Введение

ASCII art зародился в середине 80-х на платформах Amiga и Commodore 64. Развитие же этот вид компьютерного искусства получил в среде демомейкеров. Стандартный шрифт компьютера Amiga содержал очень высокие символы / и \ , из которых, располагая их в разных строках, можно было составить совершенно ровные непрерывные линии. Чаще всего такими линиями изображались названия групп, чем и объясняется, что говоря об ASCII art’е, говорят прежде всего именно о шрифтах. В самом начале, шрифты были чёткие, хорошочитаемые. Позднее, шрифты усложнялись, делались интереснее. Так впервые заговорили о школах Oldschool и Newschool.

Платформу PC искусство текстового режима облюбовало с появлением нового стандарта ANSI, давшему реальное преимущество перед Амигой. ANSI позволял задавать цвет символу, что в совокупности с имеющемеся в таблице символов блоками, породило новое мощное ответвление — ANSI art. Первой арт-группой, показавшей изобразительные возможности на высоком уровне, стала Aces of ANSI Art (A.A.A.) За которой потянулись последователи.

И только уже позже, было замеченно, что если использовать символ почти полностью заполняющий ячейку (например $), а к нему добавить менее плотный символ — то таким образом можно сглаживать общую форму (этот приём получил название «антиалиазинг» (от англ. antialiasing)) Так, в конечном итоге, зародилась современная школа рисования ASCII.

Вот пример ASCII ART:

К середине девяностых сцена полностью сформировалась, главными направлениями в рисовании в текстовом режиме, стали Ascii scene, Ansi scene и Amiga style (который, частенько, и называют Oldschool).

"…Тут важно сказать, что на PC textmode-искусство началось с имитации Amiga style, а закончилось картинками с кучей «$» и без единого «\» или «/». Писишные художники, вполне естественно, назвали старый стиль oldschool, а новый, с долларами, — newschool. Им, конечно, было невдомек, что на Amiga названия уже были зарезервированы. В результате произошла путаница, а амижные художники получили еще один повод презирать PC и все с ним связанное."

Я в своей работе представил упрощенный вариант программы, написанной на языке программирования DELPHI, предназначенной для ACSII ART.

История создания Delphi

Пожалуй, наиболее важной вехой в истории программирования, сравнимой по значимости разве что с изобретением письменности, можно считать переход от машинных кодов (тарабарщины типа 0110110101111...) к понятным простому смертному языкам программирования (типа ALGOL, FORTRAN, PL/1, Pascal), а также к широкому использованию методов структурного программирования. Программы стали модульными, состоящими из подпрограмм. Появились библиотеки готовых подпрограмм, облегчающие многие задачи, но все равно программистам хватало трудностей, особенно при разработке пользовательского интерфейса.

Качественным шагом в развитии методов структурного программирования стало изобретение объектно-ориентированного программирования (языков SmallTalk, C++, Turbo Pascal и др.). Программы стали строиться не из чудовищных по размеру процедур и функций, перерабатывающих громоздкие структуры данных, а из сравнительно простых кирпичиков-объектов, в которых были упрятаны данные и подпрограммы их обработки. Гибкость объектов позволила очень просто приспосабливать их для собственных целей, прилагая для этого минимум усилий. Программисты обзавелись готовыми библиотеками объектов, но, как и раньше, создание пользовательского интерфейса требовало уйму времени и сил, особенно когда программа должна была работать под управлением популярной операционной системы Windows и иметь графический пользовательский интерфейс.

С изобретением визуального программирования, первой ласточкой которого была среда разработки Visual Basic, создание графического пользовательского интерфейса стало под силу даже новичку. В среде Visual Basic можно было быстро создать приложение для операционной системы Windows, в котором были все присущие графическому пользовательскому интерфейсу элементы: окна, меню, кнопки, поля ввода и т.д. Все эти элементы

превратились в строительные блоки программы — компоненты — объекты, имеющие визуальное представление на стадии проектирования и во время работы.

