**C# Основы алгоритмизации**

**Типы данных**

В C# есть несколько основных типов данных, которые можно разделить на следующие категории: целочисленные, вещественные, логические, символьные и строковые типы. Ниже приведены основные типы данных с их описанием:

1. Целочисленные типы (Integral Types)

byte (8 бит): Хранит целое число от 0 до 255.

sbyte (8 бит): Хранит целое число со знаком от -128 до 127.

short (16 бит): Хранит целое число со знаком от -32,768 до 32,767.

ushort (16 бит): Хранит целое число без знака от 0 до 65,535.

int (32 бита): Хранит целое число со знаком от -2,147,483,648 до 2,147,483,647.

uint (32 бита): Хранит целое число без знака от 0 до 4,294,967,295.

long (64 бита): Хранит целое число со знаком от -9,223,372,036,854,775,808 до 9,223,372,036,854,775,807.

ulong (64 бита): Хранит целое число без знака от 0 до 18,446,744,073,709,551,615.

2. Вещественные типы (Floating-Point Types)

float (32 бита): Хранит число с плавающей запятой одинарной точности (примерно 7 знаков после запятой). Диапазон: от ±1.5 × 10^−45 до ±3.4 × 10^38.

double (64 бита): Хранит число с плавающей запятой двойной точности (примерно 15-16 знаков после запятой). Диапазон: от ±5.0 × 10^−324 до ±1.7 × 10^308.

decimal (128 бит): Хранит десятичное дробное число высокой точности (28-29 знаков после запятой). Используется в финансовых вычислениях, где важна точность.

3. Логический тип (Boolean Type)

bool (1 бит): Хранит значение логического типа, которое может быть либо true (истина), либо false (ложь).

4. Символьный тип (Character Type)

char (16 бит): Хранит один символ в кодировке Unicode. Значение находится в диапазоне от U+0000 до U+FFFF.

5. Строковый тип (String Type)

string: Хранит последовательность символов Unicode. Строки в C# являются неизменяемыми объектами (immutable), что означает, что их содержимое не может быть изменено после создания.

6. Объектный тип (Object Type)

object: Базовый тип для всех других типов данных в C#. Любой тип данных может быть приведен к типу object. Позволяет хранить данные любого типа, но требует приведения типов для использования.

7. Тип значения (Value Type)

struct: Определяет структуру, которая является типом значения, используемым для хранения связанных данных. Например, это может быть структура, представляющая точку с координатами X и Y.

8. Типы данных с указанием диапазонов (Range Types)

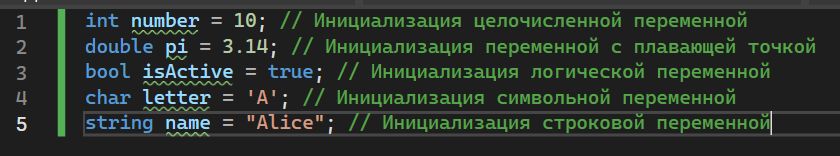
Range: Представляет диапазон значений между двумя индексами. Используется для работы с коллекциями.

Index: Представляет индекс в коллекции. Может быть использован с Range.

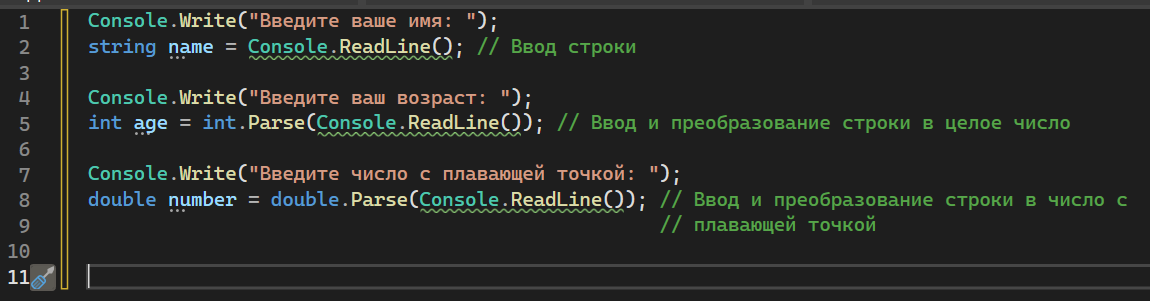
**Инициализация переменных, ввод – вывод данных**

Инициализация переменных и ввод-вывод данных выполняются с использованием определённых синтаксических конструкций. Рассмотрим основные моменты:

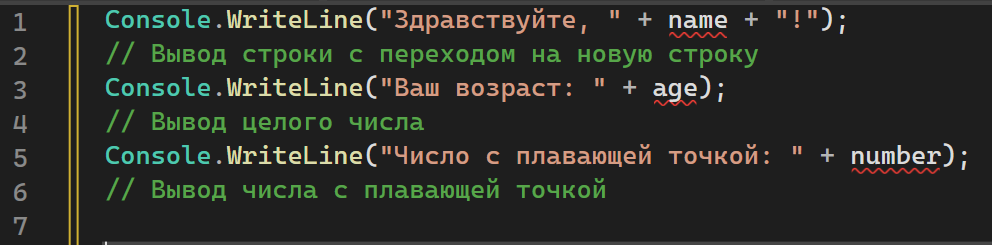
Инициализация переменных включает в себя создание переменной и присвоение ей значения.



