Сравнение PascalABC.NET и старого Паскаля

PascalABC.NET – завоевание популярности

- Компактная мощная оболочка.
- Мощный современный язык программирования, совместимый со «стандартным Паскалем».
- Сайт http://pascalabc.net с огромным количеством примеров.
- Около 2000 скачиваний в день.
- 29.03.2015 – **1 миллион скачиваний с начала проекта.**
- Включение PascalABC.NET как основного языка в ряд школьных учебников по информатике.
- С 2013 г. – активное использование на олимпиадах по программированию.

Этап всероссийской олимпиады по информатике в Москве

	2015-16						2014-15					
	Школьный		Муниципальный		Региональный		Школьный		Муниципальный		Региональный	
Всего участников (>0 баллов)	7371	100,00%	2467	100,00%	435	100,00%	5740	100,00%	1932	100,00%	478	100,00%
PascalABC.NET	3782	51,31%	1150	46,62%	64	14,71%	2723	47,44%	552	28,57%	59	12,34%
FPC	1040	14,11%	166	6,73%	29	6,67%	881	15,35%	433	22,41%	58	12,13%
Дельфи	27	0,37%	2	0,08%	4	0,92%	43	0,75%	30	1,55%	9	1,88%
Все паскали	4812	65,28%	1306	52,94%	85	19,54%	3608	62,86%	944	48,86%	102	21,34%
g++	747	10,13%	432	17,51%	211	48,51%	557	9,70%	368	19,05%	222	46,44%
gcc	242	3,28%	100	4,05%	13	2,99%	225	3,92%	87	4,50%	22	4,60%
clang++					9	2,07%	34	0,59%	20	1,04%	9	1,88%
clang							8	0,14%	4	0,21%	4	0,84%
Все C/C++	971	13,17%	530	21,48%	231	53,10%	811	14,13%	462	23,91%	244	51,05%
Python-3	859	11,65%	407	16,50%	151	34,71%	580	10,10%	379	19,62%	174	36,40%
Python-2	76	1,03%	24	0,97%	3	0,69%	38	0,66%	22	1,14%	8	1,67%
Все питоны	925	12,55%	429	17,39%	153	35,17%	611	10,64%	395	20,45%	179	37,45%
Кумир-1	219	2,97%	43	1,74%	0	0,00%	308	5,37%	31	1,60%	0	0,00%
Кумир-2	162	2,20%	26	1,05%	0	0,00%	77	1,34%	18	0,93%	0	0,00%
Все кумиры	373	5,06%	68	2,76%	0	0,00%	378	6,59%	44	2,28%	0	0,00%
Qbasic (fbc)	116	1,57%	13	0,53%	1	0,23%	240	4,18%	20	1,04%	0	0,00%
Visual Basic	36	0,49%	14	0,57%	0	0,00%	34	0,59%	10	0,52%	2	0,42%
FBC-32	16	0,22%	6	0,24%	0	0,00%	3	0,05%	0	0,00%	0	0,00%
Все бейсики	168	2,28%	33	1,34%	1	0,23%	277	4,83%	30	1,55%	2	0,42%
C#	164	2,22%	88	3,57%	7	1,61%	122	2,13%	62	3,21%	15	3,14%
Java	120	1,63%	65	2,63%	9	2,07%	58	1,01%	46	2,38%	13	2,72%
php	28	0,38%	6	0,24%	0	0,00%	18	0,31%	7	0,36%	1	0,21%
perl	2	0,03%	2	0,08%	0	0,00%	7	0,12%	3	0,16%	1	0,21%
ruby	3	0,04%	2	0,08%	0	0,00%	0	0,00%	1	0,05%	1	0,21%

(по данным региональной предметно-методической комиссии)

Сравнение версий языка Паскаль

- **Delphi XE.** Коммерческая среда. Отсутствие бесплатной версии. Оболочка, не предназначенная для обучения.
- **Turbo/Borland Pascal.** Отжившая устаревшая версия языка и среды. Нет легальной бесплатной версии.
- **Free Pascal.** Отжившая устаревшая среда. Профессиональный язык Pascal, далекий от обучения. Отсутствие в языке современных возможностей. Оболочка Lazarus, предназначенная преимущественно для создания пользовательских интерфейсов.
- **PascalABC.NET.** Современная оболочка. Язык программирования Pascal нового поколения. Основывается на мощной постоянно развивающейся платформе Microsoft.NET.

Стандартный Паскаль

- Такого не существует
- ISO-стандарт языка Паскаль есть, он – закрытый, им никто не пользуется
- То, что обычно называют «стандартным Паскалем», – это некоторое идеализированное представление о минимальном наборе конструкций языка Паскаль. Обычно у каждого это представление – своё, но связывается оно с уже не существующей версией Turbo Pascal, а также со средствами языка, существовавшими 20-30 лет назад и вредными для современного обучения программированию.
- Все языки развиваются. Те языки, которые не развиваются, – умерли
- В этой презентации

Стандартный Паскаль = Старый Паскаль

Стандартный Free Pascal

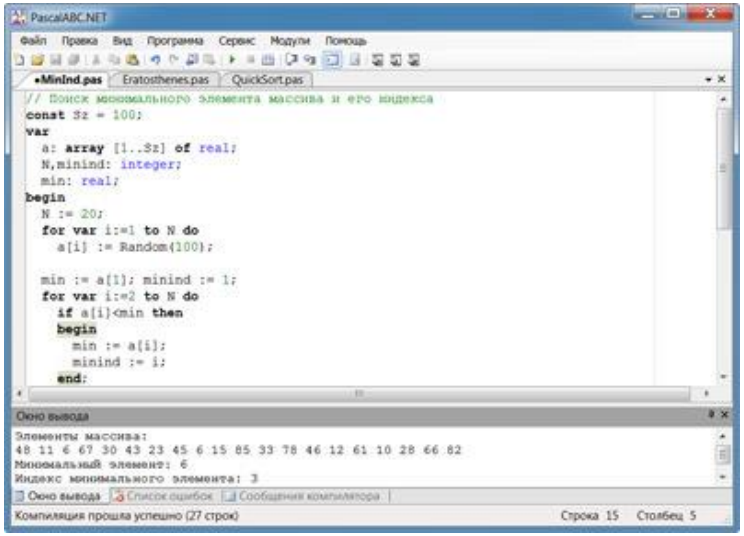
- Слухи о простоте Free Pascal сильно преувеличены. Примеры из документации «современного FP» (2015 г.):

