Калужский филиал ПГУПС

Методическая разработка

По дисциплине

«Информатика»
для специальности 08.02.10

Выполнили преподаватель
Мазина И.В.

2017

Содержание

1.Введение

В учебной программе по дисциплине «Информатика и ВТ» предусмотрены раздел с практическими занятиями по программированию на каждой специальности первого курса.

В настоящее время на занятиях по дисциплине мы рассматривали язык программирования Basic, который работал на ОС версий Windows97-2000, XP. При переходе на ОО версий Windows7 и выше, пришла необходимость подобрать язык программирования одновременно и простой по синтаксису языка, достаточно простой для студентов, начинающих программирование и работающий на данных версиях ОС. Мы выбрали язык программирования PascalABC: он является свободно распространяемым ПО, широко используемым в обучении студентов программированию и есть возможность при наличии Интернета в компьютерном классе , не ставить ПО на каждый ПК, а использовать для программирования WEB- среду разработки программ (http://pascalabc.net/WDE/)

2.Ключевые особенности PascalABC.NET

- Ряд расширений языка Pascal, в числе которых оператор foreach, внутриблочные описания переменных, автоопределение типа при описании, встроенные множества произвольных типов, case по строкам, упрощенный синтаксис модулей, методы в записях, операция new для создания объектов, определение тел методов внутри классов, целые произвольной длины, многомерные динамические массивы.
- Самые современные средства языков программирования: обобщенные классы и подпрограммы, интерфейсы, перегрузка операций, λ-выражения, исключения, сборка мусора, методы расширения, безымянные классы, автоклассы.
- Генерация эффективного кода для платформы .NET.
- Высокая совместимость с Delphi.
- Высокая скорость выполнения программ.
- Возможность доступа к огромному количеству .NET-библиотек от контейнерных классов до средств работы с сетью.
- Среда разработки с встроенным отладчиком, обеспечивающая подсказки по коду, переход к определению и реализации подпрограммы, шаблоны кода, автоформатирование кода.
- Встроенный в среду разработки дизайнер форм для быстрого создания оконных приложений.
- Простая и эффективная растровая графическая библиотека.
- Средства параллельного программирования в виде директив OpenMP.
- Встроенный электронный задачник Programming Taskbook.
- Модули исполнителей Робот и Чертежник, используемых в школьной информатике.
- Механизм проверяемых заданий, обеспечивающий автоматическую постановку и проверку заданий.
- Наличие Web-среды разработки (WDE), позволяющей запустить программу прямо из окна браузера.

- Возможность опубликовать в интернете ссылку на файл, сохраненный в Web-среде разработки.
- Возможность запуска консольного компилятора под Mono в современных версиях Linux, возможность встраивания PascalABC.NET в редактор Geany.

Система PascalABC.NET является совместной разработкой российских и немецких программистов. В России центр разработки находится в институте математики, механики и компьютерных наук Южного федерального университета.

PascalABC.NET активно используется в ряде средних и высших учебных заведений России и ближнего зарубежья. Так, на мехмате Южного федерального университета он используется для обучения 1 курса направления Информационные технологии в курсе Основы программирования, а также для обучения школьников в одной из самых больших в России Детской компьютерной школе.

Простота, современные возможности, свободный код — вот главные достоинства PascalABC.NET

3.Методические указания для выполнения практических занятий

Для выполнения практических работ нужно запустить WEB- среду разработки программ. Можно запустить главную станицу сайта http://pascalabc.net/ и воспользоваться справкой (см. левое меню) по языку программирования и примерами (кнопка «ПРИМЕРЫ») по различным темам. Этот сайт можно использовать для внеаудиторной работы студентов. Для работе в Web-среде разработки программ, перейдите по кнопке «WDE»

3.1.Структура программы

(http://pascalabc.net/downloads/pabcnethelp/index.htm)

Программа содержит ключевые слова, идентификаторы, комментарии. Ключевые слова используются для выделения синтаксических конструкций и подсвечиваются жирным шрифтом в редакторе. Идентификаторы являются именами объектов программы и не могут совпадать с ключевыми словами.

Программа на языке **PascalABC.NET** имеет следующий вид:

program *имя программы*;

раздел **uses**

раздел описаний

begin

операторы

end.

