Язык программирования PascalABC.NET 3.0 2015 год

Обзор новых возможностей

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

	2014-15						2013-14					
	Школьный		Окружной		Региональный *		Школьный		Окружной		Региональный	
Всего участников (>0 баллов)	5740	100,00%	1932	100,00%	478	100,00%	5492	100,00%	2061	100,00%	471	100,00%
* - учтены все участники												
PascalABC.Net	2723	47,44%	552	28,57%	59	12,34%	1906	34,71%	765	37,12%	78	16,56%
FPC	881	15,35%	433	22,41%	58	12,13%	1907	34,72%	552	26,78%	45	9,55%
Дельфи	43	0,75%	30	1,55%	9	1,88%	99	1,80%	26	1,26%	24	5,10%
Все паскали	3608	62,86%	944	48,86%	102	21,34%	3783	68,88%	1402	68,03%	129	27,39%
g++	557	9,70%	368	19,05%	222	46,44%	348	6,34%	58	2,81%	158	33,55%
gcc	225	3,92%	87	4,50%	22	4,60%	157	2,86%	11	0,53%	20	4,25%
clang++	34	0,59%	20	1,04%	9	1,88%	41	0,75%	306	14,85%	13	2,76%
clang	8	0,14%	4	0,21%	4	0,84%	41	0,75%	64	3,11%	3	0,64%
Все C/C++	811	14,13%	462	23,91%	244	51,05%	564	10,27%	371	18,00%	188	39,92%
Python-3	580	10,10%	379	19,62%	174	36,40%	339	6,17%	214	10,38%	162	34,39%
Python-2	38	0,66%	22	1,14%	8	1,67%	52	0,95%	21	1,02%	13	2,76%
Все питоны	611	10,64%	395	20,45%	179	37,45%	382	6,96%	235	11,40%	174	36,94%
Кумир-1	308	5,37%	31	1,60%	0	0,00%	455	8,28%	68	3,30%	2	0,42%
Кумир-2	77	1,34%	18	0,93%	0	0,00%						
Все кумиры	378	6,59%	44	2,28%	0	0,00%	455	8,28%	68	3,30%	2	0,42%
Qbasic (fbc)	240	4,18%	20	1,04%	0	0,00%	258	4,70%	52	2,52%	1	0,21%
Visual Basic	34	0,59%	10	0,52%	2	0,42%	22	0,40%	17	0,82%	1	0,21%
FBC-32	3	0,05%	0	0,00%	0	0,00%	0	0,00%	3	0,15%	0	0,00%
Все бейсики	277	4,83%	30	1,55%	2	0,42%	279	5,08%	72	3,49%	2	0,42%
C#	122	2,13%	62	3,21%	15	3,14%	72	1,31%	45	2,18%	10	2,12%
Java	58	1,01%	46	2,38%	13	2,72%	27	0,49%	20	0,97%	11	2,34%
php	18	0,31%	7	0,36%	1	0,21%	30	0,55%	9	0,44%	1	0,21%
perl	7	0,12%	3	0,16%	1	0,21%	8	0,15%	3	0,15%	2	0,42%
ruby	0	0,00%	1	0,05%	1	0,21%	3	0,05%	2	0,10%	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

Стандартный Free Pascal

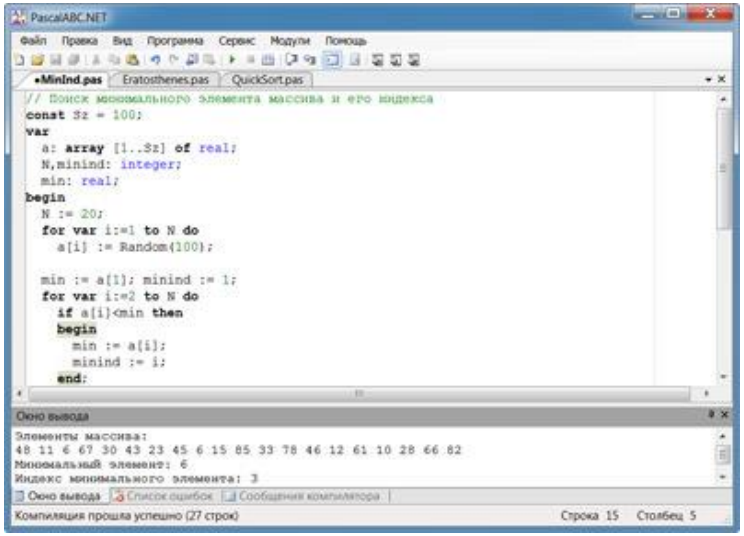
- Слухи о простоте Free Pascal сильно преувеличены. Примеры из документации «современного FP» (2015 г.):

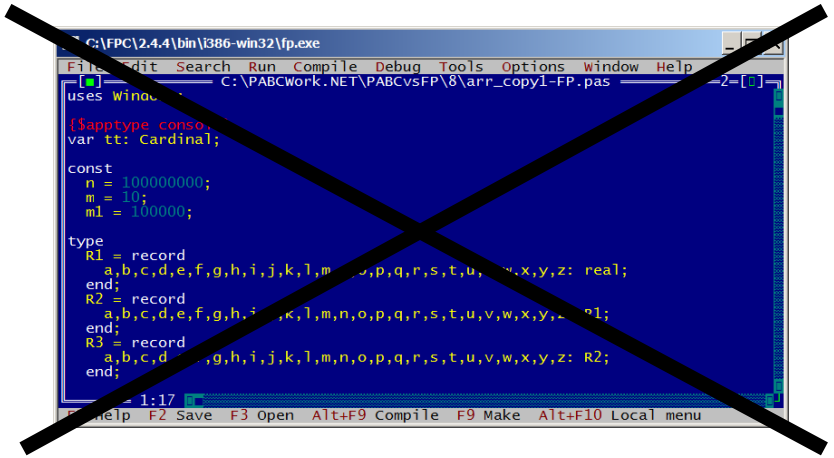
```
Procedure DoB(Out B : Integer);
begin
  B:=2;
end;

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 работы 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 vs Python

PascalABC.NET драматически опережает Python по скорости работы программ.

Ниже приводится пример с предыдущего слайда.

Python 3.0

```
import time

t1 = time.time()

n = 10000
s = 0.0
for i in range(1,n+1):
    for j in range(1,n+1):
        s += 1.0/(i*j)

print(time.time() - t1)
```

Время выполнения (Core I5-2500): 29.5 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

Это в 50 раз быстрее

PascalABC.NET и Linux

- Интегрированная среда PascalABC.NET запускается только под Windows.
- В Linux можно использовать консольный компилятор PascalABC.NET, интегрировав его в редактор Geany. Должна быть установлена последняя версия Mono
- Описание установки под Linux: http://pascalabc.net/wiki/index.php?title=Как_инсталлировать_PascalABC.NET_под_Linux.

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 можно, но не рекомендуется
- Программы в стиле старого Паскаля менее эффективны по скорости работы
- Ещё раз: работая в PascalABC.NET, надо писать в стиле PascalABC.NET. Далее мы обоснуем это многочисленными примерами кода
- Обычно на слайде слева будет содержаться 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

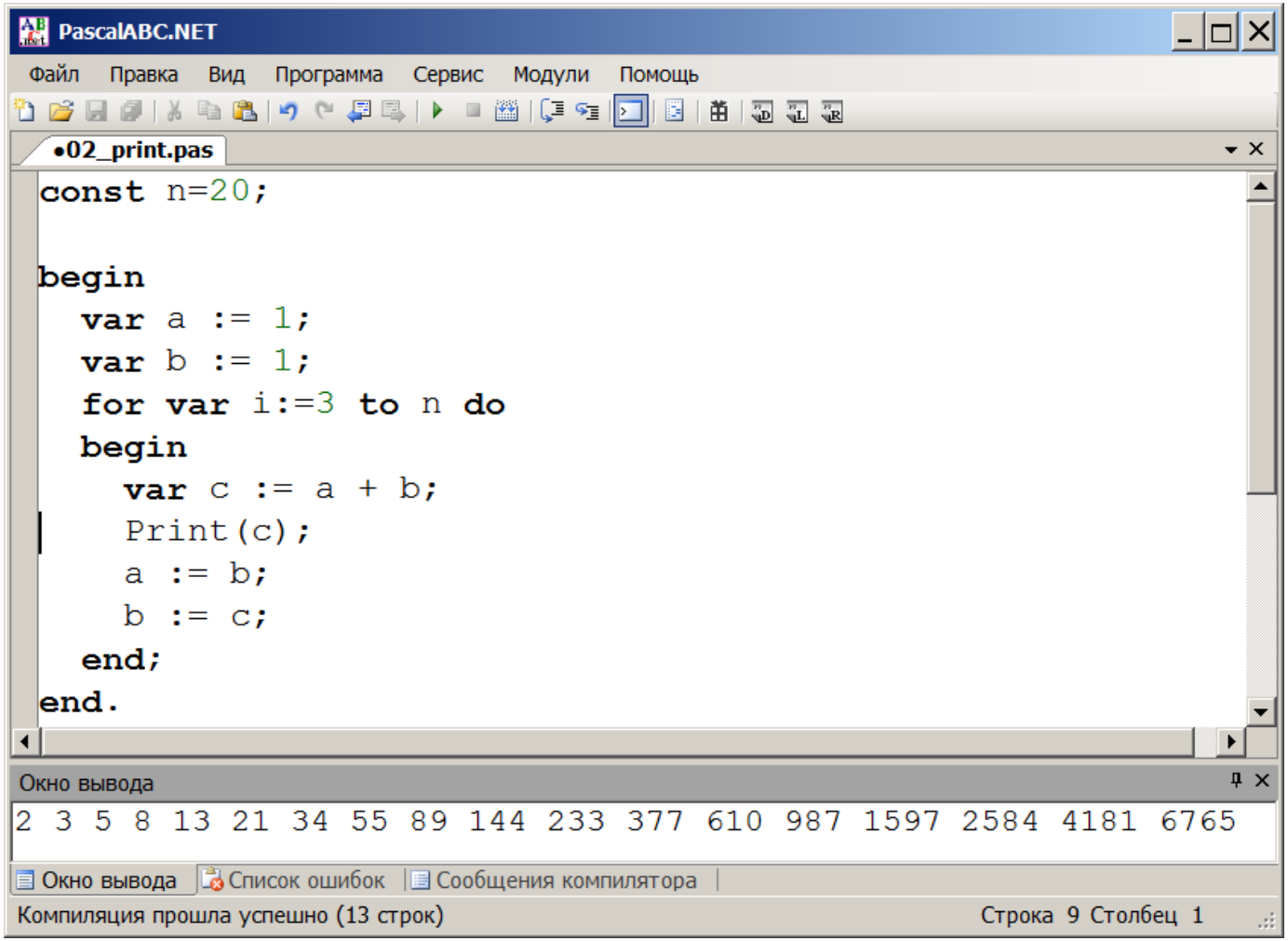
```
var a,b,t,vmin,vmax: integer;
begin
  write('Введите a:');
  readln(a);
  write('Введите b:');
  readln(b);
  if a<b then
    vmin := a
  else vmin := b;
  if a>b then
    vmax := a
  else vmax := b;
  writeln(vmin,' ',vmax);
  t := a;
  a := b;
  b := t;
  writeln(a,' ',b);
end.
```

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.
```

Использование Print в цикле

Print удобно использовать в цикле для вывода последовательностей.



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 – {}.

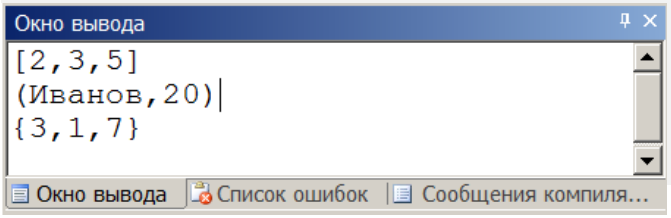
На устаревшем Паскале для вывода составных типов необходимо писать нагруженный деталями код.

