МИНОБРНАУКИ РОССИИ

Орский гуманитарно-технологический институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования «Оренбургский государственный университет»
(Орский гуманитарно-технологический институт (филиал) ОГУ)

Кафедра программного обеспечения

Методические указания
для обучающихся по освоению дисциплины

«Б.1.Б.14 Программирование»

Уровень высшего образования

БАКАЛАВРИАТ

Направление подготовки
09.03.03 Прикладная информатика
(код и наименование направления подготовки)

Прикладная информатика в экономике
(наименование направленности (профиля) образовательной программы)

Тип образовательной программы
Программа академического бакалавриата

Квалификация
Бакалавр

Форма обучения
Очная

Год начала реализации программы (набора)
2014, 2015, 2016

г. Орск 2017

Методические указания для обучающихся по освоению дисциплины «Б.1.Б.14 Программирование» предназначены для обучающихся очной формы обучения направления подготовки 09.03.03 Прикладная информатика, профиля «Прикладная информатика в экономике»

Составитель _____ О.В. Подсобляева

Методические указания рассмотрены и одобрены на заседании кафедры программного обеспечения, протокол № 9 от «07» июня 2017 г.

Заведующий кафедрой программного обеспечения _____ Е.Е.Сурина

1 Методические указания по проведению лекционных занятий

Лекционные занятия в высшем учебном заведении являются основной формой организации учебного процесса и должны быть нацелены на выполнение ряда задач:

- ознакомить студентов со структурой дисциплины;
- изложить основной материал программы курса дисциплины;
- ознакомить с новейшими подходами и проблематикой в данной области;
- сформировать у студентов потребность к самостоятельной работе с учебной, нормативной и научной литературой.

Лекционное занятие представляет собой систематическое, последовательное, монологическое изложение преподавателем-лектором учебного материала, как правило, теоретического характера.

Цель лекции – организация целенаправленной познавательной деятельности студентов по овладению программным материалом учебной дисциплины.

Чтение курса лекций позволяет дать связанное, последовательное изложение материала в соответствии с новейшими данными науки, сообщить слушателям основное содержание предмета в целостном, систематизированном виде.

В ряде случаев лекция выполняет функцию основного источника информации, когда новые научные данные по той или иной теме не нашли отражения в учебниках.

Организационно-методической базой проведения лекционных занятий является рабочий учебный план направления подготовки. При подготовке лекционного материала преподаватель обязан руководствоваться учебными программами по дисциплинам кафедры, тематика и содержание лекционных занятий которых представлена в рабочих программах, учебно-методических комплексах.

При чтении лекций преподаватель имеет право самостоятельно выбирать формы и методы изложения материала, использовать различные технические средства обучения.

Рекомендации по работе студентов с конспектом лекций.

Изучение дисциплины студенту следует начинать с проработки рабочей программы, особое внимание, уделяя целям и задачам, структуре и содержанию курса.

При конспектировании лекций студентам необходимо излагать услышанный материал кратко, своими словами, обращая внимание, на логику изложения материла, аргументацию и приводимые примеры. Необходимо выделять важные места в своих записях. Если непонятны какие-либо моменты, необходимо записывать свои вопросы, постараться найти ответ на них самостоятельно. Если самостоятельно не удалось разобраться в материале, впоследствии необходимо либо на следующей лекции, либо на лабораторном занятии или консультации обратиться к ведущему преподавателю за разъяснениями.

Успешное освоение курса предполагает активное, творческое участие студента путем планомерной, повседневной работы. Лекционный материал следует просматривать в тот же день. Рекомендуемую дополнительную литературу следует прорабатывать после изучения данной темы по учебнику и материалам лекции.

Каждая тема имеет свои специфические термины и определения. Усвоение материала необходимо начинать с усвоения этих понятий. Если какое-либо понятие вызывает затруднения, необходимо посмотреть его суть и содержание в словаре (Интернете), выписать его значение в тетрадь для подготовки к занятиям.

При подготовке материала необходимо обращать внимание на точность определений, последовательность изучения материала, аргументацию, собственные примеры, анализ конкретных ситуаций. Каждую неделю рекомендуется отводить время для повторения пройденного материала, проверяя свои знания, умения и навыки по контрольным вопросам и тестам.

2 Методические указания по лабораторным и практическим работам

Изучение дисциплины «Программирование» предполагает посещение обучающимися не только лекций, но и лабораторных работ. Лабораторные работы со студентами предназначены для проверки усвоения ими теоретического материала дисциплины.

Основные цели лабораторных работ:

- закрепить основные положения дисциплины;
- проверить уровень усвоения и понимания студентами вопросов, рассмотренных на лекциях и самостоятельно изученных по учебной литературе;
- научить пользоваться нормативной и справочной литературой для получения необходимой информации о конкретных технологиях;
- оказать помощь в приобретении навыков расчета точностных характеристик;
- восполнить пробелы в пройденной теоретической части курса и оказать помощь в его усвоении.

Для контроля знаний, полученных в процессе освоения дисциплины на лабораторных занятиях обучающиеся выполняют задания реконструктивного уровня и комплексное практическое задание.

Целью выполнения задания реконструктивного уровня и комплексного задания студентами является систематизация, закрепление и расширение теоретических знаний, полученных в ходе изучения дисциплины.

Ниже приводятся общие методические указания, которые относятся к занятиям по всем темам:

- в начале каждого лабораторного занятия необходимо сформулировать цель, поставить задачи;
- далее необходимо проверить знания студентами лекционного материала по теме занятий;
- в процессе занятия необходимо добиваться индивидуальной самостоятельной работы студентов;
- знания студентов периодически контролируются путем проведения текущей аттестации (рубежного контроля), сведения о результатах которой доводятся до студентов и подаются в деканат;
- время, выделенное на отдельные этапы занятий, указанное в рабочей программе, является ориентировочным; преподаватель может перераспределить его, но должна быть обеспечена проработка в полном объеме приведенного в рабочей программе материала;
- на первом занятии преподаватель должен ознакомить студентов с правилами поведения в лаборатории и провести инструктаж по охране труда и по пожарной безопасности на рабочем месте;
- преподаватель должен ознакомить студентов со всем объемом лабораторных работ и требованиями, изложенными выше;
- преподаватель уделяет внимание оценке активности работы студентов на занятиях, определению уровня их знаний на каждом занятии.

