DELPHI

Советы программистов

Второе издание,
дополненное

под редакцией В. Озерова

Санкт-Петербург
2004

Delphi. Советы программистов

под редакцией В. Озерова

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Редактор	*В. Овчинников*
Корректура	*С. Журавина*
Верстка	*А. Дорошенко*
Художник	*С. Борин*

Озеров В.
Delphi. Советы программистов. – СПб: Символ-Плюс, 2004. – 976 с., ил.
ISBN 5-93286-037-5

Это издание представляет собой коллекцию ответов на нетрадиционные вопросы программирования на Delphi, нестандартных решений, интересных идей. Примеры рабочего кода, которые создавали многие программисты, собирались более двух лет и охватывают широкий круг вопросов: реализацию математических алгоритмов и работу с функциями Windows API, применение массивов, работу с графикой, а также управление рабочим столом, реестром, папками и файлами Windows, форматирование дискет, взаимодействие с аппаратным обеспечением. Значительное внимание уделено базам данных: таблицам dBASE и Paradox, настройке Delphi для работы с базами данных, подключению сервера Oracle или InterBase, особенностям использования SQL. Те, кто интересуется мультимедиа, найдут в книге советы по работе со звуком.

Рассмотрены создание компонентов с нужными свойствами, а также способы изменения или дополнения уже созданных. Этой теме посвящена самая большая глава сборника. Примеры, имеющие отношение к классам Delphi, помогут понять особенности взаимодействия в MDI- и SDI-приложениях, освоить создание новых форм и управление ими. Сборник также содержит советы по работе с Интернетом и применению механизма OLE для обмена данными в приложениях.

ISBN 5-93286-037-5

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 6.01.2004. Формат 70x100 1/16 . Печать офсетная.
Объем 61 печ. л. Доп. тираж 2000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Посвящается моей жене и дочке

Введение

Эта книга – не справочник и не учебник, и ее материал рассчитан на подготовленного читателя, хорошо освоившего приемы работы в среде Delphi. Если более точно, то книга рассчитана на программистов среднего или более высокого уровня.

Перед вами коллекция ответов на нетрадиционные вопросы программирования на Delphi, нестандартных решений, хитростей и интересных замыслов. Для практической пользы дела приведены конкретные примеры кода, позволяющие донести идею или полностью ответить на заданный вопрос.

Не удивляйтесь, обнаружив в книге код для Delphi 1 или даже для Turbo Pascal. Сам Pascal практически не изменился, а идеи, реализация и технология живы до сих пор. Для того чтобы найти описание какой-либо функции, можно заглянуть в электронный справочник Delphi, а «Советы» скорее предназначены для поиска общего решения.

Составитель не вносил изменений в листинги авторов и, по возможности, в сопровождающие их описания.

История проекта

Начало «Советам по Delphi» было положено в 1998 году, когда автор стал искать на просторах Интернета и копить многочисленные FAQ, посвященные Delphi. Но пользоваться ими было жутко неудобно, все они имели разный формат, содержали дублирующий текст и претендовали на полный охват всех аспектов программирования. Само собой, чтобы найти нужный ответ, нужно было заглянуть в каждый из файлов. О систематизации и поиске в ту пору задумывались очень немногие, поэтому автору удобнее было после удачного поиска и реализации кода «откладывать» совет в отдельную папочку с именем, совпадающим с тематикой совета.

На одном из этапов был внедрен формат справочной системы MS HTML Help, имеющий развитую систему поиска, закладок и древовидный каталог разделов. На систематизацию материала ушел месяц, но это значительно облегчило дальнейшее добавление новых советов.

Этот опыт завершился успехом, и в 1999 году первая версия «Советов» была выложена в Интернете.[1] Волею пиратов сборник попал на компакт-диск с

[1] Сначала на сервере XOOM по адресу *http://members.xoom.com/kuliba/*, потом на *http://www.webmachine.ru/delphi*.

новой версией Delphi и был растиражирован по всей России (и не только). Была организована система подписки, которая осуществлялась простой отправкой письма автору. Письма с просьбой о подписке идут автору до сих пор. Поскольку количество подписчиков резко возросло и перевалило за тысячу, было принято решение наладить полноценный оффлайн-сервис, в связи с чем примеры пополнялись и выкладывались с периодичностью в две недели, а советы, присланные читателями, обрабатывались и сразу включались в сборник. По всему было видно, что получился оффлайн-клуб программистов, нуждавшихся в заочном общении.

По истечении двух лет, когда количество советов приблизилось к двум тысячам, появилась идея обобщить этот труд, переведя его в иное качество. Для книги был отобраны лучшие примеры, и то, что вы держите в руках, по объему составляет примерно половину исходного материала. Надеюсь, что лучшую половину.

Структура книги

Книга состоит из 13 глав.

- Глава 1 «Алгоритмы». Здесь вы найдете материал и примеры, относящиеся к вопросам преобразования типов, реализации математических алгоритмов и работе с датами, а небольшая часть советов посвящена кодированию информации.
- Глава 2 «API». В главе приведен материал, касающийся работы с функциями Windows API. Не секрет, что многие функции API остались «за бортом» VCL и не имеют соответствующей инкапсуляции. Как изменить системное время? Как завершить работу Windows? Как получить список запущенных приложений? Круг рассматриваемых тем невелик, но в данной главе много примеров.
- Глава 3 «Pascal». Основное внимание в этой главе уделено работе с массивами. Показаны способы создания динамических массивов и использование адресной арифметики. Не оставлена без внимания работа с графической информацией. Как загрузить графический файл или нарисовать ломаную линию, вывести на экран надпись под углом или сделать ее прозрачной?
- Глава 4 «Базы данных». В данной главе собрана информация, необходимая для работы с базами данных. Большая часть материала посвящена работе с уже давно известными Paradox и dBASE. Но рассмотрена масса «тонких» мест, незаслуженно оставленных без внимания в других книгах: особенности доступа, запросы и восстановление информации. После ознакомления с материалом этой главы работа с базами данных станет для читателя обыденным делом.
- Глава 5 «BDE». Как настроить Delphi для работы с базами данных или подключить сервер Oracle или InterBase? Каковы особенности использования SQL? Ответы на эти и другие вопросы можно найти в этой главе.

- Глава 6 «Мультимедиа». Глава не так велика, но дает возможность познакомиться с принципами использования звукового сопровождения в приложениях.
- Глава 7 «Аппаратное обеспечение». Как из приложения определить конфигурацию компьютера, управлять клавиатурой или мышью, воспользоваться модемом? Эти и многие другие вопросы рассмотрены в данной главе.
- Глава 8 «Операционная система». Здесь описывается работа с Windows: управление рабочим столом, реестром, папками и файлами, форматирование дискет и т. д.
- Глава 9 «Компоненты». Это самая большая глава сборника. Здесь читатель узнает, как создавать компоненты с нужными свойствами или внести изменения или дополнения в уже существующие; как исправить ошибки в стандартном наборе или понять принцип и особенности использования компонентов.
- Глава 10 «Классы». Здесь приведен материал, имеющий отношение к классам Delphi. Эта глава поможет понять механизм взаимодействия в MDI- и SDI-приложениях, создание новых форм и управление ими. Обсуждается работа с буфером обмена Clipboard.
- Глава 11 «Интернет». Несколько советов для работы с Интернетом.
- Глава 12 «OLE». В примерах этой небольшой главы рассмотрен механизм обмена информацией в приложениях.
- Глава 13 «Часто задаваемые вопросы (FAQ)». Данные вопросы взяты из различных конференций, связанных с программированием в среде Delphi. Решения специально помещены в отдельную главу, т. к. почти не содержат поясняющего текста и могут быть трудными для начинающих программистов. Но не отбрасывайте их. Воспользуйтесь отладчиком и посмотрите, как они работают. И через некоторое время исчезнет еще одно темное пятно. Этот материал создан группой поддержки разработчиков Borland Developer Support и может быть получен с сайта *community.borland.com* (на английском языке). Печатается с разрешения фирмы Borland. Перевод выполнен Олегом Кулабуховым.

Предупреждение

В книге приведено большое количество исходного кода. Составитель и технические рецензенты проверили большинство решений и трюков и надеются, что у читателя не будет проблем с этими программами.

Тексты наиболее интересных примеров можно скачать по адресу *http://www.symbol.ru./library/delphi_secrets/*.

Составитель не отвечает за последствия применения приведенного кода. Используйте его на свой страх и риск. Если ваш компьютер взорвется из-за какого-нибудь совета, не обвиняйте составителя и не шлите гневные письма.

Составитель не претендует на авторство приведенных в книге решений. Автор или тот, кто прислал решение, указан после совета в квадратных скобках. В разделе «Перечень авторов» приведены их электронные адреса (возможно, на данный момент некоторые из них устарели).

Группа подготовки издания приложила максимум усилий к тому, чтобы установить авторство примеров, приведенных в книге. Мы сожалеем, что для некоторых советов автор не указан, и заранее приносим извинения тем, кто обнаружит в книге свои советы без подписи. Тот, кто прислал данный совет, забыл указать свое имя или источник информации, из которого это решение было почерпнуто. Сообщения о подобных случаях отправляйте составителю по адресу *webmaster@webmachine.ru* для внесения изменений в электронный вариант справочника и следующее издание книги. Многие советы взяты из новостных групп, в которых код передается «из рук в руки», переделывается и усовершенствуется. Для такого «народного творчества» авторство также не указано.

Соглашения, принятые в данной книге

Чтобы читателю легче было ориентироваться в материале, в листингах программ, а также для выделения в тексте программных элементов, имен файлов и папок используется моноширинный шрифт.

От составителя

Занимаясь программированием уже более десяти лет, я пришел к мысли создать книгу, которая реально передавала бы опыт, накопленный сообществом программистов, как новичкам, так и маститым его представителям. Фрагменты информации с примерами разбросаны по Интернету и новостным группам. Большая их часть не систематизирована и не переведена на русский язык, а поиск конкретной информации отнимает неоправданно много времени. Программисты, отчаявшись ответить на свои вопросы самостоятельно, обязательно задавали их через какой-то промежуток времени другим программистам. Данная книга представляет собой подборку наиболее удачных, по мнению составителя, ответов на чаще всего задаваемые вопросы и в этом смысле является результатом многолетнего труда программистского сообщества.

Перечень рассмотренных тем ограничен лишь стремлением автора как-то систематизировать информацию. Реально же никаких рамок не существует, и круг решаемых средствами Delphi задач в среде Windows безграничен. Существует множество изданий и справочников, описывающих технологии и функции системы. Материал же данной книги лежит в несколько иной плоскости. Составитель стремился дать программисту идею, путь решения задачи. Между тем, приведенный код вполне работоспособен и может подсказать решение описываемой проблемы. Другой целью было желание привести решения для наиболее типичных задач, с которыми сталкивается программист. Отправной точкой для такого круга задач служили вопросы, с

завидной периодичностью задаваемые в новостных группах. И, наконец, была предпринята попытка представить подборку наиболее интересных с точки зрения программирования материалов.

Удалась книга или нет, решать вам, читатель. Составитель с благодарностью примет в свой адрес любые замечания и сообщения об ошибках. Это позволит сделать следующее издание книги более полезным и качественным.

Благодарности

Хотя книги часто пишутся писателями в одиночку, эта книга не такая. Это коллективный труд многих программистов, благодаря энтузиазму которых и желанию помочь другим данный сборник появился на свет.

Я выражаю искреннюю благодарность Скорикову Владимиру, без которого этой книги просто не было бы. Из огромного количества материала он выбрал наиболее интересные и достойные публикации, сделал первую вычитку и проверку кода. Его энтузиазм и желание помочь сделали свое дело – книга перед вами. Код Владимира также приведен в книге.

Второй человек, без которого книга не увидела бы свет – Тугаев Олег с его искренним желанием мне помочь. Он потратил все лето только на то, чтобы проверить программный код, который приведен в книге. Его советы также включены в данную книгу. Благодарность за понимание и огромное терпение адресована и его семье.

Отдельное спасибо читателям, приславшим мне письма с указанием на ошибки и правильным кодом. Они были моими самыми дотошными рецензентами.

Я признателен всем, кто помог мне преодолеть трудности и сделать выход этой книги состоявшимся событием. Прежде всего это относится к читателям моих «Советов по Delphi» в электронном виде. Они задавали мне ритм, подбрасывали головоломки и находили новые темы. Это моя команда, мои читатели, и они же авторы большинства советов. Благодарю корпорацию Borland, выпустившую столь замечательный продукт – Delphi.

Благодарю издательство «Символ-Плюс», предложившее мне издать книгу и терпеливо ожидавшее от меня материал.

Своих соседей за долготерпение во время моих ночных работ под недостаточно тихую временами музыку.

Наконец, я хочу поблагодарить всех своих домашних. Мои жена и дочка старались соблюдать тишину и не жаловались, даже когда я работал над книгой в воскресные дни и праздники. Теперь, после выхода книги, я смогу сходить с ними в обещанные театр и зоопарк.

Спасибо моим родителям, переживавшим за мой первый опыт в создании книги.

1

Алгоритмы преобразования

Преобразование шестнадцатеричной строки в целое

Как шестнадцатеричное значение преобразовать в целое?

Строка вида "6A0F" преобразуется в число 27151.


```
function HexToLong(HexStr: string): Integer;
begin
  Result := 0;
  try
    Result := StrToInt('$' + HexStr);
  except
    on E: EConvertError do ShowMessage('Convert error');
    on E: EIntOverflow do ShowMessage('Out of range');
  end
end;
```

Преобразование целого в шестнадцатеричную строку

Как целое значение преобразовать в шестнадцатеричное?

Число 27151 преобразуется в строку вида "6A0F".

```
HexStr := IntToHex(Value, Digits);
```

Преобразование ASCII в шестнадцатеричное значение

Строка представляет собой массив ASCII-символов. Как организовать преобразование по аналогии с Delphi-функциями Ord и Chr? Процедура BytesToHexStr преобразует набор байт [0,1,1,0] в строку «30313130», функция BytesToHex выполняет аналогичные действия, HexStrToBytes выполняет обратное преобразование, а HexBytesToChar переводит шестнадцатеричную строку в строку символов.

```
unit HexStr;

interface

procedure BytesToHexStr(var hHexStr: String; PBA: PByteArray; InputLength: Word);
function BytesToHex(PBA: PByteArray; Size: Word): string;
procedure HexStrToBytes(s: string; PBA: PByteArray);
procedure HexBytesToChar(var Response: String; PBA: PByteArray; InputLength: Word);

implementation

procedure BytesToHexStr(var hHexStr: String; PBA: PByteArray; InputLength: Word);
const
  HexChars: array[0..15] of Char = '0123456789ABCDEF';
var
  i, j: Word;
begin
  SetLength(hHexStr, InputLength * 2);
  j := 1;
  for i:= 0 to InputLength - 1 do begin
    hHexStr[j] := HexChars[PBA[i] shr 4];
    inc(j);
    hHexStr[j] := HexChars[PBA[i] and 15];
    inc(j);
  end;
end;

function BytesToHex(PBA: PByteArray; Size: Word): string;
var
  i: Word;
begin
  Result := '';
  for i := 0 to Size - 1 do
    Result := Result + IntToHex(Ord(PBA[i]), 2);
end;
```

```
procedure HexStrToBytes(s: string; PBA: PByteArray);
var
  i, j: Word;
begin
  j := 1;
  for i := 0 to (Length(s) div 2) - 1 do begin
    PBA[i] := StrToInt('$' + Copy(s, j, 2));
    inc(j, 2);
  end;
end;

procedure HexBytesToChar(var Response: String; PBA: PByteArray; InputLength: Word);
var
  i: Word;
  c: byte;
begin
  SetLength(Response, InputLength);
  for i := 0 to (InputLength - 1) do begin
    c := PBA[i] and $f;
    if c > 9 then Inc(c, $37) else Inc(c, $30);
    Response[i+1] := Chr(c);
  end;
end;

end.
```

Преобразование двоичного числа в десятичное

Как преобразовать двоичное значение в десятичное?

