Практические занятия

НЕТИПИЗИРОВАННЫЕ ФАЙЛЫ

Цель: выработать навыки по работе в системе программирования PascalABC.NET, научиться работать с нетипизированными файлами.

Задачи:

1. Изучить нетипизированные файлы.
2. Выполнить примеры задач в среде PascalABC.NET как в виде консольного и оконного проектов.

1. Нетипизированные файлы

Работа с нетипизированными файлами во многом схожа с работой с типизированными файлами. Разница в том, что в нетипизированном файле тип данных не объявляется, поэтому в какую переменную вы считаете данные из файла тот тип и получите, а так же какую переменную впишите в файл, столько байт вы и займёте в файле. Для работы с нетипизированными файлами используются те же процедуры и функции, что и с типизированными. Так же есть указатель, который перемещается по файлу с шагом в один байт, и который можно устанавливать в любое место. Однако если вы считаете данные в переменную, которая занимает 2 байта, то указатель переместиться не на одну позицию, как в типизированном файле, а на две позиции и наоборот, если запишите переменную, занимающую 2 байта, то указатель тоже переместиться на две позиции.

Суть нетипизированного файла заключается в том, что вы можете вписывать в файл данные разных типов, соответственно можно и считывать из файла данные разных типов. При этом настоятельно рекомендую следить за тем, какие данные, и в какой последовательности были вписаны в файл. В такой же последовательности и такие же данные необходимо будет извлекать оттуда.

Пример:

```
var f:file;
    s:string;
    b:integer;
    r:real;
begin
  assign(f,'c:\file.dat');
  rewrite(f);
  write(f,'Петр');
  write(f,'Сергеевич');
  write(f,'Орлов');
```

```
  Write(f,25);
  Write(f,'Инженер');
  Write(f,75600.35);
  close(f);
  reset(f);
  read(f,s);
  Writeln('      Имя - ',s);
  read(f,s);
  Writeln(' Отчество - ',s);
  read(f,s);
  Writeln('  Фамилия - ',s);
  read(f,b);
  Writeln('  Возраст - ',b,' лет');
  read(f,s);
  writeln('Должность - ',s);
  read(f,r);
  Writeln('    Доход - ',r,' рублей в месяц');
  close(f);
end.
```

Результат работы программы:

Как видно, структура созданного файла очень похожа на тип данных запись. Однако есть разнице. В данном примере мы вписываем в файл строки переменной длины, что невозможно было бы реализовать, используя тип данных запись. Тем самым мы можем сэкономить значительное количество памяти. К тому же, далее мы можем вписывать в файл ещё какие-либо не повторяющиеся данные.

Ещё раз отметим, следите за тем какие данные, в какой последовательности были вписаны в файл. Так же их и придётся извлекать оттуда. Если что-то перепутаете, то может произойти ошибка ввода вывода. Поэтому вся ответственность за соответствие данных при работе с нетипизированными файлами лежит полностью на программисте.

Далее приведём процедуры и функции, которыми можно работать с нетипизированными файлами, без примеров, так как тут всё аналогично типизированным файлам, только следует помнить, что файловый указатель перемещается с шагом в один байт.

function **Eof**(f: FileType): boolean; – Возвращает True, если достигнут конец файла f.

procedure **Truncate**(f: file); – Усекает нетипизированный файл f, отбрасывая все элементы с позиции файлового указателя.

function **FileSize**(f: file): LongInt; – Возвращает количество байт в нетипизированном файле f.

function **FilePos**(f: file): LongInt; – Возвращает текущую позицию файлового указателя в нетипизированном файле f.

procedure **Seek**(f: file; n: LongInt); – Устанавливает текущую позицию файлового указателя в нетипизированном файле f на байт с номером n.

Всё то, что было написано выше во всех параграфах справедливо к языку Pascal в целом, т.е. эти знания вы можете применять как в среде разработки PascalABC.NET, так и в TurboPascal. То что было сказано выше про нетипизированные файлы справедливо только в среде программирования PascalABC.NET.

В изначальном варианте Паскаля, в TurboPascal, работа с нетипизированными файлами организована несколько иначе. И кроме того был ещё один смысл их использования – это скоростной обмен данными между жёстким диском и оперативной памятью.