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.
```



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

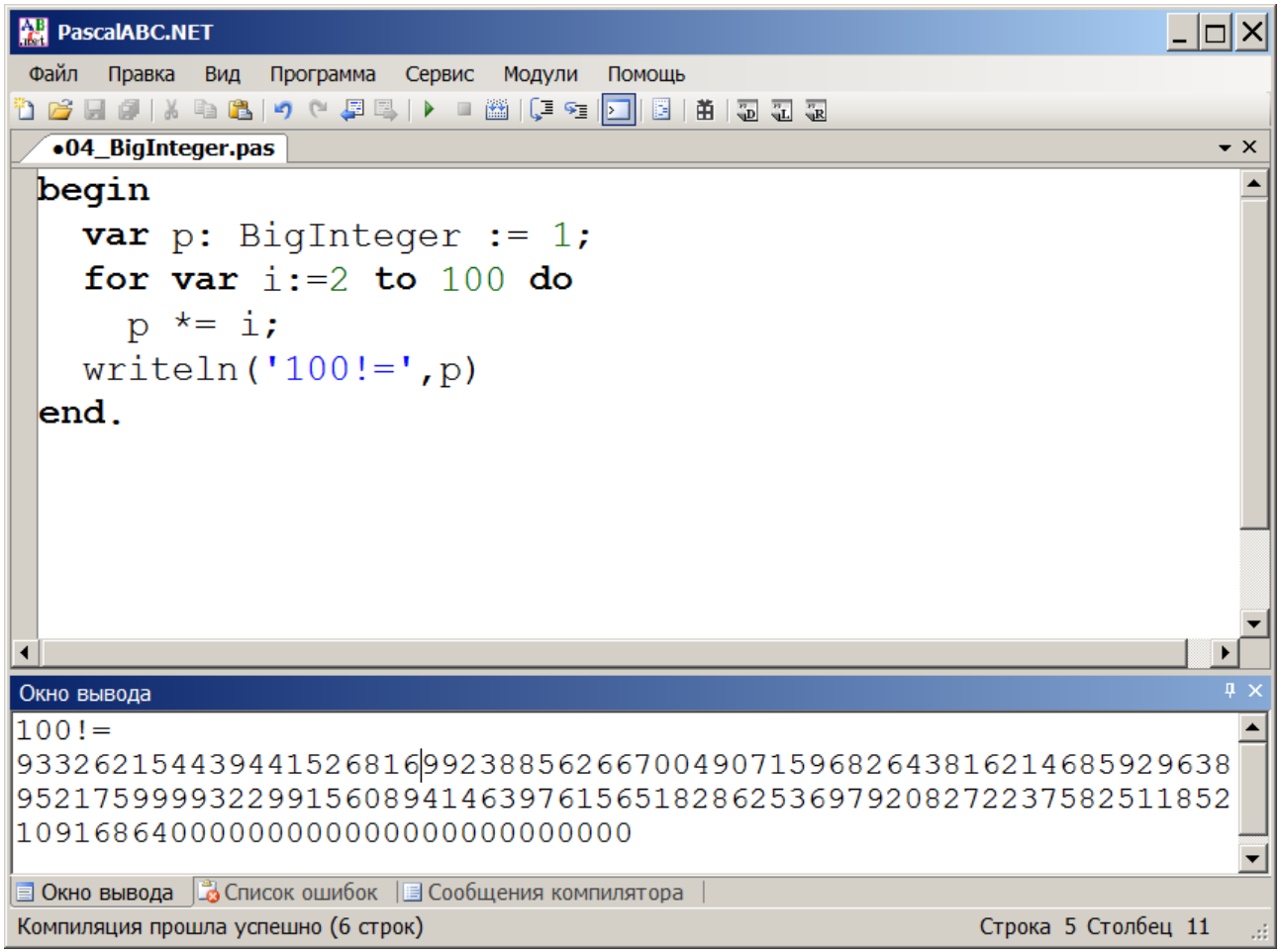
```
var Country: string;