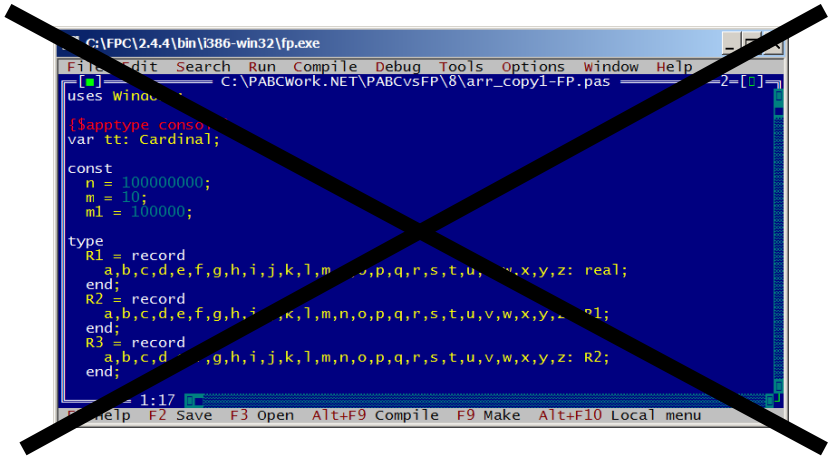
```
type
  MyItemClass = objcclass external;
  TMyObjectHelper = class helper(TObjectHelper) for TMyObject
    procedure SomeOtherMethod;
  end;
  generic TList<_T>=class(TObject)
    type public
      TCompareFunc = function(const Item1, Item2: _T): Integer;
    var public
      data : _T;
    procedure Add(item: _T);
    procedure Sort(compare: TCompareFunc);
  end;
  TB = Specialize TList <string>;
```

- Ни одной из этих возможностей нет в «стандартном» Паскале
- Эти конструкции тяжеловесны, несовременны, плохо читаются.

PascalABC.NET – это:

- Современная, простая, и мощная среда разработки.
- Язык программирования **нового поколения**, сочетающий простоту классического языка Паскаль, ряд современных расширений и огромные возможности платформы .NET.
- PascalABC.NET – это способ изучать современное программирование сегодня и завтра.
- PascalABC.NET – это не тот язык Паскаль, которому учили вашего отца и деда.





PascalABC.NET vs Free Pascal

PascalABC.NET опережает Free Pascal по скорости выполнения программ на большинстве тестов.

Ниже приводится пример со всеми включенными оптимизациями.

Free Pascal 3.0

```
uses Windows;

{$apptype console}

var
  tt: Cardinal;
  n,i,j: integer;
  s: real;
begin
  tt := GetTickCount;
  n := 10000;
  s := 0.0;
  for i:=1 to n do
  for j:=1 to n do
    s := s + 1.0/(i*j);
  writeln(GetTickCount-tt);
end.
```

Время выполнения (Core I5-2500): 0.71 с

PascalABC.NET 3.0

```
begin
  var n := 10000;
  var s := 0.0;
  for var i:=1 to n do
  for var j:=1 to n do
    s += 1.0/(i*j);
  writeln(Milliseconds);
end.
```

Время выполнения (Core I5-2500): 0.64 с

PascalABC.NET: основные нововведения в синтаксисе

- PascalABC.NET имеет ряд ключевых нововведений в синтаксисе языка, которые используются практически в каждой программе
- Это:
  - Операторы += и *=
  - Внутриблочные переменные
  - Инициализация при описании
  - Автоопределение типа
  - for var i
- Программировать в стиле старого Паскаля не рекомендуется
- Программы в стиле старого Паскаля менее эффективны по скорости работы и прививают отживший стиль программирования 90-х годов.
- Далее представлены многочисленные примерами кода. Обычно на слайде слева содержится код на устаревшем Паскале, а справа – легковесный код на PascalABC.NET с той же функциональностью

Операторы += и *=

Модифицированные операторы присваивания += и *= встречаются во многих языках (в том числе и во Free Pascal) и воспринимаются проще, чем традиционные a := a + 2 и a := a * 2.

Здесь же иллюстрируется инициализация переменной при описании.

Старый Паскаль

```
var a: integer;

begin
  a := 1;
  a := a + 2;
  a := a * 2;
end.
```

PascalABC.NET

```
var a: integer := 1;

begin
  a += 2; // увеличить на 2
  a *= 2; // увеличить в 2 раза
end.
```

Внутриблочные переменные

Переменные следует описывать как можно ближе к месту их первого использования. Временные переменные обязательно описывать внутри блока, чтобы не захламлять раздел описания. Переменные, используемые для одной цели на протяжении всей программы, допустимо описывать в разделе описания (до begin). Это подчёркивает их глобальность.
Параметры цикла for следует обязательно описывать в заголовке цикла (конструкция for var). При таком описании параметры цикла недоступны вне тела цикла.

