МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

О.Г. Савченко

Лекционный ресурс электронного курса «Информатика» (система дистанционного обучения MOODLE)

Электронное интерактивное учебное пособие

С А М А Р А

2011

Автор: **Савченко Ольга Геннадьевна**

Савченко О.Г., Лекционный ресурс электронного курса «Информатика» (система дистанционного обучения «Moodle») [Электронный ресурс] : электрон. интерактив. учеб. пособие / О.Г. Савченко; Минобрнауки России, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т). - Электрон. текстовые и граф. дан. (1,6 Мбайт). - Самара, 2011. - 1 эл. опт. диск (CD-ROM). - Систем. требования: ПК Pentium; Windows 98 или выше.

Представлен комплект лекций по курсу «Информатика», использованный как лекционный ресурс в системе дистанционного обучения MOODLE. Система размещена на сайте кафедры общей информатики СГАУ. СОД MOODLE предназначена для интерактивного освоения материалов курса и дает возможность контроля текущих знаний студентов.
Лекции включают основные положения дисциплины «Информатика». Даются основы программирования на языке высокого уровня Турбо Паскаль, рассматриваются многочисленные примеры решения стандартных задач.
Учебный курс ориентирован на 2 факультет, 1 курс, II семестр (очное) для подготовки:

- бакалавров по направлениям подготовки 151900.62, 151000.62, 141100.62, 080100.62
- студентов по специальностям 160700.65

Подготовлено на кафедре общей информатики СГАУ.

ЛЕКЦИЯ №1

Информатика-это совокупность знаний и навыков, необходимых для решения задач на ЭВМ.
Появление ЭВМ потребовало разработок новых методов и приемов выполнения вычислений с учетом специфики работы ЭВМ. Компьютер (ЭВМ) может работать только под управлением какой-либо программы. Поэтому программирование является одной из составляющих курса информатики.

История развития ЭВМ.

Первые ЭВМ использовались исключительно для решения математических задач. Современные компьютеры являются основой информационной среды человечества. Теперь их основные назначения:

- Сбор;
- Хранение;
- Преобразование различной информации.

Первая ЭВМ появился в 1944 году, и получила название «ЭНИАК». Она содержала 18 тысяч электронных ламп; занимала площадь 130 квадратных метров; потребляла 150 кВт энергии; скорость вычисления была приблизительно равна 1000 операций в секунду. В машине использовалась десятичная система счисления. В 1947 году в СССР появился МЭСМ - компьютер. Он был изобретен в монастыре под руководством будущего академика Лебедева. Большой вклад в развитие ЭВМ внес ученый австриец Фоннэйман. Анализируя работу первой ЭВМ, Фоннэйман внес ряд конструктивных предложений, которые получили названия «Архитектура Фоннэймана».

Основные предложения:

1. Машина должна работать в двоичной системе счисления;
2. И программа и исходные данные должны храниться в оперативной памяти вместе;
3. Команды программы, как и данные должны так же представляться в двоичных кодах;
4. Память ЭВМ должна иметь иерархическую структуру.

Кроме этого Фоннэйман предложил процесс программирования разделить на два этапа.
На первом этапе разрабатывалась подробная блок-схема алгоритма решения задачи.
На втором этапе эта блок-схема по четким правилам переводится в команды программы.
В дальнейшем эта идея получила развитие транслятора.

Поколение ЭВМ.

Термин «поколение» связан с элементной базой (электрическое устройство, из которого собирают ЭВМ).

1 поколение – ламповые ЭВМ (период 50х годов)

- Низкая надежность;
- Потребляли много энергии.

Любой электрический прибор может перегореть либо в момент включения или в момент выключения. Такие машины нельзя было ставить на движущиеся объекты.

2 поколение – транзисторные ЭВМ (60е – 70е г.г.)

- Аналог ламповому диоду;
- Значительно уменьшаются габариты;
- Повысилась надежность;
- Уменьшилось потребление энергии.

