Н.Б. Культин

Pascal Next 3.1

Описание языка программирования с примерами

Май 2024

Оглавление

Введение

Эта книга представляет собой описание нового языка программирования Pascal Next.

Книга адресована тем, кто знаком с основами программирования, знает с какой-либо язык программирования, имеет навык разработки компьютерных программ начального уровня.

Цель книги показать возможности языка программирования Pascal Next и интегрированной среды разработки Pascal Next.

Pascal Next

Pascal Next - среда разработки и компилируемый язык **программирования для начинающих программистов**, ориентированные на решение **задачи обучения основам программирования**.

В основе синтаксиса языка **Pascal Next** лежит синтаксис "классического" Pascal.

Компилятор Pascal Next создает выполняемый Win32 файл.

Загрузить среду разработки Pascal Next можно с сайта www.pascal-next.ru

Автор языка программирования Pascal Next, разработчик компилятора Pascal Next и среды разработки Pascal Next- Н.Б. Культин.

В настоящее время пользователям доступна версия Pascal Next 3.1.

Pascal Next 3.1 – что нового?

Язык программирования

- Можно обявить тип-массив
- Можно объявить запись
- Можно обявить массив записей
- В качестве параметров процедур и функций можно использовать массивы, в том числе массивы записей.
- Теперь для выбора варианта действия можно использовать инструкцию `case`
- Строковая константа может содержать управляющую последовательность \n.

Среда разработки

- Среда разработки запоминает 5 последних имен файлов, с которыми работал пользователь.

Структура программы

Программа PascalNext представляет собой последовательность разделов:

- раздел объявления меток
- раздел объявления именованных констант
- раздел объявления типов
- раздел объявления глобальных переменных
- раздел объявления функций и процедур программиста
- раздел объявления локальных переменных
- раздел выполняемых инструкций программы

Начало раздела объявления меток обозначается с ключевым словом `label`, раздела именованных констант - словом `const`, раздела типов – словом `type`, раздела глобальных переменных – словом `var`.

Раздел объявления функций и процедур не имеет особой метки (ключевого слова, обозначающего начало раздела). Он состоит из объявлений (текста) функций и процедур. Объявление функкции начинается ключевым словом `function`, объявление процедуры – словом `procedure`.

Радел объявления локальных переменных программы начинается ключевым словом `var`.

Раздел выполняемых инструкций начинается ключевым `begin` и заканчивается словом `end`, за которым находится точка.

В общем случае программа выглядит так:

```
program Имя()
  label
      // здесь объявления меток

  const
      // здесь объявления именованных констант

  type
      // здесь объявления типов пограммиста

  var
      // здесь объявления глобальных переменных

   // здесть функции и процедуры

  var
      // здесь объявления локальных переменных

  begin
   // здесь выполняемые инструкции программы

  end.
```

Структура простой программы

Программа Pascal Next представляет собой совокупность процедур и функций.

Главная процедура, с инструкций которой начинается выполнение программы, обозначается идентификатором program. Все остальные процедуры обозначаются идентификатором procedure, функции – идентификатором function.

Простейшая программа представляет собой одну единственную процедуру program и в общем случае выглядит так:

```
program имя()
var
    // здесь объявления переменных
begin
    // здесь выполняемые инструкции
end.
```

Пример:

```
// пересчет веса из фунтов в килограммы
program p1()
var
    fnt: float;  // вес в фунтах
    kg: float;   // вес в килограммах
begin
    write('Вес в фунтах >');
    readln(fnt);
    kg := fnt * 0.495; // 1 кг = 495 гр

    writeln(fnt:6:2, ' фнт. = ', kg:6:3, 'кг');

    writeln('Press <Enter>');
    readln;
end.
```

Перед разделом `var` может быть раздел `const` (раздел именованных констант), в который программист может поместить объявления констант, используемых в программе.

```
// пересчет веса из фунтов в килограммы
program p1()
```

```
const
    K = 0.495; // коэф. пересчета из фунтов в кг
var
    fnt: float;  // вес в фунтах
    kg: float;   // вес в килограммах
begin
    write('Вес в фунтах >');
    readln(fnt);
    kg := fnt * K;

    writeln(fnt:6:2, ' фнт. = ', kg:6:3, 'кг');

    writeln('Press <Enter>');
    readln;
end.
```

Типы данных

integer – целые числа в диапазоне -2 147 483 648 ... 2 147 483 647

float - положительные и отрицательные числа в диапазоне от $1.5x10^{-38}$ до $3.4x10^{38}$

string – строка символов длинной до 128 символов

тип, объявленный программистом

Переменные

Все переменные программы должны быть объявлены в разделе var той процедуры или функции, в которой они используются.

Числовые

Инструкция объявления числовой переменной целого или вещественного типа в общем случае выглядит так:

`имя: тип;`

где:

имя – имя переменной;

тип – тип данных.

Примеры:

```
  sum: float;
  k: integer;
```

Допускается одной инструкцией объявить несколько переменных одинакового типа, например:

```
a,b,c: float;
```

Строковые

Инструкция объявления строковой переменной в общем виде выглядит так:

имя: string[*длина*];

где:

длина – максимальное количество символов, которое может вместить переменная.

Максимальное допустимое значение параметра *длина* при объявлении строки – 128.

Пример:

```
name: string[25];
```

Допускается одной инструкцией объявить несколько переменных одинакового типа, например:

```
firstName, lastName: string[12];
```

При объявлении строковой переменной можно использовать целую именованную константу.

Например, если в разделе const объявлена целая именованная константа LN, то объявление переменных firstName и lastName может быть таким:

```
firstName, lastName: string[LN]; // LN – целая именованная константа
```

Имя переменной

В качестве имени переменной можно использовать любую, начинающуюся с буквы и состоящую из букв и чисел последовательность символов. Помимо букв и чисел имя переменной может содержать символы “подчеркивание”.

Пример:

```
amount: integer;
x1: float;
month_salary: float;
annual_income: float;
first_name: string[20];
```

Компилятор Pascal Next **не различает** прописные и строчные буквы, т.е. нечувствителен к регистру записи идентификаторов. Таким образом, например, идентификаторы `first_name`, `FIRST_NAME` и `First_Name` обозначают один и тот же объект (переменную).

В качестве имен переменных (и других объектов программы) нельзя использовать зарезервированные слова языка программирования, а также имена встроенных процедур и функций.

Константы

Числовая константа

Числовые константы записываются обычным образом.

Примеры целых констант:

```
123
-45
0
```

Примеры вещественных констант:

```
5.0
27542.15
25.7
-34.05
0.0
```

Строковая константа

Строковая константа представляет собой заключенную в одинарные кавычки последовательность любых символов.

Примеры строковых констант:

```
'Hello, World!'
'Bart Simpson'
'(C) Nikita Kultin, 2021'
'          '
''
'100'
'99.5'
```

Помимо обычных символов, строка (строковая константа) может содержать специальную последовательность символов `\n`.

Пример:

```
'Hello,\nWorld!'
'1. Алюминий\n2. Сталь\n3. Медь'
```

Последовательность `\n` при выводе строки на экран трактуется как команда продолжить вывод символов, следующих за последовательностью `\n`, с начала следующей строки.

Например, три инструкции

```
writeln('Корни уравнения:')
writeln('x1=', x1:6:2);
```

```
  writeln('x2=', x2:6:2);
```

можно заменить одной

```
  writeln('Корни уравнения:\nx1=', x1:6:2, '\nx2=', x2:6:2);
```

Именованные константы

Именованные константы должны быть объявлены в разделе `const` программы, процедуры или функции, в которой они используются.

Объявление именованной константы выглядит так:

```
  Имя = Значение;
```

Пример:

```
const
    Copyright ='(c) Nikita Kultin, 2021'; // строковая именованная константа
    PI = 3.1415925; // вещественная именованная константа
    HB = 7;         // целая именованная константа
    NL = 25;        // целая именованная константа
```

После объявления именованная константа может использоваться в программе как обычная константа, в том числе в разделе объявления переменных.

Пример использования именованных констант при объявлении переменных:

```
    matrix array[1..HB,1..HB] of float; // HB – именованная константа
    students array[1..HB] of string[NL]; // HB, NL – именованные константы
    name: string[NL]; // NL – именованная константа
```

Пример использования именованных констант в коде:

```
    sq := PI*r*r; // PI – именованная константа

    for i:=1 to HB do // HB – именованная константа
        for j:=1 to HB do
           matrix[i,j]:=0;
        end;
    end;
```

Логический тип

В Pascal Next нет логического (boolean) типа данных, однако, его легко можно смоделировать, определив в программе целые именованные константы TRUE (истина) и FALSE (ложь) со значениями `1` и

0 соответственно. После этого, вместо переменных логического типа можно использовать переменные целого типа, трактуя их как логические.

Пример

```
// псевдо-логический тип
program p()
const
    // "логические" константы
    TRUE = 1;
    FALSE = 0;

    HB = 10;
var
    a:array[1..HB] of integer; // массив чисел
    r: integer;                // число, которое надо найти в массиве
    found: integer;            // признак, что число есть в массиве (найдено)
    i: integer;
begin
    for i:= 1 to HB do
        a[i] := Random(HB);
    end;

    write('Number list: ');
    for i:= 1 to HB-1 do
        a[i] := Random(HB);
        write(a[i]:3,',');
    end;

    write(a[HB]:3);

    r:= Random(HB);
    writeln('Search: ',r);

    found := FALSE; // пусть число не найдено
    i:= 1;
    repeat
        if a[i] = r then
            found := TRUE; // число найдено
        else
            i:=i+1;
        end;
    until( found = TRUE) OR (i > HB);
```

```
    writeln('i=',i);

    if found = TRUE then
        writeln('Found!');
    else
        writeln('Not found!');
    end;

    write('Press <Enter>');
    readln;

end.
```

Вывод в окно консоли

Вывод информации на экран (в окно консоли) выполняют инструкции `write` и `writeln`.

В общем виде инструкции вывода информации в окно консоли записываются так:

```
write(список_вывода);
writeln(список_вывода);
```

где:

список_вывода – разделенные запятыми имена переменных, строковые константы или выражения.

Примеры:

```
write(sum);
write('Press <Enter>');
writeln('x1=', x1, ' x2=', x2);
writeln(pound,' фунтов =', pound*0.453, ' кг.');
```

Помимо обычных символов, строка (строковая константа) может содержать специальную последовательность `\n`, которая при выводе строки на экран трактуется как команда продолжить вывод символов, следующих за последовательностью `\n`, с начала следующей строки.

Пример:

```
writeln('x1=', x1, '\nx2=', x2);
```

Форматированный вывод

В строке вывода после имени переменной или выражения через двоеточие можно задать формат вывода значения.

Для целых и строковых значений формат задает ширину поля вывода - количество позиций на экране, которое резервируется для вывода значения переменной.

В общем виде форматированный вывод целых и строковых значений задается так:

имя:*n*

где:

имя – имя переменной , значение которой надо вывести на экран;

n – ширина поля вывода (целая константа).