Старый Паскаль

```
var
  i,n: integer;
  a,p,s: real;
begin
  read(a,n);
  a := 1;
  p := 1.0;
  for i:=1 to n do
    p := p * a;
  writeln(p);
  s := 0.0;
  for i:=1 to n do
    s := s + i*i;
  writeln(s);
end.
```

PascalABC.NET

```
var n: integer;

begin
  n := ReadInteger;

  var a := ReadReal;
  var p: real := 1;
  for var i:=1 to n do
    p *= a;
  writeln(p);

  var s := 0.0;
  for var i:=1 to n do
    s += i*i;
  writeln(s);
end.
```

Автоопределение типов

Тип переменной определяется по типу значения при описании с инициализацией. Это компактно записывается и очевидно для восприятия.

Старый Паскаль

```
var
  x: integer;
  y: real;
  z: char;
  a: array [1..3] of integer;
begin
  x := 1;
  y := 2.5;
  z := 'z';
  a[1]:=1; a[2]:=3; a[3]:=5;
end.
```

PascalABC.NET

```
begin
  var x := 1;
  var y := 2.5;
  var z := 'z';
  var a := Arr(1,3,5);
// Тип a – тот же, что возвращает Arr:
// array of integer
end.
```

Полезные стандартные подпрограммы

В PascalABC.NET имеется множество полезных стандартных подпрограмм. Для начинающих это Print, ReadInteger, ReadReal, Min, Max, Swap.

Процедура Print разделяет элементы вывода пробелом.

Старый Паскаль

```
var a,b,t,vmin,vmax: integer;
begin
  write('Введите a:');
  readln(a);
  write('Введите b:');
  readln(b);
  if a<b then
    vmin := a
  else vmin := b;
  if a>b then
    vmax := a
  else vmax := b;
  writeln(vmin,' ',vmax);
  t := a;
  a := b;
  b := t;
  writeln(a,' ',b);
end.
```

PascalABC.NET

```
begin
  var a := ReadInteger('Введите a:');
  var b := ReadInteger('Введите b:');
  var vmin := Min(a,b);
  var vmax := Max(a,b);
  Println(vmin,vmax);
  Swap(a,b);
  Println(a,b);
end.
```

Write (что угодно)

В PascalABC.NET процедуры Write и Print выводят значение любого составного типа: массива, записи, множества.

Для вывода массивов используются [], для вывода записей – (), а для вывода множеств – {}.

На устаревшем Паскале для вывода составных типов необходимо писать нагруженный деталями код.

Старый Паскаль

```
var a: array [1..3] of integer;

var p: record
  name: string;
  age: integer;
end;

var s: set of byte;
    i: integer;

begin
  a[1] := 2; a[2] := 3; a[3] := 5;
  p.name := 'Иванов'; p.age := 20;
  s := [1,3,7];

  for i:=1 to 3 do
    write(a[i],' ');
  writeln;

  writeln(p.name,' ',p.age);

  for i:=0 to 255 do
    if i in s then
      write(i,' ');
end.
```

PascalABC.NET

```
var a: array [1..3] of integer := (2,3,5);

var p: record
  name: string;
  age: integer;
end;

var s: set of integer := [1,3,7];

begin
  p.name := 'Иванов'; p.age := 20;
  writeln(a);
  writeln(p);
  writeln(s);
end.
```

```
[2,3,5]
(Иванов,20)
{3,1,7}
```

Result в функции

Для возвращения значения из функции следует использовать переменную Result, а не устаревший синтаксис, связанный с присваиванием имени функции.

Переменная Result появилась в Delphi и используется во Free Pascal.

Старый Паскаль

```
function fact(n: integer): integer;
var p: integer;
    i: integer;
begin
  p := 1;
  for i:=1 to n do
    p := p * i;
  fact := p;
end;

var n: integer;
begin
  read(n);
  writeln('n!=',fact(n));
end.
```

PascalABC.NET

```
function fact(n: integer): integer;
begin
  Result := 1;
  for var i:=1 to n do
    Result *= i;
end;


begin
  var n := ReadInteger('Введите n:');
  writeln('n!=',fact(n));
end.
```

Case по строкам

В PascalABC.NET можно делать case по строкам. Это значительно удобнее старого стиля со вложенными if

Старый Паскаль

```
var Country: string;

begin
  read(Country);
  write('Столица: ');
  if Country = 'Россия' then
    writeln('Москва')
  else if Country = 'Франция' then
    writeln('Париж')
  else if Country = 'Италия' then
    writeln('Рим')
  else if Country = 'Германия' then
    writeln('Берлин')
  else writeln('Нет в базе данных');
end.
```

PascalABC.NET

```
begin
  var Country := ReadString;
  write('Столица: ');
  case Country of
    'Россия':   writeln('Москва');
    'Франция':  writeln('Париж');
    'Италия':   writeln('Рим');
    'Германия': writeln('Берлин');
    else writeln('Нет в базе данных');
  end;
end.
```

BigInteger

В PascalABC.NET имеется стандартный тип длинных целых BigInteger. Это позволяет решать задачи, которые в старом Паскале требовали написания большого количества кода. Такие задачи ранее предлагались в качестве олимпиадных.

Старый Паскаль

```
Много строк кода для реализации
операций с длинными целыми
```

PascalABC.NET

```
begin
  var p: BigInteger := 1;
  for var i:=2 to 100 do
    p *= i;
  writeln('100!=',p)
end.
```

```
100!=933262154439441526816992388562667004907159682643
816214685929638952175999932299156089414639761565182862
253697920827223758251185210916864000000000000000000000
0000
```

Короткие определения подпрограмм

В PascalABC.NET допускаются короткие определения для функций, задаваемых одним выражением, и для процедур с телом из одного оператора. Тип возвращаемого значения функции можно не задавать – он выводится автоматически.