3 поколение – твердые (интегральные) схемы

4 поколение – ЭВМ на основе больших интегральных схем (Бисы); сверх больших интегральных схем (сверх Бисы) и микропроцессоры. На их основе появились современные ПК.
Показателем развития ЭВМ всегда являлись суперЭВМ. Сейчас строятся как множество микропроцессоров.

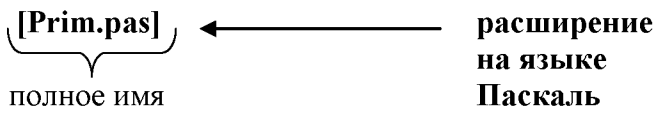
Основные назначения суперЭВМ:

1. Криптография – шифровка и дешифровка секретов государств;
2. Искусственный интеллект;
3. Моделирование сложных систем (ядерные взрывы, погода, поиск месторождений и т.д.).

ЛЕКЦИЯ №2

Основы программирования.
Файлы и маршруты.

Файл – именованная область диска, куда записывается информация определенного характера (жесткий диск)
Каждому файлу присваивается уникальное, в своем разделе, имя **[Prim.pas].**



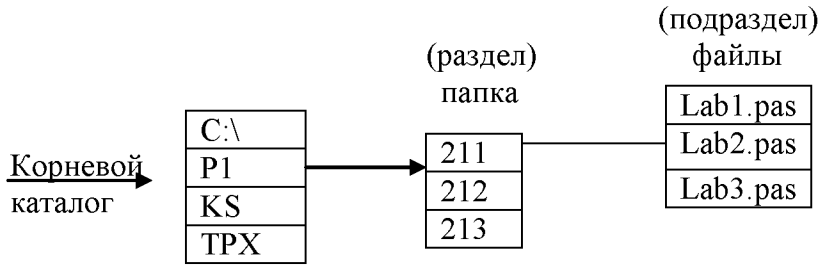
[Prim – собственное имя; .pas – расширение имени].

Расширение обычно определяет тип информации или данных файла. По расширению операционная система (ОС) может автоматически запускать обрабатывающие программы.
Именам файла лучше назначать:

- Имя файла набирать латинскими буквами;
- Начинать имя с буквы;
- Содержать имя должно **не более** 8 символов.

Пример: Lab1.pas; Lab2.pas.
….bak; ….exe → исполняемый файл (программа)
↓
Предыдущая
версия файла

Все файлы сохраняются на жестком диске [C:]



Для того чтобы найти файл требуется маршрут:

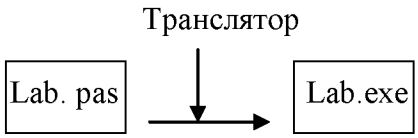
Маршрут файл → адрес файла на диске

Пример: C:\ USER_D~1\ 211\ Lab1_.pas → (маршрут)

В имени нельзя допускать «пробел», можно ставить лишь нижнее тире «_».

Трансляция программ.

Программа – последовательность команд для ЭВМ. Программа, по сути, является предельно детализированным алгоритмом.
Алгоритм – описание порядка действий, необходимых для решения поставленной задачи.
Для составления программ служат языки программирования, но компьютер понимает лишь машинный язык ⟶ двоичные коды.
Программировать в машинных кодах очень сложно, поэтому были разработаны языки высокого уровня или алгоритмические языки (Фортран, Бейсик, Паскаль, Си, С++).
Для выполнения программ на языке Паскаль используются специальные программы переводчики (трансляторы). Эта программа переводит с языка высокого уровня на машинный язык. Процесс перевода называется **трансляция**.



В результате трансляции могут создаваться файлы с расширением **.**com или **.**exe.

ЛЕКЦИЯ №3

Алфавит языка Паскаль.

