Шпаргалка для перевода конструкций PascalABC.NET на C#

Простейшая программа

PascalABC.NET

```
begin
  Println('Hello, World!');
end.
```

C#

```
using System;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

```
##

Println('Hello, World!');
```

```
using System;

Console.WriteLine("Hello World!");
```

Комментарии в коде

PascalABC.NET

```
// Однострочный комментарий до конца строки
```

```
{ Многострочный
комментарий }
```

```
(* Многострочный
комментарий *)
```

```
/// <summary>
/// Документирующий комментарий к функции
/// </summary>
function SomeFunction(param1: integer; param2: boolean);
```

C#

```
// Однострочный комментарий до конца строки
```

```
/*
 Многострочный комментарий
 */
```

```
/// <summary>
/// Документирующий комментарий к функции
/// </summary>
/// <param name="param1">Описание параметра 1</param>
/// <param name="param2">Описание параметра 2</param>
/// <returns>Описание возвращаемого значения</returns>
string SomeFunction(int param1, bool param2){}
```

Стандартные типы

PascalABC.NET

```
integer
byte
real/double
boolean
char
string
```

C#

```
int
byte
double
bool
char
string
```

Арифметические операции

PascalABC.NET

```
x + y
x - y
x * y
```

```
var x: integer := 7;
var y: integer := 2;
Println(x div y); // 3

Println(x/y);     // 3.5


Println(x mod y); // 1
```

C#

```
x + y
x - y
x * y
```

```
int x = 7;
int y = 2;
Console.WriteLine(x / y);       // 3

Console.WriteLine((double)x/y); // 3.5
Console.WriteLine(x * 1.0 / y); // 3.5

Console.WriteLine(x % y);       // 1
```

Логические операции

PascalABC.NET

```
x and y
x or y
not x
x xor y
x in a
x not in a
```

C#

```
x && y
x || y
!x
x ^ y
```

Побитовые операции

PascalABC.NET	C#
x **and** y x **or** y **not** x x **xor** y x **shl** n x **shr** n	x & y x | y ~ x x ^ y x << n x >> n

Операции сравнения

PascalABC.NET	C#
x < y (x > y) x <= y (x >= y) x = y x <> y	x < y (x > y) x <= y (x >= y) x == y x != y
a **in** -5..5 a **not in** -5..5	-5 <= a && a <= 5

Операция присваивания

PascalABC.NET	C#
var a := 10; // автовывод типа	var a = 10; // *автовывод типа*
var a: real := 10; // явное указание типа	double a = 10; // *явное указание типа*
var (a, b) := (1, 2.7); // автовывод типа **var** a := 1; **var** b := 2.7;	var (a, b) = (1, 2.7); // *автовывод типа* (int a, double b) = (1, 2.7); // *явное указание типа*
var (a, b) := (1, 2); (a, b) := (b, a); // кортежное присваивание	var (a, b) = (1, 2); (a, b) = (b, a);
var t : (integer, integer) := (1,2);	(int, int) t = (1, 2);
a += 2; a -= 2; a *= 2; a /= 2; // не применим к типу integer a := a **mod** 2; a := a **div** 2;	a += 2; a -= 2; a *= 2; a /= 2.0; // *не применим к типу int* a %= 2; a /= 2;
var t := a; a += 1; b := t; a += 1; b := a;	b = a++; // *a-- для декремента* b = ++a; // *--a для декремента*
var a: integer; **var** x, y: real; Println(a); // 0	int a; double x, y; Console.WriteLine(a); // *Ошибка [CS0165] Использование локальной переменной "a", которой не присвоено значение.*
// В PascalABC.Net нет множественного присваивания	int a, b; a = b = 666; a += b += 1;
const PI = 3.14;	const double PI = 3.14;

Чтение с консоли

PascalABC.NET	C#
var s := ReadString; **var** n := ReadlnInteger; **var** d := ReadlnReal;	using System; string s = Console.*ReadLine*(); int n = int.*Parse*(Console.*ReadLine*()); int n2 = Convert.*ToInt32*(Console.*ReadLine*()); double d = double.*Parse*(Console.*ReadLine*()); double d2 = Convert.*ToDouble*(Console.*ReadLine*());
var c := ReadChar;	char c = Convert.*ToChar*(Console.*Read*());

```pascal
var n1 := ReadInteger; // прочитается и через пробел, и на
var n2 := ReadInteger; // новой строке
```

```csharp
// В C# нет возможности читать числа через пробел (не
считая связки Console.ReadLine() + string.Split())
```

```pascal
var (y, z) := ReadReal2();
```

```csharp
// В C# нет возможности читать несколько чисел с
// распаковкой в кортеж. Похожая (неэффективная) операция:
var ints = Console
    .ReadLine()
    .Split()
    .Select(int.Parse);
```

```pascal
var x := ReadlnReal('Введите x:');
```

```csharp
Console.WriteLine("Введите x:");
double x = Convert.ToDouble(Console.ReadLine());
```

Вывод в консоль

PascalABC.NET

```pascal
Println(7);
Println(1,2,3);

// Вывод:
// 7
// 1 2 3
```

C#

```csharp
Console.WriteLine(7);
/* В C# нет возможности передать несколько параметров в
Console.WriteLine(). Вместо этого можно конкатенировать
строку на ходу или воспользоваться интерполированными
строками */
Console.WriteLine(1 + " " + 2 + " " + 3);

// Вывод:
// 7
// 1 2 3
```

```pascal
Print(7);
Print(1,2,3);

// Вывод:
// 7 1 2 3
```

```csharp
Console.Write(7);
/* В C# нет возможности передать несколько параметров в
Console.Write(). Вместо этого можно конкатенировать строку
на ходу или воспользоваться интерполированными строками */
Console.Write(1 + " " + 2 + " " + 3);

// Вывод:
// 71 2 3
```

```pascal
var name := 'Иван';
var surname := 'Иванов';
Print($'Добро пожаловать, {surname} {name}!')