Для ввода данных в консольном приложении обычно используется метод Console.ReadLine(). Этот метод считывает строку, введённую пользователем, и возвращает её в виде строки. Если требуется преобразовать строку в другой тип данных, используется соответствующий метод преобразования.

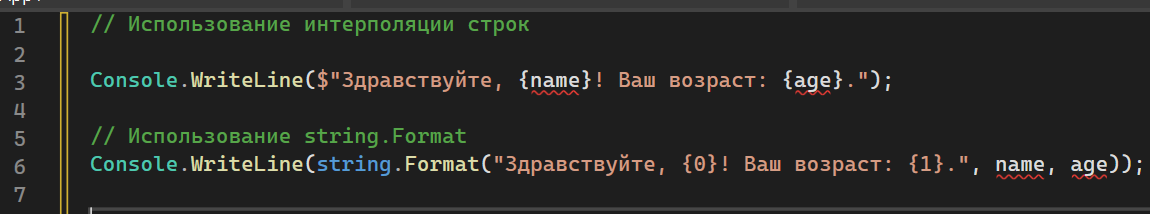


Для вывода данных в консольном приложении используется метод Console.Write() или Console.WriteLine(). Первый метод выводит текст без перехода на новую строку, а второй — с переходом на новую строку.



**Форматированный вывод данных**

Для форматированного вывода данных можно использовать интерполяцию строк или метод string.Format().



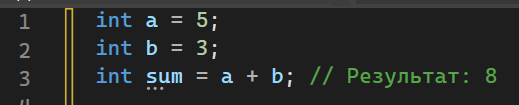
Практический пример программы: (пример 1).

**Арифметические операции**

В C# поддерживаются стандартные арифметические операции, которые можно использовать для выполнения вычислений с числами. Вот основные арифметические действия и примеры их использования:

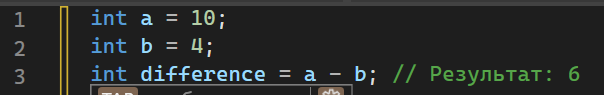
1. Сложение (+)

Операция сложения используется для суммирования двух чисел.



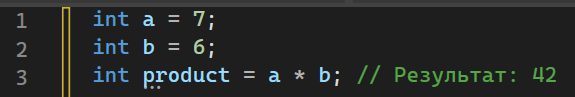
1. Вычитание (-)

Операция вычитания используется для нахождения разницы между двумя числами.



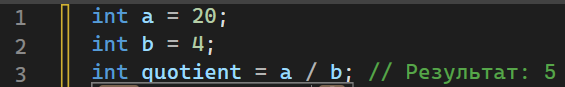
3. Умножение (\*)

Операция умножения используется для умножения двух чисел.



4. Деление (/)

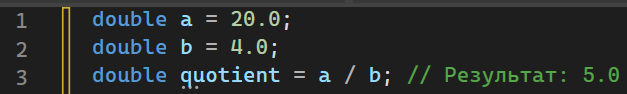
Операция деления используется для деления одного числа на другое.



**Важно:**

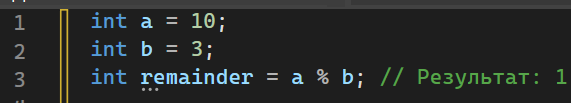
При делении целых чисел результатом также будет целое число, дробная часть будет отброшена.

Если требуется получить дробный результат, хотя бы один из операндов должен быть вещественным числом (например, double или float).



5. Остаток от деления (%)

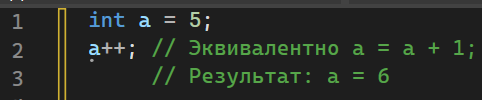
Операция получения остатка от деления возвращает остаток от деления одного числа на другое.



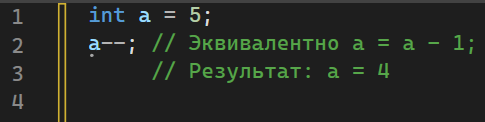
6. Инкремент (++) и декремент (--)

Эти операции используются для увеличения или уменьшения значения переменной на 1.

Пример инкремента:



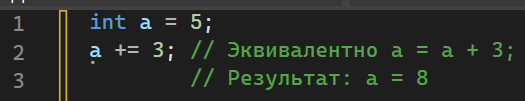
Пример декремента:



7. Комбинированные операции

C# поддерживает комбинированные операции, которые объединяют арифметическую операцию с присваиванием.

Сложение с присваиванием (+=)



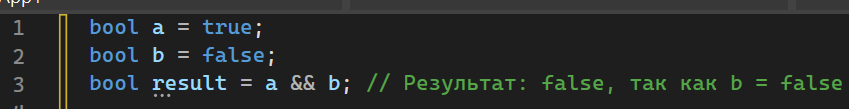
Такой же принцип подачи будет действовать и для вычитания, умножения, деления, остатка от деления с присваиванием.