Решение 1

```
function Base10(Base2: Integer): Integer; assembler;
asm
  cmp   eax, 100000000  // проверка максимального значения
  jb    @1              // значение в пределах допустимого
  mov   eax, -1         // флаг ошибки
  jmp   @exit           // выход, если -1
@1:
  push  ebx             // сохранение регистров
  push  esi
  xor   esi, esi        // результат = 0
  mov   ebx, 10         // вычисление десятичного логарифма
  mov   ecx, 8          // преобразование по формуле 10^8-1
@2:
  mov   edx, 0          // удаление разницы
  div   ebx             // eax - целочисленное деление на 10,
                        // edx - остаток от деления на 10
  add   esi, edx        // результат = результат + разность[I]
  ror   esi, 4          // перемещение разряда
  loop  @2              // цикл для всех 8 разрядов
```

```
    mov   eax, esi        // результат функции
    pop   esi             // восстановление регистров
    pop   ebx
  @exit:
  end;
```

Решение 2

Функция IntToBin выполнит преобразование десятичного числа в двоичное, а BinToInt – обратное преобразование.

```
function IntToBin(Value: Longint; Size: Integer): String;
var
  i: Integer;
begin
  Result := '';
  for i := Size downto 0 do begin
    if Value and (1 shl i) <> 0 then Result := Result + '1'
    else Result := Result + '0';
  end;
end;

function IntToBin(Value: Longint; Size: Integer): String;
var
  i: Integer;
begin
  Result := '';
  for i := Size downto 0 do begin
    if Value and (1 shl i) <> 0 then Result := Result + '1'
    else Result := Result + '0';
  end;
end;
```

Преобразование Comp в String

Были какие-то разговоры о том, что тип данных Comp *является каким-то ущербным, некорректным и что даже не существует подпрограмм, осуществляющих конвертацию* Comp *в* String *и обратно.*

Предлагаемый модуль включает в себя следующие функции: CompToStr, CompToHex, StrToComp, а также CMod и CDiv, которые являются вспомогательными и представляют собой реализацию функций MOD и DIV для типа Comp.

```
unit Compfunc;
interface

type
  CompAsTwoLongs = record
    LoL, HiL: LongInt;
  end;
```

```
const
  Two32TL: CompAsTwoLongs = (LoL:0; HiL:1);

var
  Two32: Comp ABSOLUTE Two32TL;
{ Некоторые операции могут окончиться неудачей, если значение
  находится вблизи границы диапазона Comp }

const
  MaxCompTL: CompAsTwoLongs = (LoL: $FFFFFFF0; HiL: $7FFFFFFF);

var
  MaxComp: Comp ABSOLUTE MaxCompTL;

function CMod(Divisor, Dividend: Comp): Comp;
function CDiv(Divisor: Comp; Dividend: LongInt): Comp;
function CompToStr(C: Comp): String;
function CompToHex(C: Comp; Len: Integer): String;
function StrToComp(const S: String): Comp;

implementation

uses
  SysUtils;

function CMod(Divisor, Dividend: Comp): Comp;
var
  Temp: Comp;
begin
{ Примечание: Оператор / для типа Comps ОКРУГЛЯЕТ результат,
  а не отбрасывает десятичные знаки }
  Temp := Divisor / Dividend;
  Temp := Temp * Dividend;
  Result := Divisor - Temp;
  if Result < 0 then Result := Result + Dividend;
end;

function CDiv(Divisor: Comp; Dividend: LongInt): Comp;
begin
  Result := Divisor / Dividend;
  if Result * Dividend > Divisor then Result := Result - 1;
end;

function CompToStr(C: Comp): String;
var
  Posn: Integer;
begin
  if C > MaxComp then
    Raise ERangeError.Create('Comp слишком велик для преобразования в string');
  if C < 0 then Result := '-' + CompToStr(-C)
```

```
  else begin
      Result := '';
      Posn := 0;
      while True do begin
        Result := Char(Round($30 + CMod(C, 10))) + Result;
        if C < 10 then break;
        C := CDiv(C, 10);
        Inc(Posn);
        if Posn mod 3 = 0 then Result := ',' + Result;
      end;
    end;
  end;

  function CompToHex(C: Comp; Len: Integer): String;
  begin
    if (CompAsTwoLongs(C).HiL = 0) and (Len <= 8) then
      Result := IntToHex(CompAsTwoLongs(C).LoL, Len)
    else
      Result := IntToHex(CompAsTwoLongs(C).HiL, Len - 8)
              + IntToHex(CompAsTwoLongs(C).LoL, 8)
  end;

  function StrToComp(const S: String): Comp;
  var
    Posn: Integer;
  begin
    if S[1] = '-' then Result := -StrToComp(Copy(S, 2, Length(S) - 1))
    else if S[1] = '$' then      { Шестнадцатеричная строка }
      try
        if Length(S) > 9 then begin
  { Если строка некорректна, исключение сгенерирует StrToInt }
          Result := StrToInt('$' + Copy(S, Length(S) - 7, 8));
          if Result > 10 then Result := Result + Two32;
  { Если строка некорректна, исключение сгенерирует StrToInt }
          CompAsTwoLongs(Result).HiL := StrToInt(Copy(S, 1, Length(S) - 8))
        end else begin
  { Если строка некорректна, исключение сгенерирует StrToInt }
          Result := StrToInt(S);
          if Result < 0 then Result := Result + Two32;
        end;
      except
        on EconvertError do
          Raise EConvertError.Create(S + ' некорректный Comp');
      end else begin
        Posn := 1;
        Result := 0;
        while Posn <= Length(S) do
          case S[Posn] of
              ',':  Inc(Posn);
           '0'..'9':  begin
                      Result := Result * 10 + Ord(S[Posn]) - $30;
```

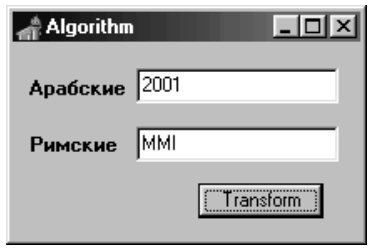
```
                Inc(Posn);
              end;
          else
            Raise EConvertError.Create(S +' некорректный Comp');
        end;
    end;
end;

end.
```

[News Group]

Преобразование арабских чисел в римские

Как преобразовать арабские числа в римские?

Функция получает в качестве параметра любую десятичную величину и возвращает результат в виде строки, содержащей римские цифры.




```
function TForm1.DecToRoman(Decimal: Integer): String;
const
 Romans: array[1..13] of String = ('I', 'IV', 'V', 'IX', 'X', 'XL', 'L',
                                   'XC', 'C', 'CD', 'D', 'CM', 'M');
 Arabics: array[1..13] of Integer = (1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500, 900, 1000);
var
  i: Integer;
begin
  result := '';
  for i := 13 downto 1 do
    while (Decimal >= Arabics[i]) do begin
      Decimal := Decimal - Arabics[i];
      result := result + Romans[i];
    end;
end;
```

Преобразование в EBCDIC

Как перекодировать строку?

Функция конвертирует любую строку. Можете доработать ее, для того чтобы она могла преобразовывать другие типы данных. Но если вам нужны

дополнительные преобразования и обработка данных, то стоит задуматься о приобретении специализированного программного обеспечения...

```
const
  a2e: array [0..255] of byte =
  (000, 001, 002, 003, 055, 045, 046, 047, 022, 005, 037, 011, 012, 013, 014, 159,
   016, 017, 018, 019, 182, 181, 050, 038, 024, 025, 063, 039, 028, 029, 030, 031,
   064, 090, 127, 123, 091, 108, 080, 125, 077, 093, 092, 078, 107, 096, 075, 097,
   240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 122, 094, 076, 126, 110, 111,
   124, 193, 194, 195, 196, 197, 198, 199, 200, 201, 209, 210, 211, 212, 213, 214,
   215, 216, 217, 226, 227, 228, 229, 230, 231, 232, 233, 173, 224, 189, 095, 109,
   121, 129, 130, 131, 132, 133, 134, 135, 136, 137, 145, 146, 147, 148, 149, 150,
   151, 152, 153, 162, 163, 164, 165, 166, 167, 168, 169, 192, 106, 208, 161, 007,
   104, 220, 081, 066, 067, 068, 071, 072, 082, 083, 084, 087, 086, 088, 099, 103,
   113, 156, 158, 203, 204, 205, 219, 221, 224, 236, 252, 176, 177, 178, 062, 180,
   069, 085, 206, 222, 073, 105, 154, 155, 171, 015, 186, 184, 183, 170, 138, 139,
   060, 061, 098, 079, 100, 101, 102, 032, 033, 034, 112, 035, 114, 115, 116, 190,
   118, 119, 120, 128, 036, 021, 140, 141, 142, 065, 006, 023, 040, 041, 157, 042,
   043, 044, 009, 010, 172, 074, 174, 175, 027, 048, 049, 250, 026, 051, 052, 053,
   054, 089, 008, 056, 188, 057, 160, 191, 202, 058, 254, 059, 004, 207, 218, 020,
   225, 143, 070, 117, 253, 235, 238, 237, 144, 239, 179, 251, 185, 234, 187, 255);

procedure StringA2E(var StringToConvert: String);
var
  Loop: integer;
begin
  for Loop := 1 to Length(StringToConvert) do
    StringToConvert[Loop] := Char(a2e[Ord(StringToConvert[Loop])]);
end;
```

[News Group]

Добавление лидирующих символов

Как в начало строки вставить символ? Количество вставляемых символов может быть различным.

Если необходимо в начало строки вставить определенный символ, например, преобразовать «1010» в «0001010», воспользуйтесь следующей функцией:

```
function PadL(s_InStr: string; i_Wide: integer; c_Chr: char): string;
begin
  while Length(s_InStr) < i_Wide do s_InStr := c_Chr + s_InStr;
  Result := s_InStr;
end;
```

[Good_mag]

Преобразование BMP в ICO

Как файл формата BMP преобразовать в формат пиктограмм – ICO?

Нужно создать две маски типа `Bitmap` – маску `AND` и маску `XOR`. Можно также передать дескрипторы масок `Handle` в функцию `CreateIconIndirect()` и использовать их в приложении:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  IconSizeX: integer;
  IconSizeY: integer;
  AndMask: TBitmap;
  XorMask: TBitmap;
  IconInfo: TIconInfo;
  Icon: TIcon;

begin
{ Получаем размеры пиктограммы }
  IconSizeX := GetSystemMetrics(SM_CXICON);
  IconSizeY := GetSystemMetrics(SM_CYICON);
{ создаем маску "And" }
  AndMask := TBitmap.Create;
  AndMask.Monochrome := true;
  AndMask.Width := IconSizeX;
  AndMask.Height := IconSizeY;
{ рисуем на маске "And" }
  AndMask.Canvas.Brush.Color := clWhite;
  AndMask.Canvas.FillRect(Rect(0, 0, IconSizeX, IconSizeY));
  AndMask.Canvas.Brush.Color := clBlack;
  AndMask.Canvas.Ellipse(4, 4, IconSizeX - 4, IconSizeY - 4);
{ Рисуем для проверки }
  Form1.Canvas.Draw(IconSizeX * 2, IconSizeY, AndMask);
{ Создаем маску "Xor" }
  XorMask := TBitmap.Create;
  XorMask.Width := IconSizeX;
  XorMask.Height := IconSizeY;
{ Рисуем на маске "Xor" }
  XorMask.Canvas.Brush.Color := ClBlack;
  XorMask.Canvas.FillRect(Rect(0, 0, IconSizeX, IconSizeY));
  XorMask.Canvas.Pen.Color := clRed;
  XorMask.Canvas.Brush.Color := clRed;
  XorMask.Canvas.Ellipse(4, 4, IconSizeX - 4, IconSizeY - 4);
{ Рисуем для проверки }
  Form1.Canvas.Draw(IconSizeX * 4, IconSizeY, XorMask);
{ Создаем пиктограмму }
  Icon := TIcon.Create;
  IconInfo.fIcon := true;
  IconInfo.xHotspot := 0;
  IconInfo.yHotspot := 0;
  IconInfo.hbmMask := AndMask.Handle;
```

```
  IconInfo.hbmColor := XorMask.Handle;
  Icon.Handle := CreateIconIndirect(IconInfo);
{ Удаляем временные bitmap }
  AndMask.Free;
  XorMask.Free;
{ Рисуем для проверки }
  Form1.Canvas.Draw(IconSizeX * 6, IconSizeY, Icon);
{ Присваиваем пиктограмму приложению }
  Application.Icon := Icon;
{ Заставляем перерисоваться }
  InvalidateRect(Application.Handle, nil, true);
{ Освобождаем пиктограмму }
  Icon.Free;
end;
```

[Vozny Alexander]

Преобразование ICO в BMP

Нужно преобразовать пиктограмму (ICO) в формат BMP. Как это выполнить с помощью Delphi?

Решение 1

Воспользуйтесь небольшой процедурой:

```
procedure ConvertIcoToBmp(Icon: TIcon; FilePath: string);
var
  Bitmap: TBitmap;

begin
  Bitmap := TBitmap.Create;
  Icon.LoadFromFile(FilePath + '.ICO');
  Bitmap.Width := Icon.Width;
  Bitmap.Height := Icon.Height;
  Bitmap.Canvas.Draw(0, 0, Icon);
  Bitmap.SaveToFile(FilePath + '.BMP');
  Bitmap.Free;
end;
```

Решение 2

Чтобы преобразовать `Icon` в `Bitmap`, воспользуйтесь классом `TImageList`. Для обратного преобразования замените метод `AddIcon` на `Add` и метод `GetBitmap` – на `GetIcon`.

```
function Icon2Bitmap(Icon: TIcon): TBitmap;
begin
  with TImageList.Create(nil) do begin
    AddIcon (Icon);
    Result := TBitmap.Create;
```

```
        GetBitmap (0, Result);
    Free;
  end;
end;
```

[Astafiev]

Примечание

Во втором решении необходимо определиться с размерами получаемого изображения.

Преобразование BMP в JPEG

Как преобразовать файл формата BMP в формат JPEG?

Image1 – компонент TImage, содержащий растровое изображение. Используйте следующий фрагмент кода для конвертации изображения в файл формата JPEG:

```
procedure ConvertBmpToJpeg(Image1: TImage; FilePath: string);
var
  MyJpeg: TJpegImage;
begin
  MyJpeg := TJpegImage.Create;
{ Чтение изображения из файла }
  Image1.Picture.LoadFromFile(FilePath + '.BMP');
{ Назначение изображения объекту }
  MyJpeg.Assign(Image1.Picture.Bitmap);

{ Сохранение изображения на диске }
  MyJpeg.SaveToFile(FilePath + '.JPG');
  MyJpeg.Free;
end;
```

Примечание

Не забудьте добавить в uses *модуль* jpeg*, в котором описан тип* TjpegImage.

Арифметика времени

Как правильно работать с переменными, использующими время?

Работа с временными величинами в Delphi очень проста, если воспользоваться встроенными функциями преобразования. Определите глобальные переменные Hour, Minute, Second и инициализируйте их следующим образом:

```
Hour := EncodeTime(1, 0, 0, 0);
Minute := EncodeTime(0, 1, 0, 0);
Second := EncodeTime(0, 0, 1, 0);
```

Если вы предпочитаете константы, определите их следующим образом:

```
Hour = 3600000 / MSecsPerDay;
Minute = 60000 / MSecsPerDay;
Second = 1000 / MSecsPerDay;
```

Теперь, для того чтобы добавить 240 минут к переменной TDateTime, просто напишите (не забывайте о типах переменных):

```
T := T + 240 * Minute;
```

Примечание

В первом случае переменные должны иметь тип TTime *(подключите модуль* SysUtils*).*

Арифметика дат

Кто-нибудь пробовал написать функцию, возвращающую для определенной даты день недели?

Delphi содержит в своем арсенале функцию DayOfWeek, возвращающую целочисленный результат в диапазоне от 1 до 7:

```
function DayOfWeekStr(S: TDateTime): string;
begin

  case DayOfWeek(S) of
    1: Result := 'Воскресенье';
    2: Result := 'Понедельник';
    3: Result := 'Вторник';
    4: Result := 'Среда';
    5: Result := 'Четверг';
    6: Result := 'Пятница';
    7: Result := 'Суббота';
  end;

end;
```

[Иванов Андрей]

Номер месяца по его имени

Как получить номер месяца, если известно его название?

Воспользуйтесь циклом обхода элементов глобального массива LongMonthNames:

```
function GetMonthNumber(Month: String): Integer;
begin
  for Result := 1 to 12 do
```

```
      if Month = LongMonthNames[Result] then Exit;
  Result := 0;
end;
```

Примечание

Функция `GetMonthNumber` *возвращает номера месяцев от 1 до 12. Название месяца задается в формате текущей локализации Windows, например «Август», «Мау». При неправильном названии месяца возвращается значение 0.*

Получение элемента даты

Как из даты выделить нужный элемент?

Используйте универсальную функцию возврата значения элемента даты (год, месяц, день, квартал):

```
function RetDate(inDate: TDateTime; inTip: integer): integer;
var
  xYear, xMonth, xDay: word;
begin
  Result := 0;
  DecodeDate(inDate, xYear, xMonth, xDay);

  case inTip of
    1:  Result := xYear;                  // год
    2:  Result := xMonth;                 // месяц
    3:  Result := xDay;                   // день
    4:  if xMonth < 4 then Result := 1    // квартал
        else if xMonth < 7 then Result := 2
        else if xMonth < 10 then Result := 3
        else Result := 4;
  end;
end;
```

[Галимарзанов Фанис]

Год четырьмя цифрами

Необходимо отображать год четырьмя цифрами. Как это можно сделать?

Данный код преобразует дату из короткого представления `DD/MM/YY` в формат длинной даты `DD/MM/YYYY`.

Если `DD/MM/YY` – входное поле, а `DD/MM/YYYY` – поле базы данных, то приведенный выше код можно использовать при задании другого формата даты, с его соответствующим переопределением в панели управления.

```
LongDate := FormatDateTime('dd/mm/yyyy', StrToDate(ShortDate));
```

Преобразование даты в количество секунд

Мне нужно сравнить даты с точностью до секунд. Как это можно сделать?

Функция EncodeDate возвращает объект TDateTime, который определен как double. Это позволяет работать с датами и временем как с обычными числами – складывать, вычитать и т. д. При этом дата хранится в целой части числа, а время в дробной. Берем две даты (TDateTime), находим разность, а полученный результат преобразуем с помощью DateTimeToStr или чего-то подобного. Если же отбросить целую часть разницы двух чисел, то получим разницу в пределах одних суток. Все зависит от вашей фантазии...

Вычисление даты Пасхи

Как определить дату Пасхи?

Решение 1

Воспользуйтесь функцией CalcEaster. Далее приводится два варианта ее применения.