Форматированный вывод вещественных значений в общем виде задается так:

имя:*n*:*m*

где:

имя – имя переменной;

n – ширина поля вывода (целая константа);

m - количество цифр дробной части.

Примеры:

```
writeln('x1=', x1:9:3, 'x2=', x2:9:3); // x1 и x2 – вещественного типа
writeln(name:15, salary:12:3); // переменная name строковая, salary - вещественная
writeln(pound:5:2,' фунтов =', pound*0.453:6:3, ' кг.'); // выражение pound*0.453
вещественного типа
```

Ввод данных

Ввод данных с клавиатуры обеспечивает инструкция `readln`, которая в общем виде записывается так:

`readln(`*имя*`);`

где:

имя – имя переменной, значение которой надо получить от пользователя во время работы программы.

Примеры:

```
  readln(name);
  readln(salary);
```

ВНИМАНИЕ! При вводе вещественных значений в качестве десятичного разделителя следует использовать **точку**. Если при вводе вещественного значения вместо точки будет введена запятая, ошибка (исключение) не возникает, но дробная часть будет отброшена.

Инструкция присваивания

Инструкция присваивания выглядит так:

имя := *выражение*;

где:

имя – имя переменной или элемента массива;

выражение – выражение, значение которого присваивается переменной или элементу массива.

Выражение состоит из *операндов* и *операторов*. Операнды это – объекты, над которыми выполняется действие, операторы – символы, обозначающие действия.

В качестве операнда выражения могут использоваться константы, переменные, элементы массивов, функции.

Примеры:

```
n := 0;
k := k + 1;
a[i] := b[i];
densyty := mat[i].density;
n := Random(100);
s := s + c[j];
m := sum / k;
name := 'James Bond';
firstName := Substr(name,1,p);
```

Арифметические операторы

Арифметические операторы:

Оператор	Действие	Тип операндов	Тип выражения
+	сложение	integer, float	integer – если оба операнда integer; float – если один из операндов float
-	вычитание	integer, float	integer – если оба операнда integer; float – если один из операндов float
*	умножение	integer, float	integer – если оба операнда integer; float – если один из операндов float
/	деление	integer, float	float

DIV	целая часть частного	integer	integer
MOD	остаток от деления как целое	integer	integer

Оператор `+` применим к операндам строкового типа. Результат применения оператора "сложение" к операндам строкового типа – конкатенация строк-операндов.

Приоритет операторов

Значение выражения вычисляется слева направо, при этом следует учитывать, что операторы умножения и деления имеют более высокий приоритет, чем операторы сложения и вычитания.

Для задания нужной последовательности вычисления значения выражения следует использовать скобки.

Пример:

```
R = (R1 * R2)/(R1 + R2);
```

Выбор действия (инструкция if)

Выбор действия в зависимости от выполнения некоторого условия реализуется при помощи инструкции `if`.

Инструкция выбора одного из двух возможных вариантов действия записывается так:

```
if условие  then
      // здесь инструкции, которые должны быть выполнены,
      // если условие выполняется (истинно)
  else
      // здесь инструкции, которые должны быть выполнены,
      // если условие НЕ выполняется (ложно)
end;
```

Пример:

```
if t = 1 then
    r := r1+r2;
 else
    r := r1*r2/(r1+r2);
end;
```

Если при выполнении условия надо выполнить некоторое действие, а в случае, если условие не выполняется, это действие надо пропустить и перейти к следующей инструкции программы, то инструкция `if` записывается так:

```
if условие then
  // здесь инструкции, которые будут выполнены,
  // если условие выполняется (истинно)
end;
```

Пример:

```
if a[i] < a[i+1] then
  b:=a[i];
  a[i]:=a[i+1];
  a[i+1]:=b;
end;
```

Выбор действия (инструкция case)

Выбор действия в зависимости от значения переменной или выражения **целого типа** можно реализовать при помощи инструкции `case`.

Инструкция `case` в общем виде записывается так:

```
case селектор of
   d_1 : do
           // здесь инструкции, которые будут
           // выполнены, если значение выражения селектор
           // принадлежит диапазону d_1
         end;

   d_i : do
           // здесь инструкции, которые будут
           // выполнены, если значение выражения селектор
           // принадлежит диапазону d_i
         end;

   else do
           // здесь инструкции, которые будут
           // выполнены, если значение выражения-селектора
           // не принадлежит ни одному из указанных диапазонов
         end;
end;
```

где:

селектор – переменная или выражение целого типа;

d_i – диапазон.

Диапазон задается так:

```
k1 .. k2
```

где

`k1` и `k2` – целые неотрицательные константы, удовлетворяющие условию `k1` <= `k2`.

Вместо диапазона можно указать целую константу.

Если между `do` и `end`, находящихся после `else`, нет ни одной инструкции, т.е. в случае, если значение селектора не принадлежит ни одному из указанных диавпазонов никаких действий выполнять не надо, инструкция `case` записывается так:

```
case селектор of
   d_1 : do
           // здесь инструкции, которые будут
           // выполнены, если значение выражения селектор
           // принадлежит диапазону d_1
         end;

   d_i : do
           // здесь инструкции, которые будут
           // выполнены, если значение выражения селектор
           // принадлежит диапазону d_i
         end;
end;
```

Примеры:

```
case month of
    3..5: do
              season := 'spring';
          end;
    6..8: do
              season := 'summer';
          end;
    9..11: do
              season := 'atumn';
          end;
    else : do
             season := 'winter';
          end;
```

```
  end;
```

```
case n of
    1: do
          title :='Aluminium';
          density := 2.71;
       end;
    2: do
          title :='Cooper';
          density := 8.94;
       end;
    3: do
          title :='Steel';
          density := 7.86;
       end;
end;
```

Множественный выбор

Множественный выбор (выбор одного действия из нескольких возможных) можно реализовать при помощи вложенных инструкций `if` или инструкции `case`

Приведенные ниже инструкции показывают, как можно реализовать выбор одного действия из четырех возможных вариантов при помощи инструкций `if`.

```
    if условие1 then
        // здесь инструкции, которые будут выполнены,
        // если условие1 истинно
    else
        if условие2 then
            // здесь инструкции, которые будут выполнены,
            // если уловие1 ложно, а условие2 истинно
        else
            if условие3 then
                // здесь инструкции, которые будут выполнены,
                // если условия условие1 и условие2 ложны, а условие3 истинно
            else
                // здесь инструкции, которые будут выполнены,
                // если ни одно из условий условие1, условие2 или условие3
                // НЕ выполняется
            end;
```

```
        end;
    end;
```

Пример:

```
  if ( n = 1) then
      density := 2.7;
  else
     if ( n = 2) then
       density := 7.6;
     else
        if ( n = 3) then
           density := 8.7;
        else
           writeln('Неправильный номер материала');
        end;
     end;
  end;
```

Условие

Условие это – выражение логического типа, которое может принимать одно из двух значений: Истина или Ложь.

Различают простое и сложное условия.

Простое условие

Простое условие в общем виде записывается так:

op1 оператор_сравнения op2

где:

op1 и *op2* – сравниваемые операнды, в качестве которых могут выступать константы, переменные, функции или выражения.

Операторы сравнения:

=	равно
>	больше
>=	больше или равно
<	меньше

<=	меньше или равно
!=	не равно

Примеры простых условий:

```
a[i+1] < a[i]
d != 0
pos(' ', st) = 1
name = 'simpson'
```

Сложное условие

Сложное условие в общем виде записывается так:

усл1 логический_оператор усл2

где:

усл1 и *усл2* – выражения логического типа, в качестве которых могут выступать простые или сложные условия.

Логические операторы:

AND	логическое И
OR	логическое ИЛИ
NOT	логическое НЕ

Примеры сложных условий:

```
  x >= x1 AND x <= x2
  NOT((x < x1) OR (x > x2))
  sum >=1000 and sum <10000
  name = 'Bart' OR name = 'Homer'
```

Цикл for

Инструкция цикла `for` в общем виде записывается так:

```
for сч := start to finish do
     // инструкции, которые надо выполнить несколько раз
end;
```

где:

сч – счетчик циклов (переменная целого типа);

start и *finish* – выражения целого типа (в простейшем случае – целые константы), определяющие, соответственно, начальное и конечное значение счетчика циклов.

Примеры:

```
for i:=1 to 10 do
    writeln(i:2, ' Hello, World!');
end;
```

```
for i:=1 to n do
    writeln(i:2, ' Hello, World!');
end;
```

Цикл While

Инструкция цикла `while` (цикл с предусловием) в общем виде записывается так:

```
while условие do
    // здесь инструкции, которые будут выполняться до тех пор,
    // пока условие истинно
end;
```

где:

условие – простое или сложное условие выполнения инструкций, находящихся между словами `do` и `end`.

Пример:

```
  i := 1;
  while i <= 10 do
      writeln(i:2, ' Hello, World!');
      i := i + 1;
  end;
```

Цикл Repeat

Инструкция цикла `repeat` (цикл с постусловием) в общем виде записывается так:

```
repeat
  // здесь инструкции, которые будут выполняться до тех пор,
  // пока условие ложно
until условие;
```

где:

условие – простое или сложное условие **завершения** цикла (прекращения выполнения инструкций, находящихся между словами `repeat` и `until`).

Пример:

```
i := 1;
repeat
    writeln(i:2, ' Hello, World!');
    i := i + 1;
until i > 10;
```

Инструкция Goto

Инструкция `goto` (безусловный переход) в общем виде записывается так:

```
goto метка
```

где:

метка – идентификатор инструкции, к которой необходимо выполнить переход.

Метка представляет собой любую начинающуюся буквой и состоящую из букв и цифр строку.

Метка записывается перед инструкцией, к которой надо выполнить переход, и отделяется от этой инструкции двоеточием.

Метка должна быть объявлена в разделе объявления меток в той процедуре или функции, в которой она используется. Начало раздела объявления меток помечает ключевое слово `label`.

Раздел объявления меток предшествует разделу объявления констант или, если раздел `const` отсутствует, разделу объявления переменных.

Пример:

```
// вычисление НОД - наибольшего общего
// делителя двух целых чисел
program p1()
label
    m1,m2;
var
    a,b: integer; // числа
    n: integer;   // НОД
begin
    a:=12;
```

```
    b:=18;
    writeln('a=',a:2,' b=',b:2);

m1: if a = b then
        n:=a;
        goto m2;
    end;
    if a > b then
       a:= a-b;
       goto m1;
    else
       b:= b-a;
       goto m1;
    end;

m2: writeln('Наибольший общий делитель:', n);
    write('Press <Enter>');
    readln;
end.
```

Одномерный массив

Объявление одномерного массива в общем виде выглядит так:

имя: **array**[1..*НВ*] **of** *тип*;

где:

имя – имя массива

НВ – верхняя граница диапазона индекса массива (количество элементов массива)

тип – тип элементов массива (integer, float, string или тип-запись)

Внимание! Максимальное допустимое количество элементов одномерного массива 255

Примеры:

```
  Salary: array[1..15] of float;  // массив вещественных чисел
  nPacients: array[1..31] of integer; // массив целых чисел
  Students: array[1..25] of strings[15]; // массив строк
  materials: array[1..10] of Material;   // массив записей
```

Допускается одной инструкцией объявить несколько массивов одинакового типа и размера, например:

```
  gold, silver, bronze: array[1 ..10] of integer; // три массива целых чисел
```

```
  students_1,  students_2: array[1 .. 30] of string[25];  // два массива строк
```

При объявлении одномерного массива в качестве верхней границы диапазона индекса можно использовать целую именованную константу.