На лабораторных работах решаются задачи из всех разделов изучаемой дисциплины.

Лабораторная работа №1
Среда Turbo Pascal

Для входа в систему Турбо-Паскаль на рабочем столе Windows выберите пиктограмму (значок) с надписью Turbo Pascal… или Borland Pascal… и дважды щелкните левой кнопкой «мыши». Если на вашем компьютере такого значка нет, тогда сначала загрузите NC (или подобную ей оболочку) и произведите поиск файла BP.EXE или TURBO.EXE (обычно Alt-F7), затем запустите найденный файл. Перед вами появится соответствующий интерфейс (рис. 1). Если на экране появилась какая-то «не ваша» программа, ее можно закрыть Alt-F3.

Переключаться между окнами-программами можно клавишей F6, распахивать окно или возвращать его размеры - F5.

Первоначально текст программы имеет имя NONAME00. Но это имя можно изменить. Сохраните текст программы на диске под именем T01.PAS. Для этого с помощью клавиш Alt + F перейдите в меню, выберите пункт "Save as…" (записать под новым именем) и в окне ввода задайте имя программы T01.PAS.

Примечание То же самое можно выполнить, войдя в главное меню с помощью клавиши F10, выбрать пункт меню File, а затем "Save as…".

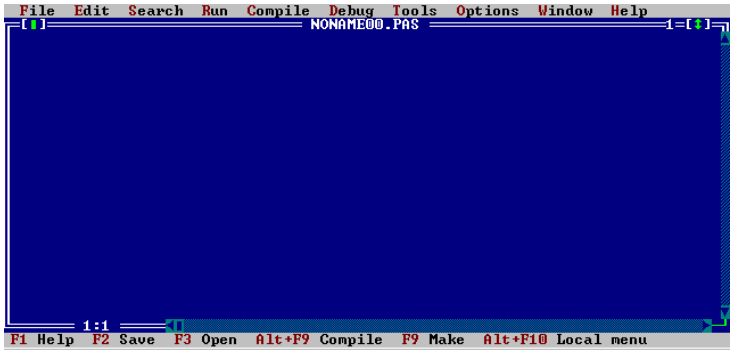

Рисунок 1. Среда Turbo Pascal

Выполнить программу можно, нажав Ctrl+F9.

Если программа содержит ошибки, то компиляция будет остановлена с выдачей соответствующего сообщения и установкой курсора в то место текста программы, где компилятор обнаружил ошибку:

В случае корректного выполнения результаты работы программы можно увидеть, переключившись в окно просмотра с помощью Alt+F5.

Для окончания работы с системой необходимо нажать Alt+X.

Структура Pascal-программы.

Программа на языке Паскаль всегда состоит из двух основных частей: описания данных, с которыми оперируют действия и описания последовательности самих действий, которые необходимо выполнить. Таким образом, Pascal-программа делится на следующие разделы:

заголовок программы
раздел описания констант
раздел описания типов

Разделы друг от друга отделяются точкой с запятой. Рассмотрим коротко основные из разделов.

1. **Заголовок программы имеет вид**

Program Program_Name; где Program_Name - имя программы.

Например,

Program Example;

Заголовок программы в языке Turbo Pascal является необязательным и никак не влияет на ее выполнение.

2. **В разделе описания констант** задаются имена, являющиеся в программе синонимами некоторых значений.

Const Const_Name = Const_Value; где Const_Name - имя, которому ставится в соответствие значение Const_Value.

При наличии более одной константы они разделяются точкой с запятой, например:

Const T='error'; This_Year=1996;

Теперь, если в программе встретятся имена T и This_Year, они заменятся на 'error' и 1996 соответственно.

3. **В разделе описания типов** программист может определять свои собственные типы данных, отличные от предоставляемых стандартных типов данных.

Type Type_Name = Type_Value; где Type_Name - имя определяемого типа Type_Value.

Например:

Type Days = (Mo, Tu, We, Th, Fr, Sa, Su); Letters = 'a'..'z';

4. В программе на языке Pascal любая переменная должна быть определена в **разделе описания переменных**.

Var Var_Name : Var_Type; где Var_Name - имя определяемой переменной, Var_Type - ее тип.

5. **Процедуры и функции** нами будут рассмотрены отдельно. Основное их назначение - обеспечить поддержку структурного программирования "сверху вниз", упростить и сделать компактнее программу.

6. **Раздел операторов** единственный является обязательным для Pascal-программы.

Begin
оператор1;
оператор2;
…
операторN;
End.

Раздел начинается служебным словом **Begin** и заканчивается **End** с точкой. В нем, как ясно из названия, находятся операторы, разделенные точкой с запятой. Перед служебнным словом "End" точка с запятой не ставится.

В языке Pascal имеются следующие виды операторов:

- оператор процедуры
- оператор присваивания
- условный оператор
- составной оператор
- оператор варианта
- оператор цикла с предусловием
- оператор цикла с постусловием
- оператор цикла с параметром
- присоединения
- пустой оператор

Стандартные арифметические функции

функция	Возвращаемые результат
Abs	абсолютное значение
Arctan	арктангенс
Cos	косинус
Pi	число "Пи"
Round	округление
Sin	синус
Sqr	квадрат
Sqrt	квадратный корень

Практическая работа

1) Пример программы:

```
Program Example_Of_Easy_Program; {1}
Type Valid_Age = 1 .. 99; {2}
Var Age: Valid_Age; {3}
Begin {4}
  Write ('Введите ваш возраст: '); {5}
  Read (Age); {6}
  Case Age Of {7}
  1..10 : WriteLn ('Вам изучать Паскаль еще рано.'); {8}
  11..14 : WrileLn ('Вам пора изучать язык Паскаль!'); {9}
  15..17 : WriteLn ('Как? Вы еще не изучили язык Паскаль?' {10}
             +' Скорей же!'); {11}
    Else Writeln ('Даже в таком почтенном возрасте '+ {12}
    'изучить Паскаль еще не поздно!') {13}
  End {Case} {14}
End. {15}
```