Среда программирования PascalABC.NET была создана для обучения принципам программирования. Она ориентирована на программирование под операционную систему Windows. Поэтому PascalABC.NET отличается от «изначального» TurboPascal. В данной работе за основу был выбран язык PascalABC.NET, поэтому принципы работы с нетипизированными файлами, которые заложены в TurboPascal не раскрыты.

Рассмотрим, как с помощью нетипизированного файла можно определить структуру переменной типа String. Строка может быть различного размера, соответственно и количество занимаемой ей памяти может быть разное. Поэтому, узнать, каким образом храниться переменная типа String в памяти компьютера, будет интересно.

Сделать это можно следующим образом: запишем переменную типа String в нетипизированный файл, затем откроем этот файл как типизированный и выведем его на экран в виде последовательности байтов. Потом проанализируем эту последовательность.

Код программы:

```
var fnet:file;
  fbyte:file of byte;
  s:string;
```

```
   b:byte;

begin
  s:='АБВГД';
  Assign(fnet,'c:\Exp.str');
  Rewrite(fnet);
  write(fnet,s);
  Close(fnet);
  Assign(fbyte,'c:\Exp.str');
  Reset(fbyte);
  while not Eof(fbyte) do
    begin
      read(fbyte,b);
      write(b,'  ');
    end;
  Close(fbyte);
end.
```

Результат работы программы:

Проанализировав результат работы программы, можно прийти к следующим заключениям: данная переменная занимает всего 6 байт памяти. Первый байт содержит количество символов в строке, последующие байты являются символами.

Для проверки такого утверждения допишем в программу следующий код:

```
    .....................
  writeln;
  Reset(fbyte);
  read(fbyte,b);
  write(b,'  ');
  while not Eof(fbyte) do
    begin
      read(fbyte,b);
      write(Chr(b):3,'  ');
    end;
  Close(fbyte);
end.
```

Результат работы программы:

Как видно из результата, наши рассуждения были верны. Получается, что в PascalABC.NET переменная типа String в памяти храниться следующим образом: первый байт содержит количество символов в строке, последующие байты содержат символы строки.

Познавать программирование можно не только, читая справочную информацию и какую-либо литературу, но и проводя какие-либо эксперименты. Причём такой способ, может быть, и не всегда экономичен с точки зрения затраченного времени, тем не менее, он больше развивает ваш интеллект. Плюсом ко всему та информация, которая добыта вами в результате каких-либо опытов, лучше запоминается и будет лучше вами понята.

На каждый, пройдённый вами материал, придумывайте свои примеры программ и экспериментируйте над ними. Такой способ обучения даёт наилучшие результаты.

2. Подпрограммы для работы с файлами

Далее будут приведены ещё ряд процедур и функций для работы с файлами, каталогами и дисками. Все они применимы ко всем трём типам файлов.

procedure **Erase**(f: FileType);

Удаляет файл, связанный с файловой переменной f. При использовании данной процедуры удаляемый файл не должен быть открыт, или использоваться другой программой:

```
var f:file of Byte;
begin
  assign(f,'c:\123.int');
  erase(f);
end.
```

Функции для работы с именами файлов:

procedure **Rename**(f: FileType; newname: string); – переименовывает файл, связанный с файловой переменной f, давая ему новое имя, находящееся в строке newname. При использовании данной процедуры так же переименовываемый файл не должен быть открыт, или использоваться другой программой:

```
var f:file of Byte;
begin
  assign(f,'c:\123.int');
  rename(f,'c:\124.int');
```

```
end.
```

function **ExtractFileName**(fname: string): string; – выделяет имя файла из полного имени файла fname.

function **ExtractFileExt**(fname: string): string; – выделяет расширение из полного имени файла fname.

function **ExtractFilePath**(fname: string): string; – выделяет путь из полного имени файла fname.

function **ExtractFileDir**(fname: string): string; – выделяет имя диска и путь из полного имени файла fname.