// Вывод:
// Добро пожаловать, Иванов Иван!
```

```csharp
var name = "Иван";
var surname = "Иванов";
Console.WriteLine($"Добро пожаловать, {surname} {name}!");

// Вывод:
// Добро пожаловать, Иванов Иван!
```

```pascal
var name := 'John';
var salary := 1470.55;

Println($'Name:{name, 7}, salary:{salary, 0:f1}');

// Name:   John, salary:1470,6
```

```csharp
string name = "John";
double salary = 1470.55;

Console.WriteLine($"Name:{name, 7}, salary:{salary,
0:f1}");
// Name:   John, salary:1470,5
```

```pascal
// Print и Write выводят составные данные
Print((1..10).Batch(3).Numerate.ToHashSet)

// Вывод:
// {(1,[1,2,3]),(2,[4,5,6]),(3,[7,8,9]),(4,[10])}
// {} - множество, [] - массив, () - класс, кортеж, запись
```

```csharp
// В C# выводится только тип внешнего контейнера
```

Условный оператор

PascalABC.NET

```pascal
if x < 4 then
 Println('x is less than 4');
```

C#

```csharp
if (x < 4)
    Console.WriteLine("x is less than 4");
```

```pascal
if x < 4 then
  Println('x is less than 4')
else if x > 4 then
  Println('x is greater than 4');
else
  Println('x equals 4')
```

```csharp
if (x < 4)
    Console.WriteLine("x is less than 4");
else if (x > 4)
    Console.WriteLine("x is greater than 4");
else
    Console.WriteLine("x equals 4");
```

```pascal
if x < 4 then
begin
  x := 10;
  Println('x was less than 4');
end;
```

```csharp
if (x < 4)
{
    x = 10;
    Console.WriteLine("x was less than 4");
}
```

Условная операция

PascalABC.NET

```
var x := 7;
var desc := x = 7 ? 'seven' : 'not seven';

// Другой вид условной операции
var x := 7;
var desc := if x = 7 then 'seven' else 'not seven';
```

C#

```
var x = 7;
var desc = x == 7 ? "seven" : "not seven";
```

Оператор выбора

PascalABC.NET

```
var dayOfWeek := ReadInteger;

case dayOfWeek of
  1:    Println('Понедельник :(');
  2..5: Println('Будни');
  6, 7: Println('Выходной день');
  else  Println('Такого дня не бывает');
end;
```

C#

```
using static System.Console;
...

int dayOfWeek = int.Parse(ReadLine());
switch (dayOfWeek)
{
    case 1: WriteLine("Понедельник :(");
        break;
    case >= 2 and < 6: WriteLine("Будни"); // C# 9.0
        break;
    case 6 or 7: WriteLine("Выходной день"); // C# 9.0
        break;
    default: WriteLine("Такого дня не бывает");
        break;
}
```

```
var dayOfWeek := ReadInteger;
var happiness := 0;

case dayOfWeek of
  1: begin
      Println('Понедельник :(');
      happiness -= 100;
    end;
  2..5: begin
      Println('Будни');
      happiness -= 1;
    end;
  6, 7: begin
      Println('Выходной день');
      happiness += 1000;
    end;
else Println('Такого дня не бывает');
end;
```

```
using static System.Console;
...

int dayOfWeek = int.Parse(ReadLine());
int happiness = 0;
switch (dayOfWeek)
{
    case 1: {WriteLine("Понедельник :(");}
        happiness -= 100;
        break;
    case >= 2 and < 6: WriteLine("Будни"); // C# 9.0
        happiness--;
        break;
    case 6 or 7: WriteLine("Выходной день"); // C# 9.0
        happiness += 1000;
        break;
    default: WriteLine("Такого дня не бывает");
        break;
}
```

```
// В PascalABC.Net нет возможности использовать оператор
case в качестве выражения.
```

```
int value = int.Parse(Console.ReadLine());
string description = value switch
{
    <= 0 => "Less than or equal to 0",
    > 0 and < 10 => "More than 0 but less than 10",
    _ => "More than or equal to 10"
};
```

Циклы for и loop

PascalABC.NET

```
for var i := 1 to 5 do
  Print(i);
// Вывод:
// 1 2 3 4 5
```

C#

```
for (int i = 1; i <= 5; i++)
    Console.Write($"{i} ");
// Вывод:
// 1 2 3 4 5
```

```
for var i := 1 to 3 do
begin
  Print(i);
  Print(i * 2);
end;
// Вывод:
// 1 2 2 4 3 6
```