На первой строке помещен заголовок. Строка 2 содержит раздел описания типов, в котором описан интервальный тип Valid_Age. В строке 3 описывается переменная Age, которая имеет тип Valid_Age. 4-я строка - начало раздела операторов, которые размещены в строках с 5-й по 14-ю. Программа содержит операторы вызова процедур Read, Write и оператор варианта Case, который располагается на строках 7-14 и, в свою очередь, содержит операторы вызова процедур Writeln (строки 8-13). Строковые константы в строках 10 и 12 полностью не уместились, поэтому были перенесены на строки 11 и 13 соответственно. Строка 14 содержит комментарий, поясняющий, что End относится к оператору варианта. В строке 15 заканчивается раздел операторов.

2) Наберите следующие программы, сохраните их на диске:

а)
```
Program Example_1;
var m, x, b, k, a: real;
begin
    clrscr;
        write (' Введите значение m => ');
      readln (m);
      write (' Введите значение x => ');
      readln (x);
      write (' Введите значение b => ');
      readln (b);
    if x*b = 0 then
          write (' Деление на 0!')
        else begin
          k:=  sqr(x) * (m-b) / sin (x);
           a:= (k/x - m/b) * sqrt ( abs (x) );
      writeln ( 'Значение k = ', k:5:5 );
      writeln ( 'Значение a = ', a:5:5 );
            end;
    readln
end.
```

б)
```
Program Example_2;
Var a, k, x: Real;
Begin
Write ('Введите через пробел значения a k x : ');
Read (a,k,x);
   x := x * Pi / 180;
   y := (sin (x) - 5 * a) / (k-b);
Writeln ('y=', y:1:3)
End.
```

в)
```
Program Example_3 ;
Var A, B: Integer;
Begin
Write ('Введите числа A, B - ');
Readln (A, B);
If  A > B then


WriteLn ('Число A – больше')
  else
WriteLn ('Число B – больше');
WriteLn ('Программа завершила свою работу')
End.
```

Контрольные вопросы:

1) Какова структура Pascal-программы?

2) Назовите назначения основных разделов программы.

3) Для чего в языке Pascal используется символ точка с запятой?

4) Какие существуют виды операторов?

5) Перечислите основные арифметические функции.

Для защиты лабораторной работы №1 необходимо уметь объяснить работу каждой программы

Лабораторная работа №2

Условный оператор. Составной оператор.

Иногда бывает необходимо предусмотреть в программе "развилку", т.е. в зависимости от истинности или ложности какого-то условия выполнить соответствующие действия.

В языке Pascal условный оператор имееет вид:

IF условие THEN оператор1 ELSE оператор2

Он выполняется следующим образом:

1) вычисляется выражение (его результат должен быть типа Boolean);

2) если результат - True, то выполняется оператор1;

3) если результат - False, то выполняется оператор2.

Существует также сокращенная форма условного оператора:

IF условие THEN оператор

В этом случае, если условие истинно, то выполняется оператор, если же оно ложно, то оператор не выполняется.

Пример программы: найти максимальное из двух чисел - a и b

```
Program IfExample;
Var a, b : Real;
Begin
    Write ('Введите 2 числа: ');
Read (a, b);
    If a > b   Then Write ('Первое')
    Else Write ('Второе');
Writeln (' число больше.')
End.
```

Обратите внимание, что внутри условного оператора, как следует из формы его записи, разделитель НЕ СТАВИТСЯ (нет точки с запятой после вызова процедуры Write перед служебным словом Else).

Для облегчения чтения программы условный оператор удобно оформлять так:

```
IF условие
    THEN оператор1
    ELSE оператор 2
```

Условные операторы могут быть также "вложены" друг в друга, а также содержать сложные условия, составленные с помощью логических операций (not, or, and, xor).

В случае вложенных условных операторов *Else* всегда будет относиться к ближайшему к нему слева *If*. Например, следующий оператор присваивает переменной x максимальное из трех чисел a, b и c:

```
If a > b and a > c Then x := a
Else If b > c Then x := b
Else x := c
```

После *Then* или *Else* мы можем записать лишь один оператор. Как быть, если нам необходимо выполнить несколько действий подряд в зависимости от условия? Язык *Pascal* предоставляет такую возможность с использованием составного оператора, который имеет вид:

BEGIN оператор1; оператор2; . . . END

В этом случае последовательность операторов, заключенных между служебными словами *Begin* и *End* (они называются операторными скобками) воспринимаются языком *Pascal* как один оператор. Таким образом, ограничение в один оператор после *Then* или *Else* можно обойти, применив составной оператор.

Пример программы: решить квадратное уравнение с коэффициентами a, b и c.

```
Program IfExample_1;
Var a, b, c, d, x1, x2: Real;
Begin
Write ('Введите коэффициенты a, b и c: ');
Read (a, b, c);
d := Sqr (b) - 4 * a * c;
If d < 0 Then Writeln ('Корней нет.')
Else Begin
x1 := (-b - Sqrt (d)) / 2 / a;
x2 := (-b + Sqrt (d)) / 2 / a;
Writeln ('Корни уравнения ', x1:1:1, ' ', x2:1:1)
End
End.
```

Пример программы: определить день недели по номеру

```
Program IfExample_1
Var i: Integer;
Begin
Write ('Номер дня недели: '); Read (i);
If i=1 Then Writeln ('Понедельник.')
Else If i=2 Then Writeln ('Вторник.')
Else If i=3 Then Writeln ('Среда.')
Else If i=4 Then Writeln ('Четверг.')
Else If i=5 Then Writeln ('Пятница.')
Else If i=6 Then Writeln ('Суббота.')
Else If i=7 Then Writeln ('Воскресенье.')
Else Writeln ('Вы ошиблись. Нет такого дня.')
End.
```

Контрольные вопросы:

1. Для чего используется условный оператор?
2. Какой вид имеет полная и сокращенная форма условного оператора?
3. Как работает условный оператор?
4. Назовите особенности использования вложенных условных операторов.
5. Что такое составной оператор?