Пример программы с использованием арифметических операций: (пример 2).

**Логические операции в С#**

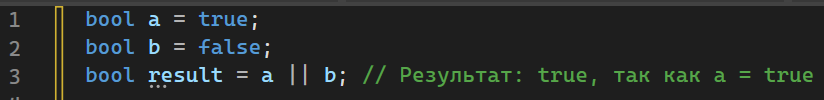
1. Логическое И (&&)

Операция возвращает true, если оба операнда равны true.



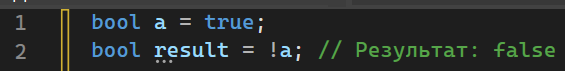
2. Логическое ИЛИ (||)

Операция возвращает true, если хотя бы один из операндов равен true.



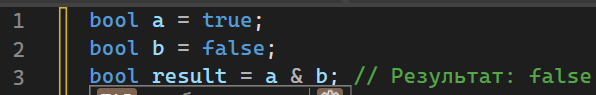
3. Логическое НЕ (!)

Операция возвращает противоположное значение операнда. Если операнд равен true, результатом будет false, и наоборот.



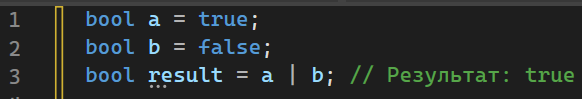
4. Побитовая операция И (&)

Операция выполняется побитово и возвращает true, если оба операнда равны true. Отличие от логического И (&&) заключается в том, что при использовании & оба операнда будут проверены, даже если первый операнд равен false.



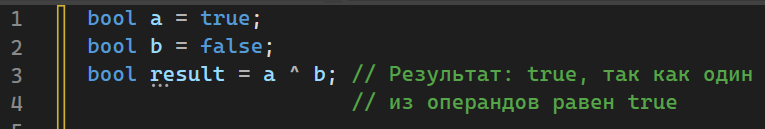
5. Побитовая операция ИЛИ (|)

Операция выполняется побитово и возвращает true, если хотя бы один из операндов равен true. Отличие от логического ИЛИ (||) заключается в том, что при использовании | оба операнда будут проверены, даже если первый операнд равен true.



6. Исключающее ИЛИ (^)

Операция возвращает true, если один из операндов равен true, но не оба одновременно.



Практический пример с логическими операциями: (пример 3).

**Алгоритмы с условиями в C#**

Условия в C# используются для управления потоком выполнения программы. Они позволяют выполнять различные действия в зависимости от выполнения определённых условий. Основные конструкции для работы с условиями включают if, else if, else, и switch. Рассмотрим примеры алгоритмов с использованием этих конструкций. (пример 4 - 7).

**Циклы в C#**

В C# циклы используются для многократного выполнения блока кода, пока выполняется определенное условие. Основные типы циклов в C# включают for, while, do-while, и foreach. Рассмотрим каждый из них с примерами. (пример 8 - 12).

**Массивы в С#**

В C# массивы используются для хранения набора значений одного типа. Они позволяют работать с группами данных под одним именем, при этом доступ к каждому элементу массива осуществляется через индекс.

Инициализация массивов

1.1. Создание массива с фиксированным размером

При создании массива нужно указать его размер (количество элементов).

**int[] numbers = new int[5]; // Массив из 5 целых чисел**

1.2. Инициализация массива значениями

Массив можно сразу инициализировать значениями, не указывая размер.

**int[] numbers = new int[] { 1, 2, 3, 4, 5 };**

1.3. Упрощенная инициализация

Можно не указывать тип при создании и инициализации массива, если тип уже известен.

**int[] numbers = { 1, 2, 3, 4, 5 };**

2. Доступ к элементам массива

Элементы массива индексируются с 0, то есть первый элемент имеет индекс 0, второй — 1 и так далее.

**int firstNumber = numbers[0]; // Доступ к первому элементу**

**numbers[1] = 10; // Изменение второго элемента**

Примеры работы с массивами: (13 - 16)

4. Многомерные массивы

Массивы в C# могут быть многомерными (например, двумерные массивы, которые можно рассматривать как матрицы). (пример 17).

5. Зубчатые массивы (массив массивов)

Зубчатый массив — это массив, элементы которого являются массивами (но каждый из них может иметь разный размер). (пример 18).

6. Заполнение массива случайными числами:

Заполнение массива случайными значениями в C# можно легко реализовать с помощью класса Random. (пример 19 - 22).

**Сортировка массивов**

Сортировка массивов в C# осуществляется различными методами, включая встроенные функции и ручные алгоритмы. Рассмотрим несколько способов сортировки массивов.

Самый простой способ отсортировать массив — использовать встроенный метод Array.Sort, также можно использовать реверсную сортировку, сортировку пузырьком или быструю сортировку (пример 23 - 27).