Старый Паскаль

```
function Sqr3(x: integer): integer;
begin
  Sqr3 := x*x*x;
end;

function CircleLen(r: real): real;
begin
  CircleLen := 2 * Pi * r;
end;

function Hypot(a,b: real): real;
begin
  Hypot := sqrt(a*a + b*b);
end;

function Len(x1,y1,x2,y2: real): real;
begin
  Len := Hypot(x2-x1,y2-y1);
end;

begin
  Writeln(Sqr3(2),' ',CircleLen(1));
  Writeln(Hypot(3,4),' ',Len(1,1,3,4));
end.
```

PascalABC.NET

```
function Sqr3(x: integer) := x*x*x;

function CircleLen(r: real): real := 2 * Pi * r;

function Hypot(a,b: real) := sqrt(a*a + b*b);

function Len(x1,y1,x2,y2: real) := Hypot(x2-x1,y2-y1);

function DigitCount(x: integer) := abs(x).ToString.Length;

function Минимум(a,b,c: real): real := Min(Min(a,b),c);

procedure Вывод<ЛюбойТип>(x: ЛюбойТип) := Println(x);

begin
  Println(Sqr3(2),CircleLen(1));
  Println(Hypot(3,4),Len(1,1,3,4));
  Println(DigitCount(-1234));
  Вывод(Минимум(5,3,8));
end.
```

Русские идентификаторы

В PascalABC.NET можно использовать русские идентификаторы. Это может улучшать восприятие, а также использоваться для начального обучения программированию

Старый Паскаль

```
Нечем похвастать
```

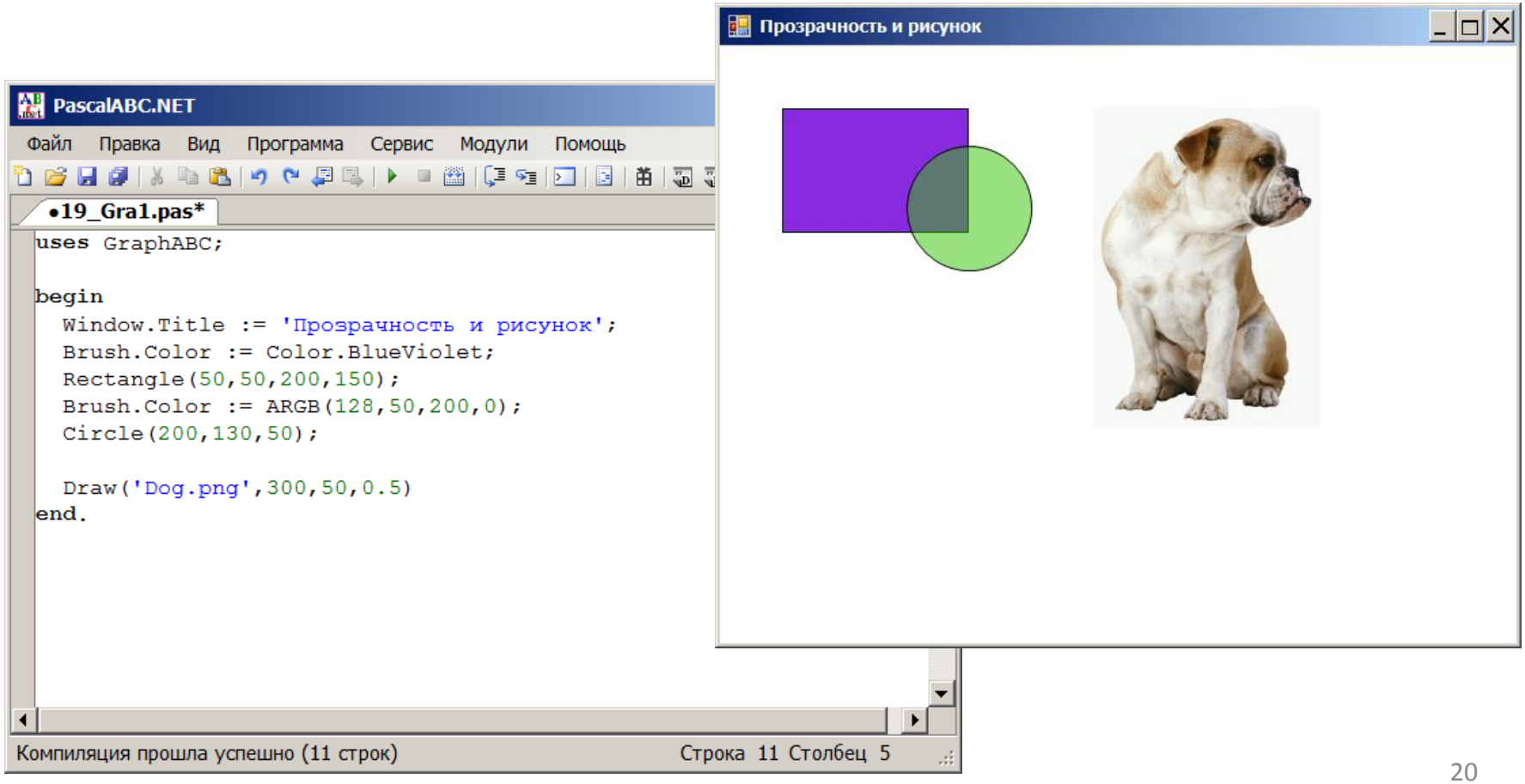
PascalABC.NET

```
type
  цел = integer;
  вещ = real;

procedure Вывод(число: вещ) := Println(число);

function Диапазон(нач,кон: цел) := Range(нач,кон);

begin
  var сумма: вещ := 0.0;
  var количество: цел := 10;

  for var i:=1 to количество do
    сумма += 1/i;

  Вывод(сумма);

  Вывод(Диапазон(1,количество).Sum(знач -> 1/знач));
end.
```

Графический модуль

В PascalABC.NET имеется простой по использованию и мощный по возможностям графический модуль GraphABC. На скриншоте – пример с прозрачностью и выводом изображения.