Например, если в разделе `const` объявлены целые именованные константы `HB` и `NL`, то объявление массива `students` может быть таким:

```
Students: array[1..HB] of strings[NL]; // HB и NL – целые именованные константы
```

Именованные константы, использованные в инструкции объявления массива, удобно использовать в инструкциях его обработки.

Например:

```
    for i:=1 to HB do
       writeln(Students[i]);
    end;
```

Если в программе объявлен тип-массив, то объявление одномерногомассива может выглядеть так:

```
  имя: тип-массив;
```

где:

имя – имя массива;

тип-массив – объявленный в разделе `type` программы тип-массив.

Пример:

```
  type
      vector = array[1..10] of float;  // vector – тип-массив
  var
      v1: vector; // v – переменная типа vector,
                  // т.е. одномерный вещественный массив
```

Внимание! При работе с большим количеством массивов или с массивами большой размерности следует учитывать, что суммарный размер данных (памяти, занимаемой переменными программы, в том числе и массивами) и кода программы не может превышать 64K.

Двумерный массив

Объявление двумерного массива в общем виде выглядит так:

```
  имя: array[1..NR,1..NC] of тип;
```

где:

имя – имя массива;

NR – количество строк (row - строка) - верхняя граница диапазона индекса строки массива;

NC – количество столбцов (column - столбец) - верхняя граница диапазона индекса столбца массива;

тип – тип элементов массива (integer, float, string или тип-запись).

Внимание! Максимальное допустимое количество строк и количество столбцов двумерного массива **255**.

Примеры объявления двумерных массивов:

```
// двумерный массив целых чисел (25 строк, 12 столбцов)
Salary: array[1..25,1..12] of integer;

// двумерный массив вещественных чисел (5 строк, 8 столбцов)
Matrix: array[1..5,1..8] of float;

// двумерный массив строк (100 строк, 2 столбца)
dictionary: array[1..100,1..2] of string[20];
```

Допускается одной инструкцией объявить несколько массивов одинакового типа и размера, например:

```
// два двумерных массива вещественных чисел
Matrix1, Matrix2: array[1..5,1..8] of float;
```

При объявлении двумерного массива в качестве верхних границ диапазонов индексов можно использовать целые именованные константы.

Например, если в разделе const объявлены целые именованные константы NR и NC, то объявление массива matrix может быть таким:

```
Matrix: array[1..NR,1..NC] of float;// NR и NC – целые именованные константы
```

Именованные константы, использованные в инструкции объявления массива, удобно использовать в инструкциях его обработки.

Например:

```
for i:=1 to NR do
   for j:=1 to NC do
        matrix[i,j]:=0;
   end;
end;
```

Если в программе объявлен тип-массив, то объявление двумерного массива может выглядеть так:

```
имя: тип-массив;
```

где:

имя – имя массива;

тип-массив – объявленный в разделе `type` программы тип-массив.

Пример:

```
type
    matrix = array[1..3, 1..5] of float;  // matrix – тип-массив
var
    m1: vector; // m1 – переменная типа matrix,
                // т.е. двумерный вещественный массив
```

Внимание! При работе с большим количеством массивов или с массивами большой размерности следует учитывать, что суммарный размер данных (памяти, занимаемой переменными программы, в том числе и массивами) и кода программы не может превышать 64К.

Инициализация массива

В начале работы программы элементы числовых массивов имеют нулевое значение, элементы строковых массивов - значение "пустая строка".

Если программе нужен массив, значение элементов которого отличаются от значений "по умолчанию", необходимо выполнить инициализацию массива.

Инициализация массива это – присваивание требуемых значений **всем** элементам массива.

Инициализацию массива можно выполнить, указав в инструкции объявления массива список инициализации.

Инициализация одномерного массива

Инструкция объявления и инициализации одномерного массива выглядит так:

```
    name: array[1 .. N] of type = list;
```

где:

name – имя массива;

N – количество элементов массива (целая или целая именованная константа);

type – тип массива (`integer`, `float` или `string`);

list – список констант. Тип констант должен соответствовать типу массива, а их количество – размеру массива.

Примеры:

```
    material: array[1..4] of string[10] = 'Aluminum','Cooper','Gold','Steel';
    density: array[1..4] of float = 2.71, 8.94, 19.32, 7.86;
```

При инициализации `float` массива в списке инициализации допускается использовать целые константы.

При инициализации символьного массива, в случае если дина строковой константы в списке больше длины строки, указанной в инструкции объявления массива, соответствующий элемент массива будет инициализирован "обрезанной" строкой.

Например, если инструкция объявления массива выглядит так

```
    material: array[1..4] of string[6] = 'Aluminum','Cooper','Gold','Steel';
```

то элемент `material[1]` будет инициализирован значением `'Alumin'`.

Инициализация двумерного массива

Инструкция объявления и инициализации двумерного массива выглядит так:

```
    name: array[1..NR, 1..NC] of type = list;
```

где:

name – имя массива;

NR и *NC* – количество строк и столбцов массива (целые или целые именованные константы);

type – тип массива (`integer`, `float` или `string`);

list – список констант. Тип констант должен соответствовать типу массива, а их количество – количеству элементов массива. В списке инициализации сначала указывают значения для первой строки массива, затем для второй и так далее.

Примеры:

```
frasi: array[1..4, 1..3] of string[12] =
                   'buongirno','ciao','grazie',
                   'good day','good by','thank you',
                   'добрый день','до встречи','спасибо',
                   'Buenos dias','adios', 'gracias';

matrix: array[1..3,  1..4] of float = 1.5, 2.5, 3.0, 0.0,
                                      3.7, 2.0, 6.2, 1.7,
                                      0.0, 0.0, 0.0, 0.0;
```

При инициализации `float` массива в списке инициализации допускается использовать целые константы.

При инициализации символьного массива, в случае если дина строковой константы больше длины строки, указанной в инструкции объявления массива, то соответствующий элемент массива будет инициализирован "обрезанной" строкой.

Массив-параметр

Масив, объявленного программистом типа, можно использовать в качестве параметра функции или процедуры.

Массив передается в функцию (процедуру) по ссылке. Это значит, что функция (процедура) может изменить значения элементов массива.

Объявление функции (процедуры) выглядит обычным образом.

Объявление параметра-массива в объявлении функции (процедуры) выглядит так:

```
  var имя: тип
```

где:

`var` – ключевое слово, показывающее, что парамерт передается в функцию (процедуру) по ссылке;

имя – имя массива;

тип – тип-массив, объявленный программистом.

Пример:

```
Program p1()
const
    N=5;
type
    TVector = array[1..N] of integer;

// ввод массива с клавиатуры
procedure GetVector(var v: TVector, k: integer)
var
    i: integer;
begin
    for i:=1 to k do
        write(i:2,'>');
        readln(v[i]);
    end;
end;

// возвращает сумму элементов массива
function SumVector(var v: TVector, k: integer): integer
var
    i: integer;
    sum: integer;
begin
    sum:= 0;
    for i:=1 to k do
        sum :=sum + v[i];
    end;
   return sum;
end;

var
```

```
    v : TVector;
    sum: integer;
    i: integer;
begin
    GetVector(v, N); // ввод массива

    write('Массив: ');
    for i:=1 to N-1 do
        write(v[i]:3, ',');
    end;
    writeln(v[N]:3);

    sum := SumVector(v, N);
    writeln('Сумма элементов массива: ', sum:5);

    writeln('Press <Enter>');
    readln;
end.
```

Функция

Объявление функции программиста выглядит так:

```
function Имя (параметры): тип
var
    // здесь объявления локальных переменных
begin
    // здесь инструкции функции
    return значение;
end;
```

где:

Имя – имя функции;

параметры – объявление параметров функции;

тип – тип значения функции;

значение – значение функции.

Объявление каждого параметра функции выглядит так:

имя_параметра: *тип*

Пример функции:

```
  // Функция CylinderVolume вычисляет объем цилиндра.
```

```
  Function CylinderVolume(d: integer, len: integer):float
    const
      PI = 3.1415926;
    var
      volume: float;
    begin
      volume := PI*(d/2)*(d/2)*len;
      return volume;
  end;
```

Параметры, которые указываются в инструкции вызова функции, называются фактическими. Параметры передаются в функцию **по значению**. В качестве **фактического** параметра может использоваться **выражение**, тип которого соответствует типу формального параметра. В простейшем случае, в качестве фактического параметра может использоваться константа или переменная. Если формальный параметр функции вещественного типа, то в качестве фактического параметра можно использовать выражение как вещественного, так и целого типа.

```
  // примеры вызова функции
  v := CylinderVolume(diam, len); // параметры - переменные
  v := CylinderVolume(20, 1000); // параметры - константы
  v := CylinderVolume(d – 2*w, len); // параметры – выражение и переменная
```

Чтобы функция стала доступна другой функции или процедуре, в том числе главной процедуре программы (`program`), ее объявление (текст) надо поместить в текст программы перед той процедурой или функцией, которая ее использует.

Пример:

```
  // Вычисление объема полого цилиндра

  // Функция CylinderVolume вычисляет объем цилиндра.
  Function CylinderVolume(d: integer, len: integer):float
    const
      PI = 3.1415926;
    var
      volume: float;
    begin
      volume := PI*(d/2)*(d/2)*len;
      return volume;
  end;

// программа, использующая функцию CylinderVolume
Program P1()
var
     diam: integer; // диаметр
     wal: integer;  // толщина стенки
     len: integer;  // длина
```

```
     volume: float; // объем
begin
     writeln('Объем полого цилиндра');
     write('Диаметр, мм >');
     readln(diam);
     write('Толщина стенки, >');
     readln(wal);
     write('Длина, мм >');
     readln(len);

     // объем в мм куб.
     volume := CylinderVolume(diam,len) - CylinderVolume(diam-2*wal,len);

     // объем в см куб.
     volume := volume / 1000;
     writeln('Объем полого цилиндра', volume:9:2, ' см.куб.');
     write('Press <Enter>');
     readln;
end.
```

Процедура

Объявление процедуры программиста выглядит так:

```
procedure Имя(параметры)
var
    // здесь объявления локальных переменных
begin
    // здесь инструкции процедуры
end;
```

где:

Имя – имя процедуры;

параметры – объявление параметров процедуры.