Проектирование пользовательского интерфейса упростилось на порядок, однако, для профессиональных программистов язык Basic оказался явно слабоват. Отсутствие в нем контроля типов данных и механизма их расширения оказалось камнем преткновения на пути создания серьезных программ. Создание нестандартных компонентов в среде Visual Basic было крайне затруднено (для этого приходилось прибегать к другим средствам разработки, в частности, к языку C++). В общем, среда Visual Basic отлично подходила для создания прототипов приложений, но не для разработки коммерческих программных продуктов.

Мечта программистов о среде программирования, в которой бы простота и удобство сочетались с мощью и гибкостью, стала реальностью с появлением среды Delphi. Она обеспечивала визуальное проектирование пользовательского интерфейса, имела развитый объектно-ориентированный язык Object Pascal (позже переименованный в Delphi) и уникальные по своей простоте и мощи средства доступа к базам данных. Язык Delphi по возможностям значительно превзошел язык Basic и даже в чем-то язык C++, но при этом он оказался весьма надежным и легким в изучении (особенно в сравнении с языком C++). В результате, среда Delphi позволила программистам легко создавать собственные компоненты и строить из них профессиональные программы. Среда оказалась настолько удачной, что по запросам любителей C++ была позже создана среда C++Builder — клон среды Delphi на основе языка C++ (с расширенным синтаксисом).

Среда Delphi стала, по сути, лучшим средством программирования для операционной системы Windows, но программистов ждало разочарование, если возникало желание перенести программу в другую операционную систему, в частности, в операционную систему Unix.

Практически одновременно со средой программирования Delphi на свет появилась технология Java, включавшая три составляющих: одноименный язык программирования, очень похожий на язык C++, но более простой и безопасный; универсальный байт-код, в который компилировались программы на языке Java; интерпретатор (виртуальную машину) для выполнения байт-кода в любой операционной системе. Благодаря автоматическому управлению памятью — так называемой «сборке мусора» — резко повысилась надежность программ и скорость их разработки.

Поначалу на технологию Java возлагались большие надежды. Программные библиотеки для языка Java стали единым стандартом, поэтому написанные на нем программы оказались по-настоящему переносимыми. Однажды написанная и компилированная в байт-код программа могла работать на любой платформе без ограничений (единственное требование — наличие на этой платформе виртуальной машины Java).

Безграничная переносимость Java-программ родила идею сетевого компьютера и сетевых вычислений, суть которой в том, что все программы хранятся в байт-коде на серверах сети Интернет. Когда подключенный к сети пользователь запускает программу, то она сначала загружается к нему на компьютер, а затем интерпретируется. Охваченные этой идеей крупные фирмы ринулись осваивать новый рынок Java-приложений. Для языка Java появились средства визуального программирования, такие как JBuilder и Visual Age for Java. Казалось бы, бери и используй их для разработки пользовательского интерфейса и серверных программ. Но практически пропускная способность сети Интернет в лучшем случае обеспечивала оперативную загрузку на клиентские компьютеры лишь небольших по размеру программ. Кроме того, созданный на языке Java пользовательский интерфейс хронически отставал от возможностей операционной системы Windows и раздражал своей медлительностью. Поэтому технологию Java стали применять главным образом

для разработки серверных приложений. Однако и здесь цена переносимости программ оказалась очень высокой — представленные в байт-коде программы работали на порядок медленнее аналогичных программ, компилированных напрямую в команды процессора. Применение динамической компиляции, при которой программа перед выполнением преобразуется из байт-кода в команды процессора и попутно оптимизируется, улучшило положение дел, но скорость работы Java-приложений так и не смогла приблизиться к скорости работы традиционных приложений. Иными словами, переносимость программ шла в ущерб их производительности и удобству. Многие начали задумываться над целесообразностью такой переносимости программ вообще.