Пример программы:

```
var s:string;
begin
  s:='c:\Papka_1\Papka_2\file.dat';
  writeln('Полное имя файла - ',s);
  writeln('ExtractFileName(s) - ',ExtractFileName(s));
  writeln('ExtractFileExt(s) - ',ExtractFileExt(s));
  writeln('ExtractFilePath(s) - ',ExtractFilePath(s));
  writeln('ExtractFileDir(s) - ',ExtractFileDir(s));
end.
```

```
Полное имя файла - c:\Papka_1\Papka_2\file.dat
ExtractFileName(s) - file.dat
ExtractFileExt(s) - .dat
ExtractFilePath(s) - c:\Papka_1\Papka_2\
ExtractFileDir(s) - c:\Papka_1\Papka_2
```

Процедуры и функции для работы с файлами, каталогами и дисками.

function **GetDir**: string; – возвращает текущий каталог.

procedure **ChDir**(s: string); – меняет текущий каталог.

function **SetCurrentDir**(s: string): boolean; – Устанавливает текущий каталог. Возвращает True, если каталог успешно установлен.

При указании имени файла можно указывать его без пути. В таком случае файл будет искаться в текущем каталоге. Изначально текущим каталогом считается тот каталог, в котором находится сама программа. О том, где находится та программа, которую вы написали и запустили на выполнение, будет рассказано в 18 параграфе. Сейчас просто знайте,

если необходимо создать в программе файл, можете указать только его имя.

С помощью функции GetDir можете узнать, где находится ваша программа:

```
begin
  writeln('Эта программа находится в ',GetDir);
  writeln('Теперь сменим текущий каталог на C:\Program Files');
  ChDir('C:\Program Files');
  writeln('Уточняем, какой текущий каталог на данный момент - ',GetDir);
end.
```

Результат работы программы:

```
Эта программа находится в C:\PABCWork.NET
Теперь сменим текущий каталог на C:\Program Files
Уточняем, какой текущий каталог на данный момент - C:\Program Files
```

function **ChangeFileNameExtension**(name,newext: string): string; – Изменяет расширение файла с именем name на newext. Другими словами, функция возвращает имя файла name с изменённым расширением на newext.

Пример:

```
var s:string;
begin
  s:=ChangeFileNameExtension('c:\Exp.str','dat');
  writeln(s);
end.
```

Результат работы программы:

```
c:\Exp.dat
```

Следующие подпрограммы будут приведены без примеров. Думаю, вы без труда сможете с ними разобраться. Примеры их использования будет предложено создать самостоятельно в задачах для самостоятельного решения.

procedure **MkDir**(s: string); – создает каталог.

function **CreateDir**(s: string): boolean; – создает каталог. Возвращает True, если каталог успешно создан.

procedure **RmDir**(s: string); – удаляет каталог.

function **RemoveDir**(s: string): boolean; – Удаляет каталог. Возвращает True, если каталог успешно удален.

function **DeleteFile**(s: string): boolean; – удаляет файл. Если файл не может быть удален, то возвращает False.

function **FileExists**(name: string): boolean; – Возвращает True, если файл с именем name существует. Прежде чем открывать какой-либо файл, с помощью данной функции можно убедиться

function **DiskFree**(diskname: string):LongInt; – Возвращает свободное место в байтах на диске с именем diskname.

function **DiskSize**(diskname: string):LongInt; Возвращает размер в байтах на диске с именем diskname.

function **DiskFree**(disk: integer):LongInt; – Возвращает свободное место в байтах на диске disk.

function **DiskSize**(disk: integer):LongInt; – Возвращает размер в байтах на диске disk.

В обоих последних функциях: disk=0 – текущий диск, disk=1 – диск A, disk=2 – диск B, и т.д.

В данной работе познакомились с нетипизированными файлами и познакомились с системными подпрограммами для работы с файлами.

Задача

Создать программу, демонстрирующую принцип, каким образом переменная типа String храниться в памяти компьютера. В программе необходимо сформировать нетипизированный файл переменными типа String. Затем нужно вывести содержимое файла на экран в виде таблицы: в первом столбце должно находиться количество символов в строке, во втором столбце содержимое строки. Строки можно сформировать случайным образом, с помощью функции random.