```
for (int i = 1; i < 4; i++)
{
    Console.Write($"{i} ");
    Console.Write($"{i * 2} ");
}
// Вывод:
// 1 2 2 4 3 6
```

```
for var i := 5 downto 1 do
  Print(i);
// Вывод:
// 5 4 3 2 1
```

```
for (int i = 5; i >= 1; i--)
    Console.Write($"{i} ");
// Вывод:
// 5 4 3 2 1
```

```pascal
for var i := 1 to 10 step 2 do
  Print(i);
// Вывод:
// 1 3 5 7 9
```

```csharp
for (int i = 1; i <= 10; i += 2)
    Console.Write($"{i} ");
// Вывод:
// 1 3 5 7 9
```

```pascal
loop 5 do
  Print(1);
// Вывод:
// 1 1 1 1 1
```

```csharp
// В C# нет встроенного loop. Он моделируется циклом for.
for (int i = 0; i < 5; i ++)
    Console.Write("1 ");
```

Цикл foreach

PascalABC.NET

```pascal
var a: array of integer := (1, 2, 3);
foreach var x in a do
  Print(x);
// Вывод:
// 1 2 3
```

C#

```csharp
int[] a = {1, 2, 3};
foreach (var x in a)
    Console.Write($"{x} ");
// Вывод:
// 1 2 3
```

```pascal
var a: array of integer := (1, 2, 3);
foreach var x in a index i do
  Print(x, i);
// Вывод:
// 1 0 2 1 3 2
```

```csharp
// В C# нет встроенного foreach с индексом. Его можно
// промоделировать с помощью вспомогательной переменной.
```

Цикл while / repeat

PascalABC.NET

```pascal
while x > 0 do
  x -= 1;
```

C#

```csharp
while (x > 0)
    x--;
```

```pascal
while x > 0 do
begin
  x -= 1;
  Println(x);
end;
```

```csharp
while (x > 0)
{
    x--;
    Console.WriteLine(x);
}
```

```pascal
repeat
  x -= 1;
  Println(x);
until x = 0;
```

```csharp
do
{
    x--;
    Console.WriteLine(x);
} while (x > 0); // Инверсии условия нет!
```

Операторы break, continue, goto

PascalABC.NET

```pascal
var prod := 1;
loop 10 do
begin
  var x := Random(-10, 10);
  if x = 0 then
    break;
  prod *= x;
end;
```

C#

```csharp
int prod = 1;
Random r = new Random();
for (int i = 0; i < 10; i++)
{
    int x = r.Next(-10, 10);
    if (x == 0)
        break;
    prod *= x;
}
```

```pascal
var arr := |1, -4, 3, 0, 2, 7|;
var prod := 1;
foreach var elem in arr do
begin
  if elem = 0 then
    continue;
  prod *= elem;
end;
```

```csharp
var array = new [] { 1, -4, 3, 0, 2, 7 };
int prod = 1;
foreach (var elem in array)
{
    if (elem == 0)
        continue;
    prod *= elem;
}
```

```pascal
// Описание метки перед блоком begin/end.
label c1;

/// Основная программа
...
var matr: array [,] of integer;
... // заполнение матрицы

var found := False;
for var i := 0 to matr.RowCount - 1 do
  for var j := 0 to matr.ColCount - 1 do
    if mat[i, j] = 10 then
    begin
      found := True;
```

```csharp
int[,] matr;
... // заполнение матрицы

bool found = false;
for (int i = 0; i < matr.GetLength(0); i++)
    for (int j = 0; j < matr.GetLength(1); j++)
        if (matr[i, j] == k)
        {
            found = true;
            goto c1;
        }
c1: Console.WriteLine(found);
```

```
      goto c1;
    end;
c1: Println(found);
```

Стандартные методы элементарных типов

PascalABC.NET

```
// integer
var i: integer;
i.Clamp(min,max)
i.Divs(2)
i.DivsAll(2,3)
i.DivsAny(2,3)
i.IsEven
i.IsOdd
i.Print
i.Println
```

C#

```
// В C# эти методы отсутствуют
```

PascalABC.NET

```
// real
var r: real;
r.Clamp(min,max)
r.Round
r.Trunc
r.Sqrt
r.Print
r.Println
```

C#

```
// В C# эти методы отсутствуют
```

PascalABC.NET

```
// char
var c: char;
c.Code
c.IsDigit
c.ToDigit
c.Print
c.Println
```

C#

```
// В C# эти методы отсутствуют
```

Стандартные математические функции

PascalABC.NET

```
var (a, b) := (1, 2);
Swap(a, b); // a = 2, b = 1
```

C#

```
// В C# нет стандартной функции Swap
(a, b) = (b, a); // сильно медленно
```

PascalABC.NET

```
Min(1, 2, 3, 4);
Max(x, y);
```

C#

```
Math.Min(1, 2); // Только 2 значения
Math.Max(1, 2); // Только 2 значения
```

PascalABC.NET

```
Round(3.57832);     // 4
Round(3.57832, 2);  // 3.58
```

C#

```
Math.Round(3.57832);    // 4
Math.Round(3.57832, 2); // 3.58
```

PascalABC.NET

```
Sqrt(16);     //  4, тип - real.