```
function CalcEaster(Year: Word): String;
var
  B, D, E, Q: Integer;
  GF: String;
begin
  B := 225 - 11 * (Year Mod 19);
  D := ((B - 21) Mod 30) + 21;
  if D > 48 then Dec(D);
  E := (Year + (Year Div 4) + D + 1) Mod 7;
  Q := D + 7 - E;
  if Q < 32 then begin
    if ShortDateFormat[1] = 'd' then
      Result := IntToStr(Q) + '/3/' + IntToStr(Year)
    else
      Result := '3/' + IntToStr(Q) + '/' + IntToStr(Year);
  end
  else begin
    if ShortDateFormat[1] = 'd' then
      Result := IntToStr(Q - 31) + '/4/' + IntToStr(Year)
    else
      Result := '4/' + IntToStr(Q - 31) + '/' + IntToStr(Year);
  end;
{ вычисление страстной пятницы }
  if Q < 32 then begin
    if ShortDateFormat[1] = 'd' then
      GF := IntToStr(Q - 2) + '/3/' + IntToStr(Year)
    else
      GF := '3/' + IntToStr(Q - 2) + '/' + IntToStr(Year);
  end
```

```
  else begin
    if ShortDateFormat[1]= 'd' then
      GF := IntToStr(Q - 31 - 2) + '/4/' + IntToStr(Year)
    else
      GF := '4/' + IntToStr(Q - 31 - 2) + '/' + IntToStr(Year);
  end;
end;
```

Решение 2

```
function Easter(Year: Integer): TDateTime;
{ Вычисляет и возвращает день Пасхи определенного года.
  Скорректировано для предотвращения переполнения целых, если передан год >= 6554}
var
  nMonth, nDay, nMoon, nEpact, nSunday, nGold, nCent, nCorx, nCorz: Integer;
begin
{ Номер Золотого Года в 19-летнем метоническом цикле: }
  nGold := (Year mod 19) + 1;
{ Вычисляем столетие: }
  nCent := (Year div 100) + 1;
{ Количество лет, в течение которых отслеживаются високосные года...
  для синхронизации с движением солнца: }
  nCorx := (3 * nCent) div 4 - 12;
{ Специальная коррекция для синхронизации Пасхи с орбитой луны: }
  nCorz := (8 * nCent + 5) div 25 - 5;
{ Находим воскресенье: }
  nSunday := (Longint(5) * Year) div 4 - nCorx - 10;
{ Предохраняем переполнение года за отметку 6554.
  Устанавливаем Epact - определяем момент полной луны: }
  nEpact := (11 * nGold + 20 + nCorz - nCorx) mod 30;
  if nEpact < 0 then nEpact := nEpact + 30;
  if ((nEpact = 25) and (nGold > 11)) or (nEpact = 24) then nEpact := nEpact + 1;
{ Ищем полную луну: }
  nMoon := 44 - nEpact;
  if nMoon < 21 then nMoon := nMoon + 30;
{ Позиционируем на воскресенье: }
  nMoon := nMoon + 7 - ((nSunday + nMoon) mod 7);
  if nMoon > 31 then begin
    nMonth := 4;
    nDay := nMoon - 31;
  end
  else begin
    nMonth := 3;
    nDay := nMoon;
  end;
  Easter := EncodeDate(Year, nMonth, nDay);
end;
```

Использование DateTime в DBGrid

При отображении `TDateTimeField` *в* `DBGrid` *с форматированием hh:mm (для показа только времени), любая попытка изменения времени приводит (при передаче данных) к ошибке примерно такого содержания: «07:00 is not a valid DateTime». Как переслать данные в виде: «Trunc(oldDateTimevalue) + StrToTime(displaytext)»?*

Следующий обработчик событий – `TDateTimeField.OnSetText` – не слишком элегантен, но он работает.

Предположим, что имеется маска редактирования, допускающая формат `hh:mm` или `hh:mm:ss`. Тогда процедура будет иметь следующий вид:

```
procedure TForm1.Table1Date1SetText(Sender: TField; const Text: String);
var
  d: TDateTime;
  t: string;
begin
  t := Text;
  with Sender as TDateTimeField do begin
    if IsNull then d := SysUtils.Date
    else d := AsDateTime;
    AsDateTime := StrToDateTime(Copy(DateToStr(d), 1, 10) + ' ' + t);
  end;
end;
```

Примечание

Функция `Copy` *как раз и формирует постоянную дату (в формате* `dd/mm/yyyy`*), которая автоматически вводится в поле,* `t` *– вводимое время.*

[News Group]

Вычисление восхода и захода солнца и луны

Воспользуйтесь готовым проектом, состоящим из трех модулей.

Модуль `sunproject.dpr`:

```
program sunproject;

uses
  Forms,
  main in 'main.pas' {Sun};

{$R *.RES}
begin
  Application.Initialize;
  Application.Title := 'Sun';
  Application.CreateForm(TSun, Sun);
  Application.Run;
end.
```

Модуль main.dfm:

```
object Sun: TSun
  Left = 210
  Top = 106
  BorderIcons = [biSystemMenu, biMinimize]
  BorderStyle = bsSingle
  Caption = 'Sun'
  ClientHeight = 257
  ClientWidth = 299
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  Position = poDesktopCenter
  OnCreate = CreateForm
  PixelsPerInch = 96
  TextHeight = 13
  object GroupBoxInput: TGroupBox
    Left = 4
    Top = 4
    Width = 173
    Height = 93
    Caption = ' Ввод '
    TabOrder = 0
    object LabelLongitude: TLabel
      Left = 35
      Top = 44
      Width = 78
      Height = 13
      Alignment = taRightJustify
      Caption = 'Долгота (град):'
    end
    object LabelTimeZone: TLabel
      Left = 13
      Top = 68
      Width = 100
      Height = 13
      Alignment = taRightJustify
      Caption = 'Часовая зона (час):'
    end
    object LabelAtitude: TLabel
      Left = 40
      Top = 20
      Width = 73
      Height = 13
      Alignment = taRightJustify
      Caption = 'Широта (град):'
    end
```

```
    object EditB5: TEdit
      Tag = 1
      Left = 120
      Top = 16
      Width = 37
      Height = 21
      TabOrder = 0
      Text = '0'
    end
    object EditL5: TEdit
      Tag = 2
      Left = 120
      Top = 40
      Width = 37
      Height = 21
      TabOrder = 1
      Text = '0'
    end
    object EditH: TEdit
      Tag = 3
      Left = 120
      Top = 64
      Width = 37
      Height = 21
      TabOrder = 2
      Text = '0'
    end
  end
  object GroupBoxCalendar: TGroupBox
    Left = 184
    Top = 4
    Width = 109
      Height = 93
      Caption = ' Календарь '
      TabOrder = 1
      object LabelD: TLabel
        Left = 19
        Top = 20
        Width = 30
        Height = 13
        Alignment = taRightJustify
        Caption = 'День:'
      end
      object LabelM: TLabel
        Left = 13
        Top = 44
        Width = 36
        Height = 13
        Alignment = taRightJustify
        Caption = 'Месяц:'
      end
```

```
    object LabelY: TLabel
      Left = 28
      Top = 68
      Width = 21
      Height = 13
      Alignment = taRightJustify
      Caption = 'Год:'
    end
    object EditD: TEdit
      Tag = 1
      Left = 56
      Top = 16
      Width = 37
      Height = 21
      TabOrder = 0
      Text = '0'
    end
    object EditM: TEdit
      Tag = 2
      Left = 56
      Top = 40
      Width = 37
      Height = 21
      TabOrder = 1
      Text = '0'
    end
    object EditY: TEdit
      Tag = 3
      Left = 56
      Top = 64
      Width = 37
      Height = 21
      TabOrder = 2
      Text = '0'
    end
  end
  object ButtonCalc: TButton
    Left = 12
    Top = 227
    Width = 169
    Height = 25
    Caption = '&Вычислить'
    TabOrder = 2
    OnClick = ButtonCalcClick
  end
  object ListBox: TListBox
    Left = 4
    Top = 104
    Width = 289
    Height = 117
    ItemHeight = 13
```

```
      TabOrder = 3
    end
    object ButtonClear: TButton
      Left = 192
      Top = 227
      Width = 91
      Height = 25
      Caption = '&Очистить'
      TabOrder = 4
      OnClick = ButtonClearClick
    end
  end
end
```

Модуль main.pas:

```
unit main;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type
  TSun = class(TForm)
    GroupBoxInput: TGroupBox;
    LabelLongitude: TLabel;
    EditB5: TEdit;
    EditL5: TEdit;
    LabelTimeZone: TLabel;
    EditH: TEdit;
    GroupBoxCalendar: TGroupBox;
    LabelD: TLabel;
    LabelM: TLabel;
    LabelY: TLabel;
    EditD: TEdit;
    EditM: TEdit;
    EditY: TEdit;
    ButtonCalc: TButton;
    ListBox: TListBox;
    ButtonClear: TButton;
    LabelAtitude: TLabel;
    procedure Calendar;                      // Календарь
    procedure GetTimeZone;                   // Получение часового пояса
    procedure PosOfSun;                      // Получаем положение солнца
    procedure OutInform;                     // Процедура вывода информации
    procedure PossibleEvents(Hour: integer); // Возможные события
                                             // на полученный час
    procedure GetDate;                       // Получить значения даты
    procedure GetInput;                      // Получить значения широты,...
    procedure ButtonCalcClick(Sender: TObject);
```

```
    procedure CreateForm(Sender: TObject);
    procedure ButtonClearClick(Sender: TObject);
  private
    function Sgn(Value: Double): integer;    // Сигнум
  public
end;

var
  Sun: TSun;
  st: string;
  aA, aD: array [1 .. 2] of double;
  B5, H, M, Y, J3, H3, M3, C0: integer;
  D, L5, Z, Z0, Z1, A5, D5, R5, T, T0, TT, T3, L0, L2: double;
  H0, H1, H2, H7, N7, D7, M8, W8, A, B, A0, D0, A2, D1, D2, DA, DD: double;
  E, F, J, S, C, P, L, G, V, U, W, V0, V1, V2, AZ: double;

const
  P2 = Pi * 2;                      // 2 * Pi
  DR = Pi / 180;                    // Радиан на градус
  K1 = 15 * DR * 1.0027379;

implementation

{$R *.DFM}

uses
  SysUtils;

function TSun.Sgn(Value: Double): integer;
begin
  if Value = 0 then } Result := 0;
  if Value > 0 then Result := 1;
  if Value < 0 then Result := -1;
end;

procedure TSun.Calendar;
begin
  G := 1;
  if Y < 1583 then G := 0;
  D1 := Trunc(D);
  F := D - D1 - 0.5;
  J := -Trunc(7 * (Trunc((M + 9) / 12) + Y) / 4);
  if G = 1 then begin
    S := Sgn(M - 9);
    A := Abs(M - 9);
    J3 := Trunc(Y + S * Trunc(A / 7));
    J3 := -Trunc((Trunc(J3 / 100) + 1) * 3 / 4);
  end;
  J := J + Trunc(275 * M / 9) + D1 + G * J3;
  J := J + 1721027 + 2 * G + 367 * Y;
  if F >= 0 then Exit;
```

```
  F := F + 1;
  J := J - 1;
end;

procedure TSun.GetTimeZone;
begin
  T0 := T / 36525;
  S := 24110.5 + 8640184.813 * T0;
  S := S + 86636.6 * Z0 + 86400 * L5;
  S := S / 86400;
  S := S - Trunc(S);
  T0 := S * 360 * DR;
end;

procedure TSun.PosOfSun;
begin
// Фундаментальные константы (Van Flandern & Pulkkinen, 1979)
  L := 0.779072 + 0.00273790931 * T;
  G := 0.993126 + 0.0027377785 * T;
  L := L - Trunc(L);
  G := G - Trunc(G);
  L := L * P2;
  G := G * P2;
  V := 0.39785 * Sin(L);
  V := V - 0.01000 * Sin(L - G);
  V := V + 0.00333 * Sin(L + G);
  V := V - 0.00021 * TT * Sin(L);
  U := 1 - 0.03349 * Cos(G);
  U := U - 0.00014 * Cos(2 * L);
  U := U + 0.00008 * Cos(L);
  W := -0.00010 - 0.04129 * Sin(2 * L);
  W := W + 0.03211 * Sin(G);
  W := W + 0.00104 * Sin(2 * L - G);
  W := W - 0.00035 * Sin(2 * L + G);
  W := W - 0.00008 * TT * Sin(G);
// Вычисление солнечных координат
  S := W / Sqrt(U - V * V);
  A5 := L + ArcTan(S / Sqrt(1 - S * S));
  S := V / Sqrt(U);
  D5 := ArcTan(S / Sqrt(1 - S * S));
  R5 := 1.00021 * Sqrt(U);
end;

procedure TSun.PossibleEvents(Hour: integer);
var
  num: string;
begin
  st := '';
  L0 := T0 + Hour * K1;
  L2 := L0 + K1;
  H0 := L0 - A0;
```

```
  H2 := L2 - A2;
  H1 := (H2 + H0) / 2;                        // Часовой угол,
  D1 := (D2 + D0) / 2;                        // наклон в получасе
  if Hour <= 0 then V0 := S * Sin(D0) + C * Cos(D0) * Cos(H0) - Z;
  V2 := S * Sin(D2) + C * Cos(D2) * Cos(H2) - Z;
  if Sgn(V0) = Sgn(V2) then Exit;
  V1 := S * Sin(D1) + C * Cos(D1) * Cos(H1) - Z;
  A:= 2 * V2 - 4 * V1 + 2 * V0;
  B := 4 * V1 - 3 * V0 - V2;
  D := B * B - 4 * A * V0;
  if D < 0 then Exit;
  D := Sqrt(D);
  if (V0 < 0) and (V2 > 0) then st := st + 'Восход солнца в ';
  if (V0 < 0) and (V2 > 0) then M8 := 1;
  if (V0 > 0) and (V2 < 0) then st := st + 'Заход солнца в ';
  if (V0 > 0) and (V2 < 0) then W8 := 1;
  E := (-B + D) / (2 * A);
  if (E > 1) or (E < 0) then E := (-B - D) / (2 * A);
  T3 := Hour + E + 1 / 120;                   // Округление
  H3 := Trunc(T3);
  M3 := Trunc((T3 - H3) * 60);
  Str(H3:2, num);
  st := st + num + ':';
  Str(M3:2, num);
  st := st + num;
  H7 := H0 + E * (H2 - H0);
  N7 := -Cos(D1) * Sin(H7);
  D7 := C * Sin(D1) - S * Cos(D1) * COS(H7);
  AZ := ArcTan(N7 / D7) / DR;
  if (D7 < 0) then AZ := AZ + 180;
  if (AZ < 0) then AZ := AZ + 360;
  if (AZ > 360) then AZ := AZ - 360;
  Str(AZ:4:1, num);
  st := st + ', азимут ' + num;
end;

procedure TSun.OutInform;
begin
  if (M8 = 0) and (W8 = 0) then begin
    if V2 < 0 then ListBox.Items.Add('Солнце заходит весь день ');
    if V2 > 0 then ListBox.Items.Add('Солнце восходит весь день ');
  end else begin
    if M8 = 0 then ListBox.Items.Add('В этот день солнце не восходит ');
    if W8 = 0 then ListBox.Items.Add('В этот день солнце не заходит ');
  end;
end;

procedure TSun.GetDate;
begin
  D := StrToInt(EditD.text);
  M := StrToInt(EditM.text);
```

```
  Y := StrToInt(EditY.text);
end;

procedure TSun.GetInput;
begin
  B5 := StrToInt(EditB5.Text);
  L5 := -StrToInt(EditL5.Text);
  H := StrToInt(EditH.Text);
end;

procedure TSun.ButtonCalcClick(Sender: TObject);
var
  C0: integer;
begin
  GetDate;
  GetInput;
  ListBox.Items.Add('Широта: ' + EditB5.Text + ' Долгота: '
        + EditL5.Text + ' Зона: ' + EditH.Text + ' Дата: '
        + EditD.Text + '/' + EditM.Text + '/' + EditY.Text);
  L5 := L5 / 360;
  Z0 := H / 24;
  Calendar;
  T := (J - 2451545) + F;
  TT := T / 36525 + 1;                        // TT - столетия, начиная с 1900.0
  GetTimeZone;                                // Получение часового пояса
  T := T + Z0;
  PosOfSun;                                   // Получаем положение солнца
  aA[1] := A5;
  aD[1] := D5;
  T := T + 1;
  PosOfSun;
  aA[2] := A5;
  aD[2] := D5;
  if aA[2] < aA[1] then aA[2] := aA[2] + P2;
  Z1 := DR * 90.833;                          // Вычисление зенита
  S := Sin(B5 * DR);
  C := Cos(B5 * DR);
  Z := Cos(Z1);
  M8 := 0;
  W8 := 0;
  A0 := aA[1];
  D0 := aD[1];
  DA := aA[2] - aA[1];
  DD := aD[2] - aD[1];
  for C0 := 0 to 23 do begin
    P := (C0 + 1) / 24;
    A2 := aA[1] + P * DA;
    D2 := aD[1] + P * DD;
    PossibleEvents(C0);
    if st <> '' then ListBox.Items.Add(st);
```

```
    A0 := A2;
    D0 := D2;
    V0 := V2;
  end;

  OutInform;
  ListBox.Items.Add('');                    // Разделяем данные
end;

procedure TSun.CreateForm(Sender: TObject);
begin
  EditD.Text := FormatDateTime('d', Date);
  EditM.Text := FormatDateTime('m', Date);
  EditY.Text := FormatDateTime('yyyy', Date);
end;

procedure TSun.ButtonClearClick(Sender: TObject);
begin
  ListBox.Clear;
end;

end.
```

[Ермолаев Александр]

Вычисление расстояния при известных широте и долготе

Как вычислять расстояния на географической карте?

Попробуйте использовать следующий код.

Входные данные:

- `StartLat` (начальная широта) – градусы и сотые доли
- `StartLong` (начальная долгота) – градусы и сотые доли
- `EndLat` (конечная широта) – градусы и сотые доли
- `EndLong` (конечная долгота) – градусы и сотые доли

Выходные данные:

- `Distance` (расстояние) – расстояние в метрах
- `Bearing` (смещение) – смещение в градусах

Не забудьте включить модуль `Math` в список используемых модулей.

```
const
// Константы, используемые для вычисления смещения и расстояния
  D2R: Double = 0.017453;          // Константа для преобразования градусов в радианы
  R2D: Double = 57.295781;         // Константа для преобразования радиан в градусы
  a: Double = 6378137.0;           // Основные полуоси
  b: Double = 6356752.314245;      // Не основные полуоси
```

```
  e2: Double = 0.006739496742337; // Квадрат эксцентричности эллипсоида
  f: Double = 0.003352810664747;  // Выравнивание эллипсоида
var
// Передаваемые широта/долгота в градусах и сотых долях
  StartLat: double;       // Начальная широта
  StartLong: double;      // Начальная долгота
  EndLat: double;         // Конечная широта
  EndLong: double;        // Конечная долгота
// Переменные, используемые для вычисления смещения и расстояния
  fPhimean: Double;       // Средняя широта
  fdLambda: Double;       // Разница между двумя значениями долготы
  fdPhi: Double;          // Разница между двумя значениями широты
  fAlpha: Double;         // Смещение
  fRho: Double;           // Меридианский радиус кривизны
  fNu: Double;            // Поперечный радиус кривизны
  fR: Double;             // Радиус сферы Земли
  fz: Double;             // Угловое расстояние от центра сфероида
  fTemp: Double;          // Временная переменная, используемая в вычислениях
  Distance: Double;       // Вычисленное расстояния в метрах
  Bearing: Double;        // Вычисленное от и до смещение
begin
// Вычисляем разницу между двумя долготами и широтами и получаем среднюю широту
  fdLambda := (StartLong - EndLong) * D2R;
  fdPhi := (StartLat - EndLat) * D2R;
  fPhimean := ((StartLat + EndLat) / 2.0) * D2R;
// Вычисляем меридианные и поперечные радиусы кривизны средней широты
  fTemp := 1 - e2 * (Power(Sin(fPhimean), 2));
  fRho := (a * (1 - e2)) / Power(fTemp, 1.5);
  fNu := a / (Sqrt(1 - e2 * (Sin(fPhimean) * Sin(fPhimean))));
// Вычисляем угловое расстояние
  fz := Sqrt(Power(Sin(fdPhi/2.0), 2) + Cos(EndLat * D2R) *
  Cos(StartLat * D2R) * Power(Sin(fdLambda/2.0), 2));
  fz := 2 * ArcSin(fz);
// Вычисляем смещение
  fAlpha := Cos(EndLat * D2R) * Sin(fdLambda) * 1 / Sin(fz);
  fAlpha := ArcSin(fAlpha);
// Вычисляем радиус Земли
  fR := (fRho * fNu) / ((fRho * Power(Sin(fAlpha), 2))
        + (fNu * Power(Cos(fAlpha), 2)));
// Получаем смещение и расстояние
  Distance := (fz * fR);
  if ((StartLat < EndLat) and (StartLong < EndLong)) then
    Bearing := Abs(fAlpha * R2D)
  else if ((StartLat < EndLat) and (StartLong > EndLong)) then
    Bearing := 360 - Abs(fAlpha * R2D)
  else if ((StartLat > EndLat) and (StartLong > EndLong)) then
    Bearing := 180 + Abs(fAlpha * R2D)
  else if ((StartLat > EndLat) and (StartLong < EndLong)) then
    Bearing := 180 - Abs(fAlpha * R2D);
end;
```

Рисование кривых Безье

Существует ли исходный код или какая-либо информация для рисования кривых Безье?

Решение 1

Для рисования кривой Безье разделяем интервал между P1 и P2 на несколько отрезков (их количество влияет на точность воспроизведения кривой, 3 – 4 точки вполне достаточно), затем в цикле создаем массив точек, используем описанную выше процедуру с параметром t от 0 до 1 и рисуем данный массив точек при помощи функции PolyLine().