Объявление каждого параметра выглядит так:

имя_параметра: *тип*

Пример:

```
// процедура Line выводит в окно консоли n раз строку st
```

```
procedure Line(n:integer, st: string)
var
    i: integer;
begin
    for i:=1 to n do
        write(st);
    end;
    writeln;
end;
```

```
// примеры вызова процедуры
Line(k,'-');
Line(25, ch);
```

Параметры, которые указываются в инструкции вызова процедуры, называются фактическими. Параметры передаются процедуру **по значению**. В качестве **фактического** параметра может использоваться **выражение**, тип которого соответствует типу формального параметра. В простейшем случае, в качестве фактического параметра может использоваться константа или переменная. Если формальный параметр функции вещественного типа, то в качестве фактического параметра можно использовать выражение как вещественного, так и целого типа.

Чтобы процедура стала доступна другой функции или процедуре, в том числе главной процедуре программы (program), ее объявление (текст) надо поместить в текст программы перед той процедурой или функцией, которая ее использует.

Рекурсия

Pascal Next поддерживает рекурсивные **функции**.

Пример рекурсивной функции и ее использования:

```
  // функция Factorial вычисляет факториал числа n
  function Factorial(n: integer): integer
  var
      f: integer;
  begin
      if n = 1 then
           f:=1;
      else
         f:= n*Factorial(n-1);
      end;

      return f;
```

```
  end;

  // значения факториала чисел от 1 до 12
  program p28()
  var
      i: integer;
  begin
      for i:=1 to 12 do
           writeln(i:2, ' - ', Factorial(i));
      end;

      write('Press <Enter>');
      readln;
  end.
```

Глобальные переменные

Глобальными или общими называют переменные и структуры данных (массивы), которые объявлены вне процедуры или функции, но к которым у процедуры или функции есть доступ.

Глобальные переменные обычно используют для обеспечения доступа процедурам и функциям к общим данным.

Для того чтобы процедура или функция получила доступ к переменным программы, ее объявление надо поместить в объявление программы, после раздела объявления переменных.

Внимание! Вкладывать процедуры и функции можно только в **программу** (в процедуру `program`). Внутрь процедуры или функции поместить другую процедуру или функцию **нельзя**.

Пример:

```
program p29()
  const
     HB = 10;

   var
     // здесь глобальные переменные
     c: array[1..HB] of integer;


  function SumArr(m: integer, n: integer): integer
    var
      i: integer;
      sum: integer;
    begin
```

```
       sum:=0;
       for i:= m to n do
          sum := sum + c[i]; // массив с - глобальный
       end;
       return sum;
    end;

   // инициализация массива случайными числами
   procedure InitArr()
    var
       i: integer;
    begin
        for i:=1 to HB do
            c[i]:= Random(10);
        end;
    end;

// основная программа
var
   // к этим переменным у функции Sum и
   // процедуры Init доступа нет
   sum: integer;
   i: integer;

begin
    InitArr(); // инициализация массива
    sum := SumArr(1,HB); // вычислить сумму элементов массива

    for i:=1 to HB-1 do
        write(c[i]:3,','); // массив с - глобальный
    end;
    writeln(c[i]:3);

    writeln('\nSum=',sum);

    write('\nPress <Enter>');
    readln;
end.
```

Файловые операции

- Чтение строк из текстового файла
- Запись строк в текстовый файл
- Добавление строк в текстовый файл

При чтении данных из файла для преобразования строки в целое или вещественное значение следует использовать соответственно функции StrToInt и StrToFloat.

При записи данных в текстовый файл для преобразования численного значения в строку следует использовать функции IntToStr или FloatToStr.

Доступ к текстовому файлу обеспечивает объект типа `Text`.

Функции работы с файлами

Функция	Описание	Пример
reset(*имя_файла*)	Открывает текстовый файл для чтения. Возвращает дескриптор файла или -1, если имя файла задано неправильно (в указанной папке файла нет, неправильно указано имя файла).	DataFile:='C:\Temp\dat.txt'; f:=reset(DataFile);
readstring(*дискриптор_файла*)	Читает строку из текстового файла. Для преобразования строки в число следует использовать функции StrToInt или StrToFloat	name:=readstring(f);
eof(*дискриптор_файла*)	Конец файла. Возвращает -1, если достигнут конец файла	while eof(f) != -1 do end;
rewrite(*имя_файла*)	Открывает текстовый файл для перезаписи. Возвращает дескриптор файла (если файла нет, то он будет создан).	DataFile:='C:\Temp\dat.txt'; f:=rewrite(DataFile);
append(*имя_файла*)	Открывает текстовый файл для добавления. Возвращает дескриптор файла или -1, если указанного файла нет.	DataFile:='C:\Temp\dat.txt'; f:=append(DataFile);

writestring(*дискриптор_файла*, *строка*);	Записывает в текстовый файл строку символов. Для преобразования численного значения в строку следует использовать функцию IntToStr или FloatToStr	n:=100; k:=32.5;
		writestring(f,IntToStr(n));
		writestring(f,FloatToStr(k));
		writestring(f, name);

close(*дискриптор_файла*)	Закрывает открытый файл	close(f);

Имя файла может быть полным (включать путь к файлу) или коротким. Короткое имя файла предполагает, что файл данных находится в том же каталоге, где находится выполняемый (exe) файл программы.

При запуске программы из среды разработки необходимо указывать полное имя файла.

Путь к файлу, находящемуся в папке Документы следует записывать так:

```
c:\users\user_name\documents
```

где ***user_name*** – имя пользователя, в системе.

Например, если имя пользователя nikita, то полное имя файла `data.txt`, находящегося в папке Документы, выглядит так:

```
c:\users\nikita\documents\data.txt
```

Если файл находится на рабочем столе, то его имя следует записывать так:

```
c:\users\user_name\desktop\fille
```

где ***user_name*** – имя пользователя, в системе.

Например, если имя пользователя nikita, то полное имя файла data.txt, находящегося на рабочем столе, выглядит так:

```
c:\users\nikita\desktop\data.txt
```

Тип программиста

Программист может объявить свой тип данных (запись или тип-массив) и использовать их в программе также, как и стандартные типы данных.

Объявление типа должно находится в разделе `type` программы, который в общем случае следует за разделом объявления констант (начинается словом `const`) и предшествует разделу объявления переменных (начинается словом `var`).

```
const
    // здесь объявления именованных констант
type
    // здесь объявления типов
var
    // здесь объявления переменных
```

Пример:

```
const
    N = 10;
    NR = 3;
    NC = 5;

type
    // TMaterial - тип-запись
```

```
        TMaterial = record
              title: string[15]; // название материала
              density: float;    // плотность материала
        end;

        // TMatrix - тип-массив
        TMatrix = array[1..NR, 1..NC] of float;

    var
        aMaterial: TMaterial;   // запись
        TablMaterials : array[1..N] of TMaterial; // массив записей

        m1: TMatrix; // двумерный массив
```

Запись

Запись, представляет собой структуру данных, которая, в общем случае, может состоять из элементов (полей) разного типа.

В общем виде объявление типа-записи выглядит так:

```
  Имя = record
      Поле_1: Тип_1;
      Поле_2: Тип_2;
              ...
      Поле_n: Тип_n;
  end;
```

где:

Имя — имя типа-записи;

`record` — слово языка программирования Pascal Next, означающее, что далее следует объявленияполей записи;

Поле_i и *Тип_i* — имя и тип i-го поля записи.

Замечание. В сообществе Pascal-программистов принято правило - имена типов, объявляемых программистом, должны начинаться буквой `T` (первая буква слова Type - тип).

Примеры:

```
  type
    TMaterial = record
      title: string[15]; // название материала
      density: float;    // плотность материала
```

```
  end;

  TBook = record
    Title: string[40];  // название
    Autor: string[20];  // автор
    Price: float;       // цена
  end;

  TPoint = record
    x: integer;
    y: integer;
    z: integer;
  end;
```

После объявления типа-записи можно обычным образом объявить переменную-запись или массив записей .

Примеры:

```
var
  material: TMaterial;
  book: TBook;
  point: TPoint;
  books: array[1..100] of TBook;
  TablMaterials: array[1..10] of TMaterial;
```

Доступ к полям записи осуществляется по имени. Чтобы получить доступ к полю записи, надо указать имя переменной-записи и имя поля, отделив имя поля от имени переменной-записи точкой.

Примеры:

```
weight := volume * material.density;
writeln('Материал: ', material.title, 'Плотность: ', material.density:6:2);
writeln(book.Autor);
writeln(book.Title);
writeln(book.Price:6:2);
```

Доступ к полям записи-элемента массивазаписи осуществляется путем указания номера элемента массива и имени поля, которое указывается после закрывающей скобки через точку.

Примеры:

```
weight := volume * TablMaterials[k].density;

for i:=1 to n
     writeln(books[i].Title:40, books[i].Author:20, books[i].Price:7:2);
end;
```

Тип-массив

Объявление типа-массива выглядит так:

```
  имя = array[1..HB] of тип;
```

или

```
  имя = array[1..NR, 1..NC] of тип;
```

где:

имя – имя типа-массива;

HB – верхняя граница диапазона изменения индекса одномерного массива (целая или целая именованная константа);

NR и NC – соответственно количество строк и столбцов двумерного массива (целая или целая именованная константа);

тип – тип элементов массива (integer, float string или тип-запись).

Замечание. В сообществе Pascal-программистов существует правило - имена типов, объявляемых программистом, должны начинаться буквой `T` (первая буква слова Type - тип).

Примеры:

```
  type
    TVector = array[1..N] of float;
    TList = array[1..15] of string[25];
    TMatrix = array[1..N, 1..M] of float;

    TBooks = array[1..100] of TBook;   // тип "массив записей TBook"
    TablMaterials = array[1..10] of TMaterial; // тип "массив записей TMaterial"
```

После объявления типа-массива можно обычным образом объявить массив, используя объявленный тип.

Примеры:

var

```
  list1: TList;       // одномерный массив строк
  v1: TVector;        // одномерный вещественный массив
  m1, m2: TMatrix;    // двумерные вещественные массивы
  tablmat: TablMaterials; // одномерный массив записей
```

Математические функции

`Sqrt (x)` – квадратный корень

`Sin(r)` - синус, r – угол в радианах

`Cos(r)` - косинус, r – угол в радианах

`Tg(r)` - тангенс, r – угол в радианах

`Arctg(x)` - арктангенс, x – тангенс

`Round(x)` - округление

`Trunc(x)` – целая часть вещественного

`Abs(x)` - абсолютное значение

`Random(n)` – случайное число в диапазоне 1..n

Строковые функции

Строковые функции:

Length

Pos

Substr

UpCase

LowCase

Length

Length(строка) – дина строки

Функция `Length` возвращает количество символов строки, указанной в качестве параметра.

Pos

Pos(подстрока, строка) – позиция подстроки в строке

Функция `Pos` возвращает позицию первого вхождения подстроки в указанной строке. Если строка не содержит указанную подстроку, то функция возвращает ноль.

```
name := 'Bart Simpson';
p := Pos('Simpson ', name);
```

Subsrt

Substr(строка, номер_символа, дина) – подстрока

Функция `Substr` возвращает подстроку указанной строки. Параметр `номер_символа` задает позицию первого символа требуемой подстроки (начало подстроки), параметр `дина` – количество символов

подстроки. Если длина строки меньше чем значение параметра `номер_символа`, то функция возвращает пустую (не содержащую ни одного символа) строку. Если длина строки такова, что получить подстроку указанной длинны нельзя, то функция возвращает правую часть строки, начиная от символа с указанным номером.