Решение**:**

```
Program StrukturaString;
//Программа демонстрирует структуру переменных типа String
Var f:file;
    i,k:byte;
    sTemp:string;
    cTemp:char;
begin
  assign(f,'c:\Text.ntf');
  rewrite(f);
//Формируем нетипизированный файл случайными строками
  For i:=1 to random(20) do
   begin
     sTemp:='';
   //Формируем строку
```

```
    for k:=1 to random(20) do sTemp:=sTemp+chr(random(32,255));
    Write(f,sTemp);
   end;
  close(f);
//Выводим содержимое файла на экран
  Writeln('Количество символов  Содиржимое строки');
  reset(f);
  while not eof(f) do
   begin
    read(f,i);
   //Выводим количесвто символов
    write(i:19,'  ');
   //Выводим содержимое строки
    for k:=1 to i do
     begin
      read(f,cTemp);
      write(cTemp);
     end;
    writeln;
   end;
  close(f);
end.
```

Результат работы программы:

```
Количество символов Содержимое строки
                 16  Я$‹*ъ®†ћћМсдШ.Ж2
                 11  юЖ,х*Jv‰єжљ
                 15  (щ<л}7Эґ0њu>©Yx
                 18  †;vbЄљMLд~ФЪ4Ў_ИмУ
                 10  SФш{чи®n,q
                  6  ПСТґ"ц
```

Контрольные вопросы

1. Какие файлы называют нетипизированными?

2. Преимущества использования нетипизированных файлов.

3. Является ли нетипизированный файл файлом прямого доступа?

4. Возможно ли в нетипизированных файлах одновременно использовать операций чтения и записи? Если да, то почему?

5. Какая единица измерения длины записи используется в нетипизированных файлах?

6. При работе с нетипизированными файлами могут ли применяться процедуры и функции, доступные типизированным файлам?

Отчёт должен содержать:

1 Выполнение примеров задач в среде PascalABC.NET в консольном и оконном видах.

2. Ответы на вопросы.

ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ DELPHI. АНИМАЦИИ ИЗОБРАЖЕНИЙ

Цель: выработать навыки по работе в системе программирования PascalABC.NET с использованием модуля GraphABC.

Модуль GraphABC представляет собой простую графическую библиотеку. Предназначен для создания не событийных графических и анимационных программ в процедурном и частично в объектном стиле. Рисование осуществляется в специальном графическом окне, возможность рисования в нескольких окнах отсутствует. Кроме этого, в модуле GraphABC определены простейшие события мыши и клавиатуры, позволяющие создавать элементарные событийные приложения. Основная сфера использования модуля GraphABC - обучение.

Инициализация окна:

procedure SetWindowSize(w,h: integer); Устанавливает размеры клиентской части графического окна в пикселах.

procedure SetWindowTitle(s: string); Устанавливает заголовок графического окна.

Рассмотрим некоторые полезные процедуры графического окна. Остальные процедуры и функции работы с окном можно посмотреть в справочной информации Помощь-Справка-Стандартные модули-Модуль GraphABC- Подпрограммы для работы с графическим окном:

procedure ClearWindow; Очищает графическое окно белым цветом.

procedure ClearWindow(c: Color); // Очищает графическое окно цветом c. Цветовых констант достаточно много, около 136 оттенков. Основные цвета clBlue- синий, clGreen-зеленный и т.д.

Есть полезная функция, которая возвращает случайный цвет:

function clRandom: Color; //Возвращает случайный цвет

procedure CenterWindow; //Центрирует графическое окно по центру экрана

procedure Sleep(ms: integer); //Делает паузу на ms миллисекунд

procedure CloseWindow; //Закрывает графическое окно и завершает приложение.

Задача 1. Инициализировать графическое окно размером 800*600 с заголовком «Первая графическая программа». Разместить его по центру и закрасить синим цветом.

Решение:

```
uses GraphABC;
var w,h:integer;
begin
  w:=800;
  h:=600;
  SetWindowSize(w,h);
  CenterWindow;
  SetWindowTitle( 'Первая графическая программа');
  ClearWindow(clBlue);
end.
```

Задача 2. Закрасить графическое окно 20 раз случайным цветом с паузой 300 ms.

Решение:

```
uses GraphABC;
var w,h,i:integer;
begin
  w:=800;
  h:=600;
  SetWindowSize(w,h);
  CenterWindow;
  SetWindowTitle( 'Первая графическая программа');
  For i:=1 to 20 do
    begin
      ClearWindow(clRandom);
      Sleep(300);
    end;
end.
```

Задача 3 Уменьшение окна по ширине и высоте на 200 пикселей. Используйте процедуры справочной системы.