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.
```

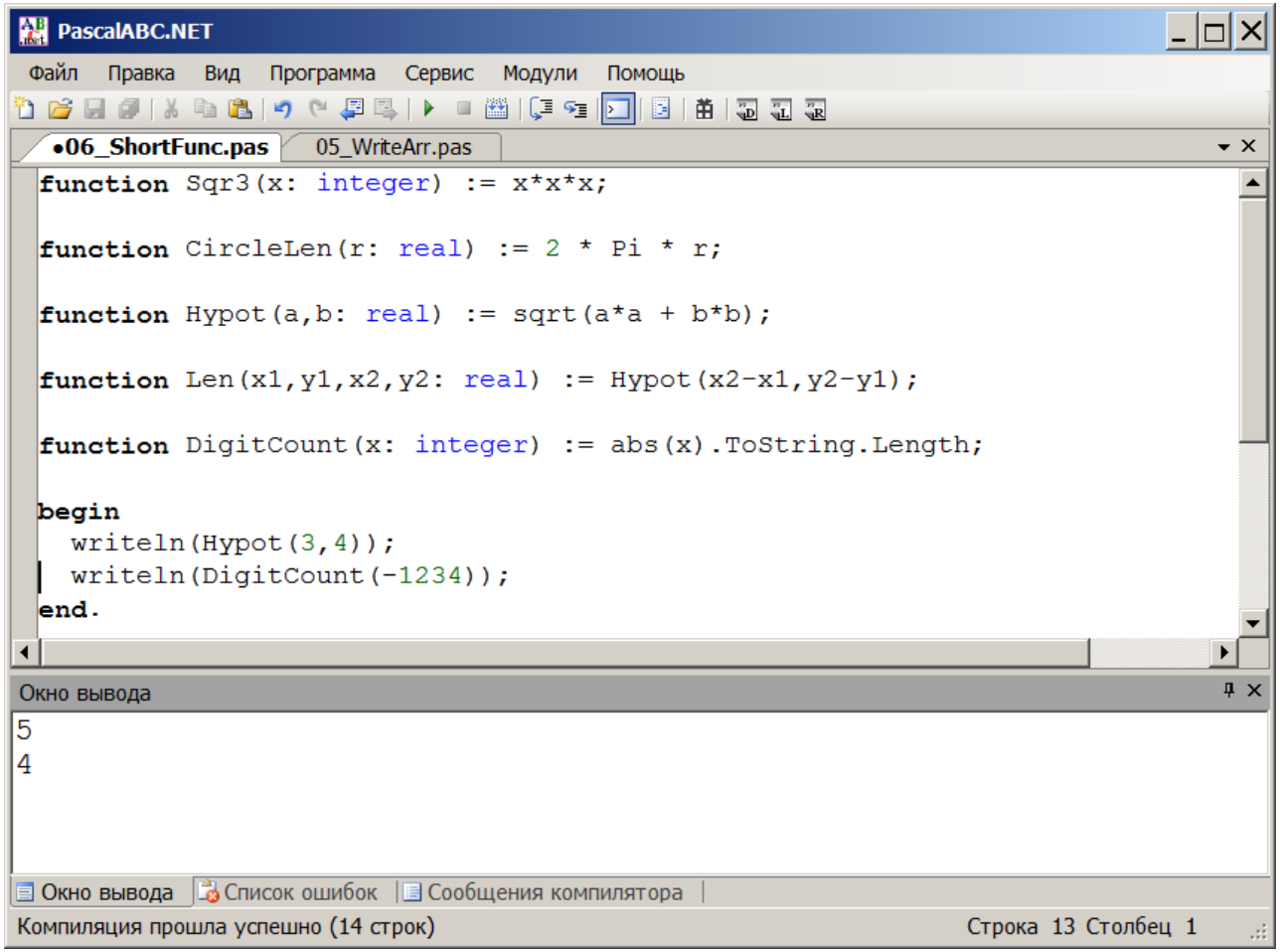
BigInteger

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.



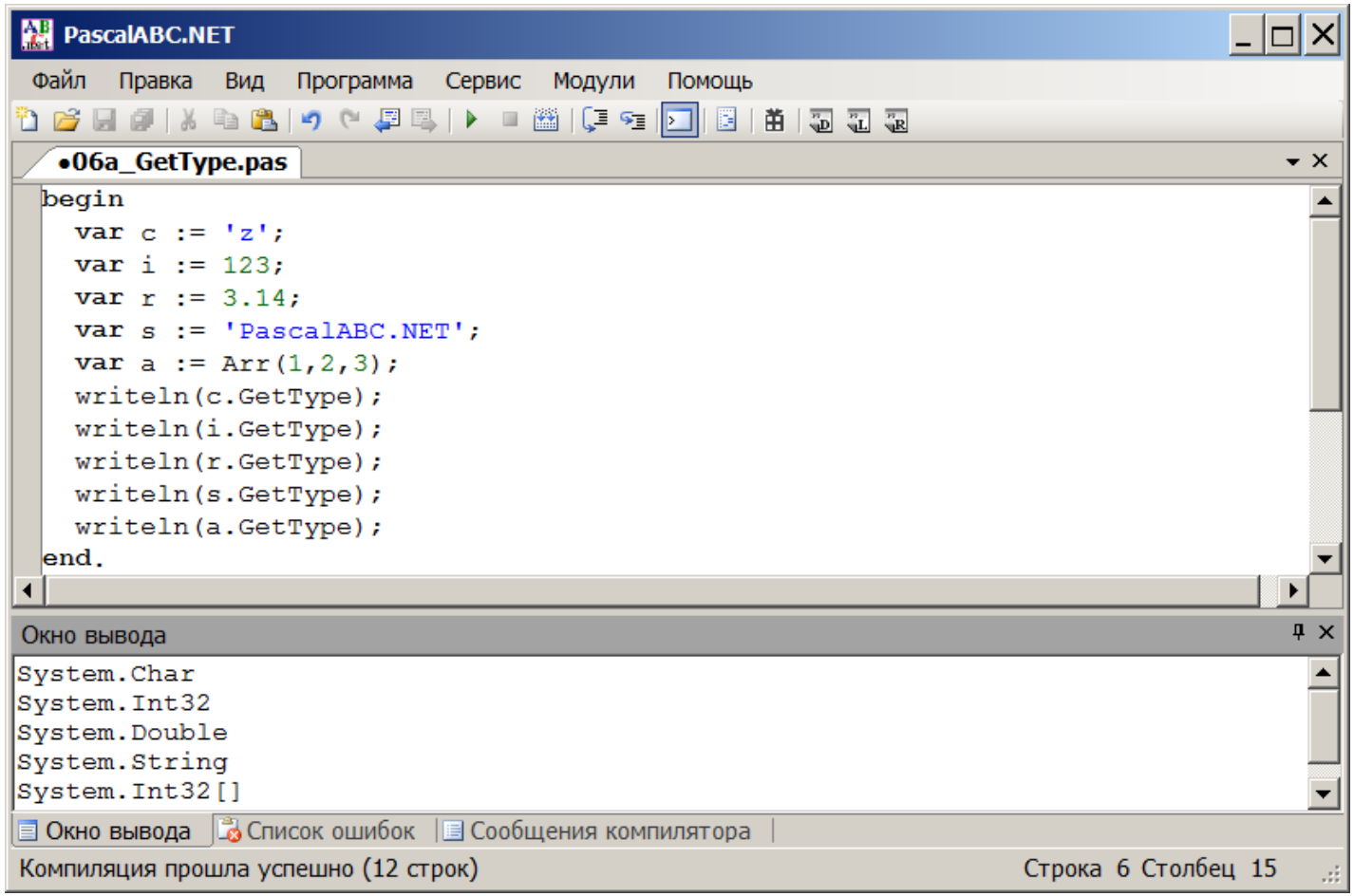
Короткие определения функций

В PascalABC.NET допускаются короткие определения для функций, задаваемых одним выражением. Для начинающих такой способ легче и позволяет написать множество простых функций, не путаясь с недостающими begin-end.



Все типы содержат методы

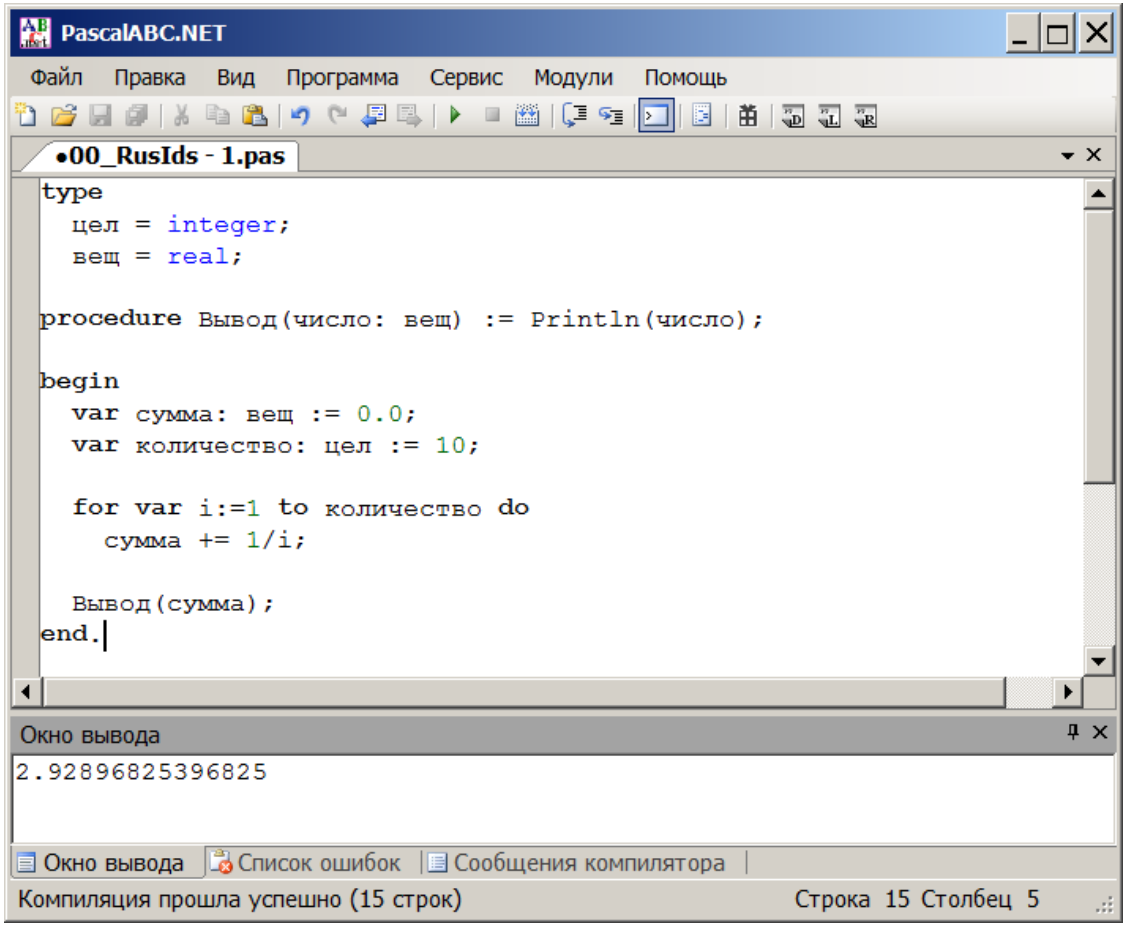
В PascalABC.NET все типы, в том числе и базовые (такие как integer, real), содержат ряд методов. В частности, это позволяет определить тип каждой переменной при выполнении, обратившись к методу GetType.



Русские идентификаторы

В PascalABC.NET можно использовать русские идентификаторы

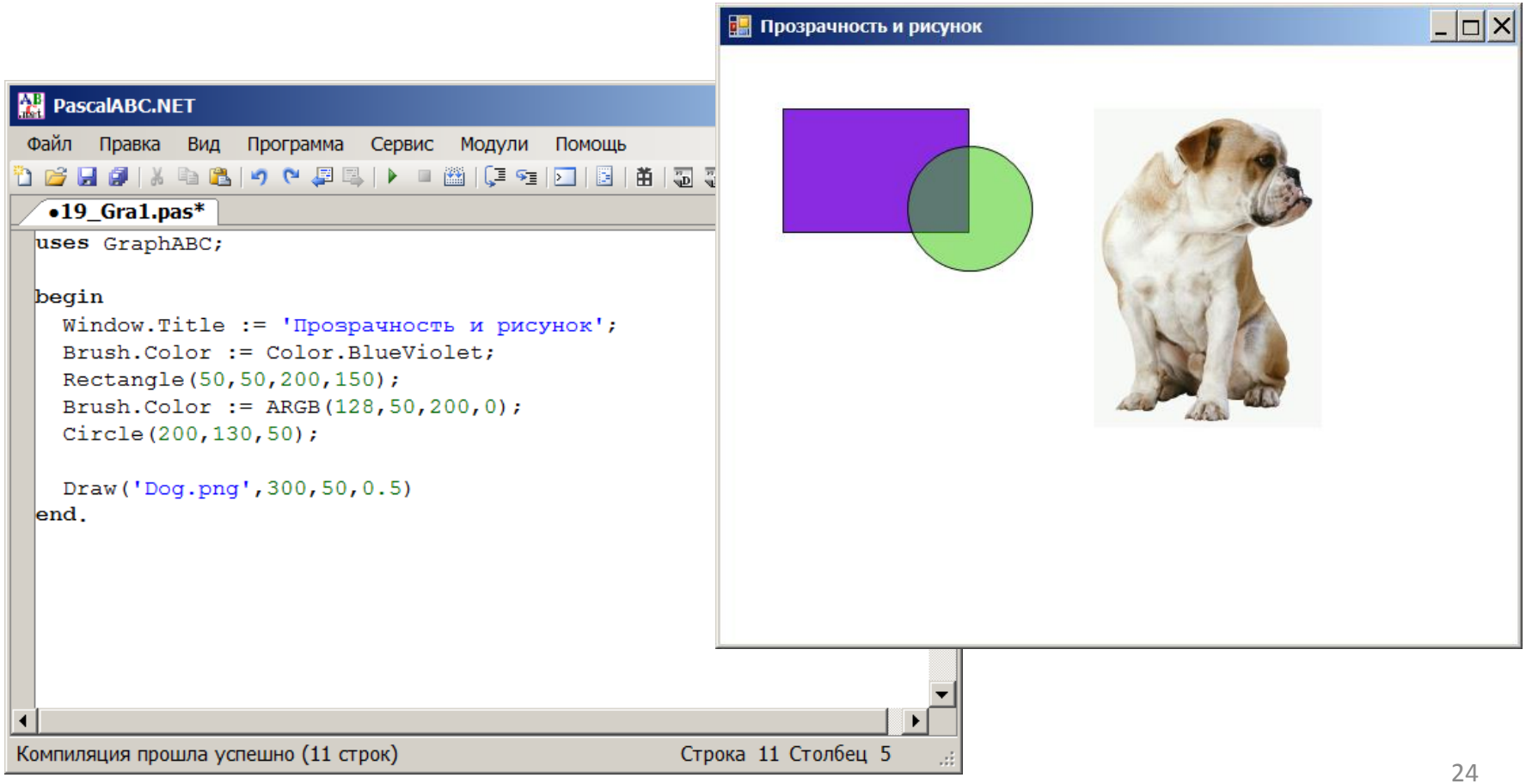
Вот как выглядит текст программы после небольших переопределений:



Графический модуль

В PascalABC.NET имеется простой по использованию и мощный по возможностям графический модуль GraphABC.

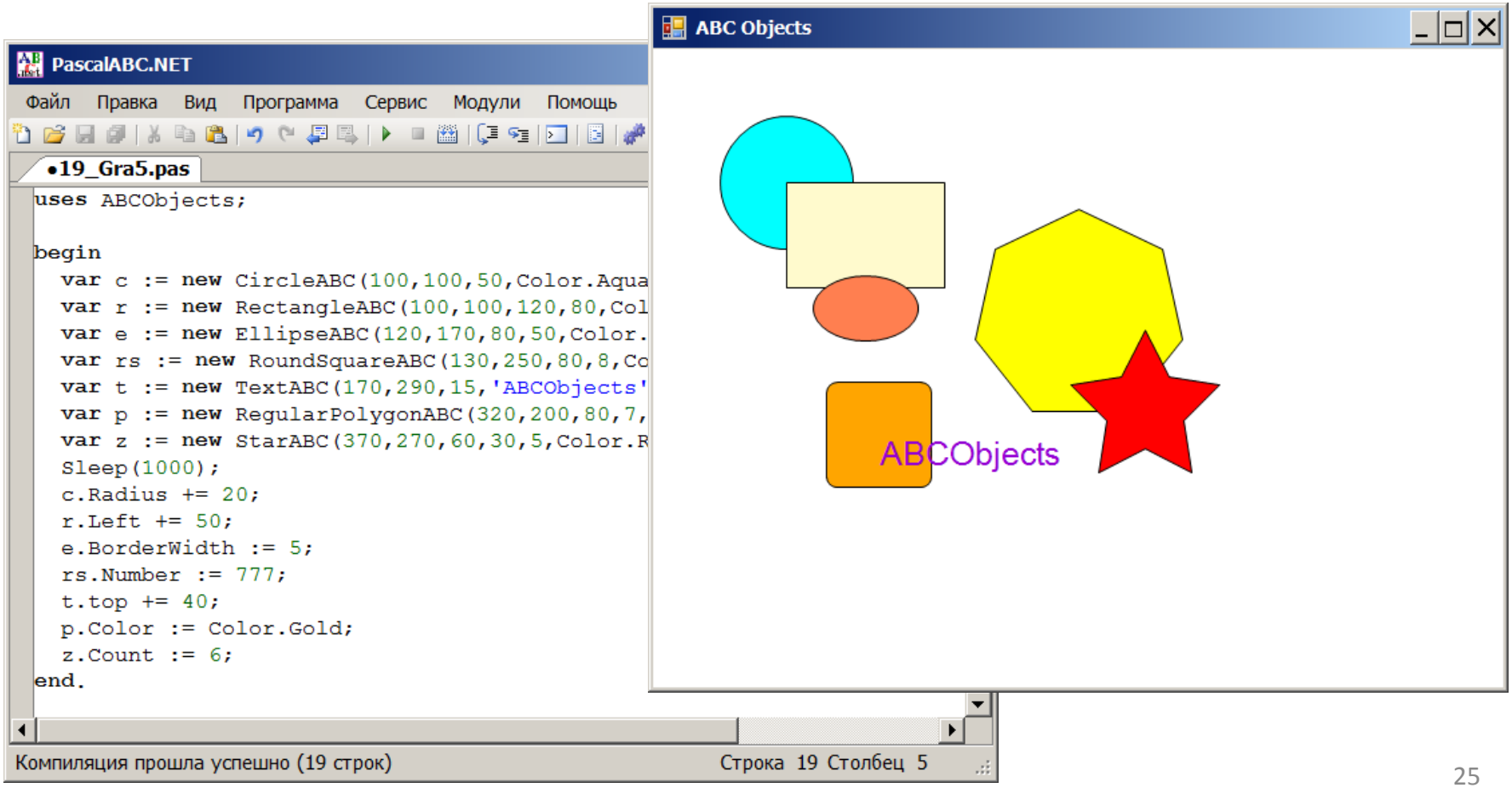
На скриншоте – пример с прозрачностью и выводом изображения.



Модуль векторной графики

В PascalABC.NET имеется простой по использованию и мощный по возможностям модуль векторной графики ABCObjects. Он позволяет создавать векторные графические объекты, при управлении свойствами которых осуществляется их правильная перерисовка на экране, не затрагивающая другие объекты.

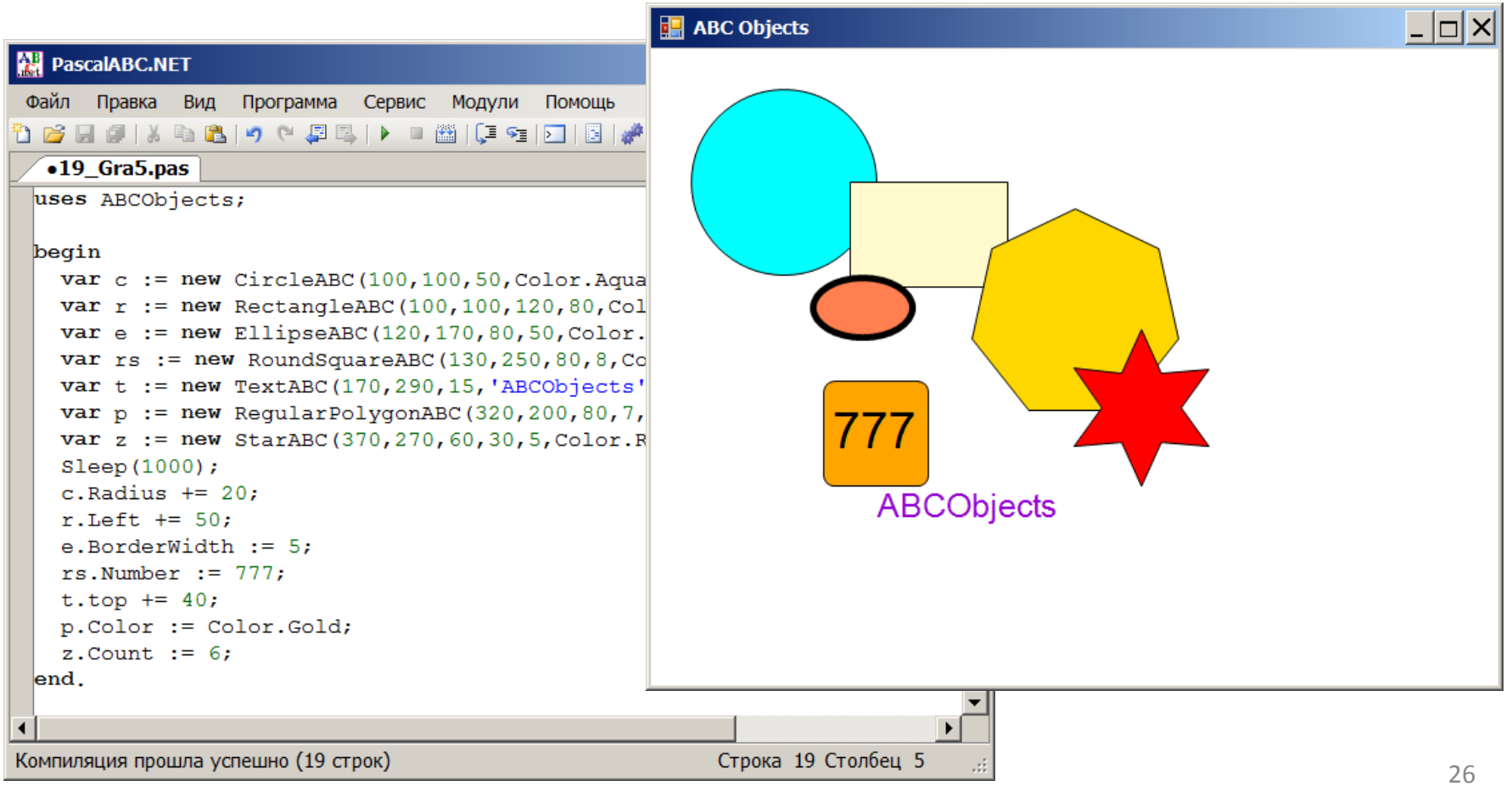
На скриншоте – пример с различными ABC объектами.



Модуль векторной графики