Пример использования строковых функций Length, Pos и Substr:

```
program p1()
var
    name: string[15];

    lastName: string[15];
    firstName: string[15];

    p: integer;   //позиция пробела в имени
begin
    name := 'Bart Simpson';
    writeln('Name: ', name);
    p:= Pos(' ', name);

    if p !=0 then
        firstName := Substr(name, 1, p-1);
        lastName:= Substr(name, p+1, Length(name)-p);
    else
        firstName := name;
        lastName:='';
    end;
    writeln('First Name: ', firstName, ' Last name: ', lastName);
    readln;
end.
```

UpCase

UpCase(строка) – строка, преобразованная к верхнему регистру

Функция `UpCase` возвращает строку, преобразованную к верхнему регистру.

Пример:

```
  name := UpCase(name);
```

LowCase

LowCase(строка) – строка, преобразованная к нижнему регистру

Функция `LowCase` возвращает строку, преобразованную к нижнему регистру.

Пример:

```
  name := LowCase(name);
```

Функции преобразования

Функции преобразования (конвертирования):

IntToStr(x)

StrToInt(s)

FloatToStr(x)

StrToFloat(s)

IntToStr

Функция `IntToStr(x)` возвращает строковое представления выражения целого типа, указанного в качестве ее параметра.

StrToInt

Функция `StrToInt(st)` преобразует строку-изображение целого числа в число, соответствующее строке-параметру. Если строка-параметр не является правильным изображением целого числа (содержит символы, отличные от цифр), то функция возвращает `0`.

FloatToStr

Функция `FloatToStr(x)` возвращает строковое представления выражения вещественного типа, указанного в качестве ее параметра.

StrToFloat

Функция `StrToFloat(st)` преобразует строку-изображение вещественного числа в число, соответствующее строке-параметру. Если строка-параметр не является правильным изображением вещественного числа (содержит символы, отличные от цифр или десятичного разделителя), то функция возвращает `0`.

Функции даты и времени

Функции даты и времени:

getDay()

getMonth()

getYear()

getDayOfWeek()

getTime()

getDay

Функция `getDay()` возвращает порядковый номер дня текущего месяца.

getMonth

Функция `getMonth()` возвращает порядковый номер месяца года.

Первый месяц года – январь, нумерация с единицы.

```
program p1()
var
    day: integer;
    month: integer;

    monthName: array[1..12] of string[10] =
            'январь','февраль','март','апрель','май','июнь',
            'июль','август','сентябрь','октябрь','ноябрь','декабрь';
begin
    day := getDay();
    month := getMonth();

    writeln('Сегодня ', day, ' ', monthName[month] );

    write('Press <Enter>');
    readln;
end.
```

getYear

Функция `getYear()` возвращает порядковый номер года.

```
program p1()
var
    day: integer;
    month: integer;
    year: integer;

begin
     day := getDay();
     month := getMonth();
     year := getYear();

     writeln('Сегодня ', day, '-', month, '-', year);
```

```
    write('Press <Enter>');
    readln;
end.
```

getDayOfWeek

Функция `getDayOfWeek()` возвращает порядковый номер дня недели.

Первым днем недели считается воскресенье. Дни недели нумеруются с нуля.

```
program p1()
var
    dayOfweek: integer;

    weekDay: array[1..7] of string[11] =
                'воскресенье', 'понедельник', 'вторник', 'среда',
                'четверг', 'пятница', 'суббота';
begin
    dayOfweek:= getDayOfWeek();
    writeln('Сегодня ', weekDay[dayOfWeek + 1]);

    write('Press <Enter>');
    readln;
end.
```

getTime

Функция возвращает количество секунд от начала текущих суток.

```
program p1()
var
   time: integer;

        hour: integer;
   min: integer;
   sec: integer;

begin
    time := getTime();
    Writeln('Секунд от начала суток: ', time);
```

```
    hour:= time div 60 div 60;
    min:=  (time - hour*3600)  div 60;
    sec:=  time- hour*3600 - min*60;

    writeln('Сейчас ', hour, ':', min, ':', sec);

    writeln('Press <Enter>');
    readln;
end.
```

Зарезервированные слова

Зарезервированные слова языка программирования Pascal Next:

and	mod
array	not
begin	of
case	or
const	procedure
div	program
do	record
else	repeat
end	return
float	string
for	then
function	to
goto	until
if	var
integer	while
label	

Pascal и Pascal Next

Основные отличия синтаксиса Pascal Next от "классического" Pascal:

	Pascal	Pascal Next

Комментарий	• многострочный комментарий	• многострочный комментарий • однострочный комментарий
Оператор сравнения "не равно "	<>	!=

Инструкция `if`	**if** *условие* **then**
	begin
	действие_1
	end
	else
	begin
	действие_2
	end;
		if *условие* **then**
		действие_1
		else
		действие_2
		end;

Инструкция `case`	case *селектор* of *диапазон_1*: begin *действие_1* end; *диапазон_2*: begin *действие_2* end; else begin *действие_k* end; end;	**case** *селектор* **of** *диапазон_1*: **do** *действие_1* **end;** *диапазон_2*: **do** *действие_2* **end;** **else do** *действие_k* **end;** **end;**

Цикл `for`	**for** i:=start to fin **do**
	begin
	действие
	end;
		for i:=start **to** fin **do**
		действие
		end;

Цикл `while`	**while** условие **do**
	begin
	действие
	end;
		while условие **do**
		действие
		end;

Цикл `repeat`	repeat *действие* until условие;	repeat *действие* until условие;
Возможность объявления именованных констант	Есть	Есть
Возможность объявления	Есть	Есть:

пользовательского типа		запись тип-массив

Свойства .exe файла

При первой компиляции программы компилятор Pascal Next создает текстовый файл (.rs), и помещает в него шаблон информации о программе (название программы, версия программы, авторские права и др.). В процессе компиляции информация из rs-файла помещается в rsrc секцию .exe файла. Это дает пользователю возможность получить информацию о программе (содержимом exe-файла), увидеть, как называется программа, кто является разработчиком, кому принадлежат авторские права, номер версии.

Информация о содержимом exe-файла отображается в окне Свойства, которое появляется на экране в результате щелчка правой кнопкой мыши на имени файла и выбора из контекстного меню команды Свойства.

rs-файл, генерируемый компилятором:

```
Company=
Description=
Version=1.0.0.3
Copyright=(C) , 2021
ProductName=
ProductVersion=1.0.0.1
```

Программист может изменить содержимое .rc файла, чтобы информация, помещаемая в exe-файл, соответствовала разрабатываемой программе.

Пример rs-файла, после внесения изменений:

```
Company=MyCompany
Description=Pound to gram converter
Version=1.0.0.1
Copyright=(C) Nikita Kultin , 2021
ProductName=Weight converter
ProductVersion=1.0.0.1
```

Внимание! При записи информации в rs-файл следует использовать буквы **латинского** алфавита.

Примеры кода

В этой главе собраны примеры программ на языке Pascal Next, демонстрирующие синтаксис и возможности языка програмирования.

Объем полого стержня

Объем полого стержня

```
// объем полого стержня (трубы)
Program P1()
const
   PI = 3.1415926;
var
    diam: integer;  // диаметр
     wal: integer;  // толщина стенки
     len: integer;  // длина
     volume: float; // объем
begin
    writeln('Объем полого цилиндра');
     write('Диаметр, мм >');
     readln(diam);
     write('Толщина стенки, мм>');
     readln(wal);
     write('Длина, мм >');
     readln(len);

    volume := PI*diam*diam/4*len - PI*(diam -2*wal)*(diam -2*wal)/4*len;
    volume := volume / 1000; // объем в см. куб.

     writeln('Объем полого цилиндра', volume:9:2, ' см.куб.');

     writeln;
     write('Press <Enter>');
     readln;
end.
```

Конвертер веса из фунтов в граммы/килограммы

Конвертер веса из фунтов в граммы/килограммы

```
// Конвертер веса из фунтов в граммы/килограммы.
program p1()
const
   K = 453.59237;
var
   Pounds:    float;   // вес в фунтах

   Grams:     integer; // вес в граммах
   Kilograms: float;   // вес в килограммах

   // вес в формате kg + g
   KG: integer;
   GR: integer;

begin

   writeln('Pounds to grams/kilograms converter');
   writeln;

   write('Pounds>');
   readln(Pounds);

   Grams := Round(Pounds * K);

   if grams < 1000 then
       writeln(Pounds:6:2, ' lb = ', Grams, ' g');
   else

        Kilograms := Grams / 1000;
        KG := Grams DIV 1000;   // или так: Trunc(kilograms);
      GR := Grams mod 1000;    // или так: Trunc((Kilograms-KG)*1000);

        write(Pounds:6:2, ' lb = ', Kilograms:6:3);
        writeln(' kg = ',KG,' kg ', GR:3, ' g');
    end;

    writeln;
    write('Press <Enter>');
    readln;
```

```
end.
```

Ток в эл. цепи состоящей из двух резисторов

Ток (A/mA) в эл. цепи состоящей из двух резисторов

```
// Ток в цепи, состоящей из двух резисторов, которые могут быть соединены
последовательно или параллельно
program p()
var
     R1,R2: float; // величины сопротивлений, Ом
     T: integer;   // тип соединения: 1 - послед.; 2 - парал.
     U: float;     // напряжение

     R: float; // сопротивление цепи
     I: float; // ток в цепи

begin
     writeln('Ток в цепи, состоящей из двух резисторов.');
     writeln;
     write('R1, Ом >');
     readln(R1);
     write('R2, Ом >');
     readln(R2);
     write('Способ соединения (1 - послед.; 2 - парал.) >');
     readln(T);
     write('U, вольт >');
     readln(U);

     if T = 1 OR T = 2 then
         if T = 1 then
             R := R1 + R2;
         else
             R := R1*R2/(R1+R2);
         end;

         writeln('Сопротивление цепи: ',R:6:2, ' Ом');

         I := U/R;
```

```
        write('I = ');
        if I < 0.1 then
            I := I * 1000;
            writeln(Round(I),' mA');
        else
            writeln(I:6:3,' A');
        end;
     else
        writeln('Ошибка! Неверно указан способ соединения.');
     end;

     write('Press <Enter>');
     readln;

end.
```