Первая строка называется ***заголовком программы*** и не является обязательной.

Раздел **uses** состоит из нескольких подряд идущих секций **uses**, каждая из которых начинается с ключевого слова **uses**, за которым следует список имен модулей и пространств имен .NET, перечисляемых через запятую.

Раздел описаний может включать следующие подразделы:

- раздел описания переменных
- раздел описания констант
- раздел описания типов
- раздел описания меток
- раздел описания процедур и функций

Данные подразделы следуют друг за другом в произвольном порядке.

Далее следует блок **begin**/**end**, внутри которого находятся операторы, отделяемые один от другого символом "точка с запятой". Среди операторов может присутствовать оператор описания переменной, который позволяет описывать переменные внутри блока.

Раздел **uses** и раздел описаний могут отсутствовать.

Например:

```
program MyProgram;
var
  a,b: integer;
  x: real;
begin
  readln(a,b);
  x := a/b;
  writeln(x);
end.
```

или

```
uses GraphABC;
begin
  var x := 100;
  var y := 100;
  var r := 50;
  Circle(x,y,r);
end.
```

3.2.Выражения и операции

Выражение - это конструкция, возвращающая значение некоторого типа. Простыми выражениями являются переменные и константы. Более сложные выражения строятся из простых с помощью операций, вызовов функций и скобок. Данные, к которым применяются операции, называются *операндами*.

В **PascalABC.NET** имеются следующие операции: @, **not**, ^, *, /, **div**, **mod**, **and**, **shl**, **shr**, +, -, **or**, **xor**, =, >, <, <>, <=, >=, **as**, **is**, **in**, а также операция new и операция приведения типа. Операции @, -, +,^, **not**, операция приведения типа и операция new являются унарными (имеют один операнд), остальные являются бинарными (имеют два операнда), операции + и - являются и бинарными и унарными.

3.3.Обзор типов

Типы в **PascalABC.NET** подразделяются на простые, структурированные, типы указателей, процедурные типы, последовательности и классы.

К ***простым*** относятся целые и вещественные типы, логический, символьный, перечислимый и диапазонный тип.

Тип данных называется ***структурированным***, если в одной переменной этого типа может содержаться множество значений.

К структурированным типам относятся массивы, строки, записи, кортежи, множества, файлы и классы.

Особым типом данных является последовательность, которая хранит по-существу алгоритм получения данных последовательности один за другим.

Все простые типы, кроме вещественного, называются ***порядковыми***. Только значения этих типов могут быть индексами статических массивов и параметрами цикла **for**. Кроме того, для порядковых типов используются функции Ord, Pred и Succ, а также процедуры Inc и Dec.

Все типы, кроме типов указателей, являются производными от типа Object. Каждый тип в **PascalABC.NET** имеет отображение на тип .NET. Тип указателя принадлежит к неуправляемому коду и моделируется типом void*.

Все типы в **PascalABC.NET** подразделяются на две большие группы: ***размерные*** и ***ссылочные***.

3.4.Оператор присваивания

Оператор присваивания имеет вид:

переменная := *выражение*

В качестве переменной может быть простая переменная, разыменованный указатель, переменная с индексами или компонент переменной типа запись. Символ := называется значком присваивания. Выражение должно быть совместимо по присваиванию с переменной.

Оператор присваивания заменяет текущее значение переменной значением выражения.

Например:

```
i := i + 1; // увеличивает значение переменной i на 1
```

В **PascalABC.NET** определены также операторы присваивания со значками +=, -=, *=, /=. Для числовых типов действие данных операторов описано здесь. Кроме того, использование операторов += и *= для строк описано здесь и операторов +=, -= и *= для множеств - здесь. Их действие для процедурных переменных описано здесь.

Операторы +=, -=, *=, /= имеют следующий смысл: a #= b означает a := a # b, где # - знак операции +, -, *, /.

Например:

a += 3; // увеличить a на 3

b *= 2; // увеличить b в 2 раза

Оператор /= неприменим, если выражение слева - целое.

Операторы +=, -=, *=, /= могут также использоваться со свойствами классов соответствующих типов в левой части.

3.4.Оператор описания переменной

В **PascalABC.NET** можно описывать переменные внутри составного оператора begin-end в специальном операторе описания переменной. Такие описания называются внутриблочными.

