Приднестровский государственный университет им. Т.Г. Шевченко

физико-математический факультет

кафедра прикладной математики и информатики

Л А Б О Р А Т О Р Н А Я Р А Б О Т А № 2

по дисциплине:
«*Системы программирования*»

Тема:
«Алгоритмы с ветвлением»

РАЗРАБОТАЛИ:
ст. преподаватель кафедры ПМиИ
Великодный В.И.
ст. преподаватель кафедры ПМиИ
Калинкова Е.В.

г. Тирасполь

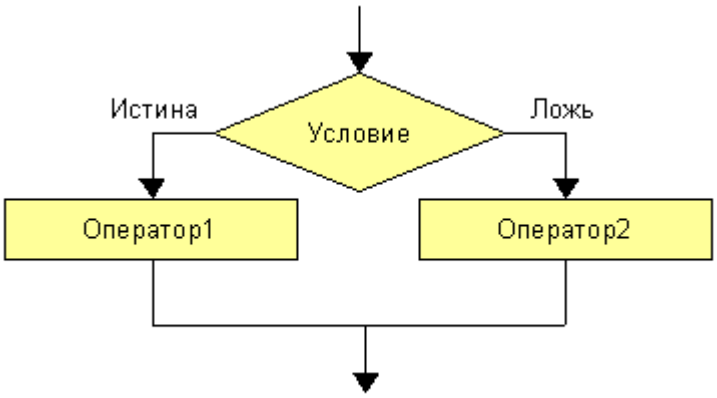
Цель работы:

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

Теоретическая часть

Условный оператор if

Условный оператор `if` используется для разветвления процесса вычислений на два направления. Блок-схема ветвления выглядит следующим образом:



Управляющая структура - ветвление

Общий вид условного 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 форма оператора
    max = a;
else
    max = b;

if (a < 0)              // Сокращенная форма оператора
    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 (x > 0)
{
    x = -x; a = 2 * b;
}
else
{
```

```
    int i = 2;
    x *= i; a = x / b;
}
```

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

Пример.

```
if (t < -20 || t > 40)      //если температура меньше -20 или больше 40
    Console.WriteLine("Вам лучше посидеть дома!");
else
    Console.WriteLine("Можете идти гулять");
```

Часто нам приходится осуществлять выбор более чем из двух вариантов. Чтобы учесть это, мы можем расширить конструкцию `if...else` конструкцией `else if`. Приведем весьма распространенный способ выбора по значению из нескольких вариантов:

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

Выражения просматриваются последовательно слева направо; как только какое-то выражение становится истинным, выполняется следующий за ним оператор, и на этом вся цепочка заканчивается. Последняя `else`-часть, как и раньше, может быть опущена.

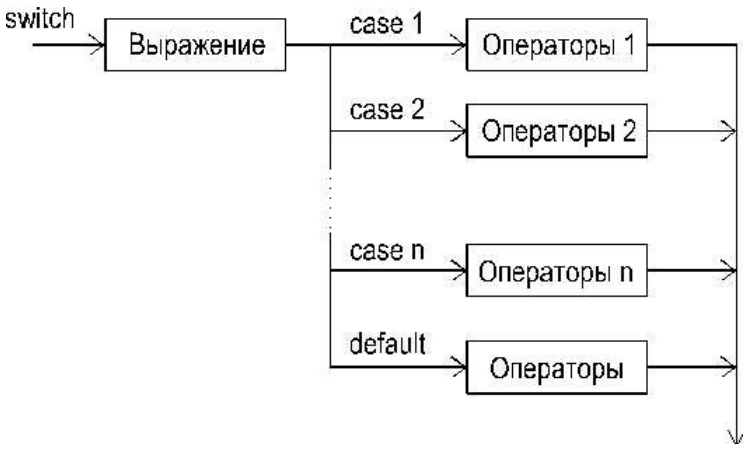
Пример.

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

Замечание. *Применяйте отступы в* ***строках****, чтобы сделать* ***структуру программы более наглядной****.*

Оператор switch

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



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. Тип выражения чаще всего целочисленный (включая `char`) или строковый. Затем управление передается первому оператору из списка, помеченному константным выражением, значение которого совпало с вычисленным. Если совпадения не произошло, 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`:

- оператор `break` выполняет выход из самого внутреннего из объемлющих его операторов `switch`, `for`, `while` и `do`;
- оператор `goto` выполняет переход на указанную после него метку, обычно это метка `case` одной из нижележащих ветвей оператора `switch`;
- оператор `return` выполняет выход из функции, в теле которой он записан.

Тернарный оператор

Тернарный оператор используется для сокращения объема кода. Им можно заменять простые по сложности операторы `if…else`. Формат оператора:

```
логическое_выражение ? выражение1 : выражение2
```

Сначала вычисляется `логическое_выражение`. Если оно истинно, то вычисляется `выражение1`, а полученный результат определяет значение всего выражения ? в целом. Если же `логическое_выражение` оказывается ложным, то вычисляется `выражение2`, и его значение становится общим для всего выражения ?.

Пример использования тернарного оператора для проверки числа на чётность:

```
using System;
namespace ConsoleApplication1
{
    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();
        }
    }
}
```

Тернарный оператор также можно использовать для присваивания значений.

Пример программы, которая находит большее число из двух вводимых:

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main()
        {
            int a, b, max;
            Console.WriteLine("Введите первое число:");
            a = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Введите второе число:");
            b = Convert.ToInt32(Console.ReadLine());
            max = a > b ? a : b;
            Console.WriteLine("Максимальное число: " + max);
            Console.ReadLine();
        }
    }
}
```

Практическая часть

Рассмотрим несколько примеров.

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

```
using System;
namespace ConsoleApplication1
{
    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("Положительных чисел нет");
            else
                Console.WriteLine("Ср. арифметическое положит.-х чисел: " + S/k);
            Console.ReadLine();
        }
    }
}
```

Пример 2. Программа реализует простейший калькулятор на 4 действия.

```
using System;
namespace ConsoleApplication1
{
    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.Read());
            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 '/':
                    res = a / b; break;
                default:
                    Console.WriteLine("Недопустимая операция");
                    flag = false; break;
            }
            if (flag) Console.WriteLine("Результат: " + res);
            Console.ReadLine();
        }
    }
}
```

Обратите внимание, что если после метки с двоеточием идет несколько операторов, то их ***необязательно помещать в блок*** с помощью скобок { }.

Пример 3. Составить программу, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений: "Рабочий день", "Суббота" или "Воскресенье".

```
using System;
namespace ConsoleApplication1
{
    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:
                    Console.WriteLine("Суббота"); break;
                case 7:
                    Console.WriteLine("Воскресенье"); break;
                default:
                    Console.WriteLine("Неверный номер");  break;
            }
            Console.ReadLine();
        }
    }
}
```

Задания для самостоятельной работы

Разработайте приложения для решения задач из сборника задач по программированию согласно вашему варианту.