Масса полого стержня. Выбор названия материала из меню

```
// масса полого стержня (трубы)
Program P1()
const
   PI = 3.1415926;
var
     diam: integer; // диаметр
     wal: integer;  // толщина стенки
     len: integer;  // длина
     n: integer;    // номер материала

     material:string[15]; //материал
     density: float; //плотность материала, гр./см.куб.

     volume: float; // объем
     mas: float;    // масса, гр.
begin
    writeln('Масса полого стержня (трубы)');
     write('Диаметр, мм >');
     readln(diam);
     write('Толщина стенки, мм>');
     readln(wal);
     write('Длина, мм >');
```

```
    readln(len);

    writeln('Выберите материал');
    writeln('1. Алюминий');
    writeln('2. Медь');
    writeln('3. Сталь');
    writeln('4. Пластик');
    write('>');
    readln(n);

    if ( n < 1 OR n > 4 ) then
        writeln('Ошибка! Неверно указан номер материала.');
    else
       if n = 1 then
           material :='Алюминий';
           density := 2.7;
       else
           if n = 2 then
               material :='Медь';
               density := 8.9;
           else
               if n = 3 then
                   material :='Сталь';
                   density := 7.856;
               else
                   material :='Пластик';
                   density := 1.9;
               end;
           end;
        end;

        writeln('');

       volume := PI*diam*diam/4*len - PI*(diam -2*wal)*(diam -2*wal)/4*len; //
объем в мм. куб.
        volume := volume / 1000; // объем в см. куб.
        mas := volume * density;

        writeln('Материал: ', material, '(',density:6:3,'гр./см.куб.)');
       writeln('Объем:', volume:9:2, ' см.куб.');
        writeln('Масса:', mas:6:2);
```

```
    end;

    writeln;
    write('Press <Enter>');
    readln;
end.
```

Масса стержня (использование инструкции case)

```
// Расчет массы стержня
program p()
var

    p: integer;           // номер выбранного материала
    material: string[10]; // название материала
    density: float;       // плотность

    diameter: integer; // диаметр
    len: integer;      // длина
    volume: float;     // объем

    mas: float;           // масса

begin
    writeln('= Расчет массы стержня =');
    repeat

        writeln('\n1. Aluminum\n2. Cooper\n3. Steel\n\n0 - Выход');
        write('\nВаш выбор>');

        readln(p);

        case p of
            1: do material := 'Aluminum'; density := 2.71; end;
            1: do material := 'Cooper';   density := 8.94; end;
            1: do material := 'Steel';    density := 7.86; end;
            //else do p := -1; end;
        end;

        if p >=1 and p <=3 then
            write('Диаметер, мм >');
```

```
            readln(diameter);
            write('Длинна, мм>');
            readln(len);

            volume:=len*(diameter/2)*(diameter/2)*3.1415926; //  объем в мм куб.
            volume := volume/1000; // объем в см. куб.
            mas:=volume * density;

            writeln('\nМатериал: ', material,' (', density:5:2,' гр./см.куб.)');
            writeln('Длинна: ', len,' мм' );
            writeln('Диаметр: ', diameter,' мм');
            writeln('Объем: ', volume:6:1,' см.куб.');
            if (mas < 1000) then
                writeln('Масса: ', mas:9:2,' гр');
            else
                writeln('Масса: ', mas/1000:9:3,' кг');
            end;

        end;

    until (p = 0);

    writeln;
    write('\nPress <Enter>');
    readln;
end.
```

Таблица тригонометрических функций sin и cos

```
// Таблица тригонометрических функций sin и cos.
// Демонстрирует использование функций sin и cos,
// использование процедур, форматированный вывод.

// рисует линию
procedure line(n:integer, ch: string)
var
        i: integer;
begin
        for i:=1 to n do
                write(ch);
        end;
```

```
        writeln;
end;

// выводит таблицу синус-косинус
procedure tabsin(p1: float, p2: float, p3: float)
var
        g: float;   // угол в градусах
        r: float;   // угол в радианах
        k: integer; // длина линии (параметр ф-и line)
begin
        k:= 43;
        writeln;
        writeln(' Таблица синусов-косинусов');
        writeln;
        Line(k,'_');
        writeln(' град.':7, '     рад.':12, '   синус':12,  '   косинус':12);
        Line(k,'-');
        g := p1;
        while g <= p2 do
          r := 3.14/180*g;
          writeln(g:7:2, r:12:6, sin(r):12:6, cos(r):12:6 );
          g:= g + p3;
        end;

        Line(k,'=');
        writeln;
end;


// главная программа
program main()
begin

    tabSin(0.0,360.0,15.0);

    write('Нажмите <Enter>');
    readln;
end.
```

Сортировка массива методом обменов

```
// Сортировка массива с использованием цикла for.
program p5()
const
    ARS = 5;
var
        a: array[1 .. ARS] of integer;
        min, max: integer;
        b: integer;

        i,j: integer;
begin

        a[1] := 8;
        a[2] :=-8;
        a[3] := -17;
        a[4] := 25;
        a[5] := 6;

        { вывод исходного массива }
        write('Source array: ');
        for i := 1 to ARS do
           write(a[i]:4);
        end;
        writeln;

        // поиск максимального элемента массива
        max:=1; // пусть первый элемент минимальный

        for i := 2 to ARS  do
           if (a[i] > a[max]) then
                max:=i;
           end;
        end;

        writeln('max=', a[max]);

        { сортировка }
        for i:= 1 to ARS do
           for j:=1 to ARS do
```

```
            if (a[j+1] < a[j]) then
                { обмен }
                b:= a[j];
                a[j] := a[j+1];
                a[j+1] := b;
            end;
         end;
      end;

      { вывод отсортированного массива }
      write('Sorted array: ');
      for i:=1 to ARS do
         write(a[i]:4);
      end;
      writeln;

      write('Нажмите <Enter>');
      readln;

end.
```

Прописью для целого числа в диапазоне от 1 до 999

```
// Формирует значение Прописью для целого числа в диапазоне от 1 до 999
function Prop(n: integer): string
var
    st: string[64];
    d1: array[1..19] of string[13] =
          'один ','два ','три ','четыре ','пять ','шесть ',
          'семь ', 'восемь ', 'девять ', 'десять ', 'одиннадцать',
          'двенадцать', 'тринадцать', 'четырнадцать', 'пятнадцать',
          'шестнадцать', 'семнадцать', 'восемнадцать', 'девятнадцать';
    d2: array[1..9] of string[12] =
          ' ', 'двадцать ','тридцать ','сорок ','пятьдесят ',
             'шестьдесят ','семьдесят ','восемьдесят ','девяносто ';
    d3: array[1..9] of string[10] =
             'сто ','двести ','триста ','четыреста ','пятьсот ',
             'шестьсот ', 'семьсот ','восемьсот ','девятьсот ';

    sot: integer; // количество сотен
    dec: integer; // количество десятков, если число больше 19
```

```
    ed: integer;  // количество единиц

begin
    st:='';

    sot := n div 100;
    if sot != 0 then
        st := d3[sot];
        n := n - sot *100;
    end;

    if N !=0 then
        if (n<=19) then
            st := st + d1[n];
        else
            dec := n div 10;
            st:=st+ d2[dec];
            ed := n - dec *10;
            if (ed !=0 ) then
                st := st+d1[ed];
            end;
        end;
    end;
    return st;
end;


// преобразует первую букву строки к верхнему регистру
function Capital(st: string):string
var
    res: string[128];
begin
    res := UpCase(substr(st,1,1)) + LowCase(substr(st,2,length(st)-1));
    return res;
end;

Program p()

var
    n: integer;
     st: string[128];
```

```
    i:integer;

begin

    writeln('Type number from 1 to 999 and press <Enter>');
     writeln('To stop typy 0 or press <Enter>');
    repeat
        writeln;
        write('>');
        readln(n);
        if (n >= 0) AND (n <1000) then
            st := Capital(Prop(n));
            writeln(st);
        end;
    until(n = 0);

    writeln;

    for i:=1 to 25 do
        n:= Random(999);
        writeln(n:3, ' - ', Capital(Prop(n)));
    end;

    write('Press <Enter>');
    readln;
end.
```

Сортировка двумерного массива

```
// Сортировка двумерного массива.
program TwoDimArrSorting ()
const
     NR=6;
    NC=16;
var
    a: array[1..NR, 1..NC] of integer;
    i,j: integer;
    key: integer; // key column index
    m: integer;   // min rows element index
    c: integer;   // row index in
```

```
      b: integer;

begin
    // random array for sorting
    for i:= 1 to NR do
        for j:=1 to NC do
            a[i,j] := Random(100);
        end;
    end;

    writeln('Source array:');
    for i:= 1 to NR do
        for j:=1 to NC do
            write(a[i,j]: 4);
        end;
        writeln;
    end;


    key:=1;
    for i:=1 to NR do // повторить столько раз, сколько строк в массиве
        // найти минимальный элемент в key столбце массива от j-ого элемента
        m:=i;
        for j:=i+1 to NR do
            if a[j,key] < a[m,key] then
                m:=j;
            end;
        end;

        if m != i then
            // обменять i-ую и m-ую строки массива
            for c:=1 to NC do
                b:=a[i,c];
                a[i,c]:=a[m,c];
                a[m,c]:=b;
            end;
        end;
    end;

    writeln;
    writeln('Key column: ', key);
```

```
    writeln('Sorted array:');
    for i:= 1 to NR do
        for j:=1 to NC do
            write(a[i,j]: 4);
        end;
        writeln;
    end;

      writeln;
    write('Press <Enter>');
    readln;
end.
```

Функция программиста Volume – объем цилиндра

```
// Функция программиста Volume - объем цилиндра.
Function CylinderVolume(d: integer, len: integer):float
const
   PI = 3.1415926;
begin
    return PI*(d/2)*(d/2)*len;
end;


// объем полого цилиндра
Program P1()
var
    diam: integer;  // диаметр
     wal: integer;  // толщина стенки
     len: integer;  // длина
     volume: float; // объем
begin
    writeln('Объем полого цилиндра');
     write('Диаметр, мм >');
     readln(diam);
     write('Толщина стенки, мм>');
     readln(wal);
     write('Длина, мм >');
     readln(len);
```

```
    // объем в мм куб.
    volume := CylinderVolume(diam,len) - CylinderVolume(diam-2*wal,len);

    volume := volume / 1000; // объем в см куб.

    writeln('\nОбъем полого цилиндра', volume:9:2, ' см.куб.');

     writeln;
     write('\nPress <Enter>');
     readln;
end.
```

Процедура и функция программиста Приветствие

```
// Процедура и функция программиста Приветствие
// процедура
// выводит приветствие на русском, итальянском, испанском или английском языке
procedure hi(lang: string)
begin
    if lang = 'ru' then
        writeln('Привет!');
    else
        if lang = 'it' then
            writeln('Ciao!');
        else
            if lang = 'es' then
                writeln('Ola!');
            else
                writeln('Hi!');
            end;
        end;
    end;
end;

// функция
// возвращает приветствие на русском, итальянском. испанском или английском языке
function GoodDay(lang: string): string
var
    msg: string[15];
begin
    if lowcase(lang) = 'ru' then
```

```
        msg:='Добрый день!';
    else
        if lowcase(lang) = 'it' then
            msg:='Buongiorno!';
        else
            if lowcase(lang) = 'es' then
                msg:='Buones dias!';
            else
                msg:='Good day!';
            end;
        end;
    end;
    return msg;
end;


program p()
var
    LangID: string[10]; // идентификатор языка:
                        // ru - русский
                        // en - английский
                        // it - итальянский
                        // es - испанский
begin
    repeat
        write('Language ID (ru, en, it, es)>');
        readln(LangID);
        if length(LangID) !=0 then
            hi(lowcase(LangID));
            writeln(GoodDay(LangID));
            else
              hi(lowcase('ru'));
            writeln(GoodDay('ru'));
        end;
    until length(LangID) = 0;

    write('Press <Enter>');
    readln;

end.
```