4 ** 2;       // 16, тип - real
4bi ** 2;     // 16, тип - BigInteger

16 ** 0.25    //  2, тип - real.
```

C#

```
Math.Sqrt(16);      //  4, тип - double.

Math.Pow(4, 2);     // 16, тип - double.


Math.Pow(16, 0.25); //  2, тип - double.
```

PascalABC.NET

```
Random(2,5); //  [2, 5]
Random;      //  [0.0, 1.0)
```

C#

```
Random r = new Random();
var i = r.Next(2, 5); // [2, 5)
var d = r.NextDouble();
```

Перечислимый тип

PascalABC.NET

```
type Season = (Winter, Spring, Summer, Autumn);
```

C#

```
enum Season { Winter, Spring, Summer, Autumn }
```

Процедурный тип / лямбда-функции

PascalABC.NET

```
// Ничего подключать не нужно

var f1: () -> integer;
var f2: integer -> integer;
var f3: (integer, integer) -> real;

var f4: () -> ();
var f5: integer -> ();
var f6: (integer, real) -> ();

var f7: Predicate<string>;
```

C#

```
using System;    // Обязательно нужно подключать

Func<int> f;
Func<int, int> f2;
Func<int, int, double> f3;

Action f4;
Action<int> f5;
Action<int, double> f6;

Predicate<string> f7; // Аналогично Func<string, bool>
```

```pascal
procedure ForAll(a: array of real; f: real->real);

/// Основная программа
...
ForAll(a, x -> x * 2);
```

```c#
void ForAll(double[] a, Func<double, double> f) {}

/// Основная программа
...
ForAll(a, x => x * 2);
```

Модули

PascalABC.NET

```pascal
unit SomeUnit; // Модуль в файле SomeUnit.pas

interface

function SumSquares(x,y): integer;

implementation

function Sqr(x: integer) := x * x;
function SumSquares (x,y): integer := Sqr(x) + Sqr(y);

end.
```

```pascal
unit SomeUnit; // Модуль упрощенной структуры – всё видимо

function Sqr(x: integer) := x * x;
function SumSquares(x,y: integer) := Sqr(x) + Sqr(y);

end.
```

```pascal
// Подключение модуля в основной программе
uses SomeUnit;
...

// Доступны только имена из секции интерфейса
// К исполняемому файлу прилинковываются только функции
модуля, вызываемые основной программой
```

C#

```c#
// В C# нет модулей.
```

Библиотеки

PascalABC.NET

```pascal
library SomeLib; // Библиотека в файле SomeLib.pas

interface

function SumSquares(x,y): integer;

implementation

function Sqr(x: integer) := x * x;
function SumSquares(x,y: integer) := Sqr(x) + Sqr(y);

end.
// Имеются также библиотеки упрощенной структуры

// Подключение библиотеки в основной программе –
// директивой компилятора
{$reference SomeLib.dll}
...
```

C#

```c#
// В C# библиотеки подключаются в окне проекта или
параметром консольного компилятора


// Основная программа:
using SomeLib; // Пространство имен библиотеки
...

// Библиотека должна лежать в папке приложения при
компиляции и при запуске
```

Процедуры и функции

PascalABC.NET

```pascal
// Процедура без параметров
procedure EmptyProc;
begin
  Println('Hello');
  Println('world');
end;

/// Основная программа
...
EmptyProc;    // Скобки можно опускать
```

```pascal
// Сокращенная форма записи процедуры без параметров.
procedure ShortEmptyProc := Println('Hello');
```

C#

```c#
// Метод без возвращаемого значения и параметров
void EmptyMethod()
{
    Console.WriteLine("Hello");
    Console.WriteLine("world");
}

/// Основная программа
...
EmptyMethod();    // Скобки обязательны
```

```c#
// Сокращённая запись метода без возвращаемого значения
void EmptyMethod() => Console.WriteLine("Hello");
```

```pascal
// Функция без параметров.
function EmptyFunc : integer;
begin
  Result := 0;
  for var i := 1 to 10 do
    Result += i;
end;
```

```c#
// Метод без входных параметров с возвращаемым целым
int EmptyIntFunc()
{
    var res = 0; // Моделирование Result из PascalABC.Net
    for (int i = 1; i <= 10; i++)
        res += i;
    return res;
}
```

```pascal
// Сокращенная запись функции (с явным указанием типа)
function ShortRegularFuncNoAuto : real := 666;

// Сокращенная форма записи функции (с автовыводом типа)
function ShortRegularFuncAuto := 666;
```

```c#
// Сокращённая запись метода с возвращаемым значением
double ShortFunc() => 666;

// В C# нет автовывода типа возвращаемого значения функции
```

```pascal
// Возвращение кортежа
function TupleFunc: (real, real) := (666, 777);
```

```c#
// Метод, возвращающий кортеж
(double, double) TupleFunction() => (666, 777);
```

```pascal
// Процедура с несколькими входными параметрами
procedure DiffInputProc(n, x: integer, d: real);


// Сокращённая запись процедуры с 2 параметрами
procedure ShortRegularFunc (a, b: real) := Print(a + b);
```