Внутриблочное описание имеет одну из форм:

var *список имен*: *тип*;

или

var *имя*: *тип* := *выражение*;

или

var *имя*: *тип* = *выражение*; // Для совместимости с Delphi

или

var *имя* := *выражение*;

Имена в списке перечисляются через запятую. Например:

```
begin
  var a1,a2,a3: integer;
  var n: real := 5;
  var s := ' ';
  ...
end.
```

3.5.Составной оператор

Составной оператор предназначен для объединения нескольких операторов в один. Он имеет вид:

```
begin
  операторы
end
```

В **PascalABC.NET** составной оператор также называется ***блоком***. (традиционно в Паскале блоком называется раздел описаний, после которого идет составной оператор; в **PascalABC.NET** принято другое решение, поскольку можно описывать переменные непосредственно внутри составного оператора).

Операторы отделяются один от другого символом ";". Ключевые слова **begin** и **end**, окаймляющие операторы, называются ***операторными скобками***.

Например:

```
s := 0;
p := 1;
for var i:=1 to 10 do
begin
  p := p * i;
  s := s + p
end
```

Перед **end** также может ставиться ";". В этом случае считается, что последним оператором перед **end** является пустой оператор, не выполняющий никаких действий.

Помимо операторов, в блоке могут быть внутриблочные описания переменных:

```
begin
  var a,b: integer;
  var r: real;
  readln(a,b);
  x := a/b;
```

```
  writeln(x);
end.
```

3.6.Оператор цикла for

Оператор цикла **for** имеет одну из двух форм:

for *переменная* := *начальное значение* **to** *конечное значение* **do**

оператор

или

for *переменная* := *начальное значение* **downto** *конечное значение* **do**

оператор

Кроме того, переменную можно описать непосредственно в заголовке цикла:

for *переменная*: *тип* := *начальное значение* **to** *или* **downto** *конечное значение* **do**

оператор

или

for var *переменная* := *начальное значение* **to** *или* **downto** *конечное значение* **do**

оператор

В последнем случае используется ***автоопределение типа*** переменной по типу начального значения. В двух последних случаях область действия объявленной переменной распространяется до конца тела цикла, которое в данном случае образует неявный блок. Вне тела цикла такая переменная недоступна, поэтому следующий цикл может использовать переменную с тем же именем:

```
for var i := 1 to 10 do
  Print(i);
for var i := 1 to 5 do
  Print(i*i);
```

Текст от слова **for** до слова **do** включительно называется ***заголовком цикла***, а оператор после **do** - ***телом цикла***. Переменная после слова **for** называется ***параметром цикла***. Для первой формы цикла с ключевым словом **to** параметр цикла меняется от начального значения до

конечного значения, увеличиваясь всякий раз на единицу, а для второй формы ключевым словом **downto** - уменьшаясь на единицу. Для каждого значения переменной-параметра выполняется тело цикла. Однократное повторение тела цикла называется ***итерацией цикла***. Значение параметра цикла после завершения цикла считается неопределенным.

Переменная-параметр цикла может иметь любой порядковый тип. При этом начальное и конечное значения должны быть совместимы по присваиванию с переменной-параметром цикла.

Например:

```
var en: (red,green,blue,white);
...
for en := red to blue do
  write(Ord(en):2);
for var c := 'a' to 'z' do
  write(c);
```

Если для цикла **for** ... **to** начальное значение переменной цикла больше конечного значения или для цикла **for** ... **downto** начальное значение переменной цикла меньше конечного значения, то тело цикла не выполнится ни разу.

Если цикл используется в подпрограмме, то переменная-параметр цикла должна быть описана как локальная. Наилучшим решением в PascalABC.NET является описание переменной в заголовке цикла.

Изменение переменной-параметра цикла внутри цикла является логической ошибкой. Например, следующий фрагмент со вложенным оператором **for** является ошибочным:

```
for i := 1 to 10 do
  i -= 1;
```

3.7.Операторы цикла while и repeat

Оператор цикла **while** имеет следующую форму:

while *условие* **do**

оператор

Условие представляет собой выражение логического типа, а оператор после **do** называется *телом цикла*. Перед каждой итерацией цикла условие вычисляется, и если оно истинно, то выполняется тело цикла, в противном случае происходит выход из цикла.