В PascalABC.NET имеется простой по использованию и мощный по возможностям модуль векторной графики ABCObjects. Он позволяет создавать векторные графические объекты, при управлении свойствами которых осуществляется их правильная перерисовка на экране, не затрагивающая другие объекты.

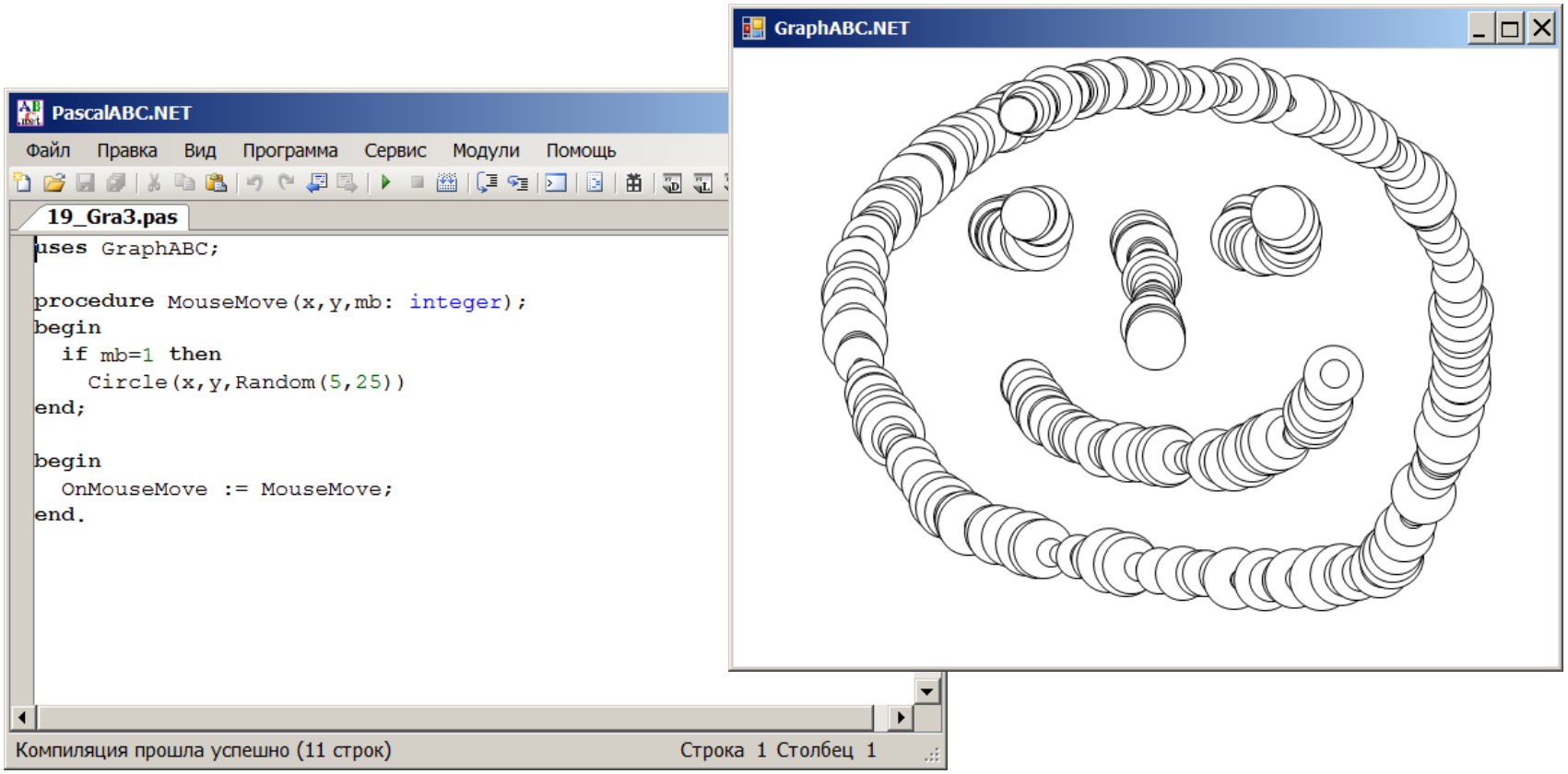
На скриншоте – пример с различными ABC объектами. После создания они меняют свойства: круг увеличивает радиус, прямоугольник и текст перемещаются, эллипс меняет толщину обводки, многоугольник меняет цвет и звезда из пятиконечной становится шестиконечной, в скругленном квадрате появляется число 777.



События мыши

Простые приложения, управляемые событиями, занимают несколько строк.

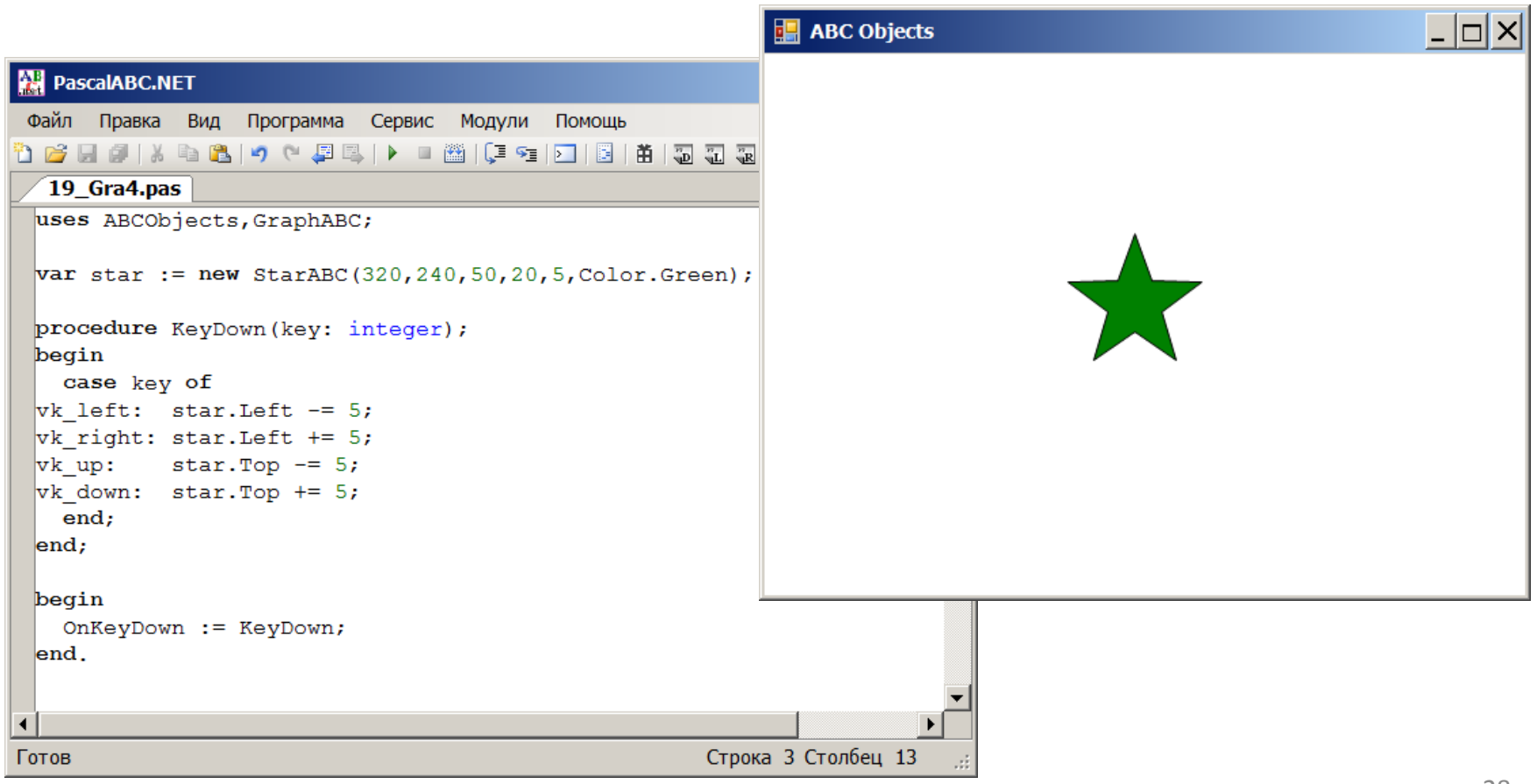
На скриншоте приведена программа рисования окружностями, обрабатывающая событие перемещения мыши (9 строк!).



События клавиатуры

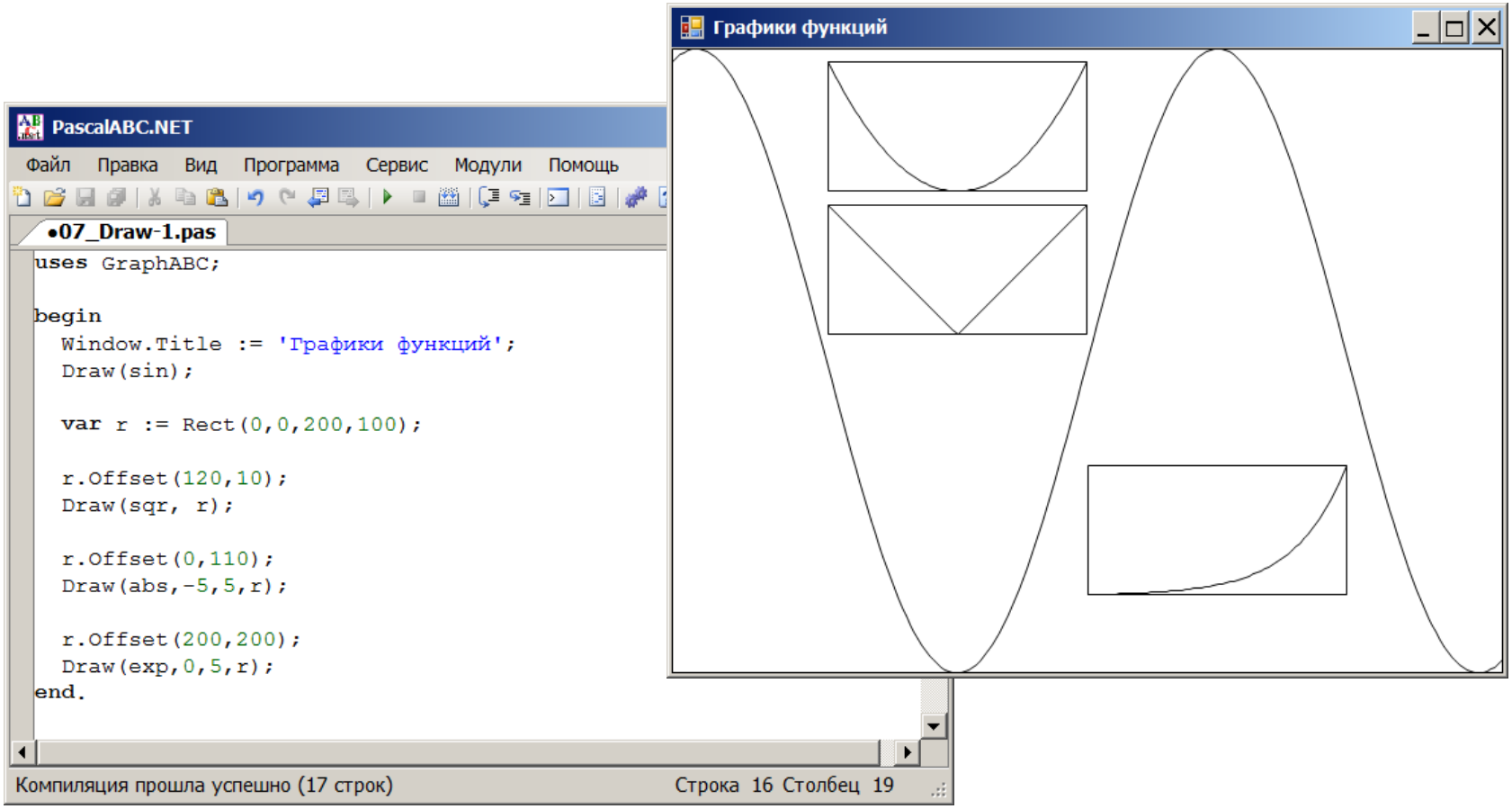
Так же просто пишутся программы, обрабатывающие события клавиатуры.

На скриншоте представлена программа, перемещающая звезду по нажатию стрелок. Сама звезда является векторным графическим объектом, при управлении свойствами Left и Top которого он меняет свои координаты.



Рисование графиков

- Графики функций в прямоугольнике рисуются очень просто, что может быть полезно при изучении темы «Функции».
- Обратите внимание, что команда рисования графика sin на полном экране предельно проста: Draw(sin)



Записи. Функция Rec

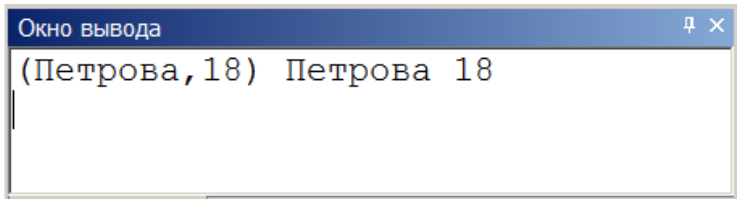
В PascalABC.NET можно создавать записи «на лету» с помощью функции Rec. Поля записи, возвращаемой функцией Rec, именуются последовательно: Item1, Item2 и т.д.

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 := Rec('Петрова',18);
  Println(p, p.Item1, p.Item2);
end.
```