Варианты индивидуальных заданий:

1	Вычислить значение функции y в точки x	$y(x)=\begin{cases}8x, x \le 0\\ \sin x, 0 < x < 3\\ \sqrt{10x}, x \ge 3\end{cases}$
2	Вычислить значение функции y в точки x	$y(x)=\begin{cases}15x^2, x > 5\\ 5,1 \le x \le 5\\ \sqrt{10x}, x < 1\end{cases}$
3	Вычислить значение функции y в точки x	$y(x)=\begin{cases}\lVert X \rVert, x \le 0\\ \sin^2 x, 0 < x < 3\\ \sqrt{1+x}, x \ge 3\end{cases}$
4	Вычислить значение функции y в точки x	$y(x)=\begin{cases}e^{2-x}, x \le 0\\ \sin^2 x, 0 < x < 3\\ \sqrt{x}+2x, x \ge 3\end{cases}$
5	Вычислить значение функции y в точки x	$y(x)=\begin{cases}\cos x^2, x \le \pi\\ \sin x, \pi < x < 2\pi\\ \sqrt{10x}, x \ge 2\pi\end{cases}$
6	Вычислить значение функции y в точки x	$y(x)=\begin{cases}tg\,2x, \pi < x \le \pi\\ \sin^2 x, \pi < x < 3\pi\\ \sqrt{1+x}, x \ge 3\pi\end{cases}$
7	Вычислить значение функции y в точки x	$y(x)=\begin{cases}(x+2)^3, x \le 0\\ \sin^2 x, 0 < x < 3\\ \sqrt{1+x}, x \ge 3\end{cases}$

Лабораторная работа №3

Оператор варианта

Иной раз при использовании условного оператора можно столкнуться с громоздкостью записи программы при решении задачи. Примером может служить задача определения дня недели по номеру (см. лабораторную работу №2).

Горадо компактнее выглядит решение этой же задачи с помощью оператора выбора, который имеет вид:

CASE выражение OF
константа1: оператор1;
константа2: оператор2;
. . .
ELSE операторN
END

Выполняется оператор так:

1) вычисляется значение выражения (оно может быть и просто переменной); 2) выполняется оператор, чья константа (ее тип дожен совпадать с типом выражения) совпадает с вычисленным значением выражения;

3) если соответствующей константы не найдено, то выполняется оператор после служебного слова ELSE.

Как и для условного оператора, существует сокращенная форма оператора варианта (без ветви ELSE).

Вернемся к программе. Теперь ее фрагмент, соответствующий анализу номера дня недели, будет выглядеть так:

```
Case i Of
1: Writeln ('Понедельник.');
2: Writeln ('Вторник.');
3: Writeln ('Среда.');
4: Writeln ('Четверг.');
```

5: Writeln ('Пятница.');
6: Writeln ('Суббота.');
7: Writeln ('Воскресенье.');
Else Writeln ('Вы ошиблись. Нет дня недели с таким номером.') End

Контрольные вопросы:

1. Для чего используется оператор варианта?
2. Какой вид имеет полная и сокращенная форма оператора варианта?
3. Как работает оператор варианта?

Варианты индивидуальных заданий:

1	Известно, что начало месяца – вторник. Составить программу, позволяющую определить день недели по числу месяца.
2	Составить программу, позволяющую определить время суток по часам. Сутки начинаются с 0 часов, каждое время суток занимает 6 часов.
3	Составить программу перевода курса рубля в евро, доллары и марки по выбору клиента
4	Известно, что начало месяца – воскресенье. Составить программу, позволяющую определить день недели по числу месяца.
5	Известно, что начало месяца – среда. Составить программу, позволяющую определить день недели по числу месяца.
6	В зависимости от вида геометрической фигуры вычислить ее площадь
7	Составить программу, позволяющую по введенному времени (в часах) выводить пожелания «С добрым утром!», «Добрый день!, «Добрый вечер!» и «Спокойной ночи!».
8	Составить программу, позволяющую по номеру месяца вывести подходящую одежду
9	Известно, что начало месяца – четверг. Составить программу, позволяющую определить день недели по числу месяца.
10	Известно, что начало месяца – пятница. Составить программу, позволяющую определить день недели по числу месяца.
11	Составить программу, позволяющую вывести по группе крови подходящую диету
12	Составить программу, позволяющую определить категорию трудоспособности по возрасту (до 16 т – ребенок, старше 16 и до 60 – трудоспособный, старше 60 – пенсионер).

Лабораторная работа №4

Оператор цикла с предусловием. Оператор цикла с предусловием

Иногда в программе необходимо многократно выполнить какое-либо действие, например, напечатать какую-нибудь таблицу. Для печати, предположим, таблицы синусов от 0 до 90 градусов с шагом в 5 градусов с точностью 0.001

Можно было бы поступить так:

```
Program TableOfSinuses;
Begin
writeln('синус 0 градусов равен ', sin (0 * Pi / 180):1:3);
writeln('синус 5 градусов равен ', sin (5 * Pi / 180):1:3);
. . .
End.
```

Многоточием отмечены пропущенные 17 процедур WriteLn. Естественно, эту же задачу можно решить короче и нагляднее с использованием оператора цикла с предусловием, который имеет вид:

WHILE выражение DO оператор

Работает оператор так:

1) вычисляется выражение (его результат должен быть типа Boolean);

2) если результат выражения - True, то выполняется оператор, а затем управление в программе передается опять на начало оператора WHILE;

3) если результат выражения - False, то происходит выход из цикла.

Например, ту же программу можно записать так:

```
Program WhileExample1;
Var Angle: Integer;
Begin
Angle := 0; {начальное значение}
While Angle <= 90 Do Begin
Writeln ('синус ', Angle:2, ' градусов равен ', Sin (Angle*Pi/180):1:3); Inc (Angle, 5)
End
End.
```

Особенностью оператора цикла с предусловием является то, что он может не выполниться ни разу, если условие продолжения выполнения цикла ложно. Поэтому он и называется оператором цикла с предусловием.