Рекурсивная функция Факториал и таблица факториалов

```
// Рекурсивная функция Факториал и таблица факториалов от 1 до 12
// Функция Факториал
function fac(n: integer): integer
var
    f: integer;
begin
    if n = 1 then
          f:=1;
    else
          f:= n*fac(n-1);
    end;

    return f;
end;


// Таблица факториалов от 1 до 12
program p28()
var
        i: integer;
begin
        for i:=1 to 12 do
             writeln(i:2, ' - ', fac(i));
        end;

        write('Press <Enter>');
        readln;
end.
```

Рекурсия. Поиск маршрутов между двумя точками графа

```
// Рекурсия. Поиск всех маршрутов между двумя точками графа
program findRoad()

var
     n: integer; //кол-во вершин графа
     map:array[1..7,1..7] of integer; // карта (граф): map[i,j] не 0, если точки i
и j соединены

     road:array[1..7] of integer; // маршрут - номера точек карты
```

```
     incl:array[1..7] of integer; // incl[i]=1, если точка с номером i включена в
road

     start: integer; // начальная точка (откуда)
     finish:integer; // конечная точка (куда)

     i: integer;
     j:integer;

     r: integer;


function step(s: integer,f: integer,p:integer): integer
               // s - точка, из которой делается шаг
               // f - точка, куда надо попасть (конечная)
               // p - номер искомой точки маршрута
var
     c:integer;// Номер точки, в которую делается очередной шаг
     r: integer;
begin

     if s=f then                // Точки s и f совпали!
          write('Маршрут: ');
          for i:=1 to p-1 do
        write(road[i],' ');
        end;
          writeln;

     else
          // Выбираем очередную точку
          for c:=1 to N do
             // Проверяем все вершины
             if(map[s,c] !=0) and (incl[c]=0)
                    // Точка соединена с текущей и не включена
                    // в маршрут
                then
                     road[p]:=c;   // Добавим точку в маршрут
                     incl[c]:=1;   // и пометим ее
                                   // как включенную
                     r:=step(c,f,p+1);
                     incl[c]:=0;
```

```
                    road[p]:=0;
                end;
            end;
      end;
end; // функция step

// Основная программа
begin
     N:=7;

     for i:=1 to N do
        for j:=1 to N do
        map[i,j]:=0;
        end;
     end;

     // ввод карты
     map[1,2]:=1;
     map[1,3]:=1;
     map[1,4]:=1;

     map[2,1]:=1;

     map[3,1]:=1;
     map[3,4]:=1;
      map[3,7]:=1;

      map[4,1]:=1;
      map[4,3]:=1;
      map[4,6]:=1;

      map[5,6]:=1;
      map[5,7]:=1;

       map[6,4]:=1;
       map[6,5]:=1;
       map[6,7]:=1;

       map[7,3]:=1;
       map[7,5]:=1;
       map[7,6]:=1;
```

```
    // показать "карту"
    for i:=1 to N do
       for j:=1 to N do
       write(map[i,j]:3);
       end;
       writeln;
    end;

    repeat
       // новый маршрут
       for i:=1 to N do
           road[i]:=0;  // нет маршрута
           incl[i]:=0;  // нет включенных точек
       end;

       writeln('Поиск маршрута');
       write('Начальная точка ->');
       readln(start);

       if start != 0 then
           write('Конечная точка ->');
           readln(finish);
           writeln;

           road[1]:=start;   // внесем точку в маршрут
           incl[start]:=1;   // и пометим ее как включенную

           r:=step(start,finish,2); // ищем вторую точку маршрута
       end;
    until start = 0;

    writeln;
    write('Для завершения нажмите <Enter>');
    readln;
end.
```

Обработка строк. Пользовательские функции Trim и Capital

```
// Обработка строк. Пользовательские функции Trim и Capital
```

```
// преобразует первую букву строки к верхнему регистру
function Capital(st: string):string
var
    res: string[128];
begin
    res := UpCase(substr(st,1,1)) + LowCase(substr(st,2,length(st)-1));
    return res;
end;


// удаляет начальные и завершающие пробелы
function Trim(st: string):string
var
    trs: string[64];   // строка без начальных и завершающих пробелов
    p: integer;        // указатель на робел в начае строки
    lch: string[1];     // последний символ строки

begin
   trs:=st;

   // убрать начальные пробелы
   p:= pos(' ',trs);
   while p = 1 do
      trs:=substr(trs,2,length(trs)-1);
       p:= pos(' ',trs);
   end;

    // убрать завершающие пробелы
   lch := substr(trs,length(trs),1);
   while lch = ' ' do
      trs:=substr(trs,1,length(trs)-1);
      lch := substr(trs,length(trs),1);
   end;

   return trs;
end;


program p1()
var
        name: string[25];
```

```
     lastName: string[15];
     firstName: string[15];

     p: integer;   //позиция пробела между first и last name

begin
     repeat
        writeln;
        write('name>');
        readln(name);

        if (Length(name) != 0) then
            name := Trim(name); // убрать начальные и завершающие пробелы
            p:= Pos(' ', name);
            if p !=0 then
                firstName := Capital(LowCase(Substr(name, 1, p-1)));
                lastName:= Capital(LowCase(Trim(Substr(name, p+1, length(name)-
p))));
            else
                firstName := Capital(LowCase(name));
                lastName:='';
            end;

            writeln('Name:', name);
            writeln('First Name:', firstName, ': Last name:', lastName, ':');
            name := firstname + ' ' + lastname;
            writeln('Name:', name, ':');
        end;
    until (length(name) = 0);

    write('Press <Enter>');
    readln;
end.
```

Криптограф. Шифрует/дешифрует текст

```
// Криптограф. Шифрует/дешифрует текст. Демонстрирует работу со строками

program crypto()
var
```

```
    alf: string[64];  // алфавит

    src: string[128]; // исходный текст
    dst: string[128]; // зашифрованный текст
    rst: string[128]; // декодированный текст


    key: string[32];  // ключ
    n: integer;  // номер буквы ключа, используемой для
         // кодирования/декодирования текущего символа сообщения

    // "код символа" это - порядковый номер символа в алфавите alf
    pk: integer;    // код символа ключа
    ps: integer;    // код символа исходного сообщения
    pd: integer;    // код символа, которым заменяется символ сообщения

    i: integer;

begin
    alf:= 'abcdefghijklnmopqrstuwvxyz0123456789 .,!?$';

    key := 'bartsimpson'; // кодовое слово должно состоять из символов алфавита

    src := 'Hello, James Bond!  Die Another Day... $100.00';

    src := LowCase(src);


    // шифруем
    n:=1;
    for i:=1 to length(src) do

        ps:= pos(substr(src,i,1),alf);
        if ( ps !=0) then // символ есть в алфавите?
            // да, кодируем
            pk:= pos(substr(key,n,1),alf);

            ps:= ps+pk;
            if  ps > length(alf)  then
                ps:= ps-length(alf);
            end;
```

```
        dst:=dst+substr(alf,ps,1);
    else
        // оставляем "как есть"
        dst:=dst+ substr(src,i,1);
    end;

    n := n + 1;
    if n > length(key) then
        n := 1;
    end;
end;


// дешифруем
n:=1;
for i:=1 to length(dst) do

    ps:= pos(substr(dst,i,1),alf);
    if ( ps !=0) then
        pk:= pos(substr(key,n,1),alf);

        ps:= ps-pk;
        if  ps < 1  then
            ps:= ps+length(alf);
        end;

        rst:=rst+substr(alf,ps,1);
    else
        rst:= rst+ substr(dst,i,1);
    end;

    n := n + 1;
    if n > length(key) then
        n := 1;
    end;
end;

writeln(' Source message:', src);
writeln('  Coded message:', dst);
  writeln;
```

```
    writeln('Decoded message:', rst);

    write('Press <Enter>');
    readln;

end.
```

Генератор паролей

```
// Генератор паролей

program PWGen()
const
     PWLEN = 10; // длина пароля
     N = 7 ;     // количество вариантов пароля
var
    pw: string[PWLEN];    // пароль
    alp: string[128];      // алфавит

    r: integer; // случайное число - номер символа алфавита
    i: integer; // номер генерируемого символа пароля

     up: integer; // 1 - преобразовать букву в строчную; 2 - оставить как есть

    j: integer;

begin
    // набор символов
    alp := 'abcdefghijklmnopqrstuwvxyz0123456789!$?#_';

    for j:=1 to N do
           // сгенерировать пароль
        pw := '';
        for i:= 1 to PWLEN do
            r := Random(length(alp));
            up := Random(2);
            if ( up = 1) then
                pw:=pw + Upcase(substr(alp,r,1));
             else
                 pw:=pw + substr(alp,r,1);
             end;
```

```
        end;

        writeln(j:3, '. ', pw);
    end;

    writeln;
    write('Press <Enter>');
    readln;
end.
```

Запись чисел в файл, чтение чисел из файла

```
// Запись целых чисел (строк) в файл. Чтение целых чисел из файла (чтение строк и преобразование в целое)

program p23()
var
      st: string[15];
      k: integer;       // число
      f: text;          // файл
      fn: string[64];   // имя файла
      sum: integer;     // сумма чисел
      n: integer;       // кол-во чисел
      med: float;       // среднее арифметическое

begin

      //fn:='numbers.txt';
      fn:= 'c:\users\Nikita\Desktop\data.dat';
      writeln('Файл данных: ',fn);

      // записать числа в файл
      f:=rewrite(fn);

      for k:=1 to 10 do
        //writestring(f, IntToStr(k));
        writestring(f, IntToStr( random(10)) );
      end;

      close(f);
```

```
      // читать числа из файла

      f:= reset(fn);
      if (f != -1) then
          n:=0;
          sum:=0;
          while (eof(f) != 1) do
              n:=n+1;
              k:= StrToInt(readstring(f));
              sum:= sum + k;
              writeln(k:4);
          end;

          close(f);

          if sum != 0 then
              med:= sum/n;
          end;

          writeln('Чисел в файле:',n:5);
          writeln('Сумма чисел:',sum:5);
          writeln('Среднее арифметическое:',med:6:2);
      end;

      write('Press <Enter>');
      readln;
end.
```

Вывод на экран содержимого текстового файла

```
// Чтение и вывод на экран текстового файла.

// Рисует линию
procedure Line(ch: string, n: integer)
var
    i: integer;
begin
    for i := 1 to n do
        write(ch);
    end;
```

```
    writeln;
end;

program p20()
var
        f: text;  // текстовый файл
        fn:string[64];  // имя файла

        st: string[128]; // строка, прочитанная из файла
        n: integer;  // количество строк

begin
        // файл находится в папке Документы/pas
        fn:= 'C:\Users\nikita\documents\pas\p20.pas';

        f:=reset(fn);  // открыть файл для чтения
                       // функция возвращает -1, если по каой-либо причине
                       // доступ к файлу не получен

        if f != -1 then
        writeln(fn);
        Line('-',length(fn));
            n:=0;

        while (eof(f) != 1) do  // пока не достигнут конец файла
            n:=n+1;
            st := readstring(f);    // читать строку из файла
            writeln(n:3, '  ', st);
        end;

        Line('-',length(fn));

    else
            writeln('Ошибка доступа к файлу ', fn);
            writeln('Неверное имя/путь или файл используется другим приложением');

         end;

        writeln;
        write('Press <Enter>');
        readln;
```

```
end.
```