```c#
// Метод с несколькими входными параметрами
void DiffInputMethod(int n, double d, int x)
{ /* Some code */ }

// Сокращённая запись метода с 2 входными параметрами
void Method(double a, double b) => Console.Write(a + b);
```

```pascal
// Процедура с выходным параметром
procedure InputOutputProc (n: integer; var p : real);
begin
  Print(p);
  p := 1;
  for var i := 2 to n do
    p *= i;
end;


/// Основная программа
...
var p : real;
InputOutputProc(7, p);
```

```c#
// Метод с выходным параметром
void InputOutputMethod(int n, out double p)
{
    Console.Write(p); // Ошибка компиляции, p нужно
                      // сначала инициализировать!
    p = 1;
    for (int i = 2; i <= n; i++)
        p *= i;
}

/// Основная программа
...
double p;
InputOutputMethod(7, out p);
InputOutputMethod(7, out var p2);
```

```pascal
// Процедура с входно-выходным параметром
procedure InputOutputProc (n: integer; var p : real);
begin
  p += n;
end;

/// Основная программа
...
var p : real;
InputOutputProc(7, p);
```

```c#
// Метод с параметром по ссылке
void InputOutputMethod(int n, ref double p)
{
    p += n;
}


/// Основная программа
...
double p;
InputOutputMethod(7, ref p); // Ошибка компиляции, p нужно
                             // сначала инициализировать!
```

```pascal
// Переменное число параметров
function Sum(params a: array of integer): integer;
```

```c#
// Переменное число параметров
int Sum(params int[] a)
```

```pascal
// Обобщенные процедуры и функции
procedure Swap<T>(var a,b: T);
begin
  var x := a;
  a := b;
  b := x;
end;
```

```c#
// Обобщенные процедуры и функции
void Swap<T>(ref T a, ref T b)
{
    var x = a;
    a = b;
    b = x;
}
```

Unit-тесты

PascalABC.NET

```pascal
uses NUnitABC;
// Для тестирования создаётся отдельный файл и
// используется uses для подключения NUnit.
```

```c#
global using NUnit.Framework;  // C# 10.0
// Для тестирования создаётся отдельный проект в решении,
// к нему подключается тестируемый проект и используется
// global using для подключения NUnit.
```

```pascal
[Test]
procedure Test1;
begin
  Assert.AreEqual(2 * 2, 5);
  Assert.That(2 * 2, &Is.EqualTo(5));
end;
```

```c#
[Test]
public void Test1()
{
    Assert.AreEqual(5, 2*2); //Классический стиль
    Assert.That(2*2, Is.EqualTo(5)); //Рекомендуемый стиль
}
```

```pascal
[TestCase(12, 3, 4)]
[TestCase(12, 2, 6)]
[TestCase(12, 4, 3)]
procedure DivideTest(n, d, q: integer);
begin
  Assert.AreEqual(q, n div d);
end;
```

```pascal
[TestCase(12, 3, ExpectedResult = 4)]
[TestCase(12, 2, ExpectedResult = 6)]
[TestCase(12, 4, ExpectedResult = 3)]
function DivideTest(n, d: integer): integer;
begin
  Result := n div d;
end;
```

```csharp
[TestCase(12, 3, 4)]
[TestCase(12, 2, 6)]
[TestCase(12, 4, 3)]
public void DivideTest(int n, int d, int q)
{
    Assert.That(n / d, Is.EqualTo(q));
}
```

```csharp
[TestCase(12, 3, ExpectedResult = 4)]
[TestCase(12, 2, ExpectedResult = 6)]
[TestCase(12, 4, ExpectedResult = 3)]
public int DivideTest(int n, int d)
{
    return n / d;
}
```

Одномерные массивы

PascalABC.NET

```pascal
var a: array of integer := new integer[5];

var b := |1, 2, 3, 4, 5|;


var c := Arr(1, 2, 3, 4, 5);
var d := Arr(1..5);
```

```pascal
var a1 := ArrFill(777, 5);
var a2 := ArrRandomInteger(10,1,100);
var a3 := ReadArrInteger(10);
var a4 := ArrGen(10, x -> x ** 2);
```

```pascal
var a := |1, 2, 3, 4, 5|;

// Стандартный цикл обработки элементов массива
for var i := 0 to a.Length - 1 do
  // Обработка элемента

foreach var x in a do
  // Обработка элемента
```

```pascal
// Срезы
var a := Arr(0..9);

a[:];       // [0,1,2,3,4,5,6,7,8,9]
a[2:5];     // [2,3,4]
a[2:];      // [2,3,4,5,6,7,8,9]
a[:5];      // [0,1,2,3,4]
a[^1]       // 9

a?[8:15];      // [8,9]

// Имеются срезы с шагом
a[2:9:2];   // [2,4,6,8]
a[8:1:-2];  // [8,6,4,2]
a[::3];     // [0,3,6,9]
a[::-1];    // [9,8,7,6,5,4,3,2,1,0]

// Имеются срезы на запись
a[1:3] := a[4:6];
```

```pascal
// Операции с массивами
var a := |1, 2, 3|;
var b := |4, 5, 6|;

Println(a + b); // [1,2,3,4,5,6]
Println(a*3);   // [1,2,3,1,2,3,1,2,3]
Println(2(a*2 + b*3));
Println(3 in a); // True
```

