Л а б о р а т о р н а я р а б о т а № 3

**РАБОТА С ПЕРЕМЕННЫМИ** [**БУЛЕВА ТИПА, ЛОГИЧЕСКИЕ ВЫРАЖЕНИЯ.**](#_Toc290318449)

**Цель работы:**

* Ознакомиться с переменными булевого типами,
* Освоить работу с булевыми величинами,
* Познакомиться с оператором выбора,
* Повторить операции ввода/вывода

**Краткие теоретические сведения Типы данных.**

Отдельный набор операций представляет условные выражения. Такие операции возвращают логическое значение, то есть значение типа *bool: true*, если выражение истинно, и *false*, если выражение ложно. К подобным операциям относятся операции сравнения и логические операции.

**Операции сравнения**

В операциях сравнения сравниваются два операнда и возвращается значение типа *bool - true*, если выражение верно, и *false*, если выражение неверно.

**= =**

Сравнивает два операнда на равенство. Если они равны, то операция возвращает *true*, если не равны, то возвращается *false*:

int a = 10;

int b = 4;

bool c = a = = b; // false

**!=**

Сравнивает два операнда и возвращает *true*, если операнды не равны, и *false*, если они равны.

int a = 10;

int b = 4;

bool c = a != b; // true

bool d = a!=10; // false

**<**

Операция "меньше чем". Возвращает *true*, если первый операнд меньше второго, и *false*, если первый операнд больше второго:

int a = 10;

int b = 4;

bool c = a < b; // false

**>**

Операция "больше чем". Сравнивает два операнда и возвращает *true*, если первый операнд больше второго, иначе возвращает *false*:

int a = 10;

int b = 4;

bool c = a > b; // true

bool d = a > 25; // false

**<=**

Операция "меньше или равно". Сравнивает два операнда и возвращает *true*, если первый операнд меньше или равен второму. Иначе возвращает *false*.

int a = 10;

int b = 4;

bool c = a <= b; // false

bool d = a <= 25; // true

**>=**

Операция "больше или равно". Сравнивает два операнда и возвращает *true*, если первый операнд больше или равен второму, иначе возвращается *false*:

int a = 10;

int b = 4;

bool c = a >= b; // true

bool d = a >= 25; // false

Операции <, > <=, >= имеют больший приоритет, чем == и !=.

**Логические операции**

Также в C# определены логические операторы, которые также возвращают значение типа *bool*. В качестве операндов они принимают значения типа *bool*. Как правило, применяются к отношениям и объединяют несколько операций сравнения.

**|**

Операция логического сложения или логическое ИЛИ. Возвращает *true*, если хотя бы один из операндов возвращает *true*.

bool x1 = (5 > 6) | (4 < 6); // 5 > 6 - false, 4 < 6 - true, поэтому возвращается true

bool x2 = (5 > 6) | (4 > 6); // 5 > 6 - false, 4 > 6 - false, поэтому возвращается false

**&**

Операция логического умножения или логическое И. Возвращает *true*, если оба операнда одновременно равны *true*.

bool x1 = (5 > 6) & (4 < 6); // 5 > 6 - false, 4 < 6 - true, поэтому возвращается false

bool x2 = (5 < 6) & (4 < 6); // 5 < 6 - true, 4 < 6 - true, поэтому возвращается true

**||**

Операция логического сложения. Возвращает true, если хотя бы один из операндов возвращает true.

bool x1 = (5 > 6) || (4 < 6); // 5 > 6 - false, 4 < 6 - true, поэтому возвращается true

bool x2 = (5 > 6) || (4 > 6); // 5 > 6 - false, 4 > 6 - false, поэтому возвращается false

**&&**

Операция логического умножения. Возвращает *true*, если оба операнда одновременно равны *true*.

bool x1 = (5 > 6) && (4 < 6); // 5 > 6 - false, 4 < 6 - true, поэтому возвращается false

bool x2 = (5 < 6) && (4 < 6); // 5 < 6 - true, 4 < 6 - true, поэтому возвращается true

**!**

Операция логического отрицания. Производится над одним операндом и возвращает *true*, если операнд равен *false*. Если операнд равен *true*, то операция возвращает *false*:

bool a = true;

bool b = !a; // false

**^**

Операция исключающего ИЛИ. Возвращает *true*, если либо первый, либо второй операнд (но не одновременно) равны *true*, иначе возвращает *false*

bool x5 = (5 > 6) ^ (4 < 6); // 5 > 6 - false, 4 < 6 - true, поэтому возвращается true

bool x6 = (50 > 6) ^ (4 / 2 < 3); // 50 > 6 - true, 4/2 < 3 - true, поэтому возвращается false

Здесь у нас две пары операций | и || (а также & и &&) выполняют похожие действия, однако же они не равнозначны.