Тем временем назревала революция в области серверных платформ — небывалыми темпами росла популярность операционной системы Linux.

Среда программирования Kylix

В связи со стремительным распространением операционной системы Linux возникла необходимость в эффективных средствах создания для нее программ. Таким средством стала среда Kylix (произносится «киликс») — первая среда визуального программирования для операционной системы Linux. Среда Kylix явилась полным аналогом среды Delphi и была совместима с ней по языку программирования и библиотекам компонентов. Программу, созданную в среде Delphi, можно было без изменений компилировать в среде Kylix, и наоборот. Эта возможность достигалась за счет новой библиотеки компонентов, которая взаимодействовала с операционной системой не напрямую, а через промежуточный программный слой, скрывающий разницу в работе компонентов в той или иной операционной системе. Программисты получили возможность создавать программы сразу для двух самых популярных операционных систем: Windows и Linux. Фактически вместо принципа абсолютной переносимости программ была предложена идея разумной переносимости.

Постепенно пришло понимание того, что в эпоху Интернет способность программ к взаимодействию в сети не менее (а порой более!) важна, чем возможность их переноса на различные платформы. Такая способность была обеспечена за счет стандартизации протоколов обмена данными в сети Интернет и форматов этих данных. Развитие протоколов и стандартов Интернет привело к рождению технологии Web-сервисов, которая ставила своей задачей максимально упростить создание программ, взаимодействующих по принципу клиент-сервер в глобальной сети. Поддержка технологии Web-сервисов была изящно встроена в системы Delphi и Kylix, в результате разработчики программ получили в руки еще один очень важный инструмент.

Несмотря на трудности и уроки Java-технологии, программисты не желали отказываться от идеи создания полностью переносимых программ. Вместе с тем их совершенно не устраивала необходимость платить производительностью и удобством программ за переносимость. Работы по разрешению этого противоречия привели к появлению на свет технологии под названием .NET (произносится «дот-нет»).

Технология .NET по сути явилась новой платформой, надстроенной над другими операционными системами, и этим походила на технологию Java. Однако у технологии .NET имелся ряд существенных концептуальных отличий. В частности, платформа .NET хотя и имела свой собственный новый язык программирования C# (произносится «си-шарп»), но не была привязана только к нему, позволяя писать программы на других языках. Кроме того, программы для платформы .NET компилировались не в байт-код, а в универсальный промежуточный язык, который сохранял семантику программы и был близок к ее исходному тексту (байт-код, напротив, близок к командам процессора). Программы на промежуточном языке вообще не интерпретировались, а всегда компилировались в команды процессора при запуске программы или при ее первоначальной установке на компьютер пользователя. Выполняемый код получался очень эффективным и оказывался сравнимым по быстродействию с

выполняемым кодом, полученным прямой компиляцией с языка высокого уровня в команды процессора. Немаловажно и то, что на платформе .NET стало возможным использование любых (а не только стандартных) библиотек подпрограмм и компонентов, а также всех функций операционной системы. Все это обеспечило создание быстрых и удобных программ.

Поначалу технология .NET была доступна только для семейства операционных систем Windows, но со временем этот недостаток был устранен, и на свет появилась платформа Mono — клон технологии .NET для операционных систем Linux и Unix.

Платформы .NET и Mono имеют большое будущее, поэтому фирма Borland адаптировала для них язык и среду программирования Delphi. В итоге, разработчики получили уникальную возможность — применять один и тот же язык Delphi для создания профессиональных программ для любых операционных систем и платформ: Windows, Linux, .NET, Mono. Этим, кстати, язык Delphi выгодно отличается от модного ныне языка C#, который применяется лишь для программирования на платформах .NET и Mono.

У языка Delphi есть еще одно очень важное преимущество перед остальными коммерчески успешными языками — он великолепно подходит для обучения программированию. Поэтому авторы рекомендуют его в качестве первого языка для всех учеников и студентов, собирающихся стать профессиональными программистами.