Он включает:

- Латинские буквы (A-Z, a-z) и символ «подчеркивания»;
- Арабские цифры (0-9);
- 22 специальных символа **( ) [ ] { } ‘ : = ; , ._ - + * / < > $ ^ # @ и «пробел»**

Паскаль **НЕ различает** большие и малые буквы. Русские буквы и другие символы можно использовать только в комментариях и текстовых строках.

Идентификаторы.

1. Идентификаторы в языке Паскаль – это имена констант, переменных, меток, типов, процедур и функций.
   Идентификатор содержит:
   1. только латинские буквы, цифры и низкое тире «_»;
   2. имя НЕ может начинаться с цифры (можно – a1, a_1; нельзя – 1a, #1, do);
   3. длина может быть любой, но учитываются только первые 63 символа.

Нужно помнить, что соответствующие заглавные и строчные буквы в идентификаторах и служебных словах не различаются. Таким образом, следующие три идентификатора обозначают одну и ту же переменную: primer, PRIMER, Primer.

В качестве идентификаторов нельзя использовать служебные слова. Служебные слова далее будут рассматривать по мере изучения языка.

Структура программы

1. Program **«пробел»** Prim; ⟶ заголовок программы

2. Label…
   Const…
   Type…
   Var… } раздел объявлений

3. { Описание подпрограмм }

4. Begin

5. Оператор 1
   Оператор 2
   Оператор n } исполняемая часть программы

6. End.

Каждый оператор закрывается «**;**», за отдельными исключениями (после Begin не ставится «**;**» **)**. В одной строке можно располагать несколько операторов. Вся программа закрывается end с точкой. В любом месте программы могут располагаться комментарии **{ }**. Но машина эти комментарии НЕ обрабатывает.
В своих программах после слова **Program** в обязательном месте надо ставить комментарии - № лабораторной работы, Фамилию, № группы.

Пример простой программы:

```
Program primer;   {заголовок программы}
  Var x,y,z:real;   {описание переменных}
   Begin            {начало выполняемой части программы}
    Read(x,y);      {ввод чисел x и y}
      z:=x+y;       {вычисление z}
    write(z);       {вывод на экран значения z}
  End.              {конец программы}
```

Переменные.

Служат для сохранения и обозначения данных.
Значения переменных при выполнении программ могут меняться, а константы не могут.

- Каждая переменная должна иметь уникальное имя.
- Имена не могут повторять названия операторов.
- Все переменные должны быть перечислены в разделе War с указанием их типа.

VAR <имя_переменной>: <тип_переменной>;
Например:

```
Var x: real;
    i,j: integer;
    авс: Boolean;
```

Типы данных.

Переменные могут сохранять числовые значения (целое – (тип) **Integer**, дробное – (тип) **Real**); символьные значения (один символ – (тип) **Char**); тестовые значения ((тип) **String**) и логическое значение.

Real – вещественные числа. Это числа, содержащие целую и дробную части, разделяемые точкой.
Real – 0.16; -0.5; 3.87; -875.001
Для типа REAL используются следующие арифметические операции:
+ сложение; - вычитание; * умножение; / деление
Список операций приведен в порядке повышения приоритета.

Так, например, в выражении a:=4+8/2*4, сначала выполнится деление

8/2=4, затем умножение 4*4=16, а затем сложение 4+16=20.

Выражение в скобках имеет высший приоритет и выполняется первым.

Пример:

a:=4+8/(2*4); {результат =5}

a:=(4+8)/2*4; {результат =24}

INTEGER - целые числа. Для типа INTEGER вместо операции

деления / определены две специальные операции:

DIV деление с отбрасыванием дробной части;

MOD взятие остатка от целочисленного деления.

Integer – диапазон (-32 000; +32 000)

Пример:

b:=7 div 3; {результат=2}

b:=7 mod 3; {результат=1}

BOOLEAN - логический тип.

Объекты типа BOOLEAN могут принимать только два значения: **TRUE** - "истина", и **FALSE** - "ложь".