Другая его особенность - то, что количество повторений цикла заранее не определено, в том числе может и быть бесконечным, если условие всегда истинно.

Наряду с оператором цикла WHILE существует и другой вид цикла - цикл с постусловием:

REPEAT операторы UNTIL выражение

Работает он следующим образом:

1) выполняются операторы;

2) вычисляется выражение - его результат должен быть типа Boolean;

3) если результат выражения - True, то происходит выход из цикла, если - False, то управление передается на начало цикла.

Пример использования оператора REPEAT (задача та же, что и при рассмотрении оператора цикла с предусловием):

```
Program RepeatExample;
Var Angle: Integer;
Begin
Angle:=0; {начальное значение}
Repeat
Writeln('синус ',Angle:2,' градусов равен ',Sin(Angle*Pi/180):1:3);
Inc(Angle,5)
Until Angle>90
End.
```

В отличие от While, при необходимости использования нескольких операторов в цикле Repeat нет необходимости использовать операторные скобки - их роль выполняют ключевые слова Repeat и Until.

Еще одной особенностью является то, что Repeat, в отличие от While, всегда выполнится хотя бы раз, вне зависимости от условия выхода из цикла. Поэтому его и называют оператором цикла с постусловием.

Так же, как и в операторе While, количество повторений цикла заранее не определено и может быть бесконечным, если условие выхода из цикла всегда ложно.

Контрольные вопросы:

1. Для чего используется оператор цикла с предусловием?
2. Какой вид имеет оператор цикла с предусловием?
3. Как работает цикл While?
4. Каковы особенности цикла While?
5. Для чего используется оператор цикла с постусловием?
6. Какой вид имеет оператор цикла с постусловием?
7. Как работает цикл Repeat?
8. Что общего и чем отличаются друг от друга циклы While и Repeat?

Для защиты лабораторной работы №4 необходимо уметь объяснить работу каждой программы

Лабораторная работа №5

Оператор цикла с параметром

Иногда нужно выполнить многократные действия с известным числом повторений. Например, вычислить факториал (факториалом числа N назывется произведение N!=1*2*3*...*N).

Для организации подобных вычислений используется оператор цикла с параметром:

FOR переменная := выражение1 TO выражение2 DO оператор

или

FOR переменная := выражение1 DOWNTO выражение2 DO оператор

Работает цикл FOR так:

1) вычисляются выражение1 и выражение2. Их тип должен соответствовать типу переменной (она называется счетчиком цикла);

2) переменной присваивается значение, равное выражению1;

3) если ее значение больше (для TO) или меньше (для DOWNTO) значения выражения2, то происходит выход из цикла;

4) выполняется оператор;

5) значение переменной увеличивается на 1 (для TO) или на -1 (для DOWNTO); 6) управление передается на начало цикла.

Пример программы: вычислить значение факториала числа N

```
Program ForExample;
Var I, N, F: Integer;
Begin
Write('N='); Read(N);
    F:=1; {начальное значение}
For I:=1 To N do F:=F*I;
    Writeln(N,'!=',F);
End.
```

Цикл *For* может не выполниться ни разу, если значение выражения1 больше (для *TO*) или меньше (для *DOWNTO*) значения выражения2.

Если внутри цикла необходимо выполнить несколько операторов, то можно воспользоваться операторными скобками:

Begin
оператор1;
оператор2;
… . .
End

Пример программы: Требуется *протабулировать функцию y=sin(x) на [a,b] с шагом h.* Для решения этой задачи определяется число шагов. Наберите программу, проверьте ее действие.

```
Program Tabff ;
Var x,y,a,b,h: real;
      i: integer;
Begin
readln(a,b,h); x:=a;
for i:=1 to TRUNC((b-a)/h) do
  begin
    y:=sin(x);
    writeln(x,y);
    x:=x+h
  end;
end.
```

TRUNC(x)- отбрасывание дробной части от числа x.

Задание: Протабулировать функцию Y=3sin(3x+1) на отрезке [a,b] с h.

Контрольные вопросы:

1. Для организации каких вычислений используется оператор FOR?
2. Какой вид имеет оператор цикла с параметром?
3. Как работает оператор цикла с параметром?

Для защиты лабораторной работы №5 необходимо уметь объяснить работу каждой программы

Лабораторная работа №6

Массивы

При необходимости использования большого количества однотипных данных обычно используются массивы.

Массив - эта структура данных в языке Pascal, имеющих общее имя, к элементам которой можно обращаться посредством индексов.

Количество элементов в массиве должно быть заранее определено. Массивы бывают одномерные (линейные), двумерные (прямоугольные, или, как их называют в математике, матрицы) и размерности большей двух.

Линейный массив представим в виде строки, например: A[1] A[2] A[3] A[4] A[5] A[6] A[7] A[8] одномерный массив из 8 элементов.

Матрицу можно представить в виде прямоугольной таблицы:

B[1,1] B[1,2] B[1,3] B[1,4] двумерный массив из 3 стоок и 4 столбцов
B[2,1] B[2,2] B[2,3] B[2,4]
B[3,1] B[3,2] B[3,3] B[3,4]

При описании массива необходимо указать имя, количество элементов по каждой размерности и тип его элементов.

Например, пусть изображенные выше массивы A и B содержат в себе соответственно элементы целого и вещественного типов. Тогда они должны быть описаны следующим образом:

Var A: Array [1..8] Of Integer;
B: Array [1..3,1..4] Of Real;

Здесь *Array* - служебное слово языка, обозначающее массив.

Для обращения к конкретному элементу массива используется имя массива и индексы, разделяемые запятыми и заключенные в квадратные скобки, например: A[7]:=334; {предпоследнему элементу массива A присваивается значение 334} B[2,j]:=A[i]-B[3,4]; {j-му элементу 2-й стоки массива B присваивается значение, равное разности i-го элемента массива A и нижнего левого элемента массива B}

Таким образом, индексами могут быть не только целые константы, но и любые выражения целого типа.