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

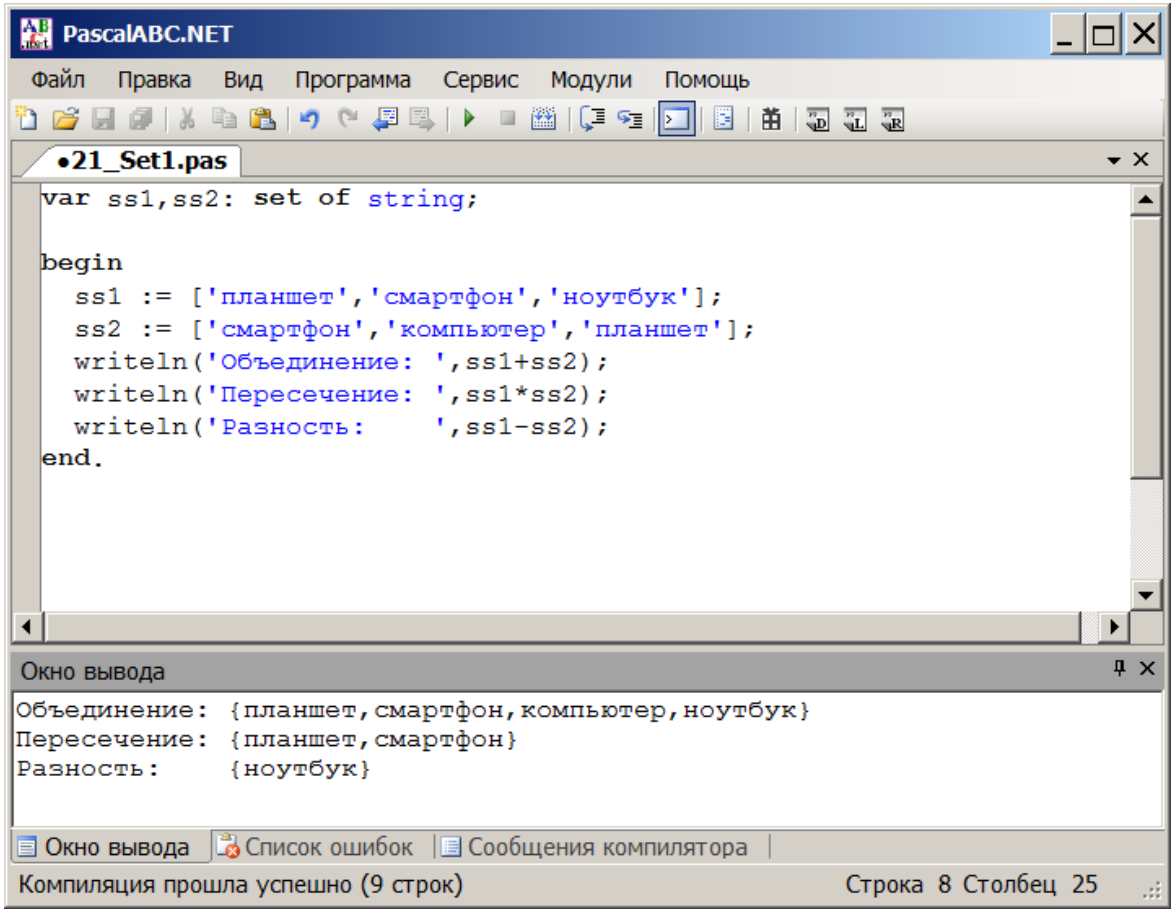
```
type
  Matr = array [1..10,1..10] of integer;

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.
- Операции над множествами удобно изучать на множествах строк.



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

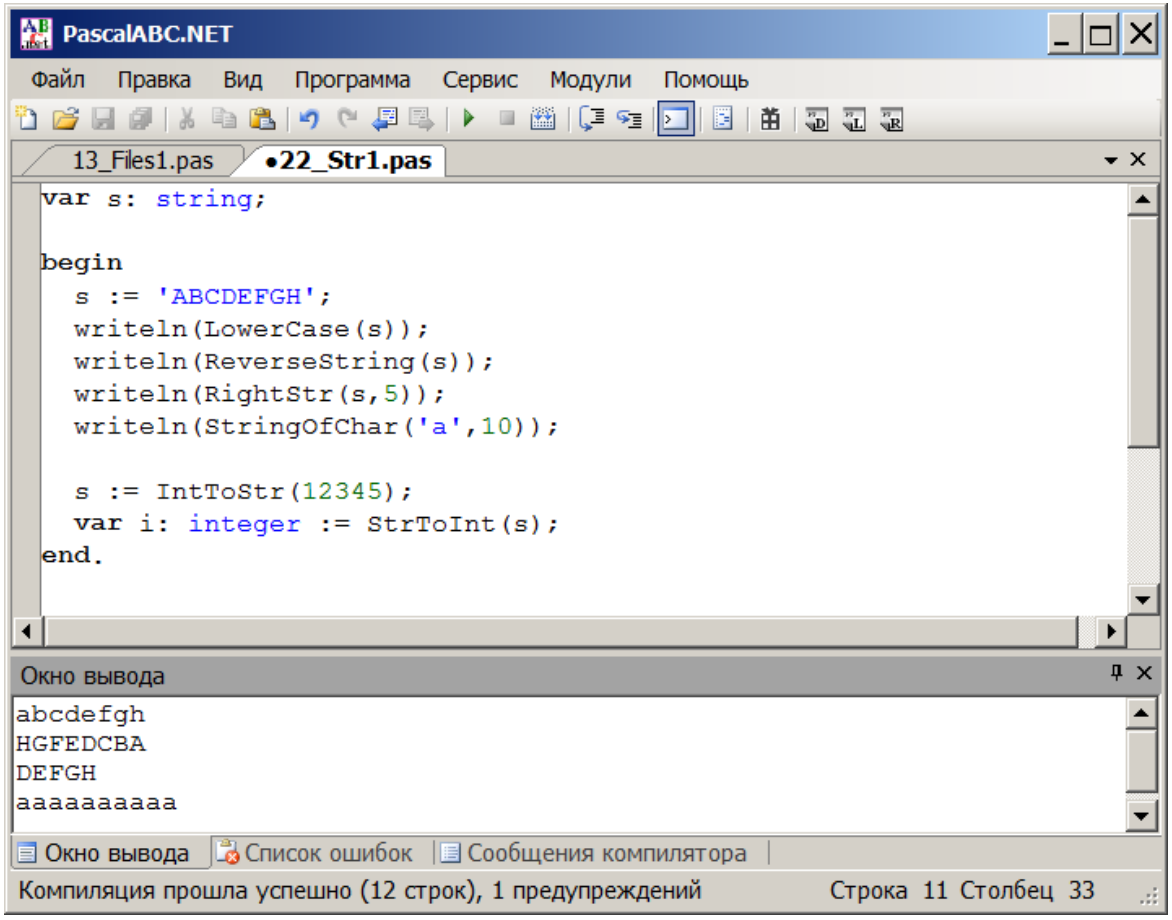
```
var s: string;
    i: integer;
begin
  s := 'ABCDEFGH';
  s := s + 'IJK';
  for i:=1 to Length(s) do
    write(s[i],' ');
  writeln;
  Str(12345,s);
  s := '';
  for i:=1 to 10 do
    s := s + 'a';
  writeln(s);
end.
```

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.
```

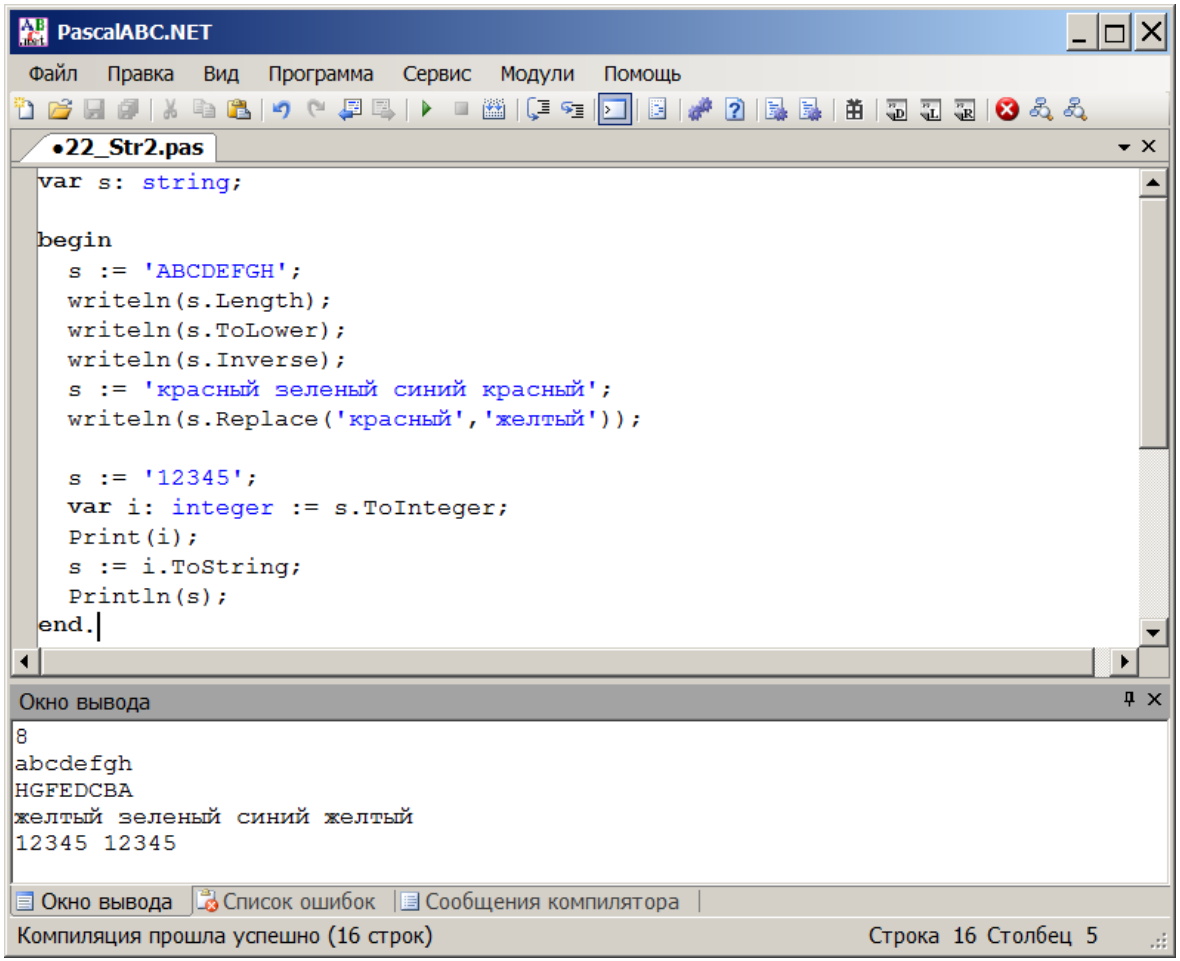
Строки. Стандартные подпрограммы

Стандартные подпрограммы Length, Copy, Insert, Delete, Pos широко используются при работе со строками – их смысл не изменился. Ряд новых функций – LowerCase, UpperCase, ReverseString, StringOfChar, IntToStr, StrToInt, TryStrToInt – взят из Delphi; их можно использовать, не подключая специальных модулей.



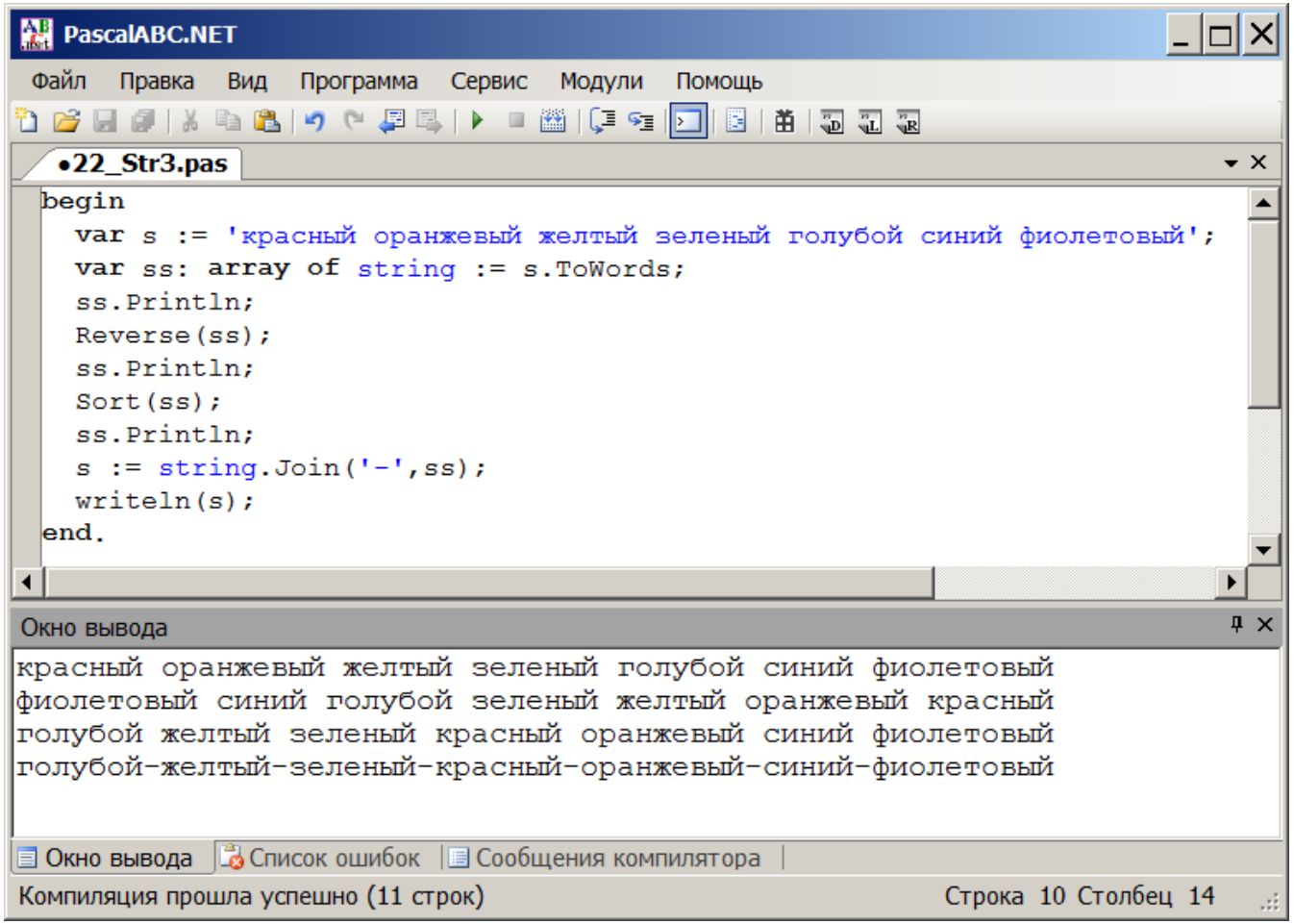
Методы строк – дополнение к стандартным подпрограммам

В строках имеется ряд стандартных методов, которые дополняют стандартные процедуры и функции.



Строки. Разбиение на слова

Стандартный метод ToWords позволяет разбить строку на слова, превратив ее в массив строк.
На старом Паскале данное действие требовало написания непростого алгоритма.
Собрать строку из массива строк можно с помощью статического метода string.Join.



Файлы

Методы, встроенные в файловые переменные, а также функции Open... открытия файла меняют стиль программирования при работе с файлами, делая его короче и понятнее.

Старый Паскаль

```
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.
```

PascalABC.NET

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

Файлы. Функция ReadLines

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. Ниже – два решения вывода текста программы на экран.

Использование последовательностей не загружает файл целиком в память, а позволяет в каждый момент хранить в памяти одну строку файла.