```pascal
// Присваивание и сравнение массивов
var a := |1, 2, 3|;
var a1 := a; // Присваивается ссылка
var a2 := Copy(a); // Создается копия
var b1 := a = a1; // Сравниваются ссылки - True
var b2 := a = a2; // Сравниваются ссылки - False
var b3 := a.ArrEqual(a2); // Сравниваются значения - True
```

C#

```csharp
int[] a = new int[5];

var b = new [] { 1, 2, 3, 4, 5 };
int[] b2 = { 1, 2, 3, 4, 5 };

// В C# нет аналога Arr().
```

```csharp
/* В C# нет аналогов генераторов массивов. Все эти функции
можно промоделировать использованием цикла for по всему
массиву и присваиванием необходимых значений */
```

```csharp
int[] a = {1, 2, 3, 4, 5};

// Стандартный цикл обработки элементов массива
for (int i = 0; i < a.Length; i++)
    // Обработка элемента

foreach (var x in a)
    // Обработка элемента
```

```csharp
// Срезы
int[] a = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

a[..];       // [0,1,2,3,4,5,6,7,8,9]
a[2..5];     // [2,3,4]
a[2..];      // [2,3,4,5,6,7,8,9]
a[..5];      // [0,1,2,3,4]
a[^1];       // 9

// В C# нет безопасных срезов

// В срезах C# нельзя задавать шаг




// Срезы на запись в C# отсутствуют
```

```csharp
// В C# отсутствуют операции с массивами
```

```csharp
// Присваивание и сравнение массивов
int[] a = {1, 2, 3};
int[] a1 = a; // Присваивается ссылка

var b1 = a == a1; // Сравниваются ссылки
```

```pascal
var a := |1, 2, 3, 4, 5|;

SetLength(a, 10);

// Методы расширения массивов
// Print, Cartesian, Permutations, Combinations, Fill,
// Find, ConvertAll, FindAll, FindIndex,
// IndexMin, IndexOf, Indices, Transform и др.
```

```csharp
int[] a = {1, 2, 3, 4, 5};

Array.Resize(ref a, 10);

/* Большинство из этих методов отсутствует в C# как методы
расширения. Некоторые, тем не менее, можно найти в классе
Array. Например: Sort, Clear, Find, Fill, FindAll,
FindIndex, IndexOf, FindLast, LastIndexOf. */
```

```pascal
// Комбинаторные методы
var a := Arr(1,3,5,7);
a.Permutations    // все перестановки
a.Cartesian(m)    // m-тая декартова степень
a.Permutations(m) // все частичные перестановки из n
элементов по m
a.Combinations(2) // все сочетания из n элементов по m
```

```csharp
// В C# у массивов комбинаторные методы отсутствуют
```

Двумерные массивы

PascalABC.NET **C#**

```pascal
var m : array [,] of real := new real[4, 3];

var m2 := new real[4,3] ((1,2,3),(4,5,6),(7,8,9),(0,1,2));

m := Matr(2,2,1,2,3,4);

4 in m; // True
```

```csharp
int[,] m = new int[4, 3];

int[,] m = {{1,2,3}, {4,5,6}, {7,8,9}, {10,11,12}};

// В C# нет аналога Matr()


// В C# нет аналога in для матриц
```

```pascal
m := Matr(|1,2,3,4|,|5,6,7,8|,|9,10,11,12|);
m := MatrGen(3,4,(i,j)->i+j+0.5);
m := MatrRandomInteger(3,4);
m := ReadMatrInteger(4, 3);
m := MatrByCol(m.Rows);
```

```csharp
/* В C# нет аналогов генераторов матриц. Все эти функции
можно промоделировать использованием вложенных циклов for
и присваиванием необходимых значений всем элементам
матрицы. */
```

```pascal
for var i := 0 to m.RowCount - 1 do
    for var j := 0 to m.ColCount - 1 do
      // Обработка элемента
```

```csharp
for (int i = 0; i < m.GetLength(0); i++)
    for (int j = 0; j < m.GetLength(1); j++)
        // Обработка элемента
```

```pascal
// Методы расширения матриц
// Row/Col, RowCount/ColCount, Rows/Cols
```

```csharp
/* В C# нет практически никаких средств для удобной работы
с матрицами. Все действия производятся с использованием
вложенных циклов и срезов. */
```

Списки

PascalABC.NET **C#**

```pascal
// Описание
var L: List<integer>;
L := new List<integer>;

// Генераторы списков
var L := Lst(1,3,5,7,9);

// Методы списков
L.Add(777);
L.Insert(0,666);
L.RemoveAt(2);

// Операции со списками
Print(3 in L); // True
var L1 := Lst(1,3,5);
var L2 := Lst (2,4,6);
L1 += 666; // L.Add(666)
L1 := L1 + L2;
L1 := L1 * 3;
```

```csharp
// Описание
using System.Collections.Generic;
List<int> L = new List<int>();

// Генераторы списков в C# отсутствуют
var L = new List<int>{1,3,5,7,9};

// Методы списков в C# совпадают с PascalABC.NET
L.Add(777);
L.Insert(0,666);
L.RemoveAt(2);