В России Borland Delphi появляется в конце 1993 г. и сразу же завоевывает широкую популярность. Новые версии выходят практически каждый год. В них реализуются все новые мастера, компоненты и технологии программирования.

Действительно, процесс разработки в Delphi предельно упрощен. В первую очередь это относится к созданию интерфейса, на который уходит 80% времени разработки программы. Вы просто помещаете нужные компоненты на

поверхность Windows-окна (в Delphi оно называется формой) и настраиваете их свойства с помощью специального инструмента (Object Inspector). С его помощью можно связать события этих компонентов (нажатие на кнопку, выбор мышью элемента в списке и т.д.) с кодом его обработки - и вот простое приложение готово. Причем разработчик получает в свое распоряжение мощные средства отладки (вплоть до пошагового выполнения команд процессора), удобную контекстную справочную систему (в том числе и по Microsoft API), средства коллективной работы над проектом, всего просто не перечислить. Вы можете создавать компоненты ActiveX без использования Microsoft IDL, расширять возможности web-сервера (скрипты на стороне сервера), практически ничего не зная об HTML, XML или ASP. Можно создавать распределенные приложения на базе COM и CORBA, Интернет- и intranet-приложения, используя для доступа к данным Borland DataBase Engine, ODBC-драйверы или Microsoft ADO. Появившаяся, начиная с Delphi 3, поддержка многозвенной технологии (multi-tiered) доступа к данным позволяет создавать масштабируемые приложения (относительно слабо зависящие от сервера БД) за счет перенесения методов обработки информации (бизнес-правил) на среднее звено.

Как уже говорилось ранее, в Delphi используется язык Object Pascal, который постоянно расширяется и дополняется Borland. Язык в полной мере поддерживает все требования, предъявляемые к объектно-ориентированному языку программирования. Как и положено строго типизированному языку, классы поддерживают только простое наследование, но зато интерфейсы могут иметь сразу несколько предков. К числу особенностей языка следует отнести поддержку обработки исключительных ситуаций (exceptions), а также перегрузку методов и подпрограмм (overload) в стиле C++. К числу удачных, на взгляд автора, относится также поддержка длинных строк в формате WideChar и AnsiChar. Последний тип (AnsiStrmg) позволяет использовать все прелести динамического размещения информации в памяти без всяких забот о ее

выделении и сборке мусора Delphi делает это автоматически. Для поклонников свободного стиля программирования имеются открытые массивы, варианты и вариантные массивы, позволяющие размещать в памяти все, что душе угодно и смешивать типы данных.

Вы можете создавать свои собственные компоненты, импортировать OCX-компоненты, создавать <шаблоны> проектов и <мастеров>, создающих <заготовки> проектов. Мало того, Delphi предоставляет разработчику интерфейс для связи ваших приложений (или внешних программ) с интегрированной оболочкой Delphi (IDE).

Таким образом, вы можете использовать Delphi для создания как самых простых приложений, на разработку которых требуется 2-3 часа, так и серьезных корпоративных проектов, предназначенных для работы десятков и сотен пользователей. Причем для этого можно использовать самые последние веяния в мире компьютерных технологий с минимальными затратами времени и сил.

Delphi — результат развития языка Турбо Паскаль, который, в свою очередь, развился из языка Паскаль. Паскаль был полностью процедурным языком, Турбо Паскаль начиная с версии 5.5 добавил в Паскаль объектно-ориентированные свойства, а Delphi — объектно-ориентированный язык программирования с возможностью доступа к метаданным классов (то есть к описанию классов и их членов) в компилируемом коде, также называемом интроспекцией. Так как все классы наследуют функции базового класса TObject, то любой указатель на объект можно преобразовать к нему, и воспользоваться методом ClassType и функцией TypeInfo, которые и обеспечат интроспекцию. Также отличительным свойством Дельфи от C++ является отсутствие возможности располагать объекты в стеке (объекты, унаследованные из Турбо Паскаля, располагаться в стеке могут) — все объекты попадают в динамически выделяемую область (*кучу*).