PascalABC.NET

```
begin
  foreach var s in ReadLines('13_Files4.pas') do
    writeln(s);
end.
// Обидно, но это - всё
```

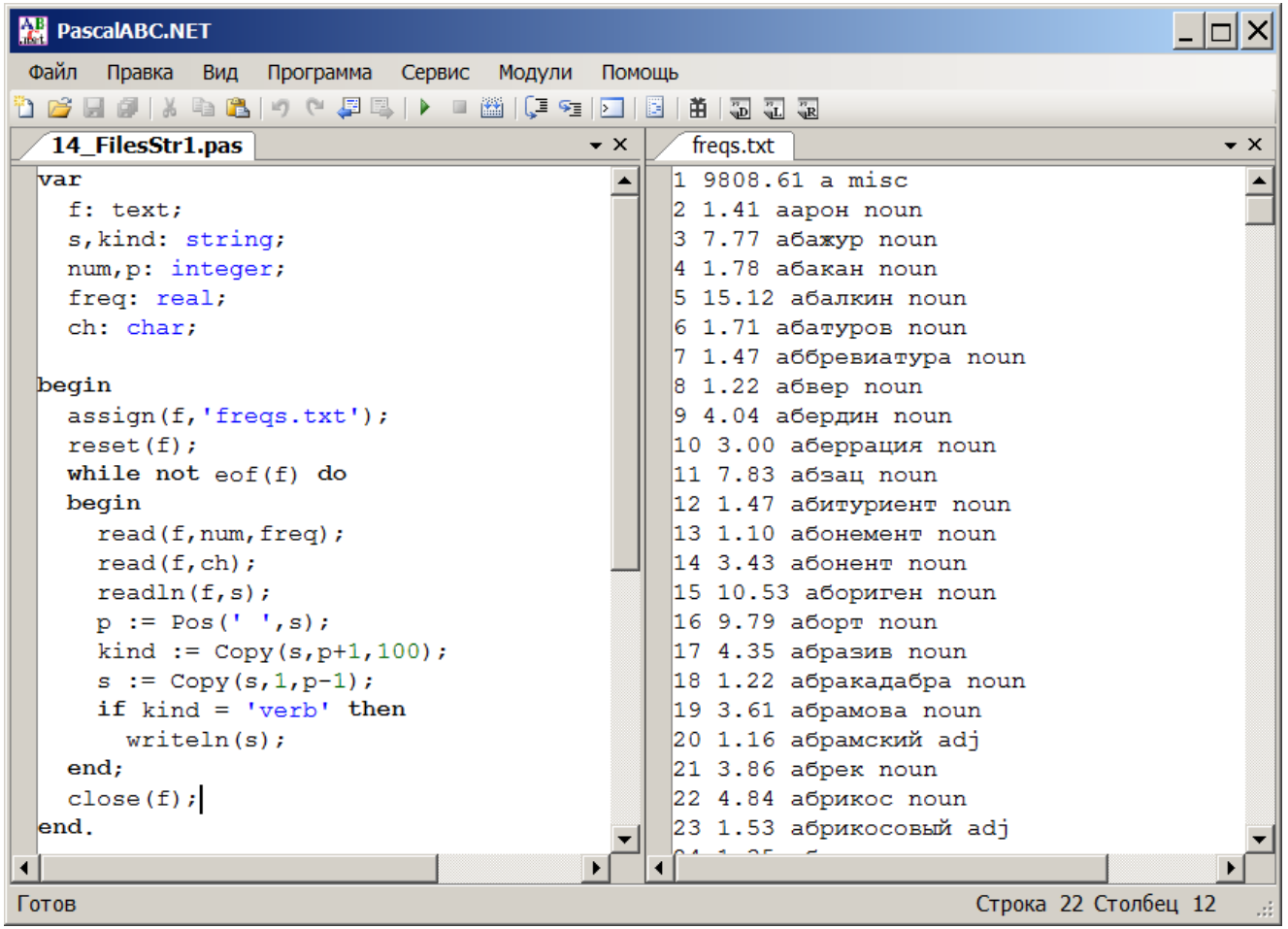
PascalABC.NET

```
begin
  ReadLines('13_Files3.pas').Print(NewLine);
end.
// Обидно, но это – ещё короче
```

Обработка строк в файлах. **Старый Паскаль**

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

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



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.
```

Списки List

- Динамических массивов недостаточно для решения ряда задач, поскольку они не умеют автоматически менять свой размер в ходе выполнения программы.
- В стандартном модуле имеется класс List (список), являющийся по-существу динамическим массивом, но имеющий более мощные возможности.
- Добавление в конец – наиболее распространенная задача, где списки проще динамических массивов.
- Добавление в конец, вставка в середину и удаление из середины реализованы как методы List.
- Списки List являются обобщенными: они могут создаваться с произвольным типом элементов, например: List<integer> или List<string>.

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 в PascalABC.NET 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.
```

Списки List. Вставка и удаление

Вставка в середину и удаление из начала списка – встроенные методы, меняющие размер списка.

Статические и динамические массивы для этой задачи неэффективны: необходимо хранить специальную переменную – заполненность массива. Кроме того, данные методы требуется писать руками.

Старый Паскаль

```
const m = 20;

var a: array [1..m] of integer;
    n,i: integer;
begin
  n := 10;
  for i:=1 to n do
    a[i] := Random(100);
  for i:=1 to n do
    write(a[i],' ');
  writeln;

  n := n + 1;
  for i:=n downto 5 do
    a[i+1] := a[i];
  a[5] := 777777;

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

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

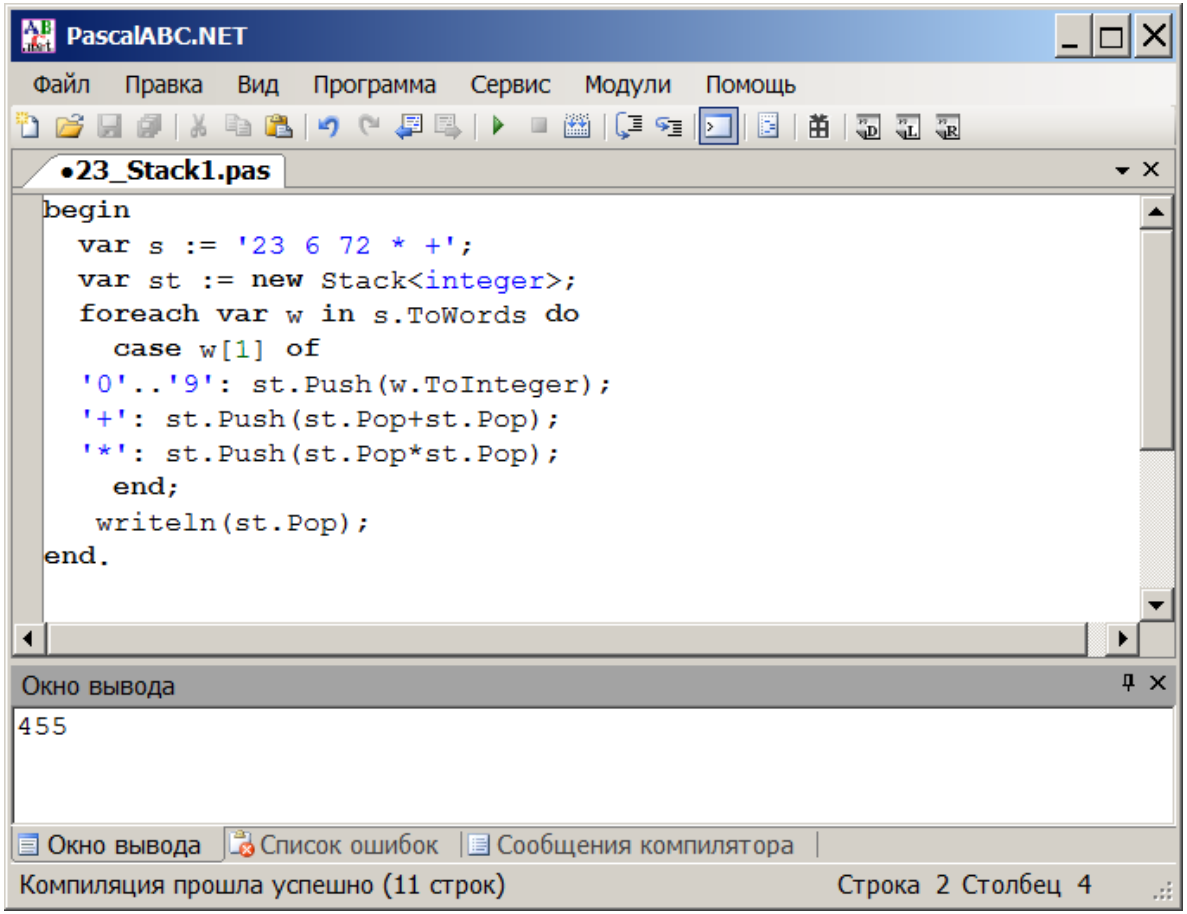
PascalABC.NET

