Организация ветвлений в программе.
Оператор выбора *Switch*

Оператор выбора *switch* предназначен для разветвления процесса вычислений по нескольким направлениям. Формат оператора:

```
switch ( <выражение> )
{
 case <константное_выражение_1>:
    [<оператор 1>];  <оператор перехода>;
 case <константное_выражение_2>:
    [<оператор 2>]; <оператор перехода>;
  ...
 case <константное_выражение_n>:
    [<оператор n>]; <оператор перехода>;
 [default: <оператор>; ]
}
```

Замечание. *Операторы, записанные в квадратных скобках, являются необязательными элементами в операторе switch. Если они отсутствуют, то могут отсутствовать и соответствующие им операторы перехода.*

Выражение, стоящее за ключевым словом *switch*, должно иметь арифметический, символьный, строковый тип или тип указатель. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящего внутри скобок *switch* или приводиться к нему. Ключевое слово *case* и расположенное после него константное выражение называют также меткой *case*.

Выполнение оператора начинается с вычисления выражения, расположенного за ключевым словом *switch*. Полученный результат сравнивается с меткой *case*. Если результат выражения соответствует метке *case*, то выполняется оператор, стоящий после этой метки, за которым *обязательно* должен следовать оператор перехода: *break, goto* и т.д. В случае отсутствия оператора перехода компилятор выдаст сообщении об ошибке. При использовании оператора *break* происходит выход из *switch* и управление передается оператору, следующему за *switch*. Если же используется оператор *goto*, то управление передается оператору, помеченному меткой, стоящей после *goto*.

Замечания.

Оператор перехода goto лучше использовать для перехода по меткам внутри switch, и не использовать его для выхода из оператора switch.

Для повышения производительности рекомендуется размещать ветви, вероятность выбора которых является наибольшей, ближе к началу. В этом случае будет на выбор требуемого варианта будет тратиться меньше времени.

Пример 1. По заданному виду арифметической операции (сложение, вычитание, умножение и деление) и двум операндам, вывести на экран результат применения данной операции к операндам.

```
static void Main()
{
    Console.Write("OPER= ");
    char oper=char.Parse(Console.ReadLine());
    bool ok=true;
    Console.Write("A= ");
    double a=double.Parse(Console.ReadLine());
    Console.Write("B= ");
    double b=double.Parse(Console.ReadLine());
    double res=0;
    switch (oper)
    {
        case '+' :
            res = a + b;
            break;          //1
        case '-' :
            res = a - b;
            break;
        case '*' :
            res = a * b;
            break;
        case ':' :
            if (b != 0)      //2
            {
                res = a / b;
                break;
            }
            else
            {
                goto default;
            }
        default:
              ok = false;
               break;
        }
        if (ok)
        {
            Console.WriteLine("{0} {1} {2} = {3}", a, oper, b, res);
        }
        else
        {
            Console.WriteLine("error");
        }
    }
```

Результат выполнения программы:

oper	x	y	rez
+	4	5	9
:	4	0	error
%	4	3	error

Задания.

1. Замените в строке 1 оператор *break*, на оператор *goto case '-'* и посмотрите, что произойдет, если в качестве операции ввести +.
2. В условном операторе *if* (см. строку 2) уберите ветку *else* и посмотрите, что произойдет. Дайте этому объяснение.
3. Решите поставленную задачу с использованием вложенных операторов *if*.

Объединение меток CASE

Если необходимо, чтобы для разных меток выполнялось одно и то же действие, то метки перечисляются через двоеточие.

Пример 2. В программе из предыдущего примера предусмотрим дополнительную возможность для знака операции деления:

```
switch (oper)
{
   case '+':
         res = a + b;
         break;
   case '-':
         res = a - b;
         break;
   case '*':
         res = a * b;
         break;
   case ':' : case '/' :          //перечисление меток
         if (b != 0)
         {
           res = a / b;
           break;
         }
         else
           {
                goto default;
           }
   default:
       ok = false;
      break;
}
```

Задание. Напишите программу, которая по признаку геометрической фигуры (п – прямоугольник, т – треугольник, к – квадрат), запрашивает необходимые данные для расчетов и выводит на экран периметр и площадь заданной фигуры.

При запуске консольного приложения на экране должен появляться список (меню) для выбора пользователя, например

```
Программа рассчитывает периметр и площадь фигур:
п – прямоугольник
т – треугольник
к – квадрат

Введите соответствующую букву (п, т, к) |
```