Де-факто Object Pascal, а затем и язык Delphi являются функциональными наращиваниями Turbo Pascal. Об этом говорят обозначения версий компилятора. Так, в Delphi 7 компилятор имеет номер версии 15.0 (Последняя версия Borland Pascal / Turbo Pascal обозначалась 7.0, в Delphi 1 компилятор имеет версию 8.0, в Delphi 2 — 9.0, и т. д. Номер версии 11.0 носит компилятор Pascal, входивший в состав среды C++Builder).

Delphi оказал огромное влияние на создание концепции языка C# для платформы .NET. Многие его элементы и концептуальные решения вошли в состав C#. Одной из причин называют переход Андерса Хейлсберга, одного из ведущих разработчиков Дельфи, из компании Borland Ltd. в Microsoft Corp.

Версия 1 была предназначена для разработки под 16-ти разрядную платформу Win16;

Версии со второй компилируют программы под 32-х разрядную платформу Win32;

Вместе с 6-й версией Delphi вышла совместимая с ним по языку и библиотекам среда Kylix, предназначенная для компиляции программ под операционную систему Linux;

Версия 8 способна генерировать байт-код исключительно для платформы .NET. Это первая среда, ориентированная на разработку мультиязычных приложений (лишь для платформы .NET);

Последующие версии (обозначаемые годами выхода, а не порядковыми номерами, как это было ранее) могут создавать как приложения Win32, так и байт-код для платформы .NET;

Delphi for .NET — среда разработки Delphi, а так же язык Delphi (Object Pascal), ориентированные на разработку приложений для .NET.

Первая версия полноценной среды разработки Delphi для .NET — Delphi 8. Она позволяла писать приложения только для .NET. В настоящее время, в Delphi 2006, можно писать приложения для .NET используя стандартную

библиотеку классов .NET, VCL для .NET. Среда также позволяет писать .NET-приложения на C# и Win32-приложения на C++. Delphi 2006 содержит функции для написания обычных приложений с использованием билиотек VCL и CLX. Delphi 2006 поддерживает технологию MDA с помощью ECO (Enterprise Core Objects) версии 3.0.

В марте 2006 года компания Borland приняла решение о прекращении дальнейшего совершенствования интегрированных сред разработки JBuilder, Delphi и C++Builder по причине убыточности этого направления. Планируется продажа IDE-сектора компании. Группа сторонников свободного программного обеспечения организовала сбор средств для покупки у Borland прав на среду разработки и компилятор (см. openDelphi.org).

Однако в ноябре того же года было принято решение отказаться от продажи IDE бизнеса. Тем не менее, разработкой IDE продуктов теперь будет заниматься новая компания - CodeGear, которая будет финансово полностью подконтрольна Borland.

Borland продолжил развитие IDE систем под именем Turbo: Turbo Delphi, Turbo Delphi for .NET, Turbo C#, Turbo C++

Пример кода на языке Delphi

```
procedure TForm2.FormCreate(Sender: TObject);
var
  {Объявление переменной типа TStrings(список строк).}
  Strings: TStrings;
begin
  {Создание(выделение памяти и т. д.) объекта типа TStringList.
   TStringList - это потомок TStrings, реализующий абстрактные методы.}
  Strings := TStringList.Create;
  try
    {Добавление строки.}
    Strings.Add('Добавляемая строка.');
    {Сохранение строк в файл.}
    Strings.SaveToFile('C:\Strings.txt');
  finally
    {Удаление объекта.}
    Strings.Free;
  end;
end;
```