Для обработки массивов целесообразно использовать циклы FOR.
Пример программы: вывести оценок ученика и его среднего балла:

```
Program ArrayExample;
Var I: Integer; {для организации цикла}
N: Integer; {количество оценок}
O: Array [1..100] Of Integer; {линейный массив для 100 (max) оценок}
S: Integer; {сумма всех оценок}
Begin
Write ('Количество оценок (не больше 100): ');
Read (N);
Writeln ('Введите ',N,' оценок через пробел. Конец ввода - Enter');
S:=0; {обнулить сумму}
For I:=0 To N Do Begin
                Read (O[I]); {считать i-ю оценку}
                 S:=S+O[I] {добавить ее к сумме}
End;
Writeln('Средний балл равен ',S/N:1:1) {вывод среднего с точностью 0.1} End.
```

При обработки массивов размерностью большей единицы обычно применяют вложеннные циклы FOR.

Пример программы: напечатать фамилию ученика с наивысшим средним баллом:

```
Program Array2Example;
Const NKids=3; {количество учащихся}
NO=5; {количество оценок}
Var O: Array [1..10,1..15] Of Integer; {массив оценок}
F: Array [1..10] Of String; {массив фамилий учеников}
S: Array [1..10] Of Real; {массив для средних оценок}
M: Integer; {для хранения индекса ученика с максимальным средним}
Max: Real; {для хранения максимального среднего балла}
I,J: Integer; {для организации циклов}
Begin
M:=1; {пусть ученик с максимальным средним баллом - первый}
Max:=0; {пусть максимальный средний балл равен 0}
For I:=1 To NKids Do Begin
```

```
S[I]:=0;
Write ('Фамилия ',I,'-го ученика: ');
ReadLn (F[I]);
WriteLn ('Введите ',NO,' оценок через пробел. Конец ввода - Enter');
        For J:=1 To NO Do Begin
Read (O[I,J]); {считаем J-ю оценку I-го ученика}
S[I]:=S[I]+O[I,J] {добавим ее к сумме оценок}
End;
S[I]:=S[I]/NO; {разделим сумму на количество для нахождения среднего}
        If S[I] > Max Then Begin {если текущий средний балл больше максимального}
M:=I; {то номер ученика с максимаольным баллом - текущий}
        Max:=S[I] {и максимальный средний балл заменим на текущий}
        End;
Readln {для подготовки к чтению следующей фамилии}
End;
WriteLn ('Максимальный средний балл имеет ',F[M])
End.
```

Кроме задач на ввод-вывод и нахождения минимального или максимального значения элементов массива, существует и задачи на сортировку элементов массива. Для этой цели обычно применяют сортировку "пузырьком", смысл которой в следующем. Пусть элементы массива расположены в хаотичном порядке. Будем сравнивать два соседних элемента, с первого до последнего: первый со вторым, второй с третьим, и т. д. При обнаружении недопустимого порядка соответствующие элементы меняем местами. На следующем проходе просматриваем все элементы до предпоследнего, т. к. последний уже находится на своем месте, он как бы "всплыл пузырьком", и т. д.

Более трудоемок способ традиционной сортировки, который заключается в том, что сначала в массиве ищется минимальный (максимальный) элемент, запоминается его индекс. На следующем шаге он меняется местами с первым (последним). И т.д. На втором проходе уже ищем минимальный (максимальный) элемент, исключив первый (последний), который теперь находится на своем месте.

Несмотря на кажущуюся сложность, готовых программ на примеры сортировки здесь не приводится. На помощь пониманию процесса сортировки может придти работа .

Пример программы: поиска максимального элемента и его индексов в одномерном массиве. Объясните действие этой программы.

```
Program Max_search;
const m=50;
type vector=array[1..m] of integer;
var  a:vector;  i,imax,max,p:integer;
begin
randomize;
writeln('Создание массива и его вывод');
 for i:=1 to m do
   begin
     p:=random(50);
     a[i]:=p-30;
     write(a[i]:4);
  end;
 max:=a[1];
 for i:=2 to m do
   if max<a[i] then begin max:=a[i]; imax:=i end;
```

```
 writeln; writeln('Значение максимального =',max);
 writeln('Индекс максимального =',max);
 writeln('Для завершения работы нажмите <Enter>');
readln
end.
```

Пример программы:

```
Program Max_search;
uses crt;
var x,i,imax,max,min,imin:integer;A:array[1..45] of integer;
begin clrscr;
randomize; writeln('Создание массива и его вывод');
for i:=1 to 45 do
begin
a[i]:=25-random(70);
write(a[i]:4);
end;
max:=a[1];
min:=a[1];
begin
for i:=1 to 45 do
if max<a[i] then begin
                  max:=a[i];
                  imax:=i;
end;writeln;end;
begin
for i:=1 to 45 do
if min>a[i] then begin
                 min:=a[i];
                 imin:=i;
                 end;writeln;end;
writeln('значение максимального=',max);
writeln('значение минимального =',min);
begin
    x:=imax;
    imax:=imin;
    imin:=x;
end;
writeln('изменённый массив');
for i:=1 to 45 do  write(a[i]:4);
end.
```

Пример программы : вычислить сумму элементов массива A, имеющих четные индексы.

```
program example;
  const  n=10;                    {кол-во элементов в массиве}
  var
       a:array[1..n] of integer;
       i:integer;                 {параметр цикла}
       s:real;                    {сумма}
 begin
```

```
randomize;                  {подключение генератора случайных чисел}
for i:=1 to 10 do
  a[i]:=random(50);         {задание исходного массива}
writeln('исходный массив');
for i:=1 to n do
   write( a [ i ], '  ' );  {распечатка исходного массива}
   writeln;                 {перевод курсора на новую строку}
 s:=0; i:=2;
 repeat
   s:=s+a[i];               {подсчет суммы}
   i:=i+2;
 until i>n;
 write('сумма=',s);         {вывод результата}
```

Контрольные вопросы:

1. В каких случаях целесообразно использовать массивы?
2. Что такое массив?
3. Как описать переменную типа массив?
4. Как обращаться к элементам массива?
5. С помощью каких конструкций удобно работать с массивами?
6. Что такое сортировка массивов?
7. Какие способы сортировки вы знаете?