В выражении z=x|y; будут вычисляться оба значения - x и y.

В выражении же z=x||y; сначала будет вычисляться значение x, и если оно равно true, то вычисление значения y уже смысла не имеет, так как у нас в любом случае уже z будет равно true. Значение y будет вычисляться только в том случае, если x равно false

То же самое касается пары операций &/&&. В выражении z=x&y; будут вычисляться оба значения - x и y.

В выражении же z=x&&y; сначала будет вычисляться значение x, и если оно равно false, то вычисление значения y уже смысла не имеет, так как у нас в любом случае уже z будет равно false. Значение y будет вычисляться только в том случае, если x равно true

Поэтому операции || и && более удобны в вычислениях, так как позволяют сократить время на вычисление значения выражения, и тем самым повышают производительность. А операции | и & больше подходят для выполнения поразрядных операций над числами.

**Оператор *switch***

Конструкция *switch…case* используется, когда необходимо, в зависимости от конкретного значения переменной или результата некоторой операции, выполнить требуемый блок кода.

Синтаксис конструкции имеет следующий вид:

**switch** (Выражение)

{

**case** Шаблон\_1:

// Блок кода выполняется, если Выражению соответствует Шаблон\_1

**break**;

**case** Шаблон\_2:

// Блок кода выполняется, если Выражению соответствует Шаблон\_2

**break**;

// ...

**default**:

// Блок кода выполняется, если среди перечисленных в case шаблонах

// нет подходящего

**break**;

}

В зависимости от версии языка *C*# допустимы различные шаблоны, которые можно использовать в блоках *case*. В *C*# 6 и более ранних версиях допустимы только константы следующих типов:

* *char*;
* *string*;
* *bool*;
* целочисленное значение;
* *enum*.

*С*# 7 и более поздние версии поддерживают упомянутый выше шаблон констант и другие виды шаблонов.

Рассмотрим простой пример работы с *switch*:

**int** n7 = 1;

**switch** (n7)

{

**case** 1:

Console.WriteLine("Case 1");

**break**;

**case** 2:

Console.WriteLine("Case 2");

**break**;

**default**:

Console.WriteLine("Default case");

**break**;

}

Оператор *switch* ищет совпадение значения переменной *n7* со значениями, указанными после *case*. Если совпадение найдено, то выполняется код внутри блока *case*. Блок *case* может закончится оператором:

* *break* – прерывание кода в конструкции *switch*;
* *goto case*– используется если необходимо перейти в другой *case*;
* *return* – завершает выполнение метода и возвращает указанное значение;
* *throw* – используется для выброса исключения.

Если среди шаблонов в *case* не было найдено подходящего, то будет выполнен код в блоке *default*.

В качестве шаблонов в *case* могут выступать:

* шаблоны констант;
* шаблоны типов;
* выражение с ключевым словом *when*.

Рассмотрим их более подробно.

**Шаблон константы**

Шаблон константы предполагает, что мы сравниваем значение выражение из *switch* с константами, которые располагаются в блоках *case*. Как уже было сказано выше, в таком варианте работы *switch*…*case* допустимы константы следующих типов:*char*, *string*, *bool*, целочисленное значение и *enum*. В первом примере, демонстрирующим работу оператора *switch*, уже был приведен вариант с шаблоном константы, приведем еще один, на этот раз, это будет работа со строками:

string s1 = "Three";

**switch** (s1.ToLower())

{

**case** "one": Console.WriteLine("Case one"); **break**;

**case** "two": Console.WriteLine("Case two"); **break**;

**case** "three": Console.WriteLine("Case three"); **break**;

**case** "four": Console.WriteLine("Case four"); **break**;

**case** "five": Console.WriteLine("Case five"); **break**;

**default**: Console.WriteLine("Default case"); **break**;

}

**Шаблон типа**

Шаблон типа предполагает сопоставление результата вычисления выражения, которое передается в *switch*, с типом, указанным в *case*:

case type var\_name

Если результат выражения можно привести к указанному **типу**, то полученное значение присваивается переменной с указанными именем – **var\_name**.