Код программы

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, Menus, StdCtrls, Gauges, ComCtrls, ExtCtrls,
ToolWin, ImgList;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    ToolBar1: TToolBar;
    Memo1: TMemo;
    ToolButton1: TToolButton;
    GroupBox1: TGroupBox;
    Image1: TImage;
    StatusBar1: TStatusBar;
    Gauge1: TGauge;
    Button1: TButton;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    OpenDialog1: TOpenDialog;
    RadioGroup1: TRadioGroup;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Label1: TLabel;
    ImageList1: TImageList;
    FontDialog1: TFontDialog;
    N7: TMenuItem;
    SaveDialog1: TSaveDialog;
    Image2: TImage;
    procedure N3Click(Sender: TObject);
```

```
    procedure N4Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure N7Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
  private
    { Private declarations }

  public
    { Public declarations }
  end;

var
  Form1: TForm1;

        implementation

uses Printers;
 {$R *.DFM}

 type
   THeaderFooterProc =
     procedure(aCanvas : TCanvas; aPageCount : integer;
     aTextrect : TRect; var Continue : boolean) of
object;


procedure TForm1.N3Click(Sender: TObject);
begin
Close;
end;

procedure TForm1.N4Click(Sender: TObject);
begin
// Открываем файл на обработку
IF OpenDialog1.Execute and
FileExists(OpenDialog1.FileName) then
begin
Image1.Picture.LoadFromFile(OpenDialog1.FileName);
StatusBar1.Panels[0].Text:='Изображение загружено.';
// Переводим картинку в монохром
Image2.Picture.Bitmap:=Image1.Picture.Bitmap;
Image1.Picture.Bitmap.Monochrome:=true;
end;
end;
```

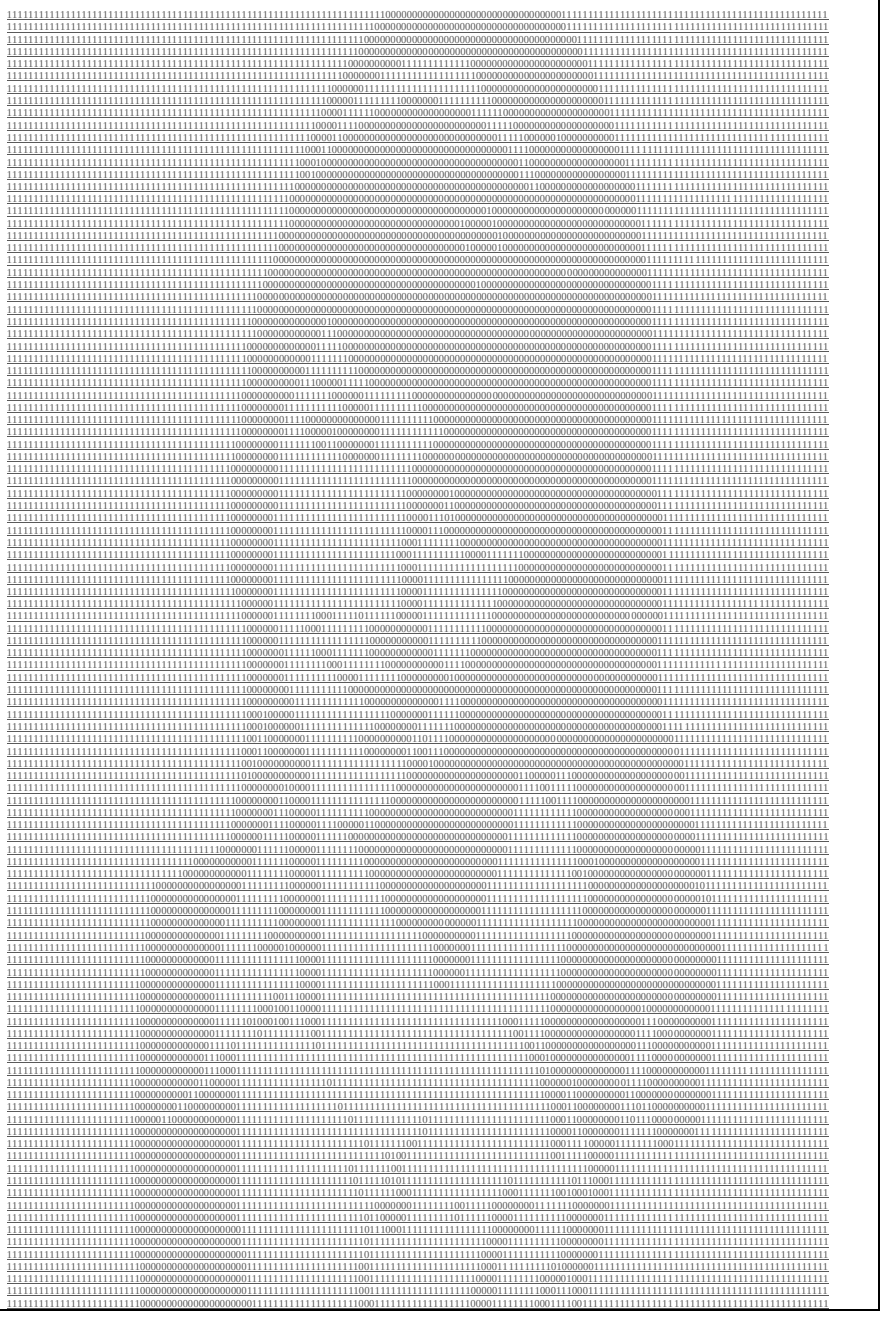
```
procedure TForm1.Button1Click(Sender: TObject);
var
x,y: integer;
col : TColor;
begin
StatusBar1.Panels[0].Text:='Идет сканирование...';
Gauge1.MaxValue:=Image1.Picture.Bitmap.Height - 1;
Gauge1.Progress:=1;
Memo1.Clear;
Memo1.Visible:=false;
//Memo1.Visible:=false;
//Первоначальное сканирование
FOR Y:=1 TO Image1.Picture.Bitmap.Height - 1 do
BEGIN
For x:=1 to Image1.Picture.Bitmap.Width - 1 do
begin
Col:=GetPixel(Image1.Canvas.Handle,x,y);