Варианты индивидуальных заданий:

1	В массиве случайных целых чисел определить процент нечетных чисел и вывести их на печать.
2	В массиве случайных целых чисел поменять местами первый отрицательный и последний положительный элементы.
3	В массиве случайных целых чисел определить номера первого и последнего отрицательного элемента.
4	В массиве случайных целых чисел вывести на печать номера элементов, которые превышают среднее арифметическое значений элементов.
5	В массиве случайных целых чисел вычислить сумму и произведение элементов, больших первого положительного элемента массива
6	В массиве случайных целых чисел вычислить сумму и произведение элементов, больших последнего элемента массива
7	В массиве случайных целых чисел поменять местами первый отрицательный и последний положительный элементы
8	В массиве случайных целых чисел вычислить сумму и произведение элементов, больших последнего элемента массива
9	В массиве случайных целых чисел найти среднее арифметическое элементов, стоящих за первым нулевым элементом.
10	В массиве случайных целых чисел поменять местами последний отрицательный и последний положительный элементы. Вывести их номера

3 Методические указания по самостоятельной работе

Для успешного освоения курса «Программирование» необходима самостоятельная работа. В настоящее время актуальными становятся требования к личным качествам современного студента – умению самостоятельно пополнять и обновлять знания, вести самостоятельный поиск необходимого материала, быть творческой личностью.

Самостоятельную работу по освоению дисциплины обучающимся следует начинать с изучения содержания рабочей учебной программы дисциплины, цели и задач, структуры и содержания курса, основной и дополнительной литературы, рекомендованной для самостоятельной работы.

Самостоятельная учебная деятельность является необходимым условием успешного обучения. Многие профессиональные навыки, способность мыслить и обобщать, делать выводы и строить суждения, выступать и слушать других, – все это развивается в процессе самостоятельной работы студентов.

Самостоятельная работа по освоению дисциплины включает:

- самостоятельное изучение разделов;
- самоподготовку (проработку и повторение лекционного материала и материала учебников и учебных пособий);
- подготовку к лабораторным работам;
- подготовку к рубежному и итоговому контролю.

Самостоятельная учебная работа – условие успешного окончания высшего учебного заведения. Она является равноправной формой учебных занятий, наряду с лекциями, семинарами, экзаменами и зачетами, но реализуемая во внеаудиторное время.

Эффективность аудиторных занятий во многом зависит от того, насколько умело студенты организуют в ходе них свою самостоятельную учебную познавательную деятельность. Такая работа также способствует самообразованию и самовоспитанию, осуществляемому в интересах повышения профессиональных компетенций, общей эрудиции и формировании личностных качеств.

Самостоятельная работа реализуется:

1. непосредственно в процессе аудиторных занятий – на лекциях, лабораторных занятиях, при проведении рубежного контроля;
2. в контакте с преподавателем вне рамок расписания – на консультациях по учебным вопросам, при ликвидации задолженностей, при выполнении индивидуальных заданий;
3. в библиотеке, дома, в общежитии, на кафедре при выполнении студентом учебных задач.

В процессе проведения самостоятельной работы необходимо производить подбор литературных источников, научной периодической печати и т.д

4 Методические указания по итоговому контролю

Итоговый контроль знаний по дисциплине «Программирование» проводится в форме экзамена. Для подготовки к итоговому контролю знаний по дисциплине «Программирование» обучающиеся используют перечень вопросов, приведенный в фонде оценочных средств. Экзамен проводится в устной форме. В экзаменационный билет включен один теоретический вопрос. На подготовку студенту отводится 20-25 минут. На дифференцированном зачете ответы обучающегося оцениваются с учетом их полноты, правильности и аргументированности с учетом шкалы оценивания.

Оценка «отлично» выставляется студенту, если он глубоко и прочно усвоил программный материал, исчерпывающе, последовательно, четко и логически его излагает, умеет тесно увязывать теорию с практикой, свободно справляется с вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении

заданий, использует в ответе профессиональные термины, правильно обосновывает принятое решение.

Оценка «хорошо» выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос, правильно применяет теоретические положения при решении практических вопросов, владеет необходимыми навыками и приемами их выполнения.

Оценка «удовлетворительно» выставляется студенту, если он имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала.

Оценка «неудовлетворительно» выставляется студенту за отсутствие знаний по дисциплине, представления по вопрос, непонимание материала по дисциплине, наличие коммуникативных «барьеров» в общении, отсутствие ответа на предложенный вопрос.

5 Учебно-методическое обеспечение дисциплины

5.1 Основная литература

1 Расолько, Г.А. Теория и практика программирования на языке Pascal : учеб. пособие [Электронный ресурс] / Г.А. Расолько, Ю.А. Кремень. – Минск : Вышэйшая школа, 2015. – 447 с. : ил. – Режим доступа : https://biblioclub.ru/index.php?page=book_view_red&book_id=459674

2 Царев, Р.Ю., Программирование на языке Си: учебное пособие. [Электронный ресурс] / Р. Ю. Царев. – Красноярск : Сиб. федер. ун-т, 2014. – 108 с. ISBN 987-5-7638-3006-4 – Режим доступа: https://biblioclub.ru/index.php?page=book_view_red&book_id=364601

3 Хиценко В.П., Основы программирования: учебное пособие [Электронный ресурс] / В.П. Хиценко. – Новосибирск: Изд-во НГТУ, 2015. – 83 с. – ISBN 978-5-7782-2706-4 – Режим доступа : https://biblioclub.ru/index.php?page=book_view_red&book_id=438365

5.2 Дополнительная литература

1 Информатика и программирование. Алгоритмизация и программирование [Текст] : учебник для вузов / под ред. Б. Г. Трусова. - Москва : Академия, 2012. - 336 с. - (Бакалавриат) - ISBN 978-5-7695-8146-5. (10)