Для задания альтернативы со значением *null* используйте следующий формат:

case null

В [документации](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/switch#pattern-matching-with-the-switch-statement) *Microsoft* приводится список условий, при которых результат выражения будет успешно приведен к указанному в *case* типу, перечислим некоторые из них:

* выражение имеет указанный тип **type**;
* результат выражения реализует интерфейс **type**;
* результат выражения имеет тип производный от **type**.

Для демонстрации создадим несколько вспомогательных классов:

**class** Transport { **public** string Name { **get**; **set**; } }

**class** Bicycle : Transport { }

**class** Moto : Transport { }

**class** Car : Transport { }

Оператор *switch…case*, работающий с этими классами, может выглядеть следующим образом:

**object** trans = new Moto() { Name = "Suzuki" };

**switch** (trans)

{

**case** Bicycle bicycle: Console.WriteLine($"Bicycle: {bicycle.Name}"); **break**;

**case** Moto moto: Console.WriteLine($"Moto: {moto.Name}"); **break**;

**case** Car car: Console.WriteLine($"Car {car.Name}"); **break**;

**case** Transport transport: Console.WriteLine($"Transport {transport.Name}"); **break**;

**case** **null**: Console.WriteLine("Transport is null!"); **break**;

}

**Выражение с ключевым словом *when***

Язык *C*#, начиная с версии 7.0, позволяет использовать в *switch…case* предложение *when* для реализации возможности использования дополнительного условия. В этом случае типы после ключевого слова *case* могут совпадать и дополнительная фильтрация будет производиться по условию после *when*.

Дополним класс *Transport*, который мы создали в разделе “Шаблон типа”, свойством *WheelsCount*:

**class** Transport

{

**public** string Name { **get**; **set**; }

**public** **int** WheelsCount { **get**; **set**; }

}

Реализуем возможность раздельной работы с тремя вариантами велосипедов:

**object** bc = new Bicycle() { Name = "Trec", WheelsCount = 1 };

**switch** (bc)

{

**case** Bicycle bicycle **when** bicycle.WheelsCount == 1: Console.WriteLine($"Bicycle: {bicycle.Name}, type - monocycle"); **break**;

**case** Bicycle bicycle **when** bicycle.WheelsCount == 2: Console.WriteLine($"Bicycle: {bicycle.Name}, type - classic"); **break**;

**case** Bicycle bicycle **when** bicycle.WheelsCount == 3: Console.WriteLine($"Bicycle: {bicycle.Name}, type - tricycle"); **break**;