```
type
  PBezierPoint = ^TBezierPoint;
  TBezierPoint = record
    X, Y: double;       // основной узел
    X1, Y1: double;     // левая контрольная точка
    Xr, Yr: double;     // правая контрольная точка
  end;

// P1 и P2 - две точки TBezierPoint, расположенные между 0 и 1:
// когда t = 0 X = P1.X, Y = P1.Y; когда t = 1 X = P2.X, Y = P2.Y;
procedure BezierValue(P1, P2: TBezierPoint; t: double; var X, Y: double);
var
  t_sq, t_cb, r1, r2, r3, r4: double;
begin
  t_sq := t * t;
  t_cb := t * t_sq;
  r1 := (1 - 3 * t + 3 * t_sq - t_cb) * P1.X;
  r2 := (3 * t - 6 * t_sq + 3 * t_cb) * P1.Xr;
  r3 := (3 * t_sq - 3 * t_cb) * P2.X1;
  r4 := (t_cb) * P2.X;
  X := r1 + r2 + r3 + r4;
  r1 := (1 - 3 * t + 3 * t_sq - t_cb) * P1.Y;
  r2 := (3 * t - 6 * t_sq + 3 * t_cb) * P1.Yr;
  r3 := (3 * t_sq - 3 * t_cb) * P2.Y1;
  r4 := (t_cb) * P2.Y;
  Y := r1 + r2 + r3 + r4;
end;
```

[Streblechenko Dmitry]

Решение 2

```
procedure Bezier(A: PlotArray; MaxContrPoints: integer; var B: PlotArray;
                 MaxIntPoints: integer);
const
  MaxControlPoints = 25;

type
  CombiArray = array[0..MaxControlPoints] of Float;
```

```
var
  N: integer; ContrPoint, IntPoint: integer;
  T, SumX, SumY, Prod, DeltaT, Quot: Float;
  Combi: CombiArray;
begin
  MaxContrPoints := MaxContrPoints - 1;
  DeltaT := 1.0 / (MaxIntPoints - 1);
  Combi[0] := 1; Combi[MaxContrPoints] := 1;
  for N := 0 to MaxContrPoints - 2 do
    Combi[N + 1] := Combi[N] * (MaxContrPoints - N) / (N + 1);
    for IntPoint := 1 to MaxIntPoints do begin
      T := (IntPoint - 1) * DeltaT;
      if T <= 0.5 then begin
        Prod := 1.0 - T; Quot := Prod;
        for N := 1 to MaxContrPoints - 1 do
          Prod := Prod * Quot;
        Quot := T / Quot;
        SumX := A[MaxContrPoints + 1, 1];
        SumY := A[MaxContrPoints + 1, 2];
        for N := MaxContrPoints downto 1 do begin
          SumX := Combi[N - 1] * A[N, 1] + Quot * SumX;
          SumY := Combi[N - 1] * A[N, 2] + Quot * SumY;
        end;
      end else begin
        Prod := T; Quot := Prod;
        for N := 1 to MaxContrPoints - 1 do
          Prod := Prod * Quot;
        Quot := (1 - T) / Quot;
        SumX := A[1, 1];
        SumY := A[1, 2];
        for N := 1 to MaxContrPoints do begin
          SumX := Combi[N] * A[N + 1, 1] + Quot * SumX;
          SumY := Combi[N] * A[N + 1, 2] + Quot * SumY;
         end;
      end;
      B[IntPoint, 1] := SumX * Prod;
      B[IntPoint, 2] := SumY * Prod;
    end;
end;
```

[News Group]

Управление битами

Как получить доступ к битам переменной и управлять их значением?

Решение 1

```
unit Bitwise;
interface
```

```
function IsBitSet(const val: longint; const TheBit: byte): boolean;
function BitOn(const val: longint; const TheBit: byte): LongInt;
function BitOff(const val: longint; const TheBit: byte): LongInt;
function BitToggle(const val: longint; const TheBit: byte): LongInt;

implementation

function IsBitSet(const val: longint; const TheBit: byte): boolean;
begin
  result := (val and (1 shl TheBit)) <> 0;
end;

function BitOn(const val: longint; const TheBit: byte): LongInt;
begin
  result := val or (1 shl TheBit);
end;

function BitOff(const val: longint; const TheBit: byte): LongInt;
begin
  result := val and ((1 shl TheBit) xor $FFFFFFFF);
end;

function BitToggle(const val: longint; const TheBit: byte): LongInt;
begin
  result := val xor (1 shl TheBit);
end;

end.
```

Решение 2

SetWord – слово, которое необходимо установить. BitNum – номер бита, который необходимо выставить согласно определениям в секции const (Bit0, Bit1 и т. д.). GetBitStat возвращает значение True, если бит установлен и, False – в противном случае.

```
const
  Bit0 = 1;
  Bit1 = 2;
  Bit2 = 4;
  Bit3 = 8;
  Bit4 = 16;
  Bit5 = 32;
  Bit6 = 64;
  Bit7 = 128;

  Bit8 = 256;
  Bit9 = 512;
  Bit10 = 1024;
  Bit11 = 2048;
  Bit12 = 4096;
  Bit13 = 8192;
```

```
  Bit14 = 16384;
  Bit15 = 32768;

procedure SetBit(SetWord, BitNum: Word);
begin
  SetWord := SetWord Or BitNum;       { Устанавливаем бит }
end;

procedure ClearBit(SetWord, BitNum: Word);
begin
  SetWord := SetWord Or BitNum;       { Устанавливаем бит }
  SetWord := SetWord Xor BitNum;      { Переключаем бит }
end;

procedure ToggleBit(SetWord, BitNum: Word);
begin
  SetWord := SetWord Xor BitNum;      { Переключаем бит }
end;

function GetBitStat(SetWord, BitNum: Word): Boolean;
begin
  GetBitStat := SetWord and BitNum = BitNum; { Если бит установлен }
end;
```

Гауссово размывание

Ядро Гауссовой функции `exp(-(x^2 + y^2))` есть разновидность формулы `f(x)*g(y)`, которая означает, что мы можем выполнить двумерную свертку, создавая последовательность одномерных сверток, – сначала мы свертываем каждую строчку изображения, затем – каждую колонку. Хороший повод для ускорения (`N^2` становится `N*2`). Любая свертка требует некоторого места для временного хранения результатов. Ниже в коде программа `BlurRow` как раз распределяет и освобождает память для каждой колонки. Вероятно, это должно ускорить обработку изображения, правда, не ясно насколько.

Поле `size` в записи `TKernel` ограничено значением 200. Если вам нужен еще больший радиус, воспользуйтесь значениями `radius = 3, 5` или другими. Для большого количества данных методы свертки на практике оказываются эффективнее преобразований Фурье (об этом свидетельствуют многочисленные опыты).

Еще один комментарий все же необходим: гауссово размывание имеет одно магическое свойство, а именно – вы можете сначала размыть каждую строчку (применить фильтр), затем каждую колонку. Такая процедура занимает значительно меньше времени, чем двумерная свертка.

Во всяком случае, можете сделать так:

```
unit GBlur2;

interface
```

```
uses
  Windows, Graphics;

type
  PRGBTriple = ^TRGBTriple;
  TRGBTriple = packed record
               b: byte;         // легче для использования, чем тип RGBTBlue...
               g: byte;
               r: byte;
  end;

PRow = ^TRow;
TRow = array[0..1000000] of TRGBTriple;

PPRows = ^TPRows;
TPRows = array[0..1000000] of PRow;

const MaxKernelSize = 100;

type
  TKernelSize = 1..MaxKernelSize;

TKernel = record
  Size: TKernelSize;
  Weights: array[-MaxKernelSize..MaxKernelSize] of single;
end;

// идея заключается в том, что при использовании TKernel мы игнорируем
// Weights (вес), за исключением Weights в диапазоне -Size..Size.
procedure GBlur(theBitmap: TBitmap; radius: double);

implementation

uses
  SysUtils;

// Создаем K (Гауссово зерно) со среднеквадратичным отклонением = radius.
// Для текущего приложения устанавливаем переменные MaxData = 255,
// DataGranularity = 1. Теперь в процедуре установим значение K.Size так,
// что при использовании K мы будем игнорировать Weights (вес)
// с наименее возможными значениями. (Малый размер нам на пользу,
// поскольку время выполнения напрямую зависит от значения K.Size.)
procedure MakeGaussianKernel(var K: TKernel; radius: double;
                             MaxData, DataGranularity: double);
var
  j: integer;
  temp, delta: double;
  KernelSize: TKernelSize;
begin
  for j := Low(K.Weights) to High(K.Weights) do begin
    temp := j/radius;
```

```
    K.Weights[j] := exp(-temp * temp / 2);
  end;

// делаем так, чтобы sum(Weights) = 1:
  temp := 0;
  for j := Low(K.Weights) to High(K.Weights) do
    temp:= temp + K.Weights[j];
  for j := Low(K.Weights) to High(K.Weights) do
    K.Weights[j] := K.Weights[j] / temp;

// теперь отбрасываем (или делаем отметку "игнорировать" для переменной Size)
// данные, имеющие относительно небольшое значение – это важно, в противном случае
// смазывание происходит с малым радиусом и в той области, которая "захватывается"
// большим радиусом...
  KernelSize := MaxKernelSize;
  delta := DataGranularity/(2 * MaxData);
  temp := 0;
  while (temp < delta) and (KernelSize > 1) do begin
    temp := temp + 2 * K.Weights[KernelSize];
    dec(KernelSize);
  end;
  K.Size := KernelSize;

// теперь для обеспечения корректности возвращаемого результата проводим ту же
// операцию с K.Size, так чтобы сумма всех данных была равна единице:
  temp := 0;
  for j := -K.Size to K.Size do
    temp := temp + K.Weights[j];
  for j := -K.Size to K.Size do
    K.Weights[j] := K.Weights[j] / temp;
end;

function TrimInt(Lower, Upper, theInteger: integer): integer;
begin
  if (theInteger <= Upper) and (theInteger >= Lower) then result := theInteger
  else if theInteger > Upper then result := Upper
  else result := Lower;
end;

function TrimReal(Lower, Upper: integer; x: double): integer;
begin
  if (x < upper) and (x >= lower) then result := trunc(x)
  else if x > Upper then result := Upper
  else result := Lower;
end;

procedure BlurRow(var theRow: array of TRGBTriple; K: TKernel; P: PRow);
var
  j, n, LocalRow: integer;
  tr, tg, tb: double;                // tempRed и др.
  w: double;
```

```
begin
  for j := 0 to High(theRow) do begin
    tb := 0;
    tg := 0;
    tr := 0;

    for n  := -K.Size to K.Size do begin
      w := K.Weights[n];
// TrimInt задает отступ от края строки...
      with theRow[TrimInt(0, High(theRow), j - n)] do begin
        tb := tb + w * b;
        tg := tg + w * g;
        tr := tr + w * r;
      end;
    end;
    with P[j] do begin
      b := TrimReal(0, 255, tb);
      g := TrimReal(0, 255, tg);
      r := TrimReal(0, 255, tr);
    end;
  end;
  Move(P[0], theRow[0], (High(theRow) + 1) * Sizeof(TRGBTriple));
end;

procedure GBlur(theBitmap: TBitmap; radius: double);
var
  Row, Col: integer;
  theRows: PPRows;
  K: TKernel;
  ACol: PRow;
  P: PRow;
begin
  if (theBitmap.HandleType <> bmDIB) or (theBitmap.PixelFormat <> pf24Bit)
  then raise exception.Create('GBlur может работать только
                               с 24-битными изображениями');
  MakeGaussianKernel(K, radius, 255, 1);
  GetMem(theRows, theBitmap.Height * SizeOf(PRow));
  GetMem(ACol, theBitmap.Height * SizeOf(TRGBTriple));

// запись позиции данных изображения:
  for Row := 0 to theBitmap.Height - 1 do
    theRows[Row] := theBitmap.Scanline[Row];

// размываем каждую строчку:
  P := AllocMem(theBitmap.Width * SizeOf(TRGBTriple));
  for Row := 0 to theBitmap.Height - 1 do
    BlurRow(Slice(theRows[Row]^, theBitmap.Width), K, P);

// теперь размываем каждую колонку
  ReAllocMem(P, theBitmap.Height * SizeOf(TRGBTriple));
  for Col := 0 to theBitmap.Width - 1 do begin
```

```
// считываем первую колонку в TRow:
    for Row := 0 to theBitmap.Height - 1 do ACol[Row] := theRows[Row][Col];
    BlurRow(Slice(ACol^, theBitmap.Height), K, P);

// теперь помещаем обработанный столбец на свое место в данные изображения:
    for Row := 0 to theBitmap.Height - 1 do theRows[Row][Col] := ACol[Row];
  end;
  FreeMem(theRows);
  FreeMem(ACol);
  ReAllocMem(P, 0);
end;

end.
```

Пример использования

```
procedure TForm1.Button1Click(Sender: TObject);
var
  b: TBitmap;
begin
  if not openDialog1.Execute then exit;
  b := TBitmap.Create;
  b.LoadFromFile(OpenDialog1.Filename);
  b.PixelFormat:= pf24Bit;
  Canvas.Draw(0, 0, b);
  GBlur(b, StrToFloat(Edit1.text));
  Canvas.Draw(b.Width, 0, b);
  b.Free;
end;
```

Имейте в виду, что 24-битные изображения при системной палитре из 256 цветов требуют некоторых дополнительных хитростей, т. к. иначе эти изображения не только теряют некоторый объем информации, но и серьезно нарушают работу фильтра.

Рисование фрактальных графов

Предлагаемое решение (исходный код для Turbo Pascal 7).

```
program Fractal;

uses
  graph, crt;

const
  GrafType = 1; {1..3}

type
  PointPtr = ^Point;
  Point = Record
```

```
    X, Y: Word;
    Angle: Real;
    Next: PointPtr
  end;

  GrfLine = array [0..5000] of Byte;
  ChangeType = array [1..30] of Record
    Mean: Char;
    NewString: String;
  end;

var
  K, T, Dx, Dy, StepLength, GrafLength: Word;
  grDriver, Xt: Integer;
  grMode, ErrCode: Integer;
  CurPosition: Point;
  Descript: GrfLine;
  StartLine: String Absolute Descript;
  ChangeNumber, Generation: Byte;
  Changes: ChangeType;
  AngleStep: Real;
  Mem: Pointer;

procedure Replace(var Stroka: GrfLine; OldChar: Char; Repl: String);
var
  I, J: Word;
begin
  if (GrafLength = 0) Or (Length(Repl) = 0) then Exit;
  I := 1;
  while I <= GrafLength do begin
    if Chr (Stroka[I]) = OldChar then begin
      for J := GrafLength downto I + 1 do
        Stroka[J + Length(Repl) - 1] := Stroka[J];
      for J := 1 to Length(Repl) do
        Stroka[I + J - 1] := Ord(Repl[J]);
      I := I + J;
      GrafLength := GrafLength + Length(Repl) - 1;
      continue;
    end;
    I := I + 1;
  end;
end;

procedure PushCoord(var Ptr: PointPtr; C: Point);
var
  P: PointPtr;
begin
  New(P);
  P^.X := C.X;
  P^.Y := C.Y;
```

```
  P^.Angle := C.Angle;
  P^.Next := Ptr;
  Ptr := P;
end;

procedure PopCoord(var Ptr: PointPtr; var Res: Point);
begin
  if Ptr <> Nil then begin
    Res.X := Ptr^.X;
    Res.Y := Ptr^.Y;
    Res.Angle := Ptr^.Angle;
    Ptr := Ptr^.Next;
  end;
end;

procedure FindGrafCoord(var Dx, Dy: Word; Angle: Real; StepLength: Word);
begin
  Dx := Round(Sin (Angle) * StepLength * GetMaxX / GetMaxY);
  Dy := Round( - Cos (Angle) * StepLength);
end;

procedure NewAngle(Way: ShortInt; var Angle: Real; AngleStep: Real);
begin
  if Way >= 0 then Angle := Angle + AngleStep
  else Angle := Angle - AngleStep;
  if Angle >= 4 * Pi then Angle := Angle - 4 * Pi;
  if Angle < 0 then Angle := 4 * Pi + Angle;
end;

procedure Rost(var Descr: GrfLine; Cn: Byte; Ch: ChangeType);
var
  I: Byte;
begin
  for I := 1 to Cn do Replace(Descr, Ch[I].Mean, Ch[I].NewString);
end;

procedure Init1;
begin
  AngleStep := Pi / 8;
  StepLength := 7;
  Generation := 4;
  ChangeNumber := 1;
  CurPosition.Next := Nil;
  StartLine := 'F';
  GrafLength := Length(StartLine);
  with Changes [1] do begin
    Mean := 'F';
    NewString := 'FF+[+F-F-F]-[-F+F+F]';
  end;
end;
```