В старом Паскале используется несовременная графика, требующая хитрых драйверов и заклинаний при подключении.



Записи и кортежи

В PascalABC.NET можно создавать записи «на лету» с помощью **кортежей**. К полям кортежа можно получить доступ по индексу

Старый Паскаль

```
type Pupil = record
  name: string;
  age: integer;
end;

var p: Pupil;

begin
  p.name := 'Петрова';
  p.age := 18;
  writeln(p.name,' ',p.age);
end.
```

PascalABC.NET

```
begin
  var p := ('Петрова',18);
  Println(p, p[0], p[1]);
end.
```

```
(Петрова,18) Петрова 18
```

Динамические массивы

При обучении следует использовать динамические массивы вместо статических. Их преимущества:

- Знают свою длину
- Легко используются как параметры подпрограмм и возвращаемые значения функций
- Имеется множество подпрограмм стандартной библиотеки для работы с динамическими массивами
- При доступе на чтение по динамическому массиву удобно использовать цикл **foreach**

Старый Паскаль

```
type IntArr = array [1..10] of integer;

procedure SquareElems(const a: IntArr;
  n: integer; var Result: IntArr);
var i: integer;
begin
  for i:=1 to n do
    Result[i] := sqr(a[i]);
end;

const m = 4;

var a,Result: IntArr;
    i: integer;
begin
  a[1] := 2; a[2] := 5;
  a[3] := 7; a[4] := 10;
  SquareElems(a,m,Result);
  for i:=1 to m do
    write(Result[i],' ');
end.
```

PascalABC.NET

```
function SquareElems(a: array of integer):
  array of integer;
begin
  SetLength(Result,a.Length);
  for var i:=0 to a.Length-1 do
    Result[i] := sqr(a[i]);
end;

var a: array of integer;

begin
  a := Arr(2,5,7,10);
  a := SquareElems(a);
  foreach var x in a do
    Print(x);
end.
```

Функции Arr для создания динамических массивов

Для создания динамических массивов используются стандартные функции Arr, ArrRandom, ArrFill, ReadArrInteger, возвращающие динамический массив соответствующего типа.

Аналогичный код на старом Паскале – длинен и ужасен.

Старый Паскаль

```
var a: array [1..10] of integer;
    b: array [1..6] of char;
    i: integer;
begin
  a[1] := 2; a[2] := 5;
  a[3] := 7; a[4] := 10;
  for i:=1 to 4 do
    write(a[i],' ');
  writeln;
  b[1] := 'a'; b[2] := 'e'; b[3] := 'i';
  b[4] := 'o'; b[5] := 'u'; b[6] := 'y';
  for i:=1 to 6 do
    write(b[i],' ');
  writeln;
  Randomize;
  for i:=1 to 10 do
    a[i] := Random(100);
  for i:=1 to 10 do
    write(a[i],' ');
  writeln;
  for i:=1 to 5 do
    a[i] := 1;
  for i:=1 to 5 do
    write(a[i],' ');
  writeln;
  for i:=1 to 5 do
    read(a[i]);
  for i:=1 to 5 do
    write(a[i],' ');
end.
```

PascalABC.NET

```
var a: array of integer;

begin
  a := Arr(2,5,7,10);
  writeln(a);
  var b := Arr('a','e','i','o','u','y');
  writeln(b);
  a := ArrRandom(10);
  writeln(a);
  a := ArrFill(5,1);
  writeln(a);
  a := ReadArrInteger(5);
  writeln(a);
end.
```

Стандартные процедуры Sort и Reverse

Стандартные процедуры Sort и Reverse позволяют решать более сложные задачи. Процедура Sort содержит эффективный алгоритм (быстрая сортировка).

При реализации этих процедур вручную ученик нередко допускает ошибки.

Код на старом Паскале – ужасен.

Старый Паскаль

```
const n = 6;
var a: array [1..n] of integer;
    i,j,t: integer;
begin
  a[1] := 2; a[2] := 5; a[3] := 3;
  a[4] := 1; a[5] := 9; a[6] := 7;
  for i:=1 to n do
    write(a[i],' ');
  writeln;
  for i:=1 to n-1 do
    for j:=n downto 2 do
      if a[j-1]>a[j] then
      begin
        t := a[j-1];
        a[j-1] := a[j];
        a[j] := t;
      end;
  for i:=1 to n do
    write(a[i],' ');
  writeln;
  for i:=1 to n div 2 do
  begin
    t := a[i];
    a[i] := a[n-i+1];
    a[n-i+1] := t
  end;
  for i:=1 to n do
    write(a[i],' ');
  writeln;
end.
```

PascalABC.NET

```
begin
  var a := Arr(2,5,3,1,9,7);
  Sort(a);
  writeln(a);
  Reverse(a);
  writeln(a);
end.
```

Методы динамических массивов

В динамические массивы встроен ряд методов, вызываемых по точке после имени переменной. Эти методы позволяют вывести элементы массива с разделителем, вычислить основные характеристики числового массива (минимум, максимум, сумма, среднее), получить по массиву отсортированный массив. Код на старом Паскале – ужасен.

Старый Паскаль

```
const n = 10;
var a,b: array [1..n] of integer;
    i,j,t,min,max,sum: integer;
    av: real;
begin
  for i:=1 to n do
    a[i] := Random(100);
  for i:=1 to n do
    write(a[i],' ');
  writeln;
  b := a;
  for i:=1 to n-1 do
    for j:=n downto 2 do
      if b[j-1]>b[j] then
      begin
        t := b[j-1];
        b[j-1] := b[j];
        b[j] := t;
      end;
  for i:=1 to n do
    write(b[i],', ');
  writeln;
  min := MaxInt; max := -MaxInt;
  sum := 0;
  for i:=1 to n do
  begin
    if a[i]<min then
      min := a[i];
    if a[i]>max then
      max := a[i];
    sum := sum + a[i];
  end;
  av := sum/n;
  writeln(min,' ',max, ' ',sum,' ',av);
end.
```

PascalABC.NET

```
begin
  var a := ArrRandom();
  a.Println;
  a.Sorted.Println(', ');
  Println(a.Min,a.Max);
  Println(a.Sum,a.Average);
end.
```

Динамические матрицы

- Матрицы (двумерные массивы) также могут быть динамическими.
- Основное преимущество динамических матриц – легкость их передачи в подпрограммы и то, что размеры матрицы совпадают с памятью, необходимой для хранения.