2 Павловская, Т. А. C/C++. Программирование на языке высокого уровня[Текст] : для магистров и бакалавров: учебник для вузов по направлению "Информатика и вычислительная техника" / Т. А. Павловская. - Москва : Питер, 2014. - 461 с. : ил. - (Учебник для вузов. Стандарт третьего поколения). - Алф. указ. : с. 450. - ISBN 978-5-496-00031-4. (15)

3 Орлов, С. А. Теория и практика языков программирования [Текст] : учебник для вузов по направлению "Информатика и вычислительная техника" / С. А. Орлов. - Санкт-Петербург : Питер, 2013. - 688 с. : ил. - (Учебник для вузов. Стандарт третьего поколения) - ISBN 978-5-496-00032-1. (5)

4 Грузина, Э.Э., Практикум по программированию. Ч. 1 [Электронный ресурс] / Э. Э. Грузина, Н. Л. Черноусова. – Кемерово : Кемеровский государственный университет, 2013. – 100 с. ISBN 987-5-8353-1605-2 (Ч.1) – Режим доступа: https://biblioclub.ru/index.php?page=book_view_red&book_id=278837

5 Долинер, Л.И., Основы программирования в среде PascalABC.NET: учебное пособие. [Электронный ресурс] / Л. И. Долинер. – Екатеринбург : Изд-во Урал. ун-та,

2014. – 128 с. ISBN 987-5-7996-1260-3 – Режим доступа: https://biblioclub.ru/index.php?page=book_view_red&book_id=275988

5.3 Периодические издания

1. Журнал «Вестник компьютерных и информационных технологий »
2. Журнал «Информационные технологии и вычислительные системы»
3. Журнал «Стандарты и качество»
4. Журнал «Прикладная информатика»

5.4 Интернет-ресурсы

5.4.1 Современные профессиональные базы данных и информационные справочные системы:

1. Информационная система «Единое окно доступа к образовательным ресурсам» - http://window.edu.ru/
2. КиберЛенинка - https://cyberleninka.ru/
3. Университетская информационная система Россия – uisrussia.msu.ru
4. Бесплатная база данных ГОСТ – https://docplan.ru/

5.4.2 Тематические профессиональные базы данных и информационные справочные системы:

1. Портал искусственного интеллекта – AIPortal
2. Web-технологии – Web-технологии
3. Электронная библиотека Института прикладной математики им. М.В. Келдыша – Электронная библиотека публикаций Института прикладной математики им. М.В. Келдыша РАН

5.4.3 Электронные библиотечные системы

1. ЭБС «Университетская библиотека онлайн» – http://www.biblioclub.ru/
2. ЭБС Znanium.com – https://znanium.com/

5.4.4 Дополнительные Интернет-ресурсы

1. https://www.ixbt.com - Интернет-издание о компьютерной технике, информационных технологиях и программных продуктах. На сайте публикуются новости IT, статьи с обзорами и тестами компьютерных комплектующих и программного обеспечения.
2. http://www.intuit.ru – ИНТУИТ – Национальный открытый университет.
3. http://citforum.ru/SE/project/arkhipenkov_lectures – Лекции по управлению программными проектами автор А. Архипенков

5.5 Программное обеспечение, профессиональные базы данных и информационные справочные системы современных информационных технологий

Тип программного обеспечения	Наименование	Схема лицензирования, режим доступа
Операционная	Microsoft Windows	Подписка Enrollment for Education Solutions

система		(EES) по государственному контракту: № 2К/17 от 02.06.2017 г.;
Офисный пакет	MicrosoftOffice	
Интернет-браузер	Internet Explorer	Является компонентом операционной системы MicrosoftWindows
	Google Chrome	Бесплатное ПО, http://www.google.com/intl/ru/policies/terms/
Интегрированная среда разработки программного обеспечения	Borland C++ 3.1 for DOS	Образовательная лицензия по государственному контракту № 34/10 от 10.12.2010 г., лицензия на рабочее место
	Microsoft Visual Studio Professional 2008	Сертификат Microsoft Open License № 46284547 от 18.12.2009 г., академическая лицензия на рабочее место
	Dev-C++	Свободное ПО http://www.gnu.org/licenses/gpl.html
	Turbo Pascal 7.0 for DOS	Образовательная лицензия по государственному контракту № 34/10 от 10.12.2010 г., лицензия на рабочее место
	PascalABC.NET	Свободное ПО http://www.pascalabc.net/litsenzionnoe-soglashenie

6 Материально-техническое обеспечение дисциплины

Учебные аудитории для проведения занятий лекционного типа, семинарского типа, для проведения групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации. Для проведения лабораторных и практических работ используются компьютерный класс (ауд. № 4-113, 4-116, 4-117), оборудованный средствами оргтехники, программным обеспечением, персональными компьютерами, объединенными в сеть с выходом в Интернет.

Аудитории оснащены комплектами ученической мебели, техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой, подключенной к сети «Интернет», и обеспечением доступа в электронную информационно-образовательную среду Орского гуманитарно-технологического института (филиала) ОГУ (ауд. № 4-307).

Наименование помещения	Материальное-техническое обеспечение
Учебные аудитории: - для проведения занятий лекционного типа, семинарского типа, - для групповых и индивидуальных консультаций; - для текущего контроля и промежуточной аттестации	Учебная мебель, классная доска, мультимедийное оборудование (проектор, экран, ноутбук с выходом в сеть «Интернет»)
Компьютерные классы № 4-113, 4-116, 4-117	Учебная мебель, компьютеры (29) с выходом в сеть «Интернет», проектор, экран, лицензионное программное обеспечение
Помещение для самостоятельной работы обучающихся, для курсового проектирования (выполнения курсовых работ)	Учебная мебель, компьютеры (3) с выходом в сеть «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Орского гуманитарно-технологического

	института (филиала) ОГУ, программное обеспечение

Для проведения занятий лекционного типа используются следующе наборы демонстрационного оборудования и учебно-наглядные пособия:

- презентации к курсу лекций.