 If RadioButton1.Checked then
  begin

  IF Col=clBlack then Memo1.Text:=Memo1.Text+'0';
  IF Col=clWhite then Memo1.Text:=Memo1.Text+'1';

  end
  else
  begin

  IF Col=clWhite then Memo1.Text:=Memo1.Text+'0' ;
  IF Col=clBlack then Memo1.Text:=Memo1.Text+'1' ;

  end;
end;
Memo1.Lines.Add('');
Gauge1.Progress:=GAuge1.Progress+1;
END;

Memo1.Visible:=true;
StatusBar1.Panels[0].Text:='Сканирование закончено.';

end;
```

```
procedure TForm1.N7Click(Sender: TObject);


begin
If SaveDialog1.Execute and
FileExists(SaveDialog1.FileName) then
Memo1.Lines.SaveToFile(SaveDialog1.FileName);
end;

procedure TForm1.N2Click(Sender: TObject);
begin
If FontDialog1.Execute then
memo1.Font:=FontDialog1.Font;
end;

end.
```

Заключение

Таким образом, мы получили алгоритм сканирования монохромного изображения и вывода простого ASCII ART. Этот алгоритм можно взять для использования его в программах, предназначенных для работы с графикой и текстом. Например, для преобразования изображения с текстом в сам текст.

Пример работы программы



Список литературы

1. Михаил Фленов «Библия Delphi»
2. Фаронов В.В. Delphi 2005. Разработка приложений для баз данных Интернета
3. Шупрута В.В. Delphi 2006 на примерах
4. Архангельский А.Я. Delphi 2006. Справочное пособие: Язык Delphi, классы, функции Win32 и NET