// Операции со списками в C# отсутствуют
```

Множества

PascalABC.NET **C#**

```pascal
// Описание
var hs := new HashSet<integer>;
var ss := new SortedSet<integer>;


// Короткие функции для множеств
var hs := HSet(1,3,5,7,9);
var ss := SSet(1,3,5,7,9);

// Методы множеств
```

```csharp
// Описание
using System.Collections.Generic;
HashSet<int> hs = new HashSet<int>();
var ss = new SortedSet<int>();

// Короткие функции для множеств в C# отсутствуют
var hs = new HashSet<int>{1,3,5,7,9};
var ss = new SortedSet<int>{1,3,5,7,9};

// Методы множеств в C# совпадают с PascalABC.NET
```

```
hs.Contains(3);
hs.Add(777);
hs.Remove(5);

// Операции с множествами (для HashSet и SortedSet)
Print(3 in hs); // True
hs += 777;
hs -= 777;
var hs1 := HSet(1,3,5);
var hs2 := HSet(3,5,7);
hs1 * hs2; // пересечение
hs1 + hs2; // объединение
hs1 - hs2; // разность
hs1 < hs2; // строгое вложение
hs1 <= hs2; // нестрогое вложение
hs1 = hs2; // поэлементное равенство множеств
```

```
hs.Contains(3);
hs.Add(777);
hs.Remove(5);

// Операции с множествами в C# отсутствуют
```

Последовательности

PascalABC.NET **C#**

```
// Описание
var sq: sequence of integer;


// Генераторы последовательностей
sq := Seq(1,3,5);
sq := Range(1,10);
sq := Range(1,10,2); // 1 3 5 7 9
sq := SeqGen(5, 1, x -> x * 2); // 1 3 5 7 9

// Операции с последовательностями
Print(3 in sq); // True
var sq1 := Seq(1,3,5);
var sq2 := Seq(2,4,6);
(sq1 + sq2).Println; // 1 3 5 2 4 6
(sq1 * 3).Println; // 1 3 5 1 3 5 1 3 5
(sq1 + 777).Println; // 1 3 5 777

// Распаковка последовательностей в переменные
var (min,min2) := ArrRandom(10).Distinct;
```

```
// Описание
using System.Collections.Generic;
IEnumerable<int> sq;

// Генераторы последовательностей в C# отсутствуют кроме
sq = Enumerable.Range(1,10);



// Операции с последовательностями в C# отсутствуют



// Распаковка последовательностей в переменные в C#
отсутствует
```

```
// Ничего подключать не нужно


var s: sequence of integer := |1, 2, 3|;
s := new List<integer>;
s := new HashSet<integer>;
s := Range(1,10);
s := SeqGen(10, 1, x -> x * x);

// Возможно использовать диапазоны как последовательности
(1..10).Print;
```

```
// Для использования контейнеров нужно подключать
using System.Collections.Generic;

IEnumerable<int> s = new [] { 1, 2, 3 };
s = new List<int>();
s = new HashSet<int>();
s = Enumerable.Range(1,10);
s = Enumerable.Range(1, 10).Select(x => x * x);

// Диапазоны нельзя использовать как последовательности
```

```
// Функции – генераторы последовательностей
function GenOddInfinity: sequence of integer;
begin
  var start := 1;
  while True do
  begin
    yield start;
    start += 2;
  end;
end;
```

```
// Функции – генераторы последовательностей
IEnumerable<int> GenOddInfinity()
{
    var start = 1;
    while (true)
    {
        yield return start;
        start += 2;
    }
}
```

```
// Большинство методов последовательностей в PascalABC.Net
// идентичны методам в C#. Есть исключения:
  .Print(delim: string := ' ')
  .Println(delim: string := ' ')
  .PrintLines
  .Sorted
  .Product
  .ForEach
  .Cartesian(second: sequence of T1)
  .ZipTuple(second: sequence of T1)
  .SplitAt(ind: integer)
  .Partition(cond: T->boolean)
  .Interleave(second: sequence of T)
  .Numerate([from: integer])
```

```
/* В C# присутствуют все методы последовательностей,
используемые в PascalABC.Net за исключением
перечисленных.*/
```

```
  .Tabulate(F: T->T1)
  .Pairwise
  .NWise(n)
  .Batch(n)
  .Incremental
  .ToHashSet
  .ToSortedSet
  .MinBy
  .MaxBy
  .CountOf(value)
  .EachCount
```

Записи

PascalABC.NET

```
type
  Person = record
    name: string;
    age: integer;

    procedure Print;
    begin
      Println($'Имя: {name}, Возраст: {age}');
    end;
  end;

/// Основная программа
...
var p := new Person;
p.Print;    // Имя: , Возраст: 0

p.name := 'Иван';
p.Print;    // Имя: Иван, Возраст: 0
```

C#

```
struct Person
{
    public string name;
    public int age;

    public void Print()
    {
        Console.WriteLine($"Имя: {name}, Возраст: {age}");
    }
}