Если *условие* всегда оказывается истинным, то может произойти *зацикливание*:

```
while 2>1 do
  write(1);
```

Оператор цикла **repeat** имеет следующую форму:

repeat

операторы

until *условие*

В отличие от цикла **while**, условие вычисляется после очередной итерации цикла, и если оно истинно, то происходит выход из цикла. Таким образом, операторы, образующие тело цикла оператора**repeat**, выполняются по крайней мере один раз.

Обычно оператор repeat используют в ситуациях, где условие нельзя проверить, не выполнив тело цикла. Например:

```
repeat
  read(x);
until x=0;
```

Если *условие* всегда оказывается ложным, то может произойти *зацикливание*:

```
repeat
  write(1);
until 2=1;
```

3.8.Условный оператор

Условный оператор имеет *полную* и *краткую* формы.

Полная форма условного оператора выглядит следующим образом:

if *условие* **then** *оператор1*

else *оператор2*

В качестве условия указывается некоторое логическое выражение. Если условие оказывается истинным, то выполняется *оператор1*, в противном случае выполняется *оператор2*.

Краткая форма условного оператора имеет вид:

if *условие* **then** *оператор*

Если условие оказывается истинным, то выполняется *оператор*, в противном случае происходит переход к следующему оператору программы.

В случае конструкции вида

if *условие1* **then**

  if *условие2* **then** *оператор1*

  else *оператор2*

else всегда относится к ближайшему предыдущему оператору **if**, для которого ветка **else** еще не указана. Если в предыдущем примере требуется, чтобы **else** относилась к первому оператору **if**, то необходимо использовать составной оператор:

```
if условие1 then
begin
  if условие2 then оператор1
end
else оператор2
```

Например:

```
if a<b then
  min := a
else min := b;
```

3.9.Практичекие занятия для студентов первого курса

Практическое занятие 4.

Тема: Решение типовых задач по линейному алгоритму.

Цель: Ознакомиться с решением задач по линейному алгоритму и командами, реализующими эти алгоритмы.

Теория:

Программа на Паскале состоит из последовательности операторов записанных в строки. Начинается программа с определения переменных, используемых в программе.

Основные типы переменных **в PascalABC.NET:**

- **Integer-**целочисленные типы. Любое число от -2147483648 до 2147483647
- **Real** - вещественный тип. Любое число от $-1.8 \cdot 10^{308}$ до $1.8 \cdot 10^{308}$.
- **Char** - символьный тип. Может содержать 1 любой символ.
- **String** - строковый тип. Может содержать произвольное кол-во символов.

Объявление переменных

Все переменные в PascalABC.NET объявляются в секции **var**, например:

```
1    var
2      i: integer;
3
4    begin
5      // Пишем саму программу
6    end.
```

Также можно объявить сразу три переменных:

```
1    var
2      a, b, c: integer;
```

Текст программы начинается словом **begin**, а заканчивается - **end.** (с точкой в конце слова)

Каждая каждый оператор заканчивается точкой с запятой **(;)**

Комментарий начинается с символов **//**

Оператор является основным элементом языка и описывает действие, которое необходимо для решения задач.

а) D = 2 * R – диаметр круга

б) L = 2 * PI * R – длина окружности

в) $S = PI * R^2$ – площадь круга

г) $P = 4 * PI * R^2$ – площадь поверхности сферы

д) $V = 4/3 * PI * R^3$ – объем шара.

Ход работы:

Задание 1. 1. Написать программу и ввести ее в компьютер.

2. Запустить браузер, перейти на страницу http://pascalabc.net/ , далее по кнопке WDE или http://pascalabc.net/WDE/ - запустить Паскаль АВС он-лайн.

Проанализировать получения решения.