Решение:

```
uses GraphABC;
var w,h,i:integer;
begin
  w:=800;
  h:=600;
  SetWindowSize(w,h);
  CenterWindow;

  SetWindowTitle( 'Первая графическая программа');
  For i:=1 to 200 do
    begin
      SetWindowWidth(w-i);
      SetWindowHeight(h-i);
      Sleep(30);
    end;
end.
```

Графические примитивы

Графические примитивы представляют собой процедуры, осуществляющие рисование в графическом окне. Рисование осуществляется текущим пером (линии), текущей кистью (заливка замкнутых областей) и текущим шрифтом (вывод строк).

Данных процедур и функций достаточно много. Их можно посмотреть в разделе Помощь-Справка-Стандартные модули- Модуль GraphABC- Графические примитивы. Вот некоторые, которые мы будем использовать в своих задачах:

procedure SetPixel(x,y: integer; c: Color); // Закрашивает пиксел с координатами (x,y) цветом c

procedure MoveTo(x,y: integer); // Устанавливает текущую позицию рисования в точку (x,y)

procedure LineTo(x,y: integer); // Рисует отрезок от текущей позиции до точки (x,y). Текущая позиция переносится в точку (x,y)

procedure Line(x1,y1,x2,y2: integer); // Рисует отрезок от точки (x1,y1) до точки (x2,y2)

procedure DrawCircle(x,y,r: integer); // Рисует окружность с центром (x,y) и радиусом r

procedure DrawRectangle(x1,y1,x2,y2: integer); // Рисует границу прямоугольника, заданного координатами противоположных вершин (x1,y1) и (x2,y2)

procedure Circle(x,y,r: integer); // Рисует заполненную окружность с центром (x,y) и радиусом r

procedure Rectangle(x1,y1,x2,y2: integer); // Рисует заполненный прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2)

procedure FloodFill(x,y: integer; c: Color); // Заливает область одного цвета цветом c, начиная с точки (x,y).

procedure TextOut(x,y: integer; s: string); // Выводит строку s в прямоугольник к координатами левого верхнего угла (x,y).

Если необходимо вывести числовое значение, то вместо строки s можно поставить переменную целого или вещественного типа.

Работа с точками

Задача 4. Закрасить точки случайного цвета в графическом окне 800*600. Количество точек 40000, координаты задаются случайным образом в пределах окна.

Решение:

```
uses GraphABC;
var w,h,i,x,y:integer;
Begin
  w:=800;
  h:=600;
  SetWindowSize(w,h);
  CenterWindow;
  SetWindowTitle( 'Точки');
  For i:=1 to 40000 do
    begin
      x:=random(800);
      y:=random(600);
      SetPixel(x,y,clRandom);
    end;
end.
```

Задача №5. Нарисовать прямую, состоящую из точек длиной в 300 пикселей параллельной оси 0X.

Решение:

Идея этой задачи реализована циклом с параметром, который выполняется 300 раз. В теле цикла изменяется координата X. Если изменять одновременно координату Y, создастся смещение прямой на некоторый угол.

```
uses GraphABC;
var w,h,i,x,y:integer;
Begin
  w:=800;
  h:=600;
  SetWindowSize(w,h);

  CenterWindow;
  SetWindowTitle( 'Прямая');
  For i:=1 to 300 do
    begin
      x:=i+200;
      y:=h div 2;
      putpixel(x,y,clBlue);
    end;
End.
```

Задача №6. Используя результат предыдущей задачи нарисовать закрашенный прямоугольник.

Идея решения задачи сводится к тому, чтобы рисовать линии одной длины и спускаться по координате Y на одну точку. Замените блок цикл с предыдущей задачи на блок представленный ниже. Блоки инициализации графики для всех программ идентичны, поэтому в дальнейшем мы их описывать не будем, а лишь сконцентрируем внимание на фрагменты решения самой задачи.

Решение:

```
for y:=100 to 300 do
    for x:=200 to 400 do
      begin
       putpixel(x,y, clBlue);
```

```
    end;
```

Нарисовать разноцветный прямоугольник.

Задача №7. Нарисовать из точек окружность радиуса R.