**case** Moto moto: Console.WriteLine($"Moto: {moto.Name}"); **break**;

**case** Car car: Console.WriteLine($"Car {car.Name}"); **break**;

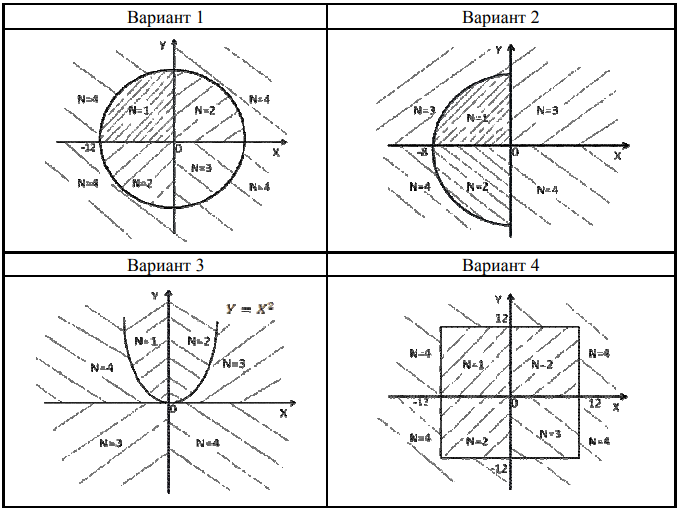
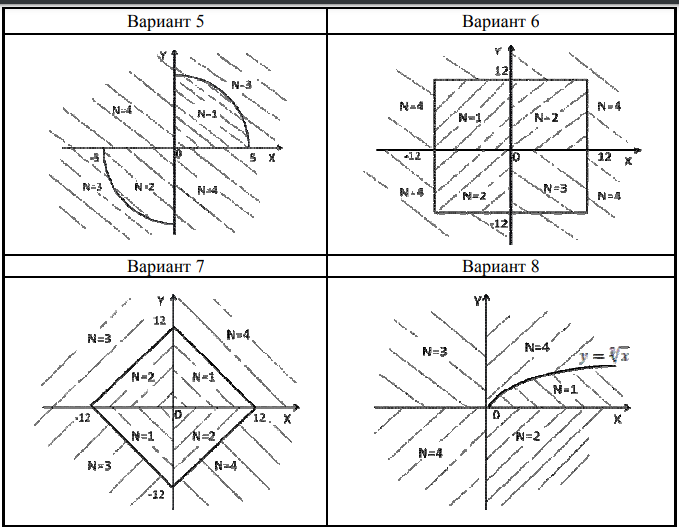
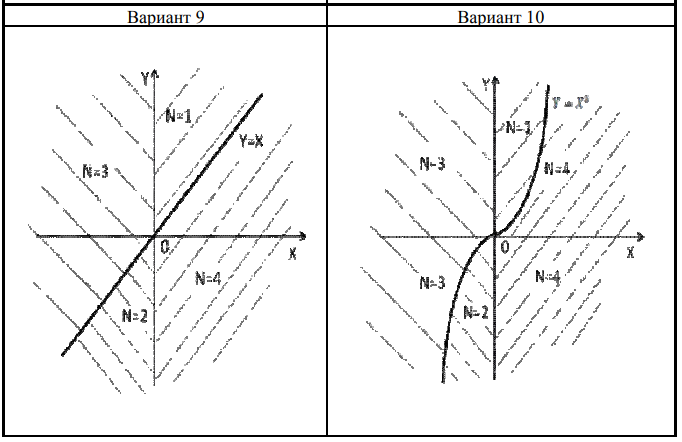
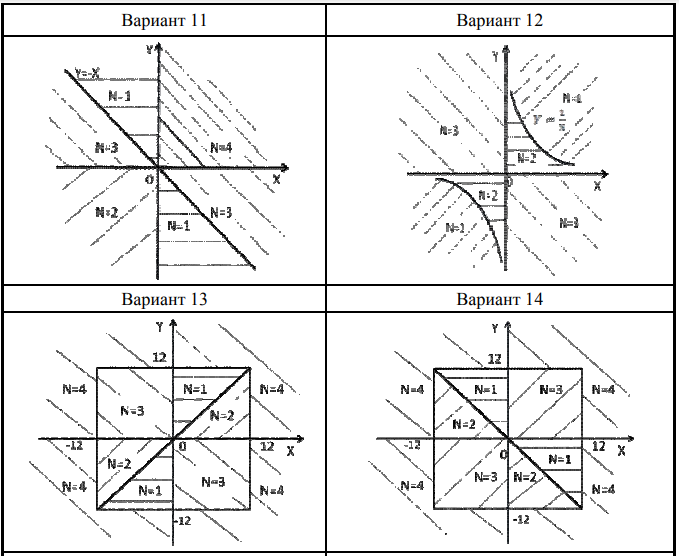
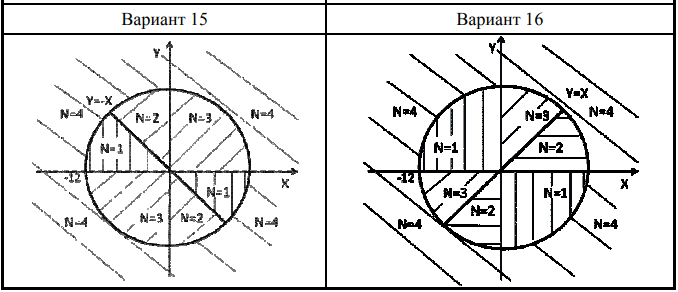
**case** Transport transport: Console.WriteLine($"Transport {transport.Name}"); **break**;

**case** **null**: Console.WriteLine("Transport is null!"); **break**;

}

**Практическая часть.**

Задание 1. Написать код, в котором для каждой области определяется переменная булева типа, определяющая находиться (true) или нет (false) в ней точка *М*(x, y) c заданными координатами (см. рисунок варианта). Границы области относить к области с наибольшим номером.

Задание 2. Преобразовать предыдущее задание так, чтобы при заданной точке и номере выбранной области получить сообщение о принадлежности этой точки область (true) или непринадлежности (false). В задании использовать оператор выбора.