Старый Паскаль

```
type
  Matr = array [1..10,1..10] of integer;

procedure Transpose(const a: Matr;  m,n: integer; var b: Matr);
var i,j: integer;
begin
  for i:=1 to n do
  for j:=1 to m do
    b[i,j] := a[j,i]
end;

var a,b: Matr;
    m,n,i,j: integer;

begin
  m := 3;
  n := 4;
  for i:=1 to 3 do
  for j:=1 to 4 do
    a[i,j] := Random(100);

  Transpose(a,m,n,b);

  for i:=1 to n do
  begin
    for j:=1 to m do
      write(b[i,j],' ');
    writeln;
  end;
end.
```

PascalABC.NET

```
function Transpose(a: array [,] of integer):
  array[,] of integer;
begin
  var m := Length(a,0);
  var n := Length(a,1);
  Result := new integer[n,m];
  for var i:=0 to n-1 do
  for var j:=0 to m-1 do
    Result[i,j] := a[j,i]
end;

begin
  var a := MatrixRandom(3,4);
  writeln(a);
  var b := Transpose(a);
  writeln(b);
end.
```

Множества

- Встроенные множества в PascalABC.NET могут иметь произвольный базовый тип

Старый Паскаль

```
Нечем похвастать
```

PascalABC.NET

```
var ss1,ss2: set of string;

begin
  ss1 := ['планшет','смартфон','ноутбук'];
  ss2 := ['смартфон','компьютер','планшет'];
  writeln('Объединение: ',ss1+ss2);
  writeln('Пересечение: ',ss1*ss2);
  writeln('Разность:    ',ss1-ss2);
end.
```

```
Объединение: {планшет,смартфон,компьютер,ноутбук}
Пересечение: {планшет,смартфон}
Разность:    {ноутбук}
```

Символы

- Символы в PascalABC.NET хранятся в кодировке Unicode
- Преобразования символ <-> код: Ord(c), Chr(i)
- Для типа char доступен ряд новых методов, вызываемых по точке

Старый Паскаль

```
{примерный эквивалент}
var c: char;
    i: integer;
begin
  c := 'a';
  if c in ['A'..'Z','a'..'z'] then
  begin
    c := Succ(c);
    c := ToUpper(c);
    c := Pred(c);
  end;
  c := '5';
  if c in ['0'..'9'] then
    i := Ord(c)-Ord('0');
  writeln(Ord('a'));
end.
```

PascalABC.NET

```
begin
  var c := 'a';
  if c.IsLetter then
  begin
    c := c.Succ;
    c := c.ToUpper;
    c := c.Pred;
  end;
  c := '5';
  var i: integer;
  if c.IsDigit then
    i := c.ToDigit;
  writeln('a'.Code);
end.
```

Строки

- Строки могут иметь произвольную длину – ограничение строки длиной 255 символов (как в старом Паскале) отсутствует.
- Строка знает свою длину: s.Length.
- Для строк можно использовать цикл foreach.
- Для строк доступны операции сложения с числами и умножения на число.

Старый Паскаль

```
var s: string;
    i: integer;
begin
  s := 'ABCDEFGH';
  s := s + 'IJK';
  for i:=1 to Length(s) do
    write(s[i],' ');
  writeln;
  Str(12345,s);
  s := '';
  for i:=1 to 10 do
    s := s + 'a';
  writeln(s);
end.
```

PascalABC.NET

```
var s: string;

begin
  s := 'ABCDEFGH';
  s += 'IJK';
  foreach var c in s do
    Print(c);
  Println;
  s := ''+12345; // число преобразуется в строку
  s := 6789 + s; // число преобразуется в строку
  writeln('a'*10); // строка повторяется 10 раз
  writeln(5*'xyz'); // строка повторяется 5 раз
end.
```

Строки. Разбиение на слова

Высокоуровневые методы строк ToWords и JoinIntoString позволяют превратить строку в массив слов и обратно.

На старом Паскале для этого необходимо писать запутанный код.

Старый Паскаль

```
Код не поместится на слайд
```

PascalABC.NET

```
begin
  var s := 'первый второй третий четвертый пятый';
  var ss := s.ToWords;
  ss.Println;
  Reverse(ss);
  ss.Println;
  Sort(ss);
  s := ss.JoinIntoString;
  writeln(s);
end.
```

```
первый второй третий четвертый пятый
пятый четвертый третий второй первый
второй первый пятый третий четвертый
```

Файлы

Методы, встроенные в файловые переменные, а также функции Open... открытия файла меняют стиль программирования при работе с текстовыми файлами, делая его короче и понятнее.
Функция ReadLines позволяет превратить файл в последовательность. При ее вызове файл открывается, а по окончании – закрывается. По последовательности мы можем совершить цикл foreach, а можем просто вывести ее с помощью метода Print с разделителем NewLine. Использование последовательностей **не загружает файл целиком в память**, а позволяет в каждый момент хранить в памяти одну строку файла.

