**Занятие 4. Арифметические и логические операции**

**Арифметические операции**

Большинство операций в программировании на С# являются стандартными математическими операциями. Есть унарные операции (выполняются над одним операндом), бинарные - над двумя операндами, а также тернарные - выполняются над тремя операндами. Операндом является переменная или значение (например, число), участвующее в операции.

Унарные арифметические операции

Унарные арифметические операции — это операции, которые производятся над одним числом: ++ (инкремент) и -- (декремент). Инкремент подразумевает под собой увеличение значения переменной на единицу, декремент — наоборот, уменьшение значения переменной на единицу.

Каждая из операций имеет две разновидности: префиксная и постфиксная. Префиксный тип операции означает, что сначала значение переменной увеличивается на 1, а затем это значение переменной передается дальше. Постфиксный тип операции означает, что сначала значение переменной передается дальше, а затем увеличивается на 1. Тут важно помнить, что инкремент и декремент хоть и находятся в правой части стейтмента и нигде не происходит изменение значения первоначальной переменной «a», но происходит неявное изменение значения этой переменной.

*int a = 8;*

*int b = ++a;*

*cout << a;// 9*

*cout << b;// 9*

*a = 8;*

*b = a++;*

*cout << a;// 9*

*cout << b;// 8*

*a = 8;*

*b = --a;*

*cout << a;// 7*

*cout << b;// 7*

*a = 8;*

*b = a--;*

*cout << a;// 7*

*cout << b;// 8*

Бинарные арифметические операции

К бинарным арифметическим операциям относятся:

\* вычитание («-»)

\* сложение («+»)

\* умножение («\*»)

\* деление («/»)

\* остаток от деления («%»)

*int a = 10;*

*int b = 7;*

*int c = a + b;// 17*

*int d = 4 - a;// -6*

*int k = b \* 5;// 35*

*int i = a / b;// 1*

*double k = 10.0 / 4;// 2.5*

*int j = a % b;// 3*

Приоритет арифметических операций

Одни операции имеют больший приоритет чем другие и поэтому выполняются вначале. Операции в порядке уменьшения приоритета:

1. ++ (инкремент), -- (декремент)

2. \* (умножение), / (деление), % (остаток от деления)

3. + (сложение), - (вычитание)

Приоритет операций следует учитывать при выполнении набора арифметических выражений. Чтобы задать свой порядок вычислений, нужно использовать ().

*int a = 8;*

*int b = 7;*

*int c = a + 5 \* ++b;*

*cout << c;// 48*

*int d = (a + 5) \* ++b;*

*cout << d;// 104*

Операции с присваиванием

Операцию присваивания можно скомбинировать с любой арифметической операцией и записать сокращенно, в виде рядом стоящих знаков присваивания и используемой арифметической операции.

*int c = 3;*

*c += 5; //8*

*c -= 2; //6*

*c /= 3; //2;*

**Логический тип данных**

В реальной жизни, перед нами очень часто возникают вопросы, на которые можно ответить однозначно: «Да» или «Нет». Яблоко является фруктом? Да! Вы видели динозавров? Нет! То есть утверждение: «Яблоко — это фрукт» - правда. А утверждение - «Вы видели динозавров» - это ложь.

Подобные [стейтменты](https://ravesli.com/urok-8-struktura-programm-s/), которые имеют только два возможных исхода: да/правда или нет/ложь, настолько распространены, что многие языки программирования добавили специальный тип для работы с ними — логический тип данных (англ. «boolean»). В языке с# для задания логического типа данных для переменной используется ключевое слово bool.

Логические переменные — это переменные, диапазон которых состоит только из двух возможный значений: true и false. Переменная типа bool занимает в памяти ровно 1 байт.

Пример кода:

*bool b1 = true;*

*bool b2 = false;*

Задание: завести две логические переменные, одной присвоить значение true, другой false. Вывести на экран значения одной перменной с помощью конкатенации строк, а другой переменной с помощью интерполяции строк.

*bool yes = true;*

*bool no = false;*

*Console.WriteLine("yes = " + yes);*

*Console.WriteLine($"no = {no}");*

**Условные выражения**

Условные выражения представляют собой некоторое условие и возвращают значение типа bool, то есть значение true (если условие истинно), или значение false (если условие ложно). К условным выражениям относятся операции сравнения и логические операции. Условные выражения используются в условных конструкциях ветвления и цикла.

### Операции сравнения

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

В С# существуют следующие операции сравнения:

>= больше или равно

> строго больше

<= меньше или равно

< строго меньше

== равно

!= не равно

Пример кода:

*int a = 4;*

*int b = 5;*

*bool result;*

*result = (a != b);*

*Console.WriteLine($”result = {result}”);*

### Логические операции

Операторысравнения используются для проверки конкретного условия и могут проверить только одно условие за один раз.

Представим ситуацию — нам нужно узнать, выиграли ли мы в лотерею, нам нужно сравнить все цифры купленного билета с выигрышными. Если в лотерее 4 цифры, то нужно выполнить 4 сравнения, все из которых должны быть true. Мы уже знаем, как сравнить отдельно 4 цифры, но как объединить все эти четыре сравнения в одно условие? С помощью логических операторов! Они позволяют проверить сразу несколько условий за раз.

В С# есть логические операции, которые позволяют объединить несколько операций сравнения в одно общее условие.

К логическим операциям относят следующие:

* && - И (конъюнкция или логическое умножение)

Результат true, если оба операнда true, в остальных случаях — false.

* || ИЛИ (дизъюнкция или логическое сложение)

Результат false, если оба операнда false, в остальных случаях — true.

* ! НЕ (отрицание)

Унарная операция. Если операнд true, то результат — false, и наоборот.

Пример:

*int a = 4;*

*int b = 5;*

*bool result;*

*result = a > b || a < b // (a больше b) логическое или (a меньше b) - true*

*result = 3 < a && a < 6 // (3 меньше a) логическое и(a меньше 6) - true*

*result = !result // логическое нет — false*

Доделываем проект

if(collision.gameObject.CompareTag("Cactus"))

{

collision.gameObject.GetComponent<AudioSource>().Play();

}

if (collision.gameObject.CompareTag("Grass"))

{

GetComponent<AudioSource>().Stop();

collision.gameObject.GetComponent<AudioSource>().Play();

}

if (collision.gameObject.CompareTag("Road"))

{

GetComponent<AudioSource>().Play();

GameObject g = GameObject.Find("ground\_grass");

if(g != null)

{

g.GetComponent<AudioSource>().Stop();

}

}

}