```
procedure Init2;
begin
  AngleStep := Pi / 4;
  StepLength := 3;
  Generation := 5;
  ChangeNumber := 2;
  CurPosition.Next := Nil;
  StartLine := 'G';
  GrafLength := Length (StartLine);
  with Changes [1] do begin
    Mean := 'G';
    NewString := 'GFX[+G][-G]';
  end;
  with Changes [2] do begin
    Mean := 'X';
    NewString := 'X[-FFF][+FFF]FX';
  end;
end;

procedure Init3;
begin
  AngleStep := Pi / 10;
  StepLength := 9;
  Generation := 5;
  ChangeNumber := 5;
  CurPosition.Next := Nil;
  StartLine := 'SLFF';
  GrafLength := Length (StartLine);
  with Changes [1] do begin
    Mean := 'S';
    NewString := '[+++G][---G]TS';
  end;
  with Changes [2] do begin
    Mean := 'G';
    NewString := '+H[-G]L';
  end;
  with Changes [3] do begin
    Mean := 'H';
    NewString := '-G[+H]L';
  end;
  with Changes [4] do begin
    Mean := 'T';
    NewString := 'TL';
  end;
  with Changes [5] do begin
    Mean := 'L';
    NewString := '[-FFF][+FFF]F';
  end;
end;
begin
  case GrafType of
```

```
    1:  Init1;
    2:  Init2;
    3:  Init3;
  end;
  grDriver := detect;
  InitGraph(grDriver, grMode, '');
  ErrCode := GraphResult;
  if ErrCode <> grOk then begin
    WriteLn('Graphics error:', GraphErrorMsg(ErrCode));
    Halt(1)
  end;
  with CurPosition do begin
    X := GetMaxX Div 2;
    Y := GetMaxY;
    Angle := 0;
    MoveTo(X, Y)
  end;
  SetColor(white);
  for K := 1 To Generation do begin
    Rost(Descript, ChangeNumber, Changes);
    Mark(Mem);
    for T := 1 To GrafLength do begin
      case Chr(Descript[T]) of
          'F':  begin
                  FindGrafCoord (Dx, Dy, CurPosition.Angle, StepLength);
                  with CurPosition do begin
                    Xt := X + Dx;
                    if Xt < 0 then X := 0
                    else X := Xt;
                    if X > GetMaxX then X := GetMaxX;
                    Xt := Y + Dy;
                    if Xt < 0 then Y := 0
                    else Y := Xt;
                    if Y > GetMaxY then Y := GetMaxY;
                    LineTo(X, Y)
                  end;
                end;
          'f':  begin
                  FindGrafCoord (Dx, Dy, CurPosition.Angle, StepLength);
                  with CurPosition do begin
                    Xt := X + Dx;
                    if Xt < 0 then X := 0 else X := Xt;
                    if X > GetMaxX then X := GetMaxX;
                    Xt := Y + Dy;
                    if Xt < 0 then Y := 0 else Y := Xt;
                    if Y > GetMaxY then Y := GetMaxY;
                    MoveTo(X, Y)
                  end;
                end;
          '+':  NewAngle(1, CurPosition.Angle, AngleStep);
          '-':  NewAngle( - 1, CurPosition.Angle, AngleStep);
```

```
          'I':  NewAngle(1, CurPosition.Angle, 2 * Pi);
          '[':  PushCoord (CurPosition.Next, CurPosition);
          ']':  begin
                  PopCoord(CurPosition.Next, CurPosition);
                  with CurPosition do MoveTo(X, Y);
                end;
      end;
    end;
    Dispose(Mem);
    Delay(1000);
  end;
  Repeat
  Until KeyPressed;
  CloseGraph;
end.
```

[Марковский Михаил]

Вращение изображения

С помощью предлагаемого программного кода реализуется быстрый и примитивный способ вращения изображения. По крайней мере, это тоже выход из положения, поскольку Windows этого делать не умеет.

```
procedure RotateRight(BitMap: TImage);
var
  FirstC, LastC, c, r: integer;

  procedure FixPixels(c, r: integer);
  var
    SavePix, SavePix2: tColor;
    i, NewC, NewR: integer;
  begin
    SavePix := Bitmap.Canvas.Pixels[c, r];
    for i := 1 to 4 do begin
      Newc := BitMap.Height - r + 1;
      Newr := c;
      SavePix2 := BitMap.Canvas.Pixels[Newc, Newr];
      Bitmap.Canvas.Pixels[Newc, Newr] := SavePix;
      SavePix := SavePix2;
      c := NewC;
      r := NewR;
    end;
  end;

begin
  if BitMap.Width <> BitMap.Height then exit;
  BitMap.Visible := False;
  with Bitmap.Canvas do begin
    FirstC := 0;
    LastC := BitMap.Width;
```

```
    for r := 0 to BitMap.Height div 2 do begin
      for c := FirstC to LastC do FixPixels(c, r);
      Inc(FirstC);
      Dec(LastC);
    end;
  end;
  BitMap.Visible := True;
end;
```

[News Group]

Примечание

Вращение происходит на 90 градусов вправо за одно выполнение процедуры. Не забудьте добавить компонент `TImage` *на форму, загрузить изображение и передать* `TImage` *в качестве параметра в процедуру вращения.*

64-битное кодирование/декодирование

Реализация алгоритма декодирования base64.

```
const
  Base64Table =
    'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/';

function Base64Decode(cStr: string): string;
var
  ResStr: string;
  DecStr: string;
  RecodeLine: array [1..76] of byte;
  f1, f2: word;
  l: integer;
begin
  l := length(cStr);
  ResStr := '';
  for f1 := 1 to l do
    if cStr[f1]='=' then RecodeLine[f1] := 0
    else RecodeLine[f1] := pos(cStr[f1], Base64Table) - 1;
  f1 := 1;
  while f1 < length(cStr) do begin
    DecStr := chr(byte(RecodeLine[f1] shl 2) + RecodeLine[f1 + 1] shr 4)
        + chr(byte(RecodeLine[f1 + 1] shl 4) + RecodeLine[f1 + 2] shr 2)
        + chr(byte(RecodeLine[f1 + 2] shl 6) + RecodeLine[f1 + 3]);
    ResStr := ResStr + DecStr;
    inc(f1, 4);
  end;
  Base64Decode := ResStr;
end;
```

[Варавва Алексей]

Защита программ перекрытием кода

Не секрет, что совершенной защиты не существует. Тем не менее, хорошая защита должна обеспечить такой уровень, чтобы на ее вскрытие нужно было затратить усилия, сравнимые с самостоятельным написанием программы. Разумеется, она должна быть многоуровневой и перекрывающейся (уровни должны работать независимо). Не забывайте, что хорошие взломщики неплохо знают Ассемблер, и высокоуровневые ухищрения от них не спасают. Следовательно, для построения высококлассной защиты с применением Ассемблера необходимо владеть последним в совершенстве. Не думайте, что вам это не подходит, т. к. слишком сложно или уже не модно. Хороший программист не пренебрегает Ассемблером и высшей математикой.

Один из методов – это перекрывающийся код. Он может показаться немного сложным для большинства из нас, но, зная несколько HEX-кодов инструкций процессора, вы тоже сможете создать небольшой по размеру перекрывающийся код. Перекрывающийся код можно сделать сколь угодно многоуровневым, а здесь я покажу лишь, в каком направлении надо «копать».

```
    temp_string := 'Den is Com';
    asm
        mov ax, $05EB
@as:  jmp @as-2
    end;
    ShowMessage('Сообщение');
```

На первый взгляд, это может озадачить, но на самом деле все очень просто. Первая инструкция заносит значение в AX. Вторая выполняет переход на значение операнда команды MOV. Код '05EB' переводится как 'JMP $+5' (помните, что слова хранятся в обратном порядке). Этот переход минует JMP и передает выполнение дальше. Вероятно, этого не будет достаточно для защиты, но технику ее создания демонстрирует.

Присваивание temp_string := 'Den is Com' существенной роли не играет, но может применяться при отладке программы, т. к. хорошо просматривается при использовании дизассемблера и отладчика. Возможно, ваши первые попытки будут приводить к частому зависанию компьютера, но не отчаивайтесь – защита того стоит. Попробуйте разработать свой способ сравнения строк (чаще всего ловятся именно эти инструкции), замаскируйте инструкции зависания компьютера и т. д.

[Den is Com]

Генерация случайного пароля

Необходимо, чтобы приложение само создавало пароли.

Возможно, данный способ пригодится. Совместимость: Delphi 5 и выше.

Внимание

Длина пароля должна быть меньше длины таблицы!

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Randomize;  // запускаем генератор случайных чисел
end;

function RandomPwd(PWLen: integer): string;
// таблица символов, используемых в пароле
const
  StrTable: string = '!#$%&/()=?@<>|{[]}\*~+#;:.-' +
                     + '_ABCDEFGHIJKLMabcdefghijklm'+
                     + '0123456789ДЦЬдцьЯNOPQRSTUVWXYZnopqrstuvwxyz';
var
  N, K, X, Y: integer;
begin
// проверяем максимальную длину пароля
  if (PWlen > Length(StrTable)) then K := Length(StrTable)- 1
  else K := PWLen;
  SetLength(result, K);              // устанавливаем длину конечной строки
  Y := Length(StrTable);             // Длина Таблицы для внутреннего цикла
  N := 0;                            // начальное значение цикла
  while N < K do begin               // цикл для создания K символов
    X := Random(Y) + 1;              // берем следующий случайный символ
// проверяем присутствие этого символа в конечной строке
    if Pos(StrTable[X], result) = 0 then begin
      inc(N);                        // символ не найден
      Result[N] := StrTable[X];      // теперь его сохраняем
    end;
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  cPwd: string;
begin
  // вызываем функцию генерации пароля из 30 символов
  cPwd := RandomPwd(30);
  // ...
end;
```

[Nikolaev Igor]

Как закодировать строку

Мне надо закодировать информацию. Как это можно сделать?

Приведенная программа демонстрирует методы кодирования и раскодирования строк:

Примечание

Мы не отвечаем за уникальность и секретность алгоритма данной функции.

```
program Crypt;

{$APPTYPE CONSOLE}

const
  C1 = 52845;
  C2 = 22719;

function Encrypt(const S: String; Key: Word): String;
var
  I: byte;
begin
  SetLength(Result, Length(S));
  for I := 1 to Length(S) do begin
    Result[I] := char(byte(S[I]) xor (Key shr 8));
    Key := (byte(Result[I]) + Key) * C1 + C2;
  end;
end;

function Decrypt(const S: String; Key: Word): String;
var
  I: byte;
begin
  SetLength(Result, Length(S));
  for I := 1 to Length(S) do begin
    Result[I] := char(byte(S[I]) xor (Key shr 8));
    Key := (byte(S[I]) + Key) * C1 + C2;
  end;
end;

var
  S: string;
begin
  Write('>');
  ReadLn(S);
  S := Encrypt(S, 12345);
  WriteLn(S);
  S := Decrypt(S, 12345);
  WriteLn(S);
  Readln;
end.
```

Как стереть самого себя

При работе происходит блокировка исполняемого файла программы на диске до момента завершения работы программы. Предлагаемый код позволяет программе стереть саму себя с диска. При этом, если программа уже загрузилась в память, то ее работа может продолжаться. Метод действия: создается временный BAT-файл во временной директории на диске, который удаляет и программу, и себя. Применяя этот код для защиты, желательно использовать хотя бы простейшую шифровку текстовых строк XOR, т. к. они

хорошо просматриваются дизассемблером, и, разумеется, хакеру не составит труда обнаружить защиту.

Все файлы проекта:

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Модуль Unit1.dfm:

```
object Form1: TForm1
  Left = 192
  Top = 107
  Width = 435
  Height = 300
  Caption = 'Form1'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Button1: TButton
    Left = 48
    Top = 200
    Width = 313
    Height = 49
    Caption = 'Del me !'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -32
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentFont = False
    TabOrder = 0
    OnClick = Button1Click
  end
  object Memo1: TMemo
```

```
    Left = 48
    Top = 16
    Width = 305
    Height = 169
    TabOrder = 1
  end
end
```

Модуль Unit1.pas:

```
unit Unit1;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls,
  SysUtils;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  Filename: String;
  aTempBuf: array[0..MAX_PATH] of char;
  bat_file: String;
  f: TextFile;
  si: TStartUpInfo;
  pi: TProcessInformation;
begin
  FileName := Application.ExeName;
  if GetShortPathName(PChar(FileName), aTempBuf, SizeOf(aTempBuf) - 1) > 0
  then FileName := StrPas(aTempBuf);
  GetEnvironmentVariable('TEMP', aTempBuf, MAX_PATH);
  bat_file := StrPas(aTempBuf) + '\' + 'delself.bat';
  if GetShortPathName(PChar(bat_file), aTempBuf, SizeOf(aTempBuf) - 1) > 0
  then bat_file := StrPas(aTempBuf);
  AssignFile(f, bat_file);
  rewrite(f);
  writeln(f, '@echo off');
```

```
    writeln(f, ':try');
    writeln(f, 'del ' + FileName);
    write(f,'if exist ' + FileName);
    writeln(f, ' goto try');
    write(f, 'del ' + bat_file);
    CloseFile(f);
    Memo1.Lines.LoadFromFile(bat_file);
    ZeroMemory(@si, SizeOf(si));
    si.cb := SizeOf(si);
    si.wShowWindow := SW_HIDE;
    si.dwFlags := STARTF_USESHOWWINDOW;
    CreateProcess(nil, PChar(bat_file), nil, nil, False, IDLE_PRIORITY_CLASS
                  or DETACHED_PROCESS, nil, nil, si, pi);
end;

end.
```

[Denis Com]

Пример защиты типа SHAREWARE

В качестве примера приведен небольшой участок программного кода, позволяющий быстро создать защиту для программ SHAREWARE, которая не влияет на функциональность самой программы, но настоятельно «просит» ее зарегистрировать и закрывает при каждом повторном запуске.

Технология данного метода заключается в том, что пользователь может запустить программу только один раз за текущий сеанс Windows.

Используйте обработчик события FormShow:

```
procedure TForm1.FormShow(Sender : TObject);
var
  atom: integer;
  CRLF: string;
begin
  if GlobalFindAtom('THIS_IS_SOME_OBSCUREE_TEXT') = 0 then
    atom := GlobalAddAtom('THIS_IS_SOME_OBSCUREE_TEXT')
  else begin
    CRLF := #10 + #13;
    ShowMessage('Данная версия предусматривает только один запуск'
              + 'в текущем сеансе Windows.' + CRLF
              + 'Для повторного запуска необходимо перезапустить Windows, или,'
              + CRLF + 'что лучше, - ' + CRLF + 'ЗАРЕГИСТРИРУЙТЕСЬ !');
    Close;
  end;
end;
```

Преимущество данного метода в том, что пользователю доступны все возможности программы, но только до момента ее закрытия или перезапуска системы. Вся хитрость заключается в сохранении некоторой строки в сис-

темных глобальных переменных («атомах») и последующей проверке ее в таблице «атомов» системы.

Перекодировка текста из DOS в Windows и наоборот

Как с помощью Delphi перекодировать текстовый файл из DOS в Windows и наоборот?

Решение 1

Str – текст для перекодировки.

```
procedure TForm1.WinToDos;
var
  Str: PChar;
begin
  Str := Memo1.Lines.GetText;            // Берем текст из TMemo как PChar
  CharToOem(Str, Str);                   // Перекодировка текста
  Memo2.Lines.SetText(Str);              // Передаем перекодированный текст
end;

procedure TForm1.DosToWin;               // С точностью до вызова функции API
var                                      // повторяем код
  Str: PChar;
begin
  Str := Memo1.Lines.GetText;
  OemToChar(Str, Str);
  Memo2.Lines.SetText(Str);
end;
```

Рекомендация: применяя функции CharToOem и OemToChar в качестве источника текста и приемника перекодированного текста, следует использовать одну и ту же переменную, иначе при перекодировке больших текстов может возникнуть ошибка типа "Access violation".

[Виталий Еремеев]

Решение 2

Используйте CharToOEM, OEMToChar, CharToOEMBuff, OEMToCharBuff.

[Nomadic]

Чтение и запись файлов UNIX

Мне нужно обрабатывать информацию, созданную в UNIX. Как эту информацию перекодировать для работы в Windows?