/// Основная программа
...
Person p = new Person();
p.Print();   // Имя: , Возраст: 0

p.name = "Иван";
p.Print();    // Имя: Иван, Возраст: 0
```

PascalABC.NET

```
type
  Person = record
    name: string;
    age: integer;

    constructor (name: string := 'Иван'; age: integer :=
18);
    begin
      Self.name := name;
      Self.age := age;
    end;

    procedure Print;
    begin
      Println($'Имя: {name}, Возраст: {age}');
    end;
  end;

/// Основная программа
...
var p := new Person;
p.Print;    // Имя: Иван, Возраст: 18

var p2 := new Person('Пётр', 20);
p2.Print;    // Имя: Пётр, Возраст: 20

var p3 := new Person('Василий');
p3.Print;    // Имя: Василий, Возраст: 18
```

C#

```
struct Person
{
    public string name;
    public int age;

    public Person(string name = "Иван", int age = 18)
    {
        this.name = name;
        this.age = age;
    }

    public void Print()
    {
        Console.WriteLine($"Имя: {name}  Возраст: {age}");
    }
}


/// Основная программа
...
Person p = new Person();
p.Print();   // Имя: Иван, Возраст: 18

Person p2 = new Person("Пётр", 20);
p2.Print();    // Имя: Пётр, Возраст: 20

Person p3 = new Person("Василий");
p3.Print();    // Имя: Василий, Возраст: 18
```

Автоклассы

PascalABC.NET

```
type
  Person = auto class
    name: string;
    age: integer;
  end;

/// Основная программа
...
// Конструктор автокласса генерируется автоматически
var p := new Person('Иван', 20);

// Вызывается сгенерированный автоматически метод ToString
```

C#

```
/* Полного аналога автоклассов в C# нет. Ближайшими по
смыслу являются записи. Подразумевается, что записи
неизменяемы. Вместо полей используются свойства с доступом
на чтение.*/
record Person
{
    public string name { get; init; }
    public int age { get; init; }
}

/// Основная программа
...
```

```
// который выводит значения полей
Println(p);
```

```
// Конструктор не генерируется, но можно использовать
// инициализатор
Person p = new Person
{
    name = "Петя",
    age = 20
};
// автоматически генерируется метод ToString()
Console.WriteLine(p);
```

```
/* При использовании позиционного синтаксиса для каждого
параметра генерируется свойство, а также генерируется
конструктор */
public record Person(string name, int age);


/// Основная программа
...
Person person = new("Nancy", 12);
```

Классы

PascalABC.NET

```
type
  Person = class
  private
    Name: string;
    Age: integer;
  public
    constructor (Name: string; Age: integer);
    begin
      Self.Name := Name;
      Self.Age := Age;
    end;
    procedure Print;
  end;

// Можно определять методы вне интерфейса класса
procedure Person.Print;
begin
  Println($'Имя: {Name}, Возраст: {Age}');
end;


/// Основная программа
...
var p := new Person('Иван', 10);
p.Print();  // Имя: Иван, Возраст: 10

var p2 := new Person;
p2.Print(); // Имя: , Возраст: 0
```

C#

```
class Person
{
    private string name;
    private int age;

    public Person(string name, int age)
    {
        this.name = name;
        this.age = age;
    }

    public void Print()
    {
        Console.WriteLine($"Имя: {name}, Возраст: {age}");
    }
}


/// Основная программа
...
Person p = new Person("Иван", 10);
p.Print();

/* Ошибка компиляции. Конструктор по умолчанию
генерируется только если нет других конструкторов */
Person p2 = new Person();
```

Строки

PascalABC.NET

```
// Строки по умолчанию индексируются с 1.
// Для индексации с 0 нужна директива {$zerobasedstrings}

{$zerobasedstrings}
var s: string := 'abcdefghijkl';
s[1:5];  // bcde
s.ToLower;

// Лексикографическое сравнение строк
s1 < s2

// Применимы и остальные методы последовательностей
s.Where(c -> char.IsPunctuation(c));

var c: char := 'c';
c.IsLetter
c in s;
s1 in s
c.ToDigit;
Ord(c); // integer(c)

var s := 'Hello';
s[2] := 'a';  // Hallo  (строки изменяемые)
```

C#

```
// Строки индексируются с 0.


var s = "abcdefghijkl";
s[1..5]; // bcde
s.ToLower();


// В C# строки нельзя сравнивать на <
s1.CompareTo(s2) // <0 если s1<s2; =0 если s1=s2; иначе >0

// Применимы и остальные методы последовательностей
s.Where(c => char.IsPunctuation(c));

char c = 'e';
char.IsLetter(c);
s.Contains(c);
s.Contains(s1);
char.GetNumericValue(c);
(int)c;

var s = "Hello";
s[2] = 'a';  // Ошибка. Строки неизменяемы.
```

``var sb := new StringBuilder(s); sb[3] := 'a'; // Индексация всегда с 0 sb.Append('!'); s := sb.ToString; // Halao!```	```var sb = new StringBuilder(s); sb[3] = 'a'; sb.Append('!'); s = sb.ToString(); // Helao!```
``// Слияние последовательности в строку var s := (1..5).JoinToString; // 1 2 3 4 5 var s1 := (1..5).JoinToString(','); // 1,2,3,4,5```	```// Слияние последовательности в строку var s = string.Join(' ',Range(1,5)); var s1 = string.Join(',',Range(1,5));```