Старый Паскаль

```
{ Решение 1 }
var
  f: Text;
  s: string;
begin
  Assign(f,'13_Files1.pas');
  Reset(f);
  while not eof(f) do
  begin
    readln(f,s);
    writeln(s);
  end;
  Close(f);
end.

{ Решение 2 - нечем похвастать}
```

PascalABC.NET

```
// Решение 1
begin
  var f := OpenRead('13_Files2.pas');
  while not f.Eof do
  begin
    var s := f.ReadlnString;
    writeln(s);
  end;
  f.Close;
end.

// Решение 2
begin
  foreach var s in ReadLines('13_Files4.pas') do
    writeln(s);
end.

// Решение 3 - ещё короче
begin
  ReadLines('13_Files3.pas').Print(NewLine);
end.
```

Обработка строк в файлах. Старый Паскаль

Задача: из файла freqs.txt (справа) вывести только глаголы (verb)

Решение на старом Паскале требует множества переменных, перегружено мелкими техническими деталями и сложно для начинающих. Оно ужасно.

Старый Паскаль

```
var
  f: text;
  s,kind: string;
  num,p: integer;
  freq: real;
  ch: char;

begin
  assign(f,'freqs.txt');
  reset(f);
  while not eof(f) do
  begin
    read(f,num,freq);
    read(f,ch);
    readln(f,s);
    p := Pos(' ',s);
    kind := Copy(s,p+1,100);
    s := Copy(s,1,p-1);
    if kind = 'verb' then
      writeln(s);
  end;
  close(f);
end.
```

freqs.txt

```
14029 31.03 напряженный adj
14030 5.45 напрямик adv
14031 9.43 напрямую adv
14032 6.43 напрячь verb
14033 13.71 напрячься verb
14034 19.10 напугать verb
14035 1.71 напугаться verb
14036 1.16 напудрить verb
14037 2.45 напускать verb
14038 1.53 напускной adj
14039 3.06 напустить verb
14040 1.04 напуститься verb
14041 3.73 напутать verb
14042 1.10 напутственный adj
14043 2.20 напутствие noun
14044 2.08 напутствовать verb
14045 1.10 напыщенный adj
14046 2.75 напяливать verb
14047 4.84 напялить verb
14048 1.78 наработать verb
14049 4.22 наравне adv
14050 1.29 нарадоваться verb
14051 2.69 нараспашку adv
```

Решения на PascalABC.NET

Решения задачи из предыдущего слайда на PascalABC.NET:

- коротки
- просты
- понятны

Решение 1

```
begin
  var f := OpenRead('freqs.txt');
  while not f.Eof do
  begin
    var ss := f.ReadlnString.ToWords;
    if ss[3] = 'verb' then
      writeln(ss[2]);
  end;
  f.Close;
end.
```

Решение 2

```
begin
  foreach var s in ReadLines('freqs.txt') do
  begin
    var ss := s.ToWords;
    if ss[3] = 'verb' then
      writeln(ss[2]);
  end;
end.
```

Задача о выделении четных чисел

Задача. Дана последовательность целых. Записать в новую последовательность только четные значения в том же порядке.

В старом Паскале требуется выделять массив максимального размера и заводить переменную – текущую заполненность этого массива. С помощью списков List в PascalABC.NET (динамические массивы с возможностью расширения) эта задача решается максимально естественно.

Старый Паскаль

```
const n = 10;
var a,b: array [1..n] of integer;
    nb: integer;
begin
  for var i:=1 to n do
    a[i] := Random(10);
  for var i:=1 to n do
    write(a[i],' ');
  writeln;

  nb := 0;
  for var i:=1 to n do
    if a[i] mod 2 = 0 then
    begin
      nb := nb + 1;
      b[nb] := a[i]
    end;

  for var i:=1 to nb do
    write(b[i],' ');
end.
```

PascalABC.NET

```
begin
  var a := ArrRandom();
  a.Println;

  var l := new List<integer>;
  foreach var x in a do
    if x mod 2 = 0 then
      l.Add(x);

  l.Println;
end.
```

Преобразование массивов

Для преобразования элементов массивов с помощью некоторой функции (например, для возведения всех элементов в квадрат) в PascalABC.NET эффективно использовать методы последовательностей, содержащиеся в любом массиве. Функция преобразования при этом задается в виде **лямбда-выражения** – функции, генерируемой «на лету».

В старом Паскале для этого приходится писать перегруженный деталями код.

Старый Паскаль

```
const n = 5;

var a,b: array [1..n] of integer;
    i: integer;

function f(x: integer): integer;
begin
  f := x*x;
end;

begin
  a[1] := 1; a[2] := 5; a[3] := 2;
  a[4] := 6; a[5] := 7;

  for i:=1 to n do
    write(a[i],' ');
  writeln;

  for i:=1 to n do
    b[i] := f(a[i]);

  for i:=1 to n do
    write(b[i],' ');
end.
```

Метод Select (проекция)

```
begin
  var a := Arr(1,5,2,6,7);
  a.Println;
  var b := a.Select(x->x*x);
  b.Println;
end.
```

```
1 5 2 6 7
1 25 4 36 49
```

В программе используется лямбда-выражение x->x*x, представляющее собой функцию возведения в квадрат.

Метод Select применяет эту функцию к каждому элементу последовательности и возвращает новую последовательность из квадратов элементов

Фильтрация массивов

Другой распространенной операцией при работе с массивами является фильтрация: отобрать в массиве элементы, удовлетворяющие некоторому условию, и вернуть их в виде другого массива.

В старом Паскале для решения этой задачи требуется выделять второй массив максимальной размерности и заполнять его частично, используя специальную переменную заполненности этого массива

Старый Паскаль

```
const n = 5;

var a,b: array [1..n] of integer;
    i,bn: integer;

function p(x: integer): boolean;
begin
  p := x mod 2 <> 0;
end;

begin
  a[1] := 1; a[2] := 5; a[3] := 2;
  a[4] := 6; a[5] := 7;
  for i:=1 to n do
    write(a[i],' ');
  writeln;

  bn := 0;
  for i:=1 to n do
    if p(a[i]) then
    begin
      bn := bn + 1;
      b[bn] := a[i];
    end;
  for i:=1 to bn do
    write(b[i],' ');
end.
```

Метод Where (фильтрация)

```
begin
  var a := Seq(1,5,2,6,7);
  a.Println;
  var b := a.Where(x->x mod 2 <> 0);
  b.Println;
end.
```

```
1 5 2 6 7
1 5 7
```

В программе используется лямбда-выражение x->x mod 2 <> 0, представляющее собой функцию-условие, определяющую нечетность числа.