```
begin
  var l := new List<integer>;
  l.AddRange(SeqRandom());
  l.Println;
  l.Insert(5,777777);
  l.Println;
  l.RemoveAt(0);
  l.Println;
end.
```

Стеки

В стандартном модуле имеется класс Stack. Стек – набор данных с операциями Push (втолкнуть на вершину стека) и Pop (вытолкнуть из вершины стека).

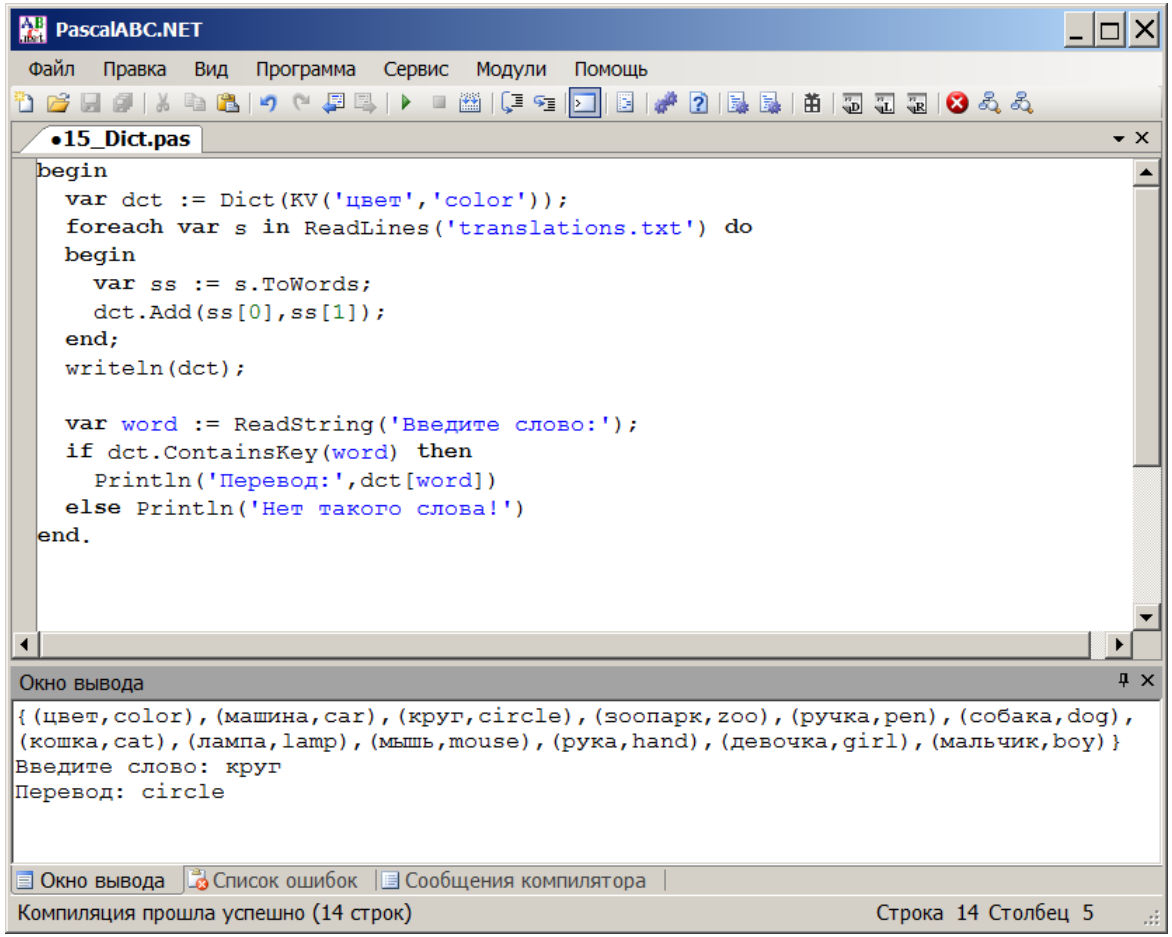
Ниже приведено решение классической задачи о вычислении выражения в польской инверсной записи (ПОЛИЗ).



Словари

Словарь – набор пар элементов вида (Ключ,Значение). Основное действие – поиск значения по ключу. Словарь можно создать функцией Dict с параметром – набором пар. Пара возвращается функцией KV.

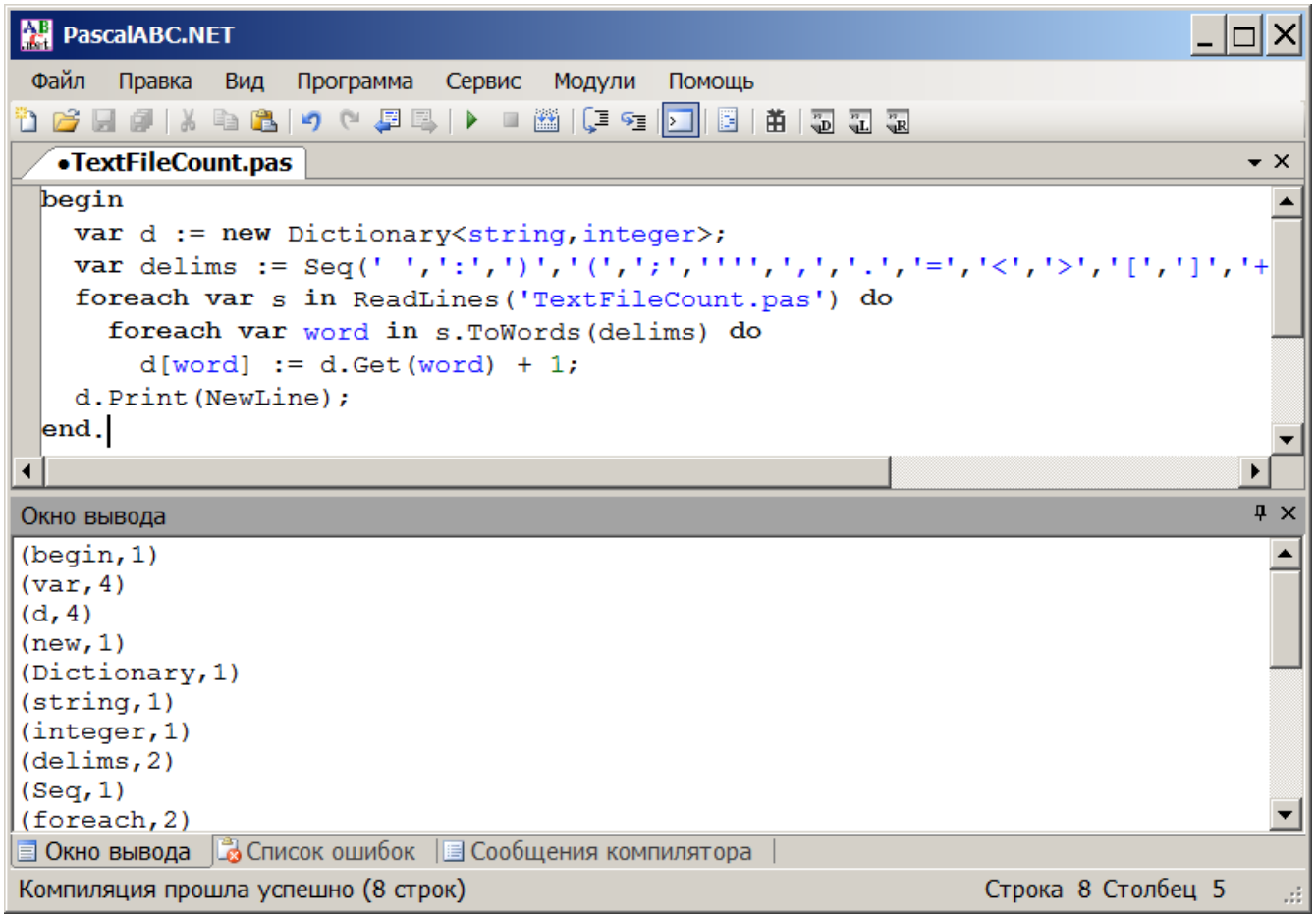
В примере на скриншоте из файла в словарь считываются слова и их переводы на английский, после чего можно обращаться к словарю и искать перевод слова.



Словари. Частотный словарь

С помощью словарей просто решается задача о подсчете всех слов в файле и для каждого слова – количества его повторений.

Вначале мы считываем файл в строку, потом разбиваем его на слова, используя delims в качестве разделителей. Наконец, мы записываем все слова в словарь и выводим его. Всё!

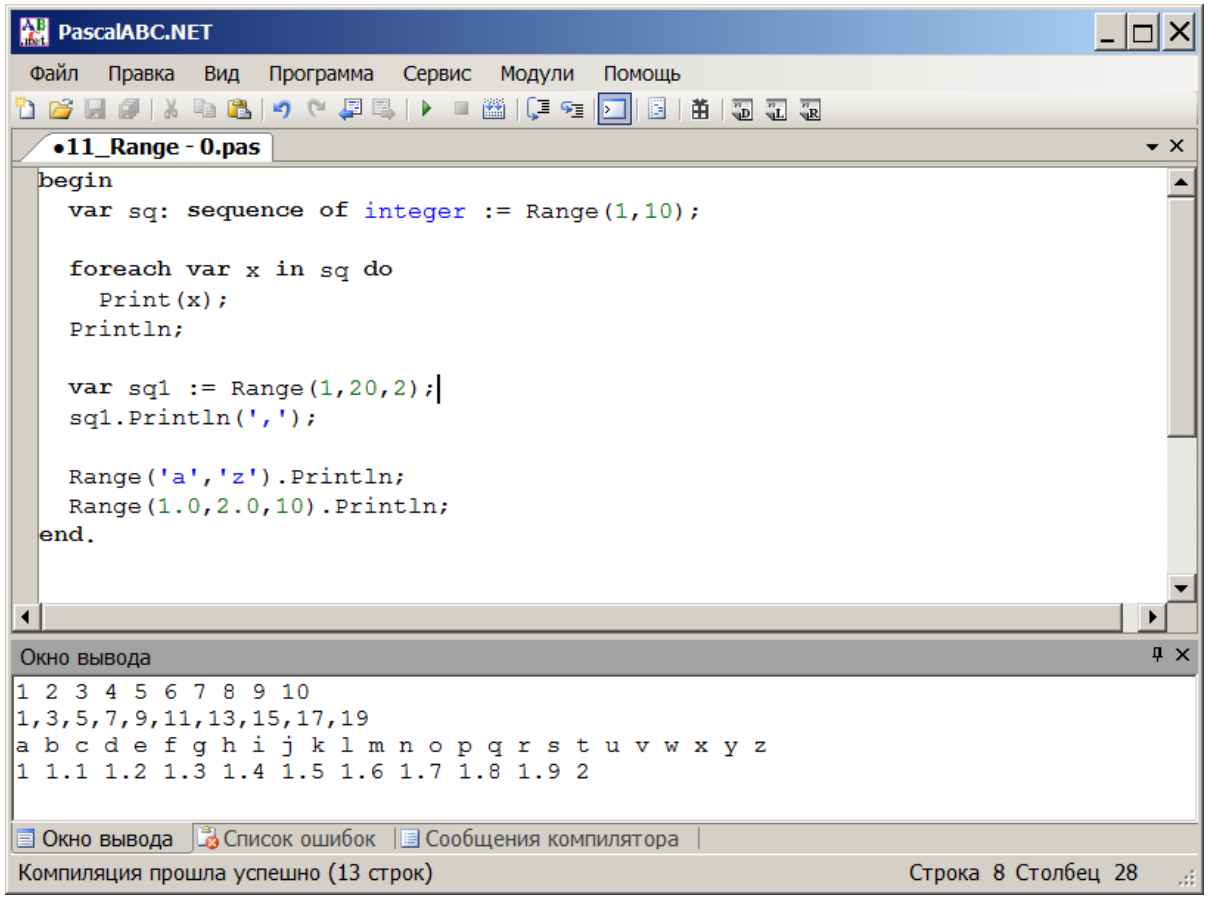


Последовательности

Последовательность в PascalABC.NET – это специальный тип **sequence of** T, представляющий собой обобщение массива, списка, множества. Элементы последовательности можно перебрать от первого до последнего (например, с помощью цикла **foreach**).

Массивы, списки, множества являются последовательностями. Но бывают и «просто» последовательности, представляющие по существу алгоритм получения элементов последовательности.

Для создания последовательностей – арифметических прогрессий используется функция Range, для вывода элементов последовательности – метод Print.



Процедурные переменные

В PascalABC.NET в переменных можно хранить процедуры и функции и передавать их в качестве параметра. Эта возможность присутствует в Delphi и Free Pascal, однако именно в PascalABC.NET она широко используется

Пример 1

```
function MyFun(x: real): real := 3*x-2;

begin
  var f := MyFun;
  writeln(f(2));
  f := sqrt;
  writeln(f(2));
end.
```

В программе используется переменная f, хранящая вещественную функцию.

Ей последовательно присваиваются функции MyFun и sqrt (стандартная), которые затем вызываются косвенно через переменную f

Пример 2

```
procedure PrintTable(f: RealFunc; a,b: real; n: integer);
begin
  var h := (b-a)/n;
  var x := a;
  for var i:=0 to n do
  begin
    writeln(x:3:1,f(x):7:2);
    x += h;
  end;
end;

begin
  PrintTable(sqr,0,5,10);
end.
```

Лямбда-выражения

Лямбда-выражения представляют собой функции, создаваемые «на лету». Они облегчают написание и восприятие текста программы. Лямбда-выражения присутствуют практически во всех современных распространенных языках программирования.

Тип простейшего лямбда-выражения имеет вид: T1->T2, что означает функцию с одним параметром типа T1, возвращающую значение типа T2.

Простейшее лямбда-выражение имеет вид: x -> 2*x+1 и задаёт функцию f(x)=2*x+1.

Ниже приведены примеры с предыдущего слайда, реализованные в виде лямбда-выражений.

Пример 1

```
var f: real->real;

begin
  f := x -> 3*x-2;
  writeln(f(2));
  f := x -> sqrt(x+1);
  writeln(f(2));
end.
```

Использование лямбда-выражений позволяет не описывать множество коротких функций, которые используются в программе один раз

Пример 2

```
procedure PrintTable(f: real->real; a,b: real;
                     n: integer);
begin
  var h := (b-a)/n;
  var x := a;
  for var i:=0 to n do
  begin
    writeln(x:3:1,f(x):7:2);
    x += h;
  end;
end;

begin
  PrintTable(x->x*x+x-1,0,5,10);
end.
```

В старом Паскале лямбда-выражения отсутствуют в принципе

Примеры использования лямбда-выражений в стандартных библиотеках

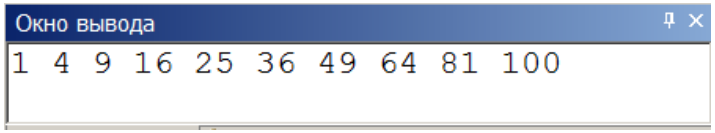
Ряд подпрограмм стандартной библиотеки в качестве параметров принимают процедурные переменные. При их вызове на этих местах можно использовать лямбда-выражения.

Пример 1

```
uses GraphABC;

begin
  Draw(x->x*sin(x));
end.
```

Пример 2

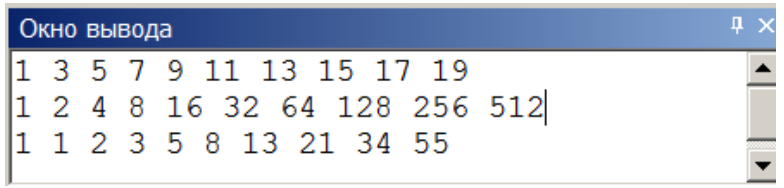
```
begin
  SeqFill(10,i->i*i).Println;
end.
```



SeqFill возвращает последовательность из 10 значений i*i, начиная с i=1

Пример 3

```
begin
  SeqGen(1,x->x+2,10).Println;
  SeqGen(1,x->x*2,10).Println;
  SeqGen(1,1,(x,y)->x+y,10).Println;
end.
```



SeqGen генерирует рекуррентные последовательности. Две первые формы с помощью лямбда-выражения задают вычисление следующего элемента по предыдущему (нечетные числа и степени двойки), а последняя форма – вычисление следующего элемента по двум предыдущим (числа Фибоначчи)

Метод Select для последовательностей

Многие методы для последовательностей (в частности, динамических массивов) содержат в качестве параметров процедурные переменные. Эта группа методов – стандартная для .NET и называется **LINQ to Objects**.

При вызове этих методов практически всегда используются лямбда-выражения.

Одним из наиболее распространенных методов является метод Select, применяющий функцию к каждому элементу последовательности и возвращающий новую последовательность.

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

Окно вывода

```
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

Метод Where для последовательностей

Другим распространенным методом для последовательностей является метод Where, обеспечивающий фильтрацию элементов по условию и возвращающий последовательность элементов исходной последовательности, удовлетворяющих фильтру.

Фильтр является функцией с одним параметром, возвращающей boolean.

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.
```

Окно вывода

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

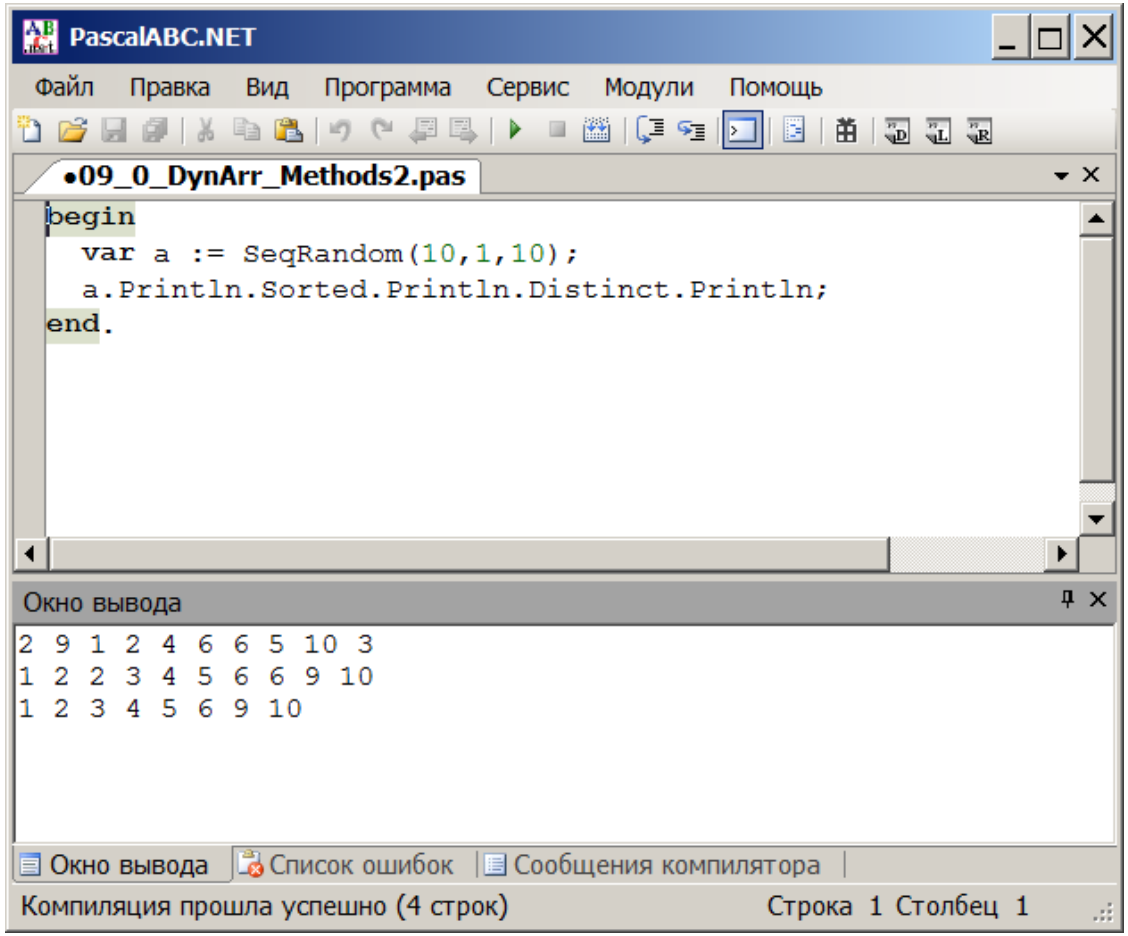
В программе используется лямбда-выражение x->x mod 2 <> 0, представляющее собой функцию-условие, определяющую нечетность числа.

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

Цепочки методов для последовательностей

Многие методы представляют собой функции, возвращающие последовательности, поэтому к ним можно применять другие методы, образуя цепочки методов.

На скриншоте случайная последовательность вначале выводится, затем сортируется и снова выводится, после чего из неё удаляются дубли (Distinct), и она снова выводится.



Range. Замена циклов

Функция Range, генерирующая последовательность, фактически является заменой циклов.

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.
```

Окно вывода

```
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->Rec(x,x*x))
    .Println(NewLine);
end.
```

Окно вывода

```
(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)
```

Range. Метод Монте-Карло для вычисления числа π

Задача. Вычислить число π методом Монте-Карло.

