Кроме примитивных типов данных в языке программирования C# есть такой тип как **enum** или **перечисление**. Перечисления представляют набор логически связанных констант.

Объявление перечисления происходит с помощью оператора **enum**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | enum название\_перечисления  {  // значения перечисления  значение1,  значение2,  .......  значениеN  } |

После оператора enum идет название перечисления. И затем в фигурных скобках через запятую перечисляются константы перечисления.

Определим простейшее перечисление:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | enum DayTime  {  Morning,  Afternoon,  Evening,  Night  } |

Здесь определено перечисление DayTime, которое имеет четыре значения: Morning, Afternoon, Evening и Night

Каждое перечисление фактически определяет новый тип данных, с помощью которых мы также, как и с помощью любого другого типа, можем определять переменные, константы, параметры методов и т.д. В качестве значения переменной, константы и параметра метода, которые представляют перечисление, должна выступать одна из констант этого перечисления, например:

|  |  |
| --- | --- |
| 1 | const DayTime dayTime = DayTime.Morning; |

Далее в программе мы можем использовать подобные переменные/константы/параметры как и любые другие:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | DayTime dayTime = DayTime.Morning;    if(dayTime == DayTime.Morning)  Console.WriteLine("Доброе утро");  else  Console.WriteLine("Привет");    enum DayTime  {  Morning,  Afternoon,  Evening,  Night  } |

### Хранение состояния

Зачастую переменная перечисления выступает в качестве хранилища состояния, в зависимости от которого производятся некоторые действия:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | DayTime now = DayTime.Evening;    PrintMessage(now); // Добрый вечер  PrintMessage(DayTime.Afternoon); // Добрый день  PrintMessage(DayTime.Night); // Доброй ночи    void PrintMessage(DayTime dayTime)  {  switch (dayTime)  {  case DayTime.Morning:  Console.WriteLine("Доброе утро");  break;  case DayTime.Afternoon:  Console.WriteLine("Добрый день");  break;  case DayTime.Evening:  Console.WriteLine("Добрый вечер");  break;  case DayTime.Night:  Console.WriteLine("Доброй ночи");  break;  }  }  enum DayTime  {  Morning,  Afternoon,  Evening,  Night  } |

Здесь метод PrintMessage() в качестве параметра принимает значение типа перечисления DayTime и зависимости от этого значения выводит определенное приведение.

Другой пример:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | DoOperation(10, 5, Operation.Add); // 15  DoOperation(10, 5, Operation.Subtract); // 5  DoOperation(10, 5, Operation.Multiply); // 50  DoOperation(10, 5, Operation.Divide); // 2    void DoOperation(double x, double y, Operation op)  {  double result = op switch  {  Operation.Add => x + y,  Operation.Subtract => x - y,  Operation.Multiply => x \* y,  Operation.Divide => x / y  };  Console.WriteLine(result);  }  enum Operation  {  Add,  Subtract,  Multiply,  Divide  } |

Здесь определено перечисление Operation, которое представляет арифметические операции. Каждый тип операций определен в виде одной из констант перечисления. И также определен метод DoOperation(), который в качестве параметров принимает два числа и тип операции в виде константы перечисления и в зависимости от этого типа возвращает из конструкции switch результат определенной операции.

### Тип и значения констант перечисления

Константы перечисления могут иметь тип. Тип указывается после названия перечисления через двоеточие:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | enum Time : byte  {  Morning,  Afternoon,  Evening,  Night  } |

Тип перечисления обязательно должен представлять целочисленный тип (byte, sbyte, short, ushort, int, uint, long, ulong). Если тип явным образом не указан, то по умолчанию используется тип **int**.

Тип влияет на значения, которые могут иметь константы. По умолчанию каждому элементу перечисления присваивается целочисленное значение, причем первый элемент будет иметь значение 0, второй - 1 и так далее. Например, возьмем выше определенное DayTime:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | DayTime now = DayTime.Morning;    Console.WriteLine((int) now); // 0  Console.WriteLine((int) DayTime.Night); // 3    enum DayTime  {  Morning,  Afternoon,  Evening,  Night  } |

Мы можем использовать операцию приведения, чтобы получить целочисленное значение константы перечисления:

|  |  |
| --- | --- |
| 1 | (int) DayTime.Night // 3 |

В то же время, несмотря на то, что каждая константа сопоставляется с определенным числом, мы НЕ можем присвоить ей числовое значение:

|  |  |
| --- | --- |
| 1 | DayTime now = 2; // ! Ошибка |

Можно также явным образом указать значения элементов, либо указав значение первого элемента:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | enum DayTime  {  Morning = 3, // каждый следующий элемент по умолчанию увеличивается на единицу  Afternoon, // этот элемент равен 4  Evening, // 5  Night // 6  } |

Но можно и для всех элементов явным образом указать значения:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | enum DayTime  {  Morning = 2,  Afternoon = 4,  Evening = 8,  Night = 16  } |

При этом константы перечисления могут иметь одинаковые значения, либо даже можно присваивать одной константе значение другой константы:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | enum DayTime  {  Morning = 1,  Afternoon = Morning,  Evening = 2,  Night = 2  } |