Метод Where применяет эту функцию к каждому элементу последовательности и возвращает новую последовательность из элементов, удовлетворяющих этому условию

Range как замена цикла for

Функция Range, генерирующая последовательность, фактически является заменой цикла for.

Чтобы найти сумму квадратов всех нечетных, меньших 100, мы либо организуем цикл, в котором переменная i пробегает нечетные значения, меньшие 100, либо вызываем функцию Range(1,100,2), которая возвращает последовательность 1,3,5,...,99. Далее к этой последовательности применяется проекция Select(i ->i*i), преобразующая исходную последовательность в последовательность квадратов, после чего полученная последовательность суммируется.

Важно отметить, что функция Range ленивая: она не хранит всю последовательность в памяти, а возвращает по одному элементу, который затем преобразуется и суммируется. Таким образом, накладные расходы по памяти отсутствуют.

Старый Паскаль

```
var sum,i: integer;

begin
  sum := 0;
  i := 1;
  while i<=100 do
  begin
    sum := sum + i*i;
    i := i + 2;
  end;
  write(sum);
end.
```

PascalABC.NET

```
begin
  var sum := Range(1,100,2).Select(i -> i*i).Sum;
  write(sum);
end.
```

Range. Простые числа

Задача. Вывести таблицу простых чисел, меньших 1000.

Решение на PascalABC.NET с Range по смыслу похоже на решение с циклами: берется последовательность чисел от 2 до 1000, в ней отбираются простые числа (IsPrime), после чего они выводятся.

Определение простоты числа столь же просто: число x считается простым если оно не делится на все (метод All) числа i от 2 до Round(sqrt(x)).

Старый Паскаль

```
function IsPrime(x: integer): boolean;
var i: integer;
begin
  for i:=2 to Round(sqrt(x)) do
    if x mod i = 0 then
    begin
      IsPrime := False;
      exit
    end;
  IsPrime := True;
end;

var i: integer;

begin
  for i:=2 to 1000 do
    if IsPrime(i) then
      write(i,' ');
end.
```

PascalABC.NET

```
function IsPrime(x: integer): boolean;
begin
  Result := Range(2,Round(sqrt(x)))
    .All(i->x mod i <> 0)
end;

begin
  Range(2,1000).Where(IsPrime).Print;
end.
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113 127 131 137 139 149
151 157 163 167 173 179 181 191 193 197 199 211 223 227
229 233 239 241 251 257 263 269 271 277 281 283 293 307
311 313 317 331 337 347 349 353 359 367 373 379 383 389
397 401 409 419 421 431 433 439 443 449 457 461 463 467
479 487 491 499 503 509 521 523 541 547 557 563 569 571
577 587 593 599 601 607 613 617 619 631 641 643 647 653
659 661 673 677 683 691 701 709 719 727 733 739 743 751
757 761 769 773 787 797 809 811 821 823 827 829 839 853
857 859 863 877 881 883 887 907 911 919 929 937 941 947
953 967 971 977 983 991 997
```

Range. Таблица значений функции

Задача. Вывести таблицу значений функции y(x)=x*x на отрезке [1,2] с шагом 0.1

Решение. Формируем с помощью Range последовательность чисел от 1 до 2 с шагом 0.1, после чего проектируем каждый элемент последовательности x на запись (x, x*x) и выводим каждую такую запись с новой строки.

Старый Паскаль

```
var a,b,x,h: real;
    i,n: integer;

begin
  a := 1.0; b := 2.0;
  n := 10;
  h := (b-a)/n;
  x := a;
  for i:=0 to n do
  begin
    writeln(x,' ',x*x);
    x := x + h;
  end
end.
```

PascalABC.NET

```
begin
  Range(1.0,2.0,10).Select(x->(x,x*x))
    .Println(NewLine);
end.
```

```
(1,1)
(1.1,1.21)
(1.2,1.44)
(1.3,1.69)
(1.4,1.96)
(1.5,2.25)
(1.6,2.56)
(1.7,2.89)
(1.8,3.24)
(1.9,3.61)
(2,4)
```

Уровни обучения программированию в PascalABC.NET

PascalABC.NET позволяет использовать несколько уровней при обучении программированию, отличающихся **расстоянием** до старого Паскаля.

- **1 уровень.** Старый Паскаль. Только базовые возможности. Программы совместимы с Turbo Pascal, Free Pascal. Данный уровень **не рекомендуется для использования** поскольку не имеет будущего.
- **Использование уровня 1** совершенно недопустимо при самостоятельном обучении: обучаемый сразу отбрасывает себя на 20 лет назад. Использование уровня 1 допустимо если таковы требования учителя (который привык работать в старых Паскаль-системах) и при решении задач ЕГЭ (если есть опасения, что недобросовестные проверяющие могут снизить балл за использование возможностей PascalABC.NET).
- **2 уровень.** Расширения PascalABC.NET, связанные с внутриблочными переменными и автоопределением типа. **Минимально рекомендуемый уровень программирования на PascalABC.NET**, отвечающий современным требованиям к коду.
- **3 уровень.** Использование стандартных подпрограмм и методов, встроенных в типы.
- **4 уровень.** Использование классов стандартной библиотеки.
- **5 уровень.** Использование цепочечных методов последовательностей и лямбда-выражений.