Для возведения в степень воспользуйтесь функцией ***POWER (x,y), где x-число, y-степень***

```
// ПЗ_4 Задание 1 Выполнил студент гр КААТ-311 Иванов Петя
var r,d,l,s,p,v,pi: real;

begin
```

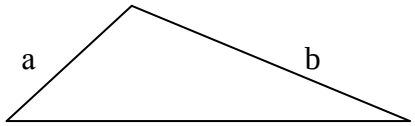
```
 write('Введите R: ');

  read(R);  // ввод радиуса
  pi:=3.11415926; // присвоение значение пи
  d := 2*r; // вычисление диаметра
  l := pi*d; //...длины окружности
  s := pi*power(r,2); //... площади....
  p := 4*pi*power(r,2); //... поверхности....
  v := 4/3*pi*power(r,3); //... объема...

 writeln('Параметры круга'); // чем отличается от оператора write?
  write('R= ',r);
  writeln(' см');
  write('D= ',d);
  writeln(' см');
  write('L= ',l);
  writeln(' см');
  write('S= ',s);
  writeln(' кв.см');
  writeln('Параметры сферы');
  write('P= ',p);
  writeln(' кв.см');
  write('V= ',v);
  writeln(' куб.см');

end.
```

Задание 2. Вычислить площадь треугольника по формуле Герона.



$$S=\sqrt{p*(p-a)*(p-b)*(p-c)}$$

$p=(a+b+c)/2$

```
// ПЗ_4 Задание 2 Выполнил студент гр КААТ-311 Иванов Петя
var a,b,c,p,s: real;

begin
  write('Введите a: ');
  readln(a);
   write('Введите b: ');
  readln(b);
   write('Введите c: ');
  readln(c);
  p:=(a+b+c)/2;  //вычисление периметра
  s:= sqr(p*(p-a)*(a-b)*(p-c));

  writeln('Площадь треугольника');

  write('s= ',s);

end.
```

Вывод:

Практическое занятие 5

Тема: Решение задач по разветвляющемуся алгоритму.

Цель: Ознакомиться с оператором, реализующим разветвляющийся алгоритм.

Теория: Для реализации различных типов переходов служат специальные операторы передачи управления

IF условие THEN A1 ELSE A2;

A1, A2 – некоторые операторы

Если условие соблюдается, то управление передается оператору под номером m, иначе выполняется оператор рядом стоящий в программе. Это оператор условного перехода в сокращенной форме. Можно пользоваться оператором условного перехода в полной форме если

В качестве условий принимается: A>B, A<>B, A<=B, A>=B, A<B, A>B, где A и B некоторые арифметические выражения.

Ход работы:

Задание 1. Вычислить значение функции.

1, 5, 9, 13	2, 6, 10, 14	3, 7, 11, 15	4, 8, 12
$Y=\begin{cases} x^2-4, \text{ если } x<5, \\ \frac{2}{x}, \text{ если } x\geq 5 \end{cases}$	$Y=\begin{cases} x^3+7, \text{если } x\leq 0, \\ 3-4x, \text{если } x>0 \end{cases}$	$Y=\begin{cases} 3x^2-4x, \text{ если } x>-1, \\ \frac{1}{2-3x}, \text{если } x\leq -1 \end{cases}$	$Y=\begin{cases} \frac{2}{x}-4, \text{ если } x<0 \\ 2x^3+3, \text{ если } x\geq 0 \end{cases}$

- Вычислить значение Y, если x=номер варианта -7
- Выведите значение Y в одну строку с надписью 'Значение функции Y = '
- Для дробных значений Y измените оператор write(y) на write(y:4:2);

Пример

```
var x,y:real;

 begin
  write('Введите x: ');

  read(x);
 writeln('Значение функции Y = ');
  if x<3 then y:= power(x,5)+0.5
  else y:=1/(x*5+3);
  write(y);
```

end.

Задание 2. Вычислить корни квадратного уравнения.

$\mathbf{ax^2 + bx + c = 0,\ a \neq 0}$

Практическое занятие 6

Тема: Решение задач по циклическому алгоритму.

Цель: Ознакомиться с операторами, реализующими циклический алгоритм.

Теория: При решении многих задач возникает необходимость повторения одних и тех же действий, но над различными значениями переменной. Такие вычисления названы **циклическими**, а многократное повторение участка – **циклами**.

Циклические алгоритмы можно реализовать с помощью оператора условного перехода **IF.** Однако существуют операторы специального предназначения для составления программ циклических алгоритмов-оператор **For**.

Формат оператора

for *переменная* := *начальное значение* **to** *конечное значение* **do**
оператор

Шаг в этом случае равен 1.