Решение. Генерируем множество случайных точек в единичном квадрате. Считаем отношение количества точек, попавших в четверть единичной окружности с центром в вершине квадрата, к общему количеству точек n. При больших n эта величина примерно площади четверти единичной окружности, то есть π/4

Старый Паскаль

```
Напишите код сами
Желтым по синему ☺
```

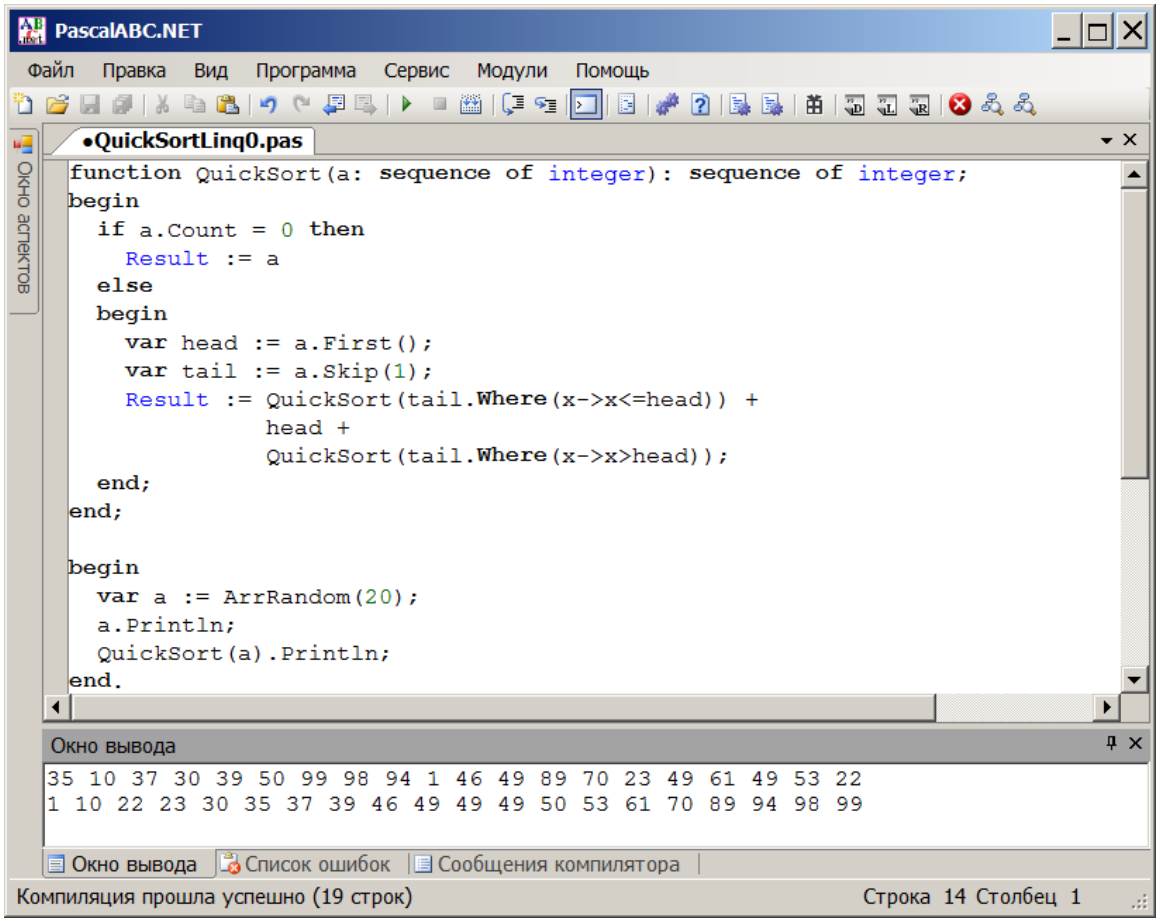
PascalABC.NET

```
begin
  var n := 10000000;
  var pp := Range(1,n)
    .Select(x->Rec(Random(),Random()))
    .Where(p->sqr(p.Item1)+sqr(p.Item2)<1)
    .Count/n*4;
  Print(pp);
end.
```

Быстрая сортировка

На данном слайде приводится решение задачи о быстрой сортировке, выполненное с использованием методов последовательностей Where, Skip и First.

Суть этого решения: пишется функция сортировки, в которой берется первый элемент последовательности, после чего все элементы разделяются на две части: те, которые меньше или равны этому элементу, и те, которые больше него. Затем к каждой части применяется та же функция сортировки.

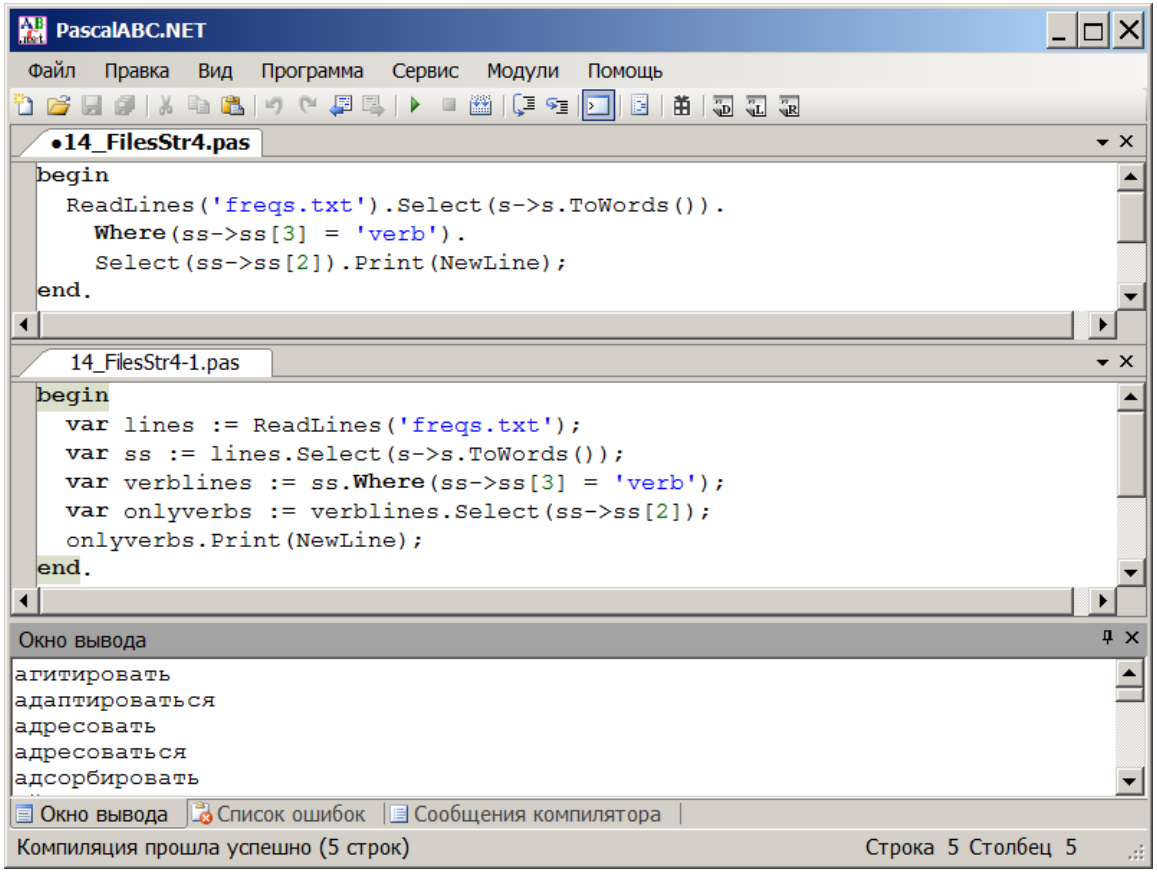


Список глаголов с помощью Select и Where

Задача о выводе списка глаголов из файла freqs.txt, решенная нами ранее, имеет простое решение с помощью методов Select и Where.

Вначале файл с помощью ReadLines превращается в последовательность строк, затем каждая строка разбивается на слова. Затем отбираются только те строки, у которых третье слово равно ‘verb’, после чего осуществляется проекция всей строки на само слово-глагол, и полученная последовательность выводится.

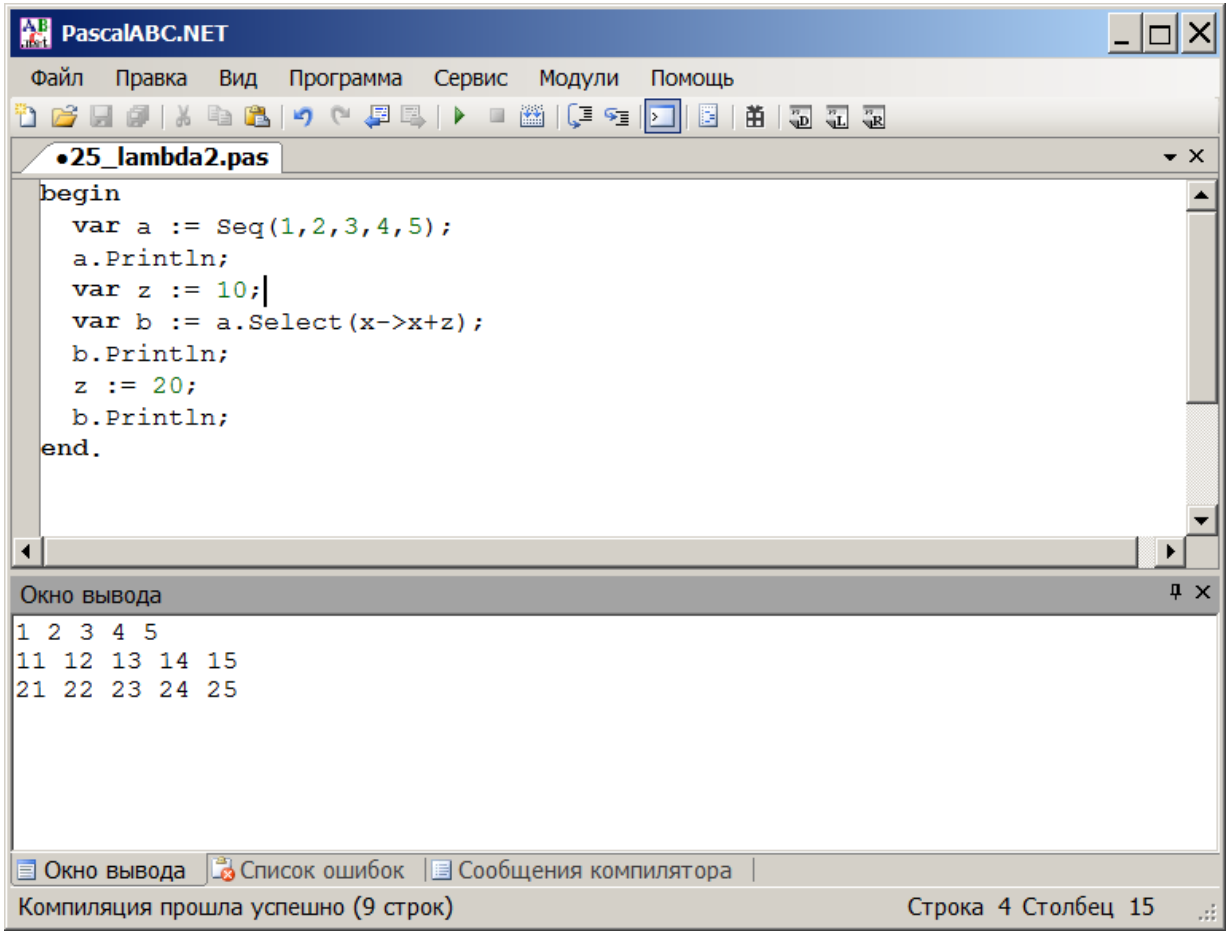
Ниже приведены два решения – с цепочечным и с пошаговым выполнением запросов.



Захват переменной

Использование в лямбда-выражении внешней переменной называется захватом этой переменной. В примере на скриншоте следует обратить внимание, что вычисление последовательности b происходит не в момент выполнения проекции a.Select(x->x*x), а в момент вывода на экран b.Println: при различных z результат вычисления разный.

Это иллюстрирует, что вычисления с последовательностями носят ленивый характер: вычисление последовательности b откладывается до выполнения команды Println.



Уровни работы с Паскалем 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.