CHAR - символы.

В переменную этого типа может быть помещен любой символ. Значение переменной CHAR задается в апострофах.

Пример: y:='y';

STRING - строки. Массив символов.

Пример: text:='жизнь прекрасна';

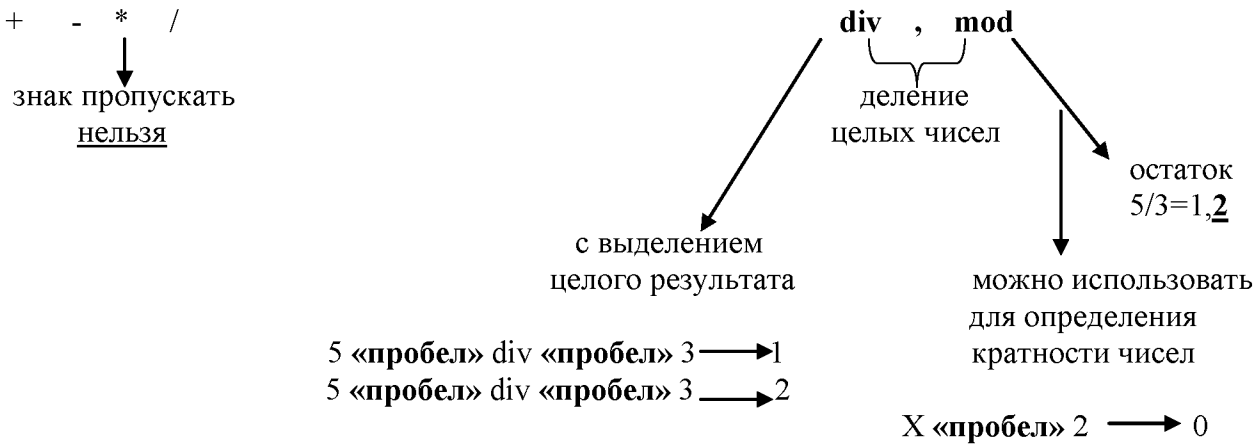
Константы

Константы описываются в разделе описаний. Описание константы начинается со слова CONST. Значение константы нельзя изменить в теле программы.

Например:

```
Const  PI=3.14;              { Раздел }
Var    a: real;              {описаний }
       begin
            a:=PI;           {Допустимое действие}
            PI:=2.7189;      {Ошибка!}
       end.
```

Арифметические выражения.



Порядок действий в арифметических выражениях задается скобками и приоритетом действий в частности: * /, потом: + -

Пример:

1. $\dfrac{a + b}{c + d}$ ⇨ (a + b) / (c + d)

2. $\dfrac{a + b}{cd}$ ⇨ (a + b) / (c * d)
или
(a + b) / c / d

Основные математические функции.
Стандартные математические функции языка Паскаль.

PI	Число П
Abs(x)	Абсолютная величина |x|
Exp(x)	Экспонента e^x
Sqr(x)	x^2
Sqrt(x)	Квадратный корень x
Ln(x)	Натуральный логарифм ln x
Sin(x)	Sin x (угол в радианах)
Cos(x)	Cos x (угол в радианах)
ArcTan(x)	Arctg x (значение в радианах)
Round(x)	Округляет число до ближайшего целого
Trunc(x)	Отбрасывает дробную часть числа
Int(x)	Целая часть числа
Frac(x)	Дробная часть числа
Randomize	Инициация счѐтчика случайных чисел
Random(x)	Случайное число в диапазоне 0…(x-1) (число x – целое)

1. аргумент может быть сложным выражением и должен находиться в круглых скобках.

- $y = \sqrt{e^{|\sin x|}}$

  y:= SQRT (Exp (ABC (Sin (x))));

- $a = \sqrt[3]{e}$

  a:= Exp (1/3 *ln (Exp (1)));

2. число открывающихся скобок = числу закрывающихся скобок;
3. название функции не является параметром.