Решение:

Окружность имеет уравнение x2 + y2 = R2. Будем циклом пробегать все значения от –R до +R. Значение Y вычисляем дважды. Ордината Y используем для построения верхней дуги, а –Y для нижней.

Функция Round округляет полученное значение. При выводе точки смещаем окружность по оси X на 300 точек, по оси Y на 200 точек от начала координат (левого верхнего угла монитора).

```
uses GraphABC;
var w,h,i,x,y,r:integer;
Begin
  w:=800;

  h:=600;
  SetWindowSize(w,h);
  CenterWindow;
  SetWindowTitle( 'Окружность');
   r:=100;
    for x:=-r to r do
      begin
        y:=Round(sqrt(sqr(r)-sqr(x)));
        putpixel(x+300,y+200,ClRed);
        putpixel(x+300,-y+200,ClBlue);
      end;
End.
```

Задача №8. Нарисовать кольцо с внешним радиусом R1 и внутренним радиусом R2

Задача аналогична предыдущей.

Задача №9. Вывести точки случайным образом, случайного цвета в область кольца.

Решение:

Так как, данные кольца вписаны в квадрат со стороной 2*r1, то достаточно пройти вложенными циклами все точки квадрата построчно и выбрать при построении только те точки, которые удовлетворяют условию попадания в кольцо (x2+y2<=r12) and (x2+y2>=r22)

```
uses GraphABC;
var w,h,i,x,y,r1,r2:integer;
Begin
  w:=800;
  h:=600;
  SetWindowSize(w,h);
  CenterWindow;
  SetWindowTitle( 'Окружность');
  r1:=100;
  r2:=80;
    for x:=-r1 to r1 do
      for y:=-r1 to r1 do
        begin
          if ((sqr(x)+sqr(y))>=sqr(r2)) and ((sqr(x)+sqr(y))<=sqr(r1))
              then putpixel(x+300,y+200,ClRandom);

        end;
End.
```

Задача №7. Построить эффект, увеличивающийся с центра до краёв экрана шар.

Решение:

Алгоритм решения задачи построен на попадание случайной точки с координатами x,y в область увеличивающейся окружности. Введём величины:

r1– внешний радиус круга взрыва;

r – увеличивающийся радиус взрыва;

n – количество случайных точек, чем больше r, тем больше точек выводим на экран;

Sleep(2); – задержка вывода на экран.

Функция random генерирует случайное положительное число, то вывод точки происходит только в положительную область декартовой

системы координат. Так как точки симметричны относительно системы координат, вместо вывода одной точки выводим 4 точки.

```
uses GraphABC;
var w,h,i,x,y,r,r1,r2,n:integer;
Begin
  w:=800;
  h:=600;
  SetWindowSize(w,h);
  CenterWindow;
  SetWindowTitle( 'Взрыв');
  r1:=200;
  r:=1;
   while r<=r1 do
     begin
       n:=1;
         while n<=r do
           begin
             x:=random(r);
             y:=random(r);
             if sqr(x)+sqr(y)<=sqr(r)
                    then
                      begin
                        putpixel(x+300,y+200,clBlue);

                        putpixel(-x+300,-y+200,clBlue);
                        putpixel(-x+300,y+200,clBlue);
                        putpixel(x+300,-y+200,clBlue);
                      end;
              Sleep(2);
              n:=n+1;
           end;
        r:=r+1;
     end;
End.
```

Задачи для самостоятельного решения.

1. Запросить номер координатной плоскости и закрасить ее точками.

2. Нарисовать прямоугольник в центре экрана под углом 45 градусов к осям координат, состоящий из 4-х прямых.

3. С использованием модуля GraphABC в PascalABC.NET разработать принципиальную схему одно трансформаторной подстанции.

4. Создать приложение, которое по заданным за равные промежутки времени значениям расхода электроэнергии строит суточный график электрической нагрузки для объекта. В расчётах, выполняемых в приложении должно учитываться и коэффициент трансформации. Исходные данные принять из примера дисциплины «Основы алгоритмизации и программирование». При разработке пользовательского интерфейса руководствоваться материалами лекции.

5. Для предыдущей задачи разработать UML-диаграмму последовательности и диаграмму переходов состояний интерфейса приложения (см. материалы лекции).