Дата и время

```
// Демонстрирует использование функций GetTime, GetDay, GetMonth, GetYear,
DayOfWeek

// Возвращает время в формате hh:mm:ss
function TimeToStr(time: integer):string
var
        st: string[8];
        hour: integer;
        min: integer;
        sec: integer;
begin
        hour:= Trunc(time/60/60);
        min:= Trunc((time - hour*3600)/60);
        sec:= time- hour*3600 -min*60;

        st:='';
        if hour < 9 then
                st:= '0';
        end;
        st:= st + IntToStr(hour) +':';

        if min < 9 then
                st:= st + '0';
        end;
        st:= st + IntToStr(min) + ':';

        if sec < 9 then
                st:= st+ '0';
        end;

      st:= st + IntToStr(sec);

        return st;
end;

// возвращет дату в формате dd/mm/yyyy
```

```
function ShortDate(day: integer, month: integer, year: integer): string
var
    st: string[10];
begin
    st:='';
    day := getDay();
    if day < 10 then
        st:='0';
    end;
    st:= st + IntToStr(day)+'/';

    month := getMonth();
    if month < 10 then
        st:=st+'0';
    end;
    st:= st+IntToStr(month)+'/';

    year := getYear();
    st:=st+ IntToStr(year);

    return st;
end;


program p1()
var
    day: integer;
    month: integer;
    year: integer;
    dayOfweek: integer;

    weekDay: array[1..7] of string[11] =
              'воскресенье', 'понедельник', 'вторник', 'среда',
              'четверг', 'пятница', 'суббота';
    monthName: array[1..12] of string[10] =
              'январь', 'февраль', 'март', 'апрель', 'май', 'июнь',
              'июль', 'август', 'сентябрь', 'октябрь', ,'декабрь';

    hour: integer;
    min: integer;
    sec: integer;
```

```
    time: integer;
    time2: integer;
    dtime: integer;

    i: integer;

begin
     writeln('Сегодня ', getDay(), ' ', monthName[getMonth()], getYear():5, ', ' ,
weekDay[getDayOfWeek()+1]);

     day := getDay();
     dayOfweek:= getDayOfWeek();
     month := getMonth();
     year := getYear();

    writeln('Сегодня ', day, ' ', monthName[month], year:5, ', ' ,
weekDay[dayOfweek+1]);
     writeln('Сегодня ', day, ' ', monthName[month] , year:5);
     writeln('Сегодня ', day, '-', month, '-', year);

    time := getTime();
    hour:= time div 60 div 60;
    min:=  (time - hour*3600)  div 60;
    sec:=  time- hour*3600 - min*60;

    writeln('Сейчас ', hour, ':', min, ':', sec);
    writeln;
    writeln;

    for i:=1 to 7 do
        writeln(i-1:2,' - ', weekDay[i]);
    end;

    writeln;

    day := getDay();
    month := getMonth();
    year := getYear();
    dayOfweek := getDayOfWeek();
    time := getTime();
```

```
    writeln('Today ', ShortDate(day, month, year), ' ', weekDay[dayOfWeek+1], '
(',dayOfWeek, ')');

    writeln;
    writeln('Now ', TimeToStr(time));

    write('Wait 15 sec and press <Enter>');
    readln;

    time2:=getTime();
    writeln('Now ', TimeToStr(time2));

    dtime:= time2- time;
    writeln('You where waiting ', TimeToStr(dtime));

        write('Press <Enter>');
        readln;
end.
```

Hangman game

```
// Hangman game
Program HangmanGame()
const
     NW = 8;   // количество слов
     LW = 15;  // максимальное количество букв в слове

     TRUE = 1;
     FALSE = 0;
var
    words: array[1..NW] of string[15]  = 'hangman', 'apple', 'pascal',
                         'russia', 'italia', 'book','notebook','pencil';

    secretWord:string[LW];
    userWord:string[LW];

    ch: string[1]; // буква, введенная пользователем

    k: integer; // количество букв, которое ввел игрок

    misses:string[15]; // буквы, которых нет в слове (8 букв + 7 запятых)
```

```
    st: string[LW];

    right:integer;

    i: integer;

    debug: integer; // 1 - режим отладки, показать секретное слово

begin

    debug := 1;

    writeln;
    writeln('Welcome to the Hangman game!');
    writeln;
    secretWord := words[Random(NW)];

    userWord := '';
    for i := 1 to Length(secretWord) do
          userWord := userWord + '-';
    end;

    if debug = 1 then
       WriteLn('Secret word:',secretWord);
       WriteLn('User word:',userWord);
    end;

    k := 0;
    repeat
        writeln;
        WriteLn('\nСлово:', UpCase(userWord));
        WriteLn('Нет букв:', UpCase(misses));
        Write('\nБуква>');

        ReadLn(ch); // the first character of the entered line

        st := '';
        right := FALSE;

        for i := 1 to Length(secretWord) do
```

```
            if substr(secretWord,i,1) = ch then
                 st := st + substr(secretWord,i,1);
                 if NOT right then
                     right := TRUE;
                 end;
            else
                 st := st + substr(userWord,i,1);
            end;

        end;
        userWord := st;

        if NOT right then
            if Length(misses) = 0 then
                  misses := misses + ch;
             else
                  misses := misses +',' + ch;
             end;
         end;

         k := k + 1;

    until  (k = 8) OR (userWord = secretWord);

    writeln;
    if (userWord = secretWord) then
            WriteLn('You are win!');
    else
            WriteLn('You are lost!');
    end;

    WriteLn('The seecret word is ', UpCase(secretWord));

    Write('\nPress <Enter>');
    Readln;

end.
```

Массив записей

```
// Пользовательский тип: запись
```

```
program p1()

const
   N = 3;
   PI=3.14;

type

   TMaterial = record
       title: string[15];      // Названиe англ.
       density: float;         // плотность
   end;

   TTabMat = array[1..N] of TMaterial;


function Menu(var materials: TTabMat): integer
var
    p: integer; // номер выбранного материала
    i: integer;
begin
    writeln('\n Материал');
    for i:=1 to N do
        writeln(i:2, '. ',materials[i].title);
    end;
    writeln('\n 0 - выход');
    write('\n Ваш выбор>');

    readln(p);
    return p;
end;

var
   // таблица материалов - одномерный массив записей
   tabmat:  TTabMat;

   diameter, len: integer; // диаметр и длина стержня

   volume, weight: float;  // объем и вес стержня

   k: integer; // номер материала
```

```
begin

   tabmat[1].title:='Aluminium';
   tabmat[1].density := 2.7;

   tabmat[2].title := 'Cooper';
   tabmat[2].density := 8.94;

   tabmat[3].title := 'Steel';
   tabmat[3].density := 7.86;

   k := Menu(tabmat);

   if (k > 0) and (k <= N)
   then
      diameter := 21;
      len:=350;

      volume := PI * (diameter /2) * (diameter /2) * len/1000;
      writeln('\n\nДлина:', len, ' мм\nДиаметер:', diameter,' мм');
      writeln('Объем:', volume:4:2, 'см.куб.');

      writeln('Материал: ', tabmat[k].title, ' (', tabmat[k].density:6:2, ' г/см
куб.)');

      weight := volume * tabmat[k].density;
      write('Масса: ');
      if weight < 1000 then
          writeln(weight:8:2, ' гр.');
      else
          writeln(weight/1000:8:2, ' кг.');
      end;

   end;


   write('\n\nPress <Enter>');
   readln;
end.
```

Вектор (тип-массив, массив как парамертр функции/процедуры)

```
// Определенный пользователем одномерный тип-массив (вектор)
// Массив, как параметр функции и процедуры
program p1()

const
    N = 5;

type
    vector = array[1..N] of integer;

// сумма элементов массива
function SumItems(var v: vector, k: integer): integer
var
   s: integer;
   i: integer;
begin
   s := 0;
   for i:=1 to k do
      s := s + v[i];
   end;

   return s;
end;


// возвращает номер максимального элемента
function MaxItem(var v: vector): integer
var
   m: integer; // номер максимального элемента
   i: integer;
begin
   m := 1;
   for i := 2 to N do
       if v[i] > v[m] then
            m := i;
        end;
   end;

   return m;
```

```
end;

// вывод вектора
procedure Print(var v: vector)
var
    i: integer;
begin
    for i:=1 to N-1 do
       write(v[i]:4,',');
    end;
    writeln(v[i]:4);
end;



var
    v1: vector;   // вектор
    sum: integer; // сумма элементов массива (вектора)
    m: integer;   // номер максимального элемента
    i: integer;

begin

   for i:=1 to N do
       v1[i] := Random(100);
   end;

   Print(v1); // вывод вектора

   sum := SumItems(v1,n);
   writeln('\nItems sum: ', sum);

    m := maxItem(v1);
    writeln('\nMax item:');
    writeln(' - index: ',m);
    writeln(' - value: ', v1[m]);


    write('\nPress <Enter>');
    readln;
end.
```

Матрица (тип-массив, массив как параметр функции/процедуры)

```
// Тип, определенный пользователем - матрица (двумерный массив)
program p1()

const
    N = 5;

type
    matrix = array[1..N, 1..N] of integer;


// инициализация матрицы
procedure MatrixInit(var m: matrix, r: integer)
var
   i,j: integer;
begin
   for i:=1 to N do
      for j:=1 to N do
         m[i,j] := Random(r);
      end;
   end;
end;

// сумма матриц
procedure MatrixSum(var m1: matrix, var m2: matrix, var m3: matrix)
var
   i,j: integer;
begin
   for i:=1 to N do
      for j:=1 to N do
         m3[i,j] := m1[i,j] + m2[i,j];
      end;
   end;
end;

// печать матрицы
procedure MatrixPrint(var m: matrix)
var
   i,j: integer;
begin
```

```
   for i:=1 to N do
      for j:=1 to N do
         write(m[i,j]:5);
      end;
      writeln;
   end;
end;



procedure Line(st:string, len: integer)
var
    i: integer;
begin
    for i:=1 to len do
        write(st);
    end;
    writeln;
end;

var
    m1,m2,m3: matrix;       // матрицы

begin

   // инициализация матриц
   MatrixInit(m1, 100);
   MatrixInit(m2, 100);

   MatrixSum(m1,m2,m3); // сумма матриц

   MatrixPrint(m1);

   Line('-', 5*n);
   MatrixPrint(m2);

   writeln('\n');
   Line('-', 5*n);
   MatrixPrint(m3);

   write('\n\nPress <Enter>');
```

```
   readln;

end.
```