Порядок выполнения действий в арифметических выражениях.

1. Вычислительная функция
2. Действие в скобках
3. Умножение и деление
4. Сложение и вычитание

ЛЕКЦИЯ №4

Оператор присвоения.

< Переменная >: =< выражение >;

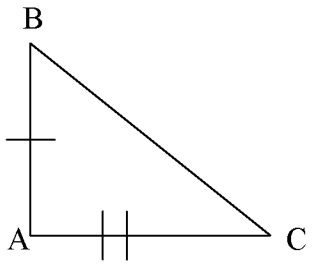
a: = 0.5
a: = b;
a: = b + c;

ОЗУ	
0,5	a
-1,16	b
+1,16	c
2	y

Знак «присвоить» не есть знак «равно».
« = » ⟶ « = ».

x: = x + 1 - такое выражение используется в циклах для накопления результата

Пример программы расчета треугольника



S = ½ AB * AC

$BC = \sqrt{AB^2 + AC^2}$

P = AB + BC + AC

Program **«пробел»** Prim;

{ Лаб.№1; расчет треугольника; }

Фамилия; группа

```
Var «пробел» AB,AC,BC,S,P:Real;
Begin
WriteLn(`Введите два катета`);        { вывести на экран }
ReadLn(AB,AC);                        { прочитать с клавиатуры }
S: =AB*AC/2;
BC: =SQRT (AB*AB+AC*AC);
P: =AB+BC+AC;
WriteLn(`Площадь треугольника=`,S:8:2);
WriteLn(`Периметр треугольника=`,P:8:2);
ReadLn;                               {временная остановка программы}
End.
```

Без изменений; Пл.=16.32 (число)

Оператор ввода / вывода.

Оператор ввода / вывода организует обмен данными между программами и ЭВМ. По умолчанию устройство **вывода** является монитор, а устройство **ввода** – клавиатура. Курсор остается на текущей строке после операции **Write** и **Read,** а после **WriteLn** и **ReadLn** переходит в начало следующей строки.

WRITE.
Процедура Write выводит на экран выражения без перевода строки. Символьные (текстовые) константы заключаются в а п о с т р о ф ы . Например, в результате выполнения следующего фрагмента:

```
A := 555;
Write('A=', a);
Write('<Конец вычислений>');
```

на экране будет напечатано следующее:
A= 555<Конец вычислений>
WRITELN. Процедура Writeln отличается от Write только тем, что она после завершения вывода переводит текущую позицию в начало следующей строки. Например:

```
A := 555;
Writeln ('A=', a);
Writeln('<Конец вычислений>');
```

В результате выполнения этого фрагмента на экране будет напечатано следующее:
A= 555
<Конец вычислений>
Переменные типа real выводятся с помощью оператора write в экспоненциальной форме. Например:
pi:=3.14;
write('PI=',pi);
На экран выведется:
C=3.1400000000E+01
Для вывода чисел в нормальной форме необходимо использовать форматный вывод: после имени переменной ставится ":" и указывается общее количество позиций под число, и, через еще одно ":" – количество позиций под дробную часть.

Например:
write('pi=',c:9:7);
На экран выведется:
pi= 3.1415926

READ.
Процедура Read выполняет ввод с клавиатуры значений переменных.
Наример:
Read(a); - в этом месте выполнение программы приостанавливается, программа ожидает ввода с клавиатуры переменной a.
READLN. Процедура Readln отличается от Read только тем, что после завершения ввода она переводит текущую позицию в начало следующей строки.

