, 2.3-0