Данный модуль позволяет читать и записывать файлы формата UNIX.

```
unit StreamFile;

interface
```

```
uses
  SysUtils;

procedure AssignStreamFile(var F: Text; Filename: String);

implementation

const
  BufferSize = 128;

type
  TStreamBuffer = Array [1..High (Integer)] of Char;

  TStreamBufferPointer = ^TStreamBuffer;

  TStreamFileRecord = record
    Case Integer of
      1: (Filehandle: Integer;
         Buffer: TStreamBufferPointer;
         BufferOffset: Integer;
         ReadCount: Integer;);
      2: (Dummy: Array [1..32] of Char)
  end;

function StreamFileOpen(var F: TTextRec): Integer;
var
  Status: Integer;
begin
  with TStreamFileRecord(F.UserData) do begin
    GetMem (Buffer, BufferSize);
    case F.Mode of
      fmInput:
          FileHandle := FileOpen(StrPas(F.Name), fmShareDenyNone);
      fmOutput:
          FileHandle := FileCreate(StrPas(F.Name));
      fmInOut:  begin
          FileHandle := FileOpen(StrPas(F.Name),
                  fmShareDenyNone or fmOpenWrite or fmOpenRead);
          if FileHandle <> -1 then
          { Перемещаемся в конец файла. }
            status := FileSeek (FileHandle, 0, 2);
          F.Mode := fmOutput;
        end;
    end;
    BufferOffset := 0;
    ReadCount := 0;
    F.BufEnd := 0;
    if FileHandle = -1 then Result := -1
    else Result := 0;
  end;
end;
```

```
function StreamFileInOut(var F: TTextRec): Integer;
  procedure Read(var Data: TStreamFileRecord);

    procedure CopyData;
    begin
      while (F.BufEnd < Sizeof(F.Buffer) - 2)
        and (Data.BufferOffset <= Data.ReadCount)
        and (Data.Buffer[Data.BufferOffset] <> #10) do begin
          F.Buffer[F.BufEnd] := Data.Buffer^[Data.BufferOffset];
          Inc(Data.BufferOffset);
          Inc(F.BufEnd);
      end;
      if Data.Buffer[Data.BufferOffset] = #10 then begin
        F.Buffer[F.BufEnd] := #13;
        Inc(F.BufEnd);
        F.Buffer[F.BufEnd] := #10;
        Inc(F.BufEnd);
        Inc(Data.BufferOffset);
      end;
    end;

  begin
    F.BufEnd := 0;
    F.BufPos := 0;
    F.Buffer := '';
    repeat
      if (Data.ReadCount = 0) or (Data.BufferOffset > Data.ReadCount)
      then begin
          Data.BufferOffset := 1;
          Data.ReadCount := FileRead(Data.FileHandle, Data.Buffer^, BufferSize);
      end;
      CopyData;
    until (Data.ReadCount = 0) or (F.BufEnd >= Sizeof(F.Buffer) - 2);
    Result := 0;
  end;
  procedure Write(var Data: TStreamFileRecord);
  var
    Status: Integer;
    Destination: Integer;
    II: Integer;
  begin
    with TStreamFileRecord(F.UserData) do begin
      Destination := 0;
      for II := 0 to F.BufPos - 1 do begin
        if F.Buffer[II] <> #13 then begin
          Inc(Destination);
          Buffer^[Destination] := F.Buffer[II];
        end;
      end;
      Status := FileWrite(FileHandle, Buffer^, Destination);
```

```
      F.BufPos := 0;
      Result := 0;
    end;
  end;

begin
  case F.Mode of
    fmInput:   read(TStreamFileRecord(F.UserData));
    fmOutput:  write(TStreamFileRecord(F.UserData));
  end;
end;

function StreamFileFlush(var F: TTextRec): Integer;
begin
  Result := 0;
end;

function StreamFileClose(var F: TTextRec): Integer;
begin
  with TStreamFileRecord (F.UserData) do begin
    FreeMem(Buffer);
    FileClose(FileHandle);
  end;
  Result := 0;
end;

procedure AssignStreamFile(var F: Text; Filename: String);
begin
  with TTextRec(F) do begin
    Mode := fmClosed;
    BufPtr := @Buffer;
    BufSize := SizeOf(Buffer);
    OpenFunc := @StreamFileOpen;
    InOutFunc := @StreamFileInOut;
    FlushFunc := @StreamFileFlush;
    CloseFunc := @StreamFileClose;
    StrPLCopy(Name, FileName, Sizeof(Name) - 1);
  end;
end;

end.
```

Перенос русского текста по слогам

Как выполнить перенос текста по слогам?

Решение

```
unit Hyper;

interface
```

```
uses
  Windows, Classes, SysUtils;

function SetHyph(pc: PChar; MaxSize: Integer): PChar;
function SetHyphString(s: String): String;
function MayBeHyph(p: PChar; pos: Integer): Boolean;

implementation

type
  TSymbol = (st_Empty, st_NoDefined, st_Glas, st_Sogl, st_Spec);
  TSymbAr = array [0..1000] of TSymbol;
  PSymbAr = ^TSymbAr;

const
  HypSymb = #$1F;
  Spaces = [' ', ',', ';', ':', '.', '?', '!', '/', #10, #13];
  GlasChar = ['Й', 'й', 'У', 'у', 'Е', 'е','Ю', 'ю',
          'А', 'а', 'О', 'о', 'Э', 'э', 'Я', 'я', 'И', 'и',
    { english }
          'e', 'E', 'u', 'U', 'i', 'I', 'o', 'O', 'a', 'A', 'j', 'J'];
  SoglChar = ['Г', 'г', 'Ц', 'ц', 'К', 'к', 'Н', 'н',
          'Ш', 'ш', 'Щ', 'Щ', 'З', 'з', 'Х', 'х',
          'Ф', 'ф', 'В', 'в', 'П', 'п', 'Р', 'р', 'Л', 'л', 'Д', 'д',
          'Ж', 'ж', 'Ч', 'ч', 'С', 'с', 'М', 'м', 'т', 'Т', 'б', 'Б'
    { english }
          'q', 'Q', 'w', 'W', 'r', 'R', 't', 'T', 'y', 'Y',
          'p', 'P', 's', 'S', 'd', 'D', 'f', 'F', 'g', 'G',
          'h', 'H', 'k', 'K', 'l', 'L', 'z', 'Z', 'x', 'X',
          'c', 'C', 'v', 'V', 'b', 'B', 'n', 'N', 'm', 'M'];
  SpecSign = ['Ы', 'ы', 'Ь', 'ь', 'Ъ', 'ъ'];

function isSogl(c: Char): Boolean;
begin
  Result := c in SoglChar;
end;

function isGlas(c: Char): Boolean;
begin
  Result := c in GlasChar;
end;

function isSpecSign(c: Char): Boolean;
begin
  Result := c in SpecSign;
end;

function GetSymbType(c: Char): TSymbol;
begin
  if isSogl(c) then begin
```

```
    Result := st_Sogl;
    exit;
  end;
  if isGlas(c) then begin
    Result := st_Glas;
    exit;
  end;
  if isSpecSign(c) then begin
    Result := st_Spec;
    exit;
  end;
  Result := st_NoDefined;
end;

function isSlogMore(c: pSymbAr; start, len: Integer): Boolean;
var
  i: Integer;
begin
  for I := Start to Len-1 do begin
    if c^[i] = st_NoDefined then begin
      Result := false;
      exit;
    end;
    if (c^[i] = st_Glas)
      and ((c^[i+1] <> st_Nodefined) or (I <> Start)) then begin
        Result := True;
        exit;
    end;
  end;
  Result := false;
end;

{ расставляем переносы }
function SetHyph(pc: PChar; MaxSize: Integer): PChar;
var
  HypBuff: Pointer;
  h: PSymbAr;
  i: Integer;
  len: Integer;
  Cur: Integer;                    { текущая позиция в результирующем массиве}
  cw: Integer;                     { Номер буквы в слове }
  Lock: Integer;                   { счетчик блокировок }
begin
  Cur := 0;
  len := StrLen(pc);
  if (MaxSize=0) or (Len=0) then begin
    Result := nil;
    Exit;
  end;
  GetMem(HypBuff, MaxSize);
  GetMem(h, len + 1);
```

```
{ заполнение массива типов символов }
  for I := 0 to len - 1 do h^[i] := GetSymbType(pc[i]);
{ собственно расстановка переносов }
  cw := 0;
  Lock := 0;
  for I := 0 to Len-1 do begin
    PChar(HypBuff)[cur] := PChar(pc)[i]; Inc(Cur);
    if I >= Len - 2 then Continue;
    if h^[i] = st_NoDefined then begin
      cw := 0;
      Continue;
    end else
      Inc(cw);
    if Lock <> 0 then begin
      Dec(Lock);
      Continue;
    end;
    if cw <= 1 then Continue;
    if not (isSlogMore(h, i + 1, len)) then Continue;
    if (h^[i] = st_Sogl) and (h^[i - 1] = st_Glas)
      and (h^[i + 1] = st_Sogl)
      and (h^[i + 2] <> st_Spec) then begin
        PChar(HypBuff)[cur] := HypSymb;
        Inc(Cur);
        Lock := 1;
    end;
    if (h^[i] = st_Glas) and (h^[i - 1] = st_Sogl)
      and (h^[i + 1] = st_Sogl)
      and (h^[i + 2] = st_Glas) then begin
        PChar(HypBuff)[cur] := HypSymb;
        Inc(Cur);
        Lock := 1;
    end;
    if (h^[i] = st_Glas) and (h^[i - 1] = st_Sogl)
      and (h^[i + 1] = st_Glas)
      and (h^[i + 2] = st_Sogl) then begin
        PChar(HypBuff)[cur] := HypSymb;
        Inc(Cur);
        Lock := 1;
    end;
    if (h^[i] = st_Spec) then begin
      PChar(HypBuff)[cur] := HypSymb;
      Inc(Cur);
      Lock:=1;
    end;

  end;
  FreeMem(h, len + 1);
  PChar(HypBuff)[cur] := #0;
  Result := HypBuff;
end;
```

```
function Red_GlasMore(p: PChar; pos: Integer): Boolean;
begin
  while p[pos] <> #0 do begin
    if p[pos] in Spaces then begin
      Result := False;
      Exit;
    end;
    if isGlas(p[pos]) then begin
      Result := True;
      Exit;
    end;
    Inc(pos);
  end;
  Result := False;
end;

function Red_SlogMore(p: PChar; pos: Integer): Boolean;
var
  BeSogl, BeGlas: Boolean;
begin
  BeSogl := False; BeGlas := False;
  while p[pos] <> #0 do begin
    if p[pos] in Spaces then Break;
    if not BeGlas then BeGlas := isGlas(p[pos]);
    if not BeSogl then BeSogl := isSogl(p[pos]);
    Inc(pos);
  end;
  Result := BeGlas and BeSogl;
end;

function MayBeHyph(p: PChar; pos: Integer): Boolean;
var
  i: Integer;
  len: Integer;
begin
  I := pos;
  Len := StrLen(p);
  Result := (Len > 3) and (i > 2) and (i < Len - 2)
    and (not (p[i] in Spaces))
    and (not (p[i + 1] in Spaces)) and (not (p[i-1] in Spaces))
    and ((isSogl(p[i]) and isGlas(p[i - 1]) and isSogl(p[i + 1])
    and Red_SlogMore(p, i + 1)) or ((isGlas(p[i]))
    and (isSogl(p[i - 1])) and (isSogl(p[i + 1]))
    and (isGlas(p[i + 2]))) or ((isGlas(p[i]))
    and (isSogl(p[i - 1])) and (isGlas(p[i + 1]))
    and Red_SlogMore(p, i + 1)) or ((isSpecSign(p[i])));
end;

function SetHyphString(s: String): String;
var
  Res: PChar;
```

```
begin
  Res := SetHyph(PChar(s), Length(s)*2);
  Result := Res;
  FreeMem(Res, Length(s)*2);
end;

end.
```

[Nomadic]

Примечание

Функция `SetHyphString` *выполняет непосредственный перенос текста по слогам, функция* `MayBeHyph` *– индицирует возможность переноса текста в указанной позиции,* `SetHyph` *– необходима для работы* `SetHyphString` *(отдельно не вызывается).*

Сумма прописью

Очень часто в финансовых приложениях сумму нужно писать прописью. Как сумму, представленную цифрой, преобразовать в строку прописью?

Решение 1

```
function TextSum(S: double): string;

  function Conv999(M: longint; fm: integer): string;
  const
    c1to9m: array [1..9] of string[6] =
      ('один', 'два', 'три', 'четыре', 'пять', 'шесть', 'семь', 'восемь','девять');
    c1to9f: array [1..9] of string[6] =
     ('одна', 'две', 'три', 'четыре', 'пять', 'шесть', 'семь', 'восемь', 'девять');
    c11to19: array [1..9] of string[12] =
       ('одиннадцать', 'двенадцать', 'тринадцать', 'четырнадцать', 'пятнадцать',
        'шестнадцать', 'семнадцать', 'восемнадцать', 'девятнадцать');
    c10to90: array [1..9] of string[11] =
       ('десять', 'двадцать', 'тридцать', 'сорок', 'пятьдесят', 'шестьдесят',
        'семьдесят', 'восемьдесят', 'девяносто');
    c100to900: array [1..9] of string[9] =
       ('сто', 'двести', 'триста', 'четыреста', 'пятьсот',
        'шестьсот', 'семьсот', 'восемьсот', 'девятьсот');

var
  s: String;
  i: Longint;
begin
  s := '';
  i := M div 100;
  if I <> 0 then s := c100to900[i] + ' ';
  M := M mod 100;
  i := M div 10;
  if (M > 10) and (M < 20) then
```

```
    s := s + c11to19[M - 10] + ' '
  else begin
    if I <> 0 then s := s + c10to90[i] + ' ';
    M := M mod 10;
    if M <> 0 then
      if fm = 0 then s := s + c1to9f[M] + ' '
      else s := s + c1to9m[M] + ' ';
  end;
  Conv999 := s;
end;

var
  i: Longint;
  j: Longint;
  r: Real;
  t: String;
begin
  t := '';
  j := Trunc(S / 1000000000.0);
  r := j;
  r := S - r*1000000000.0;
  i := Trunc(r);

  if j <> 0 then begin
    t := t + Conv999(j, 1) + 'миллиард';
    j := j mod 100;
    if (j > 10) and (j < 20) then t := t + 'ов '
    else
      case j mod 10 of
         0:  t := t + 'ов ';
         1:  t := t + ' ';
      2..4:  t := t + 'а ';
      5..9:  t := t + 'ов ';
      end;
  end;
  j := i div 1000000;
  if j <> 0 then begin
    t := t + Conv999(j, 1) + 'миллион';
    j := j mod 100;
    if (j > 10) and (j < 20) then t := t + 'ов '
    else
      case j mod 10 of
         0:  t := t + 'ов ';
         1:  t := t + ' ';
      2..4:  t := t + 'а ';
      5..9:  t := t + 'ов ';
      end;
  end;
  i := i mod 1000000;
  j := i div 1000;
  if j <> 0 then begin
```

```
    t := t + Conv999(j, 0) + 'тысяч';
    j := j mod 100;
    if (j > 10) and (j < 20) then t := t + ' '
    else
      case j mod 10 of
           0:  t := t + ' ';
           1:  t := t + 'а ';
        2..4:  t := t + 'и ';
        5..9:  t := t + ' ';
      end;
  end;
  i := i mod 1000;
  j := i;
  if j <> 0 then t := t + Conv999(j, 1);
  t := t + 'руб. ';
  i := Round(Frac(S)*100.0);
  t := t + IntToStr(i) + ' коп.';
  TextSum := t;
end;
```

[Александр]

Решение 2

```
unit Numinwrd;

interface
  function sMoneyInWords(Nin: currency): string; export;
  function szMoneyInWords(Nin: currency): PChar; export;

implementation

uses
  SysUtils, Dialogs, Math;

type
  tri = string[4];
  mood = 1..2;
  gender = (m, f);
  uns = array[0..9] of string[7];
  tns = array[0..9] of string[13];
  decs = array[0..9] of string[12];
  huns = array[0..9] of string[10];
  nums = array[0..4] of string[8];
  money = array[1..2] of string[5];
  endings = array[gender,mood,1..3] of tri;       // окончания числительных и денег

const
  units: uns = ('', 'один ', 'два ', 'три ', 'четыре ', 'пять ',
                'шесть ', 'семь ', 'восемь ', 'девять ');
  unitsf: uns = ('', 'одна ', 'две ', 'три ', 'четыре ',
                 'пять ', 'шесть ','семь ','восемь ','девять ');
```

```
  teens: tns = ('десять ', 'одиннадцать ', 'двенадцать ', 'тринадцать ',
                'четырнадцать ', 'пятнадцать ', 'шестнадцать ',
                'семнадцать ', 'восемнадцать ', 'девятнадцать ');
  decades: decs = ('', 'десять ', 'двадцать ', 'тридцать ', 'сорок ',
                   'пятьдесят ', 'шестьдесят ', 'семьдесят ',
                   'восемьдесят ', 'девяносто ');
  hundreds: huns=('', 'сто ', 'двести ', 'триста ', 'четыреста ', 'пятьсот ',
                  'шестьсот ', 'семьсот ', 'восемьсот ', 'девятьсот ');
  numericals: nums = ('', 'тысяч', 'миллион', 'миллиард', 'триллион');
  RusMon: money = ('рубл', 'копе');
  ends: endings = ((('', 'а', 'ов'), ('ь', 'я', 'ей')),
                   (('а', 'и', ''), ('йка', 'йки', 'ек')));

threadvar
  str: string;

function EndingIndex(Arg: integer): integer;
begin
  if ((Arg div 10) mod 10) <> 1 then
    case (Arg mod 10) of
       1:  Result := 1;
    2..4:  Result := 2;
    else   Result := 3;
    end
  else Result := 3;
end;

{ Число Nin прописью, как функция }
function sMoneyInWords(Nin: currency): string;

var
  g: gender;                  // род
  Nr: comp;                   // целая часть числа
  Fr: integer;                // дробная часть числа
  i, iTri, Order: longint;    // триада

  procedure Triad;
  var
    iTri2: integer;
    un, de, ce: byte;         // единицы, десятки, сотни

    function GetDigit: byte;
    begin
      Result := iTri2 mod 10;
      iTri2 := iTri2 div 10;
    end;

  begin
    iTri := trunc(Nr/IntPower(1000, i));
    Nr := Nr - int(iTri * IntPower(1000, i));
    iTri2:= iTri;
```

```
    if iTri > 0 then begin
      un := GetDigit;
      de := GetDigit;
      ce := GetDigit;
      if i = 1 then g := f

      else g := m;           // женского рода только тысяча
      str := TrimRight(str) + ' ' + Hundreds[ce];
      if de = 1 then
        str := TrimRight(str) + ' ' + Teens[un]

      else begin
        str := TrimRight(str) + ' ' + Decades[de];
        case g of
          m:  str := TrimRight(str) + ' ' + Units[un];
          f:  str := TrimRight(str) + ' ' + UnitsF[un];
        end;

      end;
      if length(numericals[i]) > 1 then begin
        str := TrimRight(str) + ' ' + numericals[i];
        str := TrimRight(str) + ends[g, 1, EndingIndex(iTri)];
      end;
    end;                  // триада 0 ?
    if I = 0 then Exit;
    Dec(i);
    Triad;
  end;

begin
  str := '';
  Nr := int(Nin);
  Fr := round(Nin*100 + 0.00000001) mod 100;
  if Nr > 0 then
    Order := trunc(Log10(Nr) / 3)
  else begin
    str := 'ноль';
    Order := 0
  end;
  if Order > High(numericals) then
    raise Exception.Create('Слишком большое число для суммы прописью');
  i := Order;
  Triad;
  str := Format('%s %s%s %.2d %s%s', [Trim(str), RusMon[1],
      ends[m, 2, EndingIndex(iTri)], Fr, RusMon[2],
      ends[f, 2, EndingIndex(Fr)]]);
  str[1] := (ANSIUpperCase(copy(str, 1, 1)))[1];

  Result := str + #0;
end;
```

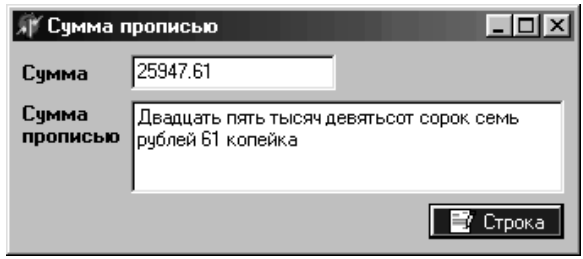
```
function szMoneyInWords(Nin: currency): PChar;
begin
  sMoneyInWords(Nin);
  Result := @(str[1]);
end;

end.
```