Задание 3. Оператор варианта (выбора)

1. Составить программу, которая в зависимости от порядкового номера дня недели (1, 2, ..., 7) выводит на экран его название (понедельник, Вторник, воскресенье).
2. Составить программу, которая в зависимости от порядкового номера дня месяца (1, 2, ..., 12) выводит на экран его название (январь, февраль, декабрь).
3. Составить программу, которая в зависимости от порядкового номера дня месяца (1, 2, 12) выводит на экран время года, к которому относится этот месяц.
4. Составить программу, которая в зависимости от порядкового номера месяца (1, 2, ..., 12) выводит на экран количество дней в этом месяце. Рассмотреть два случая:
5. год не является високосным;
6. год может быть високосным (информация об этом вводится с клавиатуры).
7. Мастям игральных карт условно присвоены следующие порядковые номера: масти "пики" - 1, масти "трефы" - 2, масти "бубны"- 3, масти "червы" - 4. По заданному номеру масти т (1 m≤4) определить название соответствующей масти.
8. Игральным картам условно присвоены следующие порядковые номера в зависимости от их достоинства: "валету" 11, "даме" 12, "королю" - 13, "тузу" 14. Порядковые номера остальных карт соответствуют их названиям ("шестерка", "девятка" и т. п.). По заданному номеру карты k (6 ≤ k≤ 14) определить достоинство соответствующей карты.
9. Мастям игральных карт условно присвоены следующие порядковые номера: масти "пики" - 1, масти "трефы" 2, масти "бубны" - 3, масти "червы" - 4, а достоинству карт: "валету" - 11, "даме" - 12 "королю" - 13, "тузу" - 14 (порядковые номера карт остальных достоинств соответствуют их названиям: "шестерка", "девятка" и т. п.). По заданным номеру масти (15 ≤ m ≤4) и номеру достоинства карты к (6 ≤ k ≤ 14) определить полное название (масть и достоинство) соответствующей карты в виде "Дама пик", "Шестерка бубен" и т. п.
10. Дано целое число k (1 ≤ k≤ 365). Определить, каким днем недели (понедельником, вторником, .... субботой или воскресеньем) является к-й день невисокосного года, в котором 1 января:
    * + - 1. понедельник;
          2. d-й день недели (если 1 января- понедельник, то d =1, если вторник - d=2 ...., если воскресенье - d=7).
11. С начала 1990 года по некоторый день прошло *n* месяцев и 2 дня. Определить название месяца (январь, февраль и т. п.) этого дня.
12. Дата некоторого дня характеризуется двумя натуральными числами: *m* (порядковый номер месяца) и *n* (число). По заданным п и т определить:
    * + - 1. дату предыдущего дня (принять, что *n* и *m* не характеризуют 1 января);
          2. дату следующего дня (принять, что *n* и *m* не характеризуют 31 декабря).

Примечание: В обеих задачах принять, что год не является високосным.

1. Дата некоторого дня характеризуется тремя натуральными числами: *g* (год), *m* (порядковый номер месяца) и *n* (число). Заданный год может быть високосным. По заданным *g*, *n, m* определить:

а) дату предыдущего дня;

б) дату следующего дня.

1. Дата некоторого дня характеризуется тремя натуральными числами: *g* (год), *m* (порядковый номер месяца) и *n* (число). Заданный год не является високосным. По заданным *g*, *n, m* определить:

а) дату предыдущего дня;

б) дату следующего дня.

1. В старояпонском календаре был принят 60-летний цикл, состоящий из пяти 12-летних подциклов. Подциклы обозначались названиями цвета: зеленый, красный, желтый, белый и черный. Внутри каждого подцикла года носили названия животных: крыса, корова, тигр, заяц, дракон, змея, лошадь, овца, обезьяна, курица, собака и свинья. Например, 1984 год - год начала очередного цикла - назывался годом зеленой крысы.
   1. Составить программу, которая по заданному номеру года нашей эры п печатает его название по старояпонскому календарю. Значение *n* ≥ 1984;
2. Составить программу, которая по заданному номеру года нашей эры п печатает его название по старояпонскому календарю. Значение *n* может быть любым натуральным числом.

**Контрольные вопросы**

1. Что такое логический тип данных?
2. Как обозначается булева переменная?
3. Какие логические операции Вам известны?