Read(A,B,C);

Только имена переменных находятся в списках оператора **Read**

0.1 «пробел» 0.2 «пробел» 0.3 «Enter»

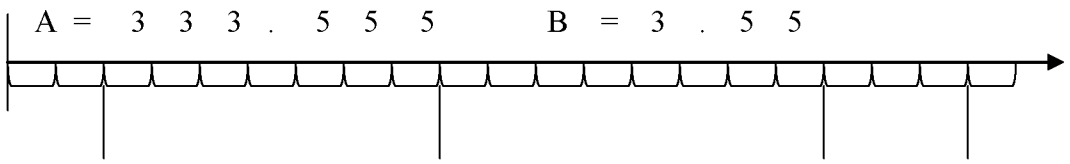
0.1 «Enter»
0.2 «Enter»
0.3 «Enter»

WriteLn (`A = `, A, `B = `, B);
(команда вывести на экран) **текст**

Значения A и B вводятся бесформатным способом.

WriteLn(`A =`, A : 7 : 3, ` «пробел» «пробел» B = `, B:4:2);

Сколько всего позиций выделяется под число
Указать размер дробной части



Перед вводом данных обязательно выводят **подсказку.**

Пример

Ввод данных в строке подсказки

```
Write (`вв числ A =`);
ReadLn (A);
```

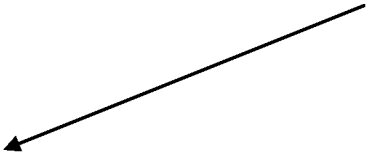
ЛЕКЦИЯ №5.

Операторы программирования.

В обычном режиме программа выполняется последовательно оператор за оператором сверху вниз. Такой порядок характеризуется как линейный. В таких программах порядок оказывается неизменным, в реальных задачах приходится учитывать какие-либо условия и выбирать варианты решения. Операторы управления позволяют менять обычный порядок действия и в одной программе организовать несколько вариантов решения. Операторы управления позволяют менять порядок действия. С их помощью создаются ветвления и циклы программы.

Оператор безусловного перехода GoTo.

Go To «пробел» < метка > **Метка** – условное число или символьное имя.