[Клюкач Олег]

Примечание

Функция sMoneyInWords *выдает результат типа* string*, а функция* szMoneyInWords *– типа* PChar*.*

Решение 3




```
unit valtostr;

interface

uses
  SysUtils, Math;

function SumStr(Sum: double):
string;

implementation

function SumStr(Sum: double): string;

type
  TSex = (Male, Female);

const
  MaxSum = 1000000000000 - 1;
  Ind: array [1..3, 1..3] of string = (('тысяча', 'тысячи', 'тысяч'),
                                       ('миллион', 'миллиона', 'миллионов'),
                                       ('миллиард', 'миллиарда', 'миллиардов'));

  Curr: array [1..2, 1..3] of string = (('рубль', 'рубля', 'рублей'),
                                        ('копейка', 'копейки', 'копеек'));

function ValToStr(Sum: word; Sex: TSex): string;

const
  f1to9m: array [1..9] of string = ('один', 'два', 'три', 'четыре', 'пять',
                                    'шесть', 'семь', 'восемь', 'девять');

  f1to9f: array [1..9] of string = ('одна', 'две', 'три', 'четыре', 'пять',
                                    'шесть', 'семь', 'восемь', 'девять');
```

```
  f11to19: array [1..9] of string = ('одиннадцать', 'двенадцать', 'тринадцать',
                                     'четырнадцать', 'пятнадцать', 'шестнадцать',
                                     'семнадцать', 'восемнадцать', 'девятнадцать');

  f10To90: array [1..9] of string = ('десять', 'двадцать', 'тридцать', 'сорок',
                                     'пятьдесят', 'шестьдесят', 'семьдесят',
                                     'восемьдесят', 'девяносто');

  f100to900: array [1..9] of string = ('сто', 'двести', 'триста', 'четыреста',
                                       'пятьсот', 'шестьсот', 'семьсот',
                                       'восемьсот', 'девятьсот');
  var
    Val: integer;
  begin
    result := '';
    Val := Sum div 100;
    if Val <> 0 then result := f100to900[Val] + ' ';
    Sum := Sum mod 100;
    Val := Sum div 10;
    if ((Sum > 10) and (Sum < 20)) then
      result := result + f11to19[Sum-10] + ' '
    else begin
      if Val <> 0 then result := result + f10to90[Val] + ' ';
      Sum := Sum mod 10;
      if Sum <> 0 then
        if Sex = Male then result := result + f1to9m[Sum] + ' '
        else result := result + f1to9f[Sum] + ' ';
    end;
  end;

var
  Sym: string;
  Val: double;
  Cnt, LastDidg: byte;
  Rub, Divisor: int64;
  Kop, Dividend: word;

begin
  result := '';
  if Sum < 0 then Exit;
  if Sum > MaxSum then
    raise Exception.Create('Превышение максимально допустимого значения');
  Val := Sum - Trunc(Sum);
  Rub := Trunc(Sum - Val);
  Kop := Round((Sum - Trunc(Sum))*100);
  if Rub = 0 then
    result := 'Ноль рублей '
  else
    for Cnt := 3 downto 0 do begin
      Divisor := Trunc(Power(1000, Cnt));
```

```
      Dividend := Rub div Divisor;
      Rub := Rub - Dividend*Divisor;
      if Dividend <> 0 then begin
        if Cnt = 1 then result := result + ValToStr(Dividend, Female)
        else result := result + ValToStr(Dividend, Male);
        if Cnt > 0 then begin
          LastDidg := Dividend mod 10;
          case LastDidg of
             0:  result := result + Ind[Cnt, 3] + ' ';
             1:  result := result + Ind[Cnt, 1] + ' ';
          2..4:  result := result + Ind[Cnt, 2] + ' ';
          5..9:  result := result + Ind[Cnt, 3] + ' ';
          end;
        end else begin
          LastDidg := Dividend mod 100;
          if ((LastDidg > 4) and (LastDidg < 21)) then
            result := result + Curr[1, 3] + ' '
          else begin
            LastDidg := Dividend mod 10;
            case LastDidg of
               0:  result := result + Curr[1, 3] + ' ';
               1:  result := result + Curr[1, 1] + ' ';
            2..4:  result := result + Curr[1, 2] + ' ';
            5..9:  result := result + Curr[1, 3] + ' ';
            end;
          end;
        end;
      end;
    end;
  if Kop = 0 then
    result := result + '00 копеек'
  else begin
    if Kop < 10 then result := result + '0' + IntToStr(Kop) + ' '
    else result := result + IntToStr(Kop) + ' ';
    if ((Kop > 4) and (Kop < 21)) then
      result := result + Curr[2, 3]
    else begin
      LastDidg := Kop mod 10;
      case LastDidg of
         0:  result := result + Curr[2, 3];
         1:  result := result + Curr[2, 1];
      2..4:  result := result + Curr[2, 2];
      5..9:  result := result + Curr[2, 3];
      end;
    end;
  end;
  Sym := AnsiUpperCase(result[1]);
  result[1] := Sym[1];
end;
end.
```

[VS]

Решение 4

Для англоязычного варианта

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    num: TEdit;
    spell: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    function trans9(num: integer): string;
    function trans19(num: integer): string;
    function trans99(num: integer): string;
    function IntToSpell(num: integer): string;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

function TForm1.IntToSpell(num: integer): string;
var
  spell: string;
  hspell: string;
  hundred: string;
  thousand: string;
  tthousand: string;
  hthousand: string;
  million: string;
begin
  if num < 10 then spell := trans9(num);
  if (num < 20) and (num > 10) then spell := trans19(num);
  if (((num < 100) and (num > 19)) or (num = 10)) then begin
    hspell := copy(IntToStr(num), 1, 1) + '0';
    spell := trans99(StrToInt(hspell));
    hspell := copy(IntToStr(num), 2, 1);
    spell := spell + ' ' + IntToSpell(StrToInt(hspell));
  end;
  if (num < 1000) and (num > 100) then begin
    hspell := copy(IntToStr(num), 1, 1);
```

```
    hundred := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 2, 2);
    hundred := hundred + ' hundred and ' + IntToSpell(StrToInt(hspell));
    spell := hundred;
  end;
  if (num < 10000) and (num > 1000) then begin
    hspell := copy(IntToStr(num), 1, 1);
    thousand := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 2, 3);
    thousand := thousand + ' thousand ' + IntToSpell(StrToInt(hspell));
    spell := thousand;
  end;
  if (num < 100000) and (num > 10000) then begin
    hspell := copy(IntToStr(num), 1, 2);
    tthousand := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 3, 3);
    tthousand := tthousand + ' thousand ' + IntToSpell(StrToInt(hspell));
    spell := tthousand;
  end;
  if (num < 1000000) and (num > 100000) then begin
    hspell := copy(IntToStr(num), 1, 3);
    hthousand := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 4, 3);
    hthousand := hthousand + ' thousand and ' + IntToSpell(StrToInt(hspell));
    spell := hthousand;
  end;
  if (num < 10000000) and (num > 1000000) then begin
    hspell := copy(IntToStr(num), 1, 1);
    million := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 2, 6);
    million := million + ' million and ' + IntToSpell(StrToInt(hspell));
    spell := million;
  end;
  IntToSpell := spell;
end;

function TForm1.trans99(num: integer): string;
var
  spell: string;
begin
  case num of
    10:  spell := 'ten';
    20:  spell := 'twenty';
    30:  spell := 'thirty';
    40:  spell := 'forty';
    50:  spell := 'fifty';
    60:  spell := 'sixty';
    70:  spell := 'seventy';
    80:  spell := 'eighty';
    90:  spell := 'ninety';
  end;
```

```
  trans99 := spell;
end;

function TForm1.trans19(num: integer): string;
var
  spell: string;
begin
  case num of
    11:  spell := 'eleven';
    12:  spell := 'twelve';
    13:  spell := 'thirteen';
    14:  spell := 'fourteen';
    15:  spell := 'fifteen';
    16:  spell := 'sixteen';
    17:  spell := 'seventeen';
    18:  spell := 'eighteen';
    19:  spell := 'nineteen';
  end;
  trans19 := spell;
end;

function TForm1.trans9(num: integer): string;
var
  spell: string;
begin
  case num of
    1:  spell := 'one';
    2:  spell := 'two';
    3:  spell := 'three';
    4:  spell := 'four';
    5:  spell := 'five';
    6:  spell := 'six';
    7:  spell := 'seven';
    8:  spell := 'eight';
    9:  spell := 'nine';
  end;
  trans9 := spell;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  numb: integer;
begin
  spell.text := IntToSpell(StrToInt(num.text));
end;
end.
```

Примечание

В данном варианте программы можно работать только с целыми числами, отсутствует написание денежной единицы.

Проверка кредитной карты

Данный модуль создан на основе алгоритма ccard Питера Миллера (Peter Miller). Автор не против бесплатного использования, но резервирует все права на данный алгоритм.

Примечание

Внесите данный модуль в список uses *любого модуля, которому необходим доступ к функции проверки кредитной карты.*

```
IsValidCreditCardNumber(CardNumber, ReturnMessage) returns Boolean
```

Таким образом можно, например, сообщить пользователю о недействительности карты.

CardNumber – строка, содержащая номер карты, которую необходимо проверить. ReturnMessage – строка, с помощью которой функция может возвратить любое сообщение (при этом старое содержимое строки стирается). Функция возвращает значение True, если номер карточки верен, и False – в противном случае.

Во входных параметрах функции допускаются тире и пробелы, если же встретятся другие символы, их можно удалить. Функция RemoveChar довольно легко реализует данную операцию. Для этого передайте ей входную строку и символ, который необходимо удалить.

Разрешаются любые изменения кода модуля для собственных целей, но в случае его распространения необходимо сообщить другим пользователям обо всех внесенных изменениях.

Вы можете воспользоваться предлагаемым модулем на свой страх и риск, не забывая при этом, что ответственность за какой-либо ущерб, причиненный данным модулем, лежит только на его пользователе.

На момент написания модуля он устойчиво работал под Delphi версий 1 и 2. Для Turbo Pascal необходимо внести некоторые несложные исправления (главным образом из-за различия реализации функций в модуле SysUtils).

```
unit Creditc;

interface
uses
  SysUtils;

function IsValidCreditCardNumber(CardNumber: String; var MessageText: String): Boolean;

implementation

const
  CardPrefixes: array[1..19] of string =
   ('2014', '2149', '300', '301', '302', '303', '304', '305', '34',
    '36', '37', '38', '4', '51', '52', '53', '54', '55', '6011');
```

```
  CardTypes: array[1..19] of String =
   ('enRoute',
    'enRoute',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'American Express',
    'Diner Club/Carte Blanche',
    'American Express',
    'Diner Club/Carte Blanche',
    'Visa',
    'MasterCard',
    'MasterCard',
    'MasterCard',
    'MasterCard',
    'MasterCard',
    'Discover');

function RemoveChar(const Input: String; DeletedChar: Char): String;
{ Данная функция удаляет все вхождения указанного символа из переданной ей строки }
var
  Index: Word;
begin
  Result := Input;
  for Index := Length(Result) downto 1 do
    if Result[Index] = DeletedChar then Delete(Result, Index, 1);
end;

function ShiftMask(Input: Integer): Integer;
{ Простая оболочка для функции сдвига битов числа }
begin
  result := (1 shl (Input - 12));
end;

{ Это, вероятно, самый запутанный код, который вы когда-либо видели.
  Основное, что делает данная функция, - извлекает каждую цифру из номера карты
  для использования в формуле проверки контрольной суммы, устанавливаемой
  компаниями. Алгоритм производит выборку, начиная от последней цифры
  и заканчивая первой. }
function ConfirmChecksum(CardNumber: String): Boolean;
var
  CheckSum: Integer;     // Содержит значение операции
  Flag: Boolean;         // флаг готовности
  Counter: Integer;      // индекс счетчика
  PartNumber: String;    // используется для извлечения каждой цифры числа
  Number: Integer;       // используется для преобразования каждой цифры в число
```

```
begin
{ получаем стартовое значение счетчика }
  Counter := Length(CardNumber);
  CheckSum := 0;
  PartNumber := '';
  Number := 0;
  Flag := false;
  while (Counter >= 1) do begin

{ получаем текущую цифру }
    PartNumber := Copy(CardNumber, Counter, 1);
    Number := StrToInt(PartNumber);
    if (Flag) then begin      { только каждую вторую цифру }
      Number := Number * 2;
      if (Number >= 10) then Number := Number - 9;
    end;
    CheckSum := CheckSum + Number;
    Flag := not(Flag);
    Counter := Counter - 1;
  end;
  result := ((CheckSum mod 10) = 0);
end;

function GetMask(CardName: String): Integer;
begin
  result := 0;                 // значение по умолчанию
  if (CardName = 'MasterCard') then result := ShiftMask(16);
  if (CardName = 'Visa') then result := (ShiftMask(13) or ShiftMask(16));
  if (CardName = 'American Express') then result := ShiftMask(15);
  if (CardName = 'Diner Club/Carte Blanche') then result := ShiftMask(14);
  if (CardName = 'Discover') then result := ShiftMask(16);
end;

function IsValidCreditCardNumber(CardNumber: String; var MessageText: String): Boolean;
var
  StrippedNumber: String;   // используется для хранения числа без
                            // дополнительных символов
  Index: Integer;           // универсальный счетчик для циклов и т.п.
  TheMask: Integer;         // число, которое мы будем использовать для маски
  FoundIt: Boolean;         // используется для индикации, когда что-либо найдено
  CardName: String;         // хранит имя типа карты
  PerformChecksum: Boolean; // тип enRoute карты, если контрольная сумма не сошлась
begin
{ сначала избавимся от пробелов и тире }
  StrippedNumber := RemoveChar(CardNumber, ' ');
  StrippedNumber := RemoveChar(StrippedNumber, '-');
{ если строка была нулевой длины, то тоже OK }
  if (StrippedNumber = '') then begin
    result := true;
    exit;
  end;
```

```
{ инициализация возвращаемых переменных }
  MessageText := '';
  result := true;
{ устанавливаем нашу переменную-флаг }
  FoundIt := false;
{ проверка правильности введенных символов в номере карты }
  for Index := 1 to Length(StrippedNumber) do begin
    case StrippedNumber[Index] of
      '0'..'9': FoundIt := FoundIt;  { другими словами не ор }
       else
         MessageText := 'Неверно введенный символ';
         result := false;
         exit;
    end;
  end;
{ теперь давайте определим тип используемой карты }
  for Index := 1 to 19 do
    if (Pos(CardPrefixes[Index], StrippedNumber) = 1) then begin
      { мы обнаружили правильный тип }
      FoundIt := true;
      CardName := CardTypes[Index];
      TheMask := GetMask(CardName);
    end;
{ если тип карты не определен, указываем на это }
  if not FoundIt then begin
    CardName := 'Unknown Card Type';
    TheMask := 0;
    MessageText := 'Неизвестный тип карты ';
    result := false;
    exit;
  end;
  if ((Length(StrippedNumber) > 28) and result) then begin   { проверка длины }
    MessageText := 'Номер слишком большой ';
    result := false;
    exit;
  end;
{ проверка длины }
  if ((Length(StrippedNumber) < 12) or ((shiftmask(length(strippednumber))
    and themask) = 0)) then begin
      messagetext := 'неверная длина числа';
      result := false;
      exit;
  end;
{ проверяем вычисление контрольной суммы }
  if (cardname = 'enroute') then performchecksum := false
  else performchecksum := true;
  if (performchecksum and (not confirmchecksum(strippednumber))) then begin
      messagetext := 'неверная контрольная сумма';
      result := false;
      exit;
  end;
```

```
{ если результат равен true, тогда все ok }
  if (result) then messagetext := 'номер верен: тип карты: ' + cardname;
{ если строка была нулевой длины, то тоже OK }
  if (strippednumber = '') then result := true;
end;

end.
```

[News Group]

Проверка ISBN

ISBN (или International Standard Book Numbers, стандартные международные номера книг) – мистические кодовые числа, однозначно идентифицирующие книги. Цель этого совета заключается в том, чтобы убрать покров таинственности, окружающий структуру ISBN, и в качестве примера разработать приложение, проверяющее правильность создания кода-кандидата на ISBN.

ISBN имеет длину тринадцать символов, которые могут быть цифрами от 0 до 9, дефисом или буквой «X». Этот код состоит из четырех частей (между которыми стоит дефис): идентификатор группы, идентификатор издателя, идентификатор книги для издателя и контрольная цифра. Первая часть (идентификатор группы) используется для обозначения страны, географического региона, языка и пр. Вторая часть (идентификатор издателя) однозначно идентифицирует издателя. Третья часть (идентификатор книги) однозначно идентифицирует данную книгу среди коллекции книг, выпущенных данным издателем. Четвертая, заключительная часть (контрольная цифра), используется в коде алгоритма другими цифрами для получения поддающегося проверке ISBN. Количество цифр, содержащихся в первых трех частях, может быть различным, но контрольная цифра всегда содержит один символ (расположенный между «0» и «9» включительно, или «X» для величины 10). Таким образом, ISBN имеет длину тринадцать символов (десять чисел плюс три дефиса, разделяющих три части ISBN).

ISBN 3-88053-002-5 можно разложить на части следующим образом:

Группа:	3
Издатель:	88053
Книга:	002
Контрольная цифра:	5

ISBN можно проверить на правильность кода с помощью простого математического алгоритма. Суть его в следующем: нужно взять каждую из девяти цифр первых трех частей ISBN (пропуская дефисы), умножить каждую отдельную цифру на ее позицию в коде ISBN (считая справа от контрольной цифры), сложить эти произведения и прибавить контрольную цифру, после

чего разделить получившееся число на одиннадцать. Если после процедуры деления остатка нет (т. е. число по модулю 11 делится без остатка), кандидат на ISBN является верным кодом ISBN. Например, используем предыдущий образец ISBN 3-88053-002-5:

ISBN:	3 8 8 0 5 3 0 0 2 5
Множитель:	10 9 8 7 6 5 4 3 2 1
Продукт:	30+72+64+00+30+15+00+00+04+05 = 220

Поскольку 220 на одиннадцать делится без остатка, рассмотренный нами кандидат на ISBN является верным кодом ISBN.

Вот пример этой методики, изложенной на Delphi:

```
function IsISBN(ISBN: String): Boolean;
var
  Number, CheckDigit: String;
  CheckValue, CheckSum, Err: Integer;
  i, Cnt: Word;
begin
{ Получаем контрольную цифру }
  CheckDigit := Copy(ISBN, Length(ISBN), 1);
{ Получаем остальную часть, ISBN минус контрольная цифра и дефис }
  Number := Copy(ISBN, 1, Length(ISBN) - 2);
{ Длина разницы ISBN должны быть 11 и контрольная цифра между 0 и 9, или X }
  if (Length(Number) = 11) and (Pos(CheckDigit, '0123456789X') > 0) then begin
    { Получаем числовое значение контрольной цифры }
    if (CheckDigit = 'X') then CheckSum := 10
    else Val(CheckDigit, CheckSum, Err);
{ Извлекаем в цикле все цифры из кода ISBN, применяя алгоритм декодирования }
    Cnt := 1;
    for i := 1 to 11 do begin
{ Действуем, если текущий символ находится между "0" и "9", исключая дефисы }
      if (Pos(Number[i], '0123456789') > 0) then begin
        Val(Number[i], CheckValue, Err);
{ Алгоритм для каждого символа кода ISBN, Cnt - n-й обрабатываемый символ }
        CheckSum := CheckSum + CheckValue * (11 - Cnt);
        Inc(Cnt);
      end;
    end;
{ Проверяем делимость без остатка полученного значения на 11 }
    if (CheckSum mod 11 = 0) then IsISBN := True
    else IsISBN := False;
  end else IsISBN := False;
end;
```

[News Group]

Генерация еженедельных списков задач

Необходима программа, которая генерировала бы еженедельные списки задач. Программа должна просто показывать количество недель в списке задач и организовывать мероприятия, не совпадающие по времени. В предлагаемом текущем планировщике имеются 12 групп и планы на 11 недель.

Решение

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    Edit1: TEdit;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

const
  maxTeams = 100;

var
  Teams: array[1..maxTeams] of integer;
  nTeams, ix, week, savix: integer;

function WriteBox(week: integer): string;
var
  str: string;
  ix: integer;
begin
  Result := Format('Неделя=%d ', [week]);
  for ix := 1 to nTeams do begin
    if odd(ix) then Result := Result + ' '
    else Result := Result + 'v';
    Result := Result + IntToStr(Teams[ix]);
  end;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  nTeams := StrToInt(Edit1.Text);
  if Odd(nTeams) then inc(nTeams);   // должны иметь номера каждой группы
  ListBox1.Clear;
  for ix := 1 to nTeams do
    Teams[ix] := ix;
  ListBox1.Items.Add(WriteBox(1));
  for week := 2 to nTeams - 1 do begin
    Teams[1] := Teams[nTeams - 1];   // используем Teams[1] в качестве
                                     // временного хранилища
    for ix := nTeams downto 2 do
      if not Odd(ix) then begin
        savix := Teams[ix];
        Teams[ix] := Teams[1];
        Teams[1] := savix;
      end;
    for ix := 3 to nTeams - 1 do
      if Odd(ix) then begin
        savix := Teams[ix];
        Teams[ix] := Teams[1];
        Teams[1] := savix;
      end;
    Teams[1] := 1;                   // восстанавливаем известное значение
    ListBox1.Items.Add(WriteBox(week));
  end;
end;

end.
```

[News Group]

Правильное округление дробных чисел

Как научить Delphi правильно округлять дробные числа?

Решение 1

```
function RoundEx(X: Double; Precision : Integer): Double;
{ Precision : 1 - до целых, 10 - до десятых, 100 - до сотых... }
var
  ScaledFractPart, Temp: Double;
begin
  ScaledFractPart := Frac(X) * Precision;
  Temp := Frac(ScaledFractPart);
  ScaledFractPart := Int(ScaledFractPart);
  if Temp >= 0.5 then ScaledFractPart := ScaledFractPart + 1;
  if Temp <= -0.5 then ScaledFractPart := ScaledFractPart - 1;
  RoundEx := Int(X) + ScaledFractPart / Precision;
end;
```

[Nomadic]

Решение 2

`Conv(2.005, 2)` возвращает `2.01`; `Conv(2.5, 0)` возвращает `3`.

```
function Conv(cs: double; numb: integer): double;
var
  db, db1, db2: double;
  i: int64;
  ii, ink, i1: integer;
  st: string;
begin
  db := cs - int(cs);
  ink := 1;
  for ii := 1 to numb do ink := ink * 10;
  db1 := db * ink;
  db2 := cs * ink * 100;
  i := trunc(int(db2) / 100);
  i1 := trunc(db2 - i * 100);
  if i1 > 49 then inc(i);
  result := i / ink;
end;
```

[Kordyum Alexandr]

Примечание

Автор этой программы, очевидно, не знал, что кроме положительных чисел есть и отрицательные. Читателям предлагается самим (ради тренировки) ликвидировать эту оплошность – программа правильно округляет только положительные числа.

Хочу подсказать еще один способ округления:

```
NumStr := FloatToStrF(Temp, ffFixed, 16, 2);
```

`NumStr` – строка приемник, `Temp` – число, `ffFixed` – формат, `16` – общая длина полученного числа, `2` – количество знаков после запятой.

Этот способ примечателен тем, что существует пять стандартных различных форматов: `ffGeneral`, `ffExponent`, `ffFixed`, `ffNumber`, `ffCurrency` – это позволяет найти себе нужный, на свой вкус. Если же вам необходимо получить результат в числовой форме, то можно написать:

```
Temp := StrToFloat(FloatToStrF(Temp, ffFixed, 16, 2));
```

Решение 3

```
function RoundFloat(R: Extended; Decimals: Integer): Extended;

var
  Factor: Extended;

begin
  Factor := Int(Exp(Decimals * Ln(10)));
```

```
  Result := Round(Factor * R) / Factor;
end;
```

[News Group]

Почему возникает ошибка при использовании функции StrToFloat

В моей программе функция `StrToFloat('32.34')` *вызывает исключение «'32.34' is not valid float». Если число записывается без десятичной точки, такой проблемы не возникает. Почему?*

В установках русской редакции Windows по умолчанию считается, что разделитель дроби – запятая. Изменить данный параметр можно либо с помощью Панели управления, либо программно:

```
DecimalSeparator := '.';
```

Или вызвать функцию так:

```
StrToFloat('32,24');
```

[Nomadic]

Эквивалент Trim$(), Mid$() и другие

Удаление конечных/начальных пробелов и левых/правых частей строк может осуществляться с помощью следующих решений:

Решение 1

```
unit TrimStr;
{$B-}

{ Автор: Bob Swart [100434, 2072]
  Доработка под Delphi: Тугаев Олег, knsprog@krintel.ru

  LTrim() - Удаляет все пробелы в левой части строки
  RTrim() - Удаляет все пробелы в правой части строки
  Trim() - Удаляет все пробелы по краям строки
  RightStr() - Возвращает правую часть строки заданной длины
  LeftStr() - Возвращает левую часть строки заданной длины
  MidStr() - Возвращает центральную часть строки заданной длины }

interface

const
  Space = #$20;

function LTrim(const Str: String): String;
function RTrim(Str: String): String;
function Trim(Str: String): String;
```

```
function RightStr(const Str: String; Size: Word): String;
function LeftStr(const Str: String; Size: Word): String;
function MidStr(const Str: String; Size: Word): String;

implementation

function LTrim(const Str: String): String;
var
  len, i: Word;
begin
  i := 1;
  len := Length(Str);
  while (i <= len) and (Str[i] = Space) do Inc(i);
  LTrim := Copy(Str, i, len)
end;

function RTrim(Str: String): String;
var
  len: Word;
begin
  len := Length(Str);
  while (len > 0) and (Str[len] = Space) do Dec(len);
  RTrim := Copy(Str, 1, len);
end;

function Trim(Str: String): String;
begin
  Trim := LTrim(RTrim(Str))
end;

function RightStr(Const Str: String; Size: Word): String;
var
  len: Word;
begin
  len := Length(Str);
  if Size >= len then Size := len
  else RightStr := Copy(Str, len - Size + 1, Size)
end;

function LeftStr(Const Str: String; Size: Word): String;
begin
  LeftStr := Copy(Str, 1, Size)
end;

function MidStr(Const Str: String; Size: Word): String;
var
  len: Word;
begin
  len := Length(Str);
  if Size >= len then Size := len
```

```
else MidStr := Copy(Str, ((len - Size) div 2) + 1, Size)
end;

end.
```

[News Group]

Решение 2

Для Mid$ используйте функцию

```
Copy(S: string; index, count: integer): string;
```

С помощью команды Copy можно осуществить операции Right$ и Left$, выполнив:

Copy(S, 1, Length) для Left$ и Copy(S, Start, 255) для Right$.

Примечание

Index *и* Length *– байтовые позиции вашего «отправного пункта», их можно получить с помощью* Pos().

Некоторые дополнительные функции:

```
const
  BlackSpace = [#33..#126];

{ Возвращает строку со всеми пробельными символами и с удаленными повторяющимися
  апострофами. }
function squish(const Search: string): string;
var
  Index: byte;
  InString: boolean;
begin
  InString := False;
  Result := '';
  for Index := 1 to Length(Search) do begin
    if InString or (Search[Index] in BlackSpace) then
      AppendStr(Result, Search[Index]);
    InString := ((Search[Index] = '''') and (Search[Index - 1] <> '\'))
      xor InString;
  end;
end;

{ Возвращает часть строки, находящейся перед первой найденной подстрокой Find
  в строке Search. Если Find не найдена, функция возвращает параметр Search. }
function before(const Search, Find: string): string;
var
  index: byte;
begin
  index := Pos(Find, Search);
```

```
  if index = 0 then Result := Search
  else Result := Copy(Search, 1, index - 1);
end;

{ Возвращает часть строки, находящейся после первой найденной подстроки Find
  в строке Search. Если Find не найдена, функция возвращает NULL. }

function after(const Search, Find: string): string;
var
  index: byte;
begin
  index := Pos(Find, Search);
  if index = 0 then Result := ''
  else Result := Copy(Search, index + Length(Find), 255);
end;

{ Возвращает первый символ последней найденной подстроки Find в строке Search. Если
  Find не найдена, функция возвращает 0. Подобна реверсированной Pos() }

function RPos(const Find, Search: string): byte;
var
  FindPtr, SearchPtr, TempPtr: PChar;
begin
  FindPtr := StrAlloc(Length(Find) + 1);
  SearchPtr := StrAlloc(Length(Search) + 1);
  StrPCopy(FindPtr, Find);
  StrPCopy(SearchPtr, Search);
  Result := 0;
  repeat
    TempPtr := StrRScan(SearchPtr, FindPtr^);
    if TempPtr <> nil then
      if (StrLComp(TempPtr, FindPtr, Length(Find)) = 0) then begin
        Result := TempPtr - SearchPtr + 1;
        TempPtr := nil;
      end else
        TempPtr := #0;
  until TempPtr = nil;
end;

{ Возвращает подстроку, вложенную между парой подстрок Front ... Back. }
function inside(const Search, Front, Back: string): string;
var
  Index, Len: byte;
begin
  Index := RPos(Front, before(Search, Back));
  Len := Pos(Back, Search);
  if (Index > 0) and (Len > 0) then
    Result := Copy(Search, Index + 1, Len - (Index + 1))
  else
    Result := '';
end;
```

```
{ Возвращает левую часть "остатка" inside() или Search. }
function leftside(const Search, Front, Back: string): string;
begin
  Result := before(Search, Front + inside(Search, Front, Back) + Back);
end;

{ Возвращает правую часть "остатка" inside() или Null. }
function rightside(const Search, Front, Back: string): string;
begin
  Result := after(Search, Front + inside(Search, Front, Back) + Back);
end;

{ Возвращает строку со всеми удаленными по краям белыми пробелами. }
function trim(const Search: string): string;
var
  Index: byte;
begin
  Index := 1;
  while (Index <= Length(Search)) and not (Search[Index] in BlackSpace) do
    Index:=Index + 1;
  Result := Copy(Search, Index, 255);
  Index := Length(Result);
  while (Index > 0) and not (Result[Index] in BlackSpace) do Index:=Index - 1;
    Result := Copy(Result, 1, Index);
end;
```

[News Group]

Примечание

Будьте внимательны – символы в строках нумеруются, начиная с 1.

Разбивка строки на слова

Вашему вниманию предлагается пара простых функций, позволяющих работать с отдельными словами в строке. Они могут использоваться для разбивки текстовых полей на отдельные слова (`for i := 1 to NumToken do ...`) с последующим их сохранением в базе данных.

Решение 1

```
function GetToken(aString: String; SepChar: Char; TokenNum: Word): String;
{ параметры:
  aString: полная строка
  SepChar: символ, служащий разделителем между словами
  TokenNum: номер требуемого слова (подстроки)
  result: искомое слово или пустая строка, если количество слов меньше значения
         'TokenNum' }
var
  Token: String;
  StrLen: Byte;
  TNum: Byte;
```

```
  TEnd: Byte;
begin
  StrLen := Length(aString);
  TNum := 1;
  TEnd := StrLen;
  while ((TNum <= TokenNum) and (TEnd <> 0)) do begin
    TEnd := Pos(SepChar, aString);
    if TEnd <> 0 then begin
      Token := Copy(aString, 1, TEnd - 1);
      Delete(aString, 1, TEnd);
      Inc(TNum);
    end
    else Token := aString;
  end;
  if TNum >= TokenNum then GetToken := Token
  else GetToken := '';
end;

function NumToken(aString: String; SepChar: Char): Word;
{ параметры:
  aString: полная строка
  SepChar: символ, служащий разделителем между словами (подстроками)
  result: количество найденных слов (подстрок) }
var
  RChar: Char;
  StrLen: Byte;
  TNum: Byte;
  TEnd: Byte;
begin
  if SepChar = '#' then RChar := '*'
  else RChar := '#';
  StrLen := Length(aString);
  Tnum := 0;
  Tend := StrLen;
  while TEnd <> 0 do begin
    Inc(TNum);
    TEnd := Pos(SepChar, aString);
    if TEnd <> 0 then aString[TEnd] := RChar;
  end;
  Result := TNum -1;
end;
```

Решение 2

```
function CopyColumn(const sstring: string; cfence: char; iindex: word): string;
var
  i, ileft: integer;
begin
  result := EmptyStr;
  if iindex = 0 then exit;
  ileft := 0;
  for i := 1 to Length(sstring) do
```

```
    if sstring[i] = cfence then begin
      Dec(iindex);
      if iindex = 0 then begin
        result := Copy(sstring, ileft + 1, i - ileft - 1);
        exit;
      end
      else ileft := i;
    end;
    Dec(iindex);
    if iindex = 0 then result := Copy(sstring, ileft + 1, Length(sstring));
end;
```

Замена подстрок

Кто-нибудь знает быстрый алгоритм поиска и замены всех найденных подстрок sub1 на sub2 в строке str?

Решение

```
function ReplaceSub(str, sub1, sub2: String): String;
var
  aPos: Integer;
  rslt: String;
begin
  aPos := Pos(sub1, str);
  rslt := '';
  while (aPos <> 0) do begin
    rslt := rslt + Copy(str, 1, aPos - 1) + sub2;
    Delete(str, 1, aPos + Length(sub1) - 1);
    aPos := Pos(sub1, str);
  end;
  Result := rslt + str;
end;
```

[Щаматис Сергей]

Первая буква каждого слова в строке прописью

Как сделать, чтобы первая буква каждого слова в строке была в верхнем регистре?

Решение

```
function proper(s: string): string;
var
  t: string;
  i: integer;
  newWord: boolean;
begin
  if s = '' then exit;
  s := lowercase(s);
  t := uppercase(s);
  newWord := true;
  for i:=1 to length(s) do begin
    if newWord and (s[i] in ['a'..'z']) then begin
```

```
      s[i] := t[i];
      newWord := false;
      continue;
    end;
    if s[i] in ['a'..'z', ''''] then continue;
    newWord := true;
  end;
  result := s;
end;
```

[News Group]

Примечание

Данный код не работает с русскими символами.

Удаление ненужных подстрок из строки

Решение 1

```
function RemoveInvalid(what, where: string): string;
// what – удаляемая подстрока, where – обрабатываемая строка
var
  tstr: string;
  n: Word;
begin
  tstr := where;
  while Pos(what, tstr) > 0 do begin
    n := Pos(what, tstr);
    tstr := Copy(tstr, 1, n - 1) +
            Copy(tstr, n + length(what), length(tstr) - length(what) - n + 1);
  end;
  Result := tstr;
end;
```

Решение 2

Используйте стандартную функцию Pascal `Delete`. Например:

```
function RemoveSubStr(what, where: string): string;
var
  tstr: string;
  n: Word;
begin
  tstr := where;
  while Pos(what, tstr) > 0 do begin
    n := Pos(what, tstr);
    Delete(tstr, n, Length(what));
  end;
  result := tstr;
end;
```

Паскалевский эквивалент StrTok

Решение 1

```
function NextToken(P: PChar; Divider: PChar): PChar;
const
  next: PChar = nil ;
begin
  if P = nil then P := next;
  if P <> nil then begin
    next := StrPos(P, Divider);
    if next <> nil then begin
      next^ := #0;
      next := @next[StrLen(Divider)];
    end;
  end;
  NextToken := P;
end;
```

[News Group]

Решение 2

```
function StrTok(Phrase: PChar; Delimeter: PChar): PChar;
const
  tokenPtr: PChar = nil;
  workPtr: PChar = nil;
var
  delimPtr: PChar;
begin
  if (Phrase <> nil) then workPtr := Phrase
  else workPtr := tokenPtr;
  if workPtr = nil then begin
    Result := nil;
    Exit;
  end;
  delimPtr := StrPos(workPtr, Delimeter);
  if (delimPtr <> nil) then begin
    delimPtr^ := Chr(0);
    tokenPtr := delimPtr + 1
  end

  else
    tokenPtr := nil;
  Result := workPtr;
end;
```

[News Group]

Корректное сравнение и арифметические действия с DWORD

Существуют ли корректные методы сравнения и выполнения арифметических действий с четырехбайтными беззнаковыми целыми числами (DWORD)?

Ничего лучшего, чем `PChar(a) < PChar(b)`, пока не придумали.

[Nomadic]