Если шаг не равен единице, используется оператор **While.** Если будет насколько оператора, то используется конструкция **begin …end**, как скобки.

Формат оператора

while *условие* ***do***
оператор

вар	функция	диапазон	шаг
1	$Y=-2x+3$	-2;2	0.3
2	$T=6x^2-2$	-3;1	0.2
3	$W=3x-7$	2;5	0.25
4	$U=5\ (2-x)$	0;4	0.5
5	$A=4+2x$	-1;1	0.1
6	$B=x^4-2$	-2;1	0.23
7	$C=(2x-7)^3$	1;5	0.4
8	$D=(x-4)^2+2$	2;6	0.42
9	$E=2-x^3$	-2;3	0.5
10	$F=3(5-x^2)$	-3;2	0.6
11	$G=3(5-x)^2$	-1;4	0.7
12	$H=72-3x^3$	-10;10	2
13	$K=2(5x+3)^2$	-5;-1	0.4
14	$Z=-7x^3+2$	-2;3	0.32

Ход работы:

1. Написать программу по циклическому алгоритму
2. Ввести программу в компьютер.
3. Отладить текст программы, выполнить программу.
4. Записать в отчет программу, результаты ее выполнения

Задание 1.

Вычислить значение функции согласно варианту с оператором For и шагом =1.
Вычислить значение функции согласно варианту.

функция	диапазон	шаг
$Y=-0,5x-3$	-2;5	0.35

ПРИМЕР

Вычислить значение функции

```
Var
  x: Real;
BEGIN
x:=-2;
While x<=5 do
begin
  Writeln('X=',x,'  Y=',-0.5*x-3); // попробуйте задать формат выводимого числа как в ПЗ_5
  x:=x+0.35;
end;
END.
```

Задание 2.

Вычислить произведение всех натуральных чисел от 1 до n.

Практическое занятие №7

Тема: Решение задач с сортировкой данных в заданном массиве.

Цель: Изучить простейшие программы с сортировкой данных в заданном массиве.

Теория: Многие задачи, которые решаются с помощью ПЭВМ, связаны с обработкой больших объемов информации, представляющей совокупность данных, объединенных единым математическим содержанием или связанных между собой по смыслу.

В программе для представления используют массивы.

Массив – это упорядоченная совокупность однотипных данных, с каждым из которых связан упорядоченный набор целых чисел, называемых индексами.

Будем рассматривать одномерные и двумерные массивы.

Массив характеризуется

- Именем
- Размерностью
- Размером.

Индексы определяют положение элементов массива.

Число индексов определяет размерность массива. Размер – это количество элементов в массиве.

Массив определяется следующим образом: **A: array [1..n] of integer (real, и т.д),**

где А– имя массива, n- размерность массива

Ход работы:

1. Написать программу по обработке массивов.
2. Ввести в компьютер и выполнить.
3. Сделать анализ результатов.

Задание 1. Пусть дан одномерный массив

0	1	2	3	4	5 – номера элементов
2	4	-3	5	7	-1 – значения элементов

Ввести в память компьютера значения элементов и вывести на экран в виде A(0) = 2, A(1) = 4 и т.д.

```
var
a: array [1..5] of real;  i: integer;

begin

  write('Введите элементы массива: ');
  for i:=1 to 5 do
    read(a[i]);
  writeln('Вывод элементов массива: ');
  for i:=1 to 5 do
    writeln('a(',i,')= ',a[i]);
end.
```

Воспользуйтесь кнопкой **Примеры** в среде разработки и найдите раздел **Массивы.**

Задание 2. Найти сумму элементов массива.

Вывод:

Для преподавателя- обработка двумерного массива

```
var
  A: array [1..3,1..3] of real;
  i,j: integer;

begin

  write('Введите элементы массива: ');
  for i:=1 to 3 do
  for j:=1 to 3 do begin
    read(a[i,j]);
    end;
  writeln('Вывод элементов массива: ');
  for i:=1 to 3 do
    for j:=1 to 3 do begin
    writeln('a(',i,';',j,')= ',a[i,j]);
    end;
end.
```

4.ЛИТЕРАТУРА

1. http://pascalabc.net/ - главная страница
2. http://pascalabc.net/WDE/ -WEB- среда разработки программ