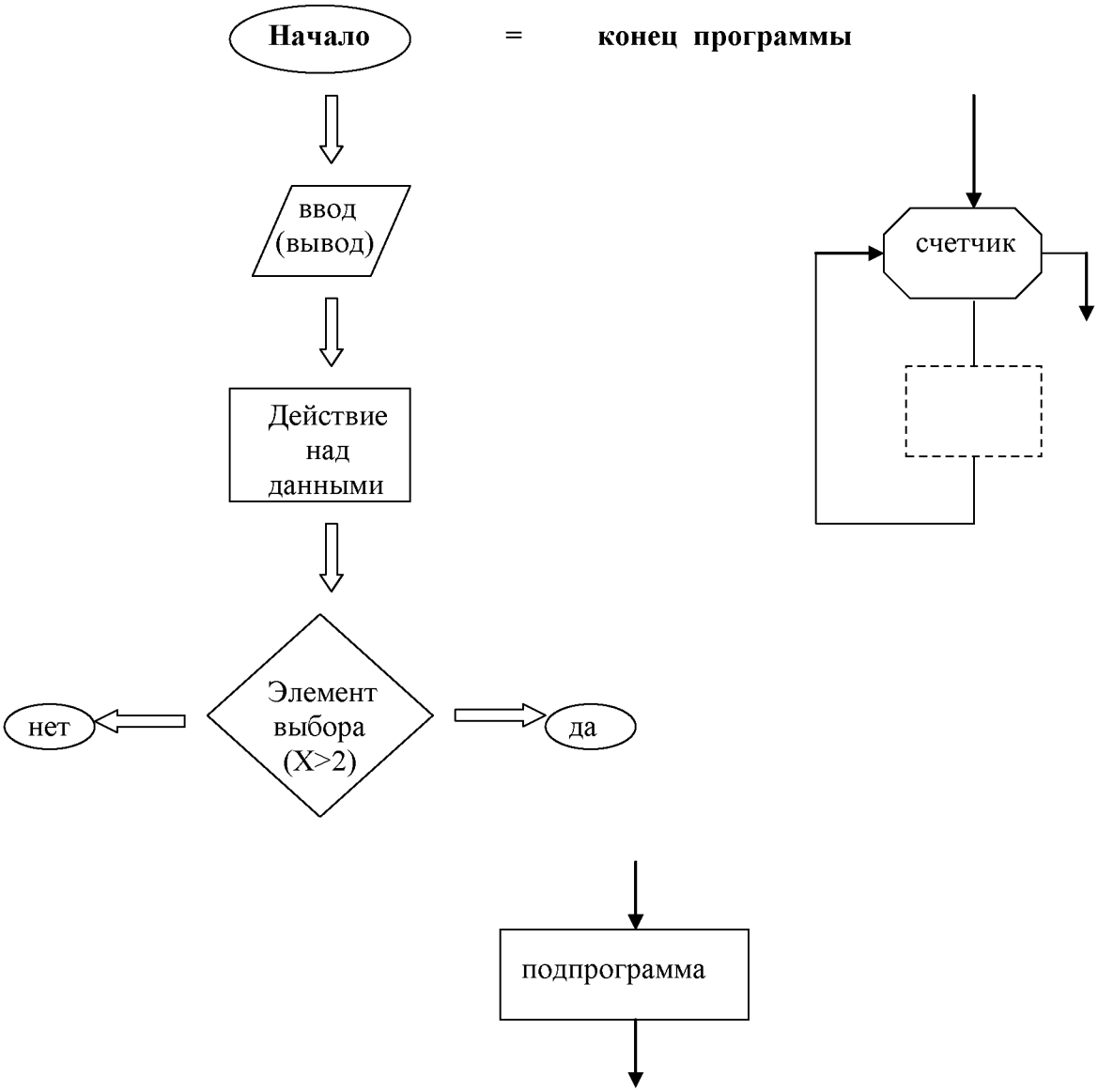


```
Program  Prim;
Label   1, 2, stop;
Var     A, B, C, D: Real;
Begin
    …..
    A: = B + C;
    Go To 1;
1:  D: = A / 2;
End.
```

Управление может передаваться вверх или вниз по программе. При передаче вверх могут возникнуть бесконечные циклы.

Алгоритм.

Алгоритм – это описание последовательности действий, выполнение которых ведет к поставленной цели.
Блок – схема – графическое представление алгоритма. Позволяет более наглядно представить структуру программы.



В блок – схеме не может быть стрелок, направленных в никуда.

Составной оператор.

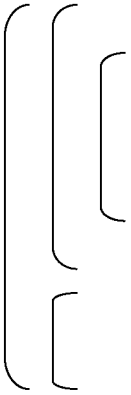
Это последовательность операторов, заключенных в операторные скобки **Begin – End.**

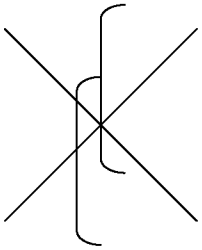
Begin

< оператор 1 >**;**
< оператор 2 >;
……..
< оператор N >;

End.

С точки зрения структуры паскаля это один оператор.





В программе количество **Begin** равно количеству **End (**за исключением оператора выбора CASE).

Условный оператор IF.

Позволяет организовывать ветвления в программе по каким-либо условиям.
Общий вид оператора:

If < условие > Then < оператор 1 >

Else < оператор 2 >;

Если условие выполняется, то выполняется только < оператор 1 >;
если условие **НЕ выполняется**, то выполняется только < оператор 2 >.

< Оператор 1 > и < оператор 2 > могут быть составными или другими операторами управления.

Пример:

$$У = \begin{cases} a + b, \text{ если } x > 0 \\ a - b, \text{ если } x \leq 0 \end{cases}$$

If x >0 Then y: = a + b
Else y: = a – b;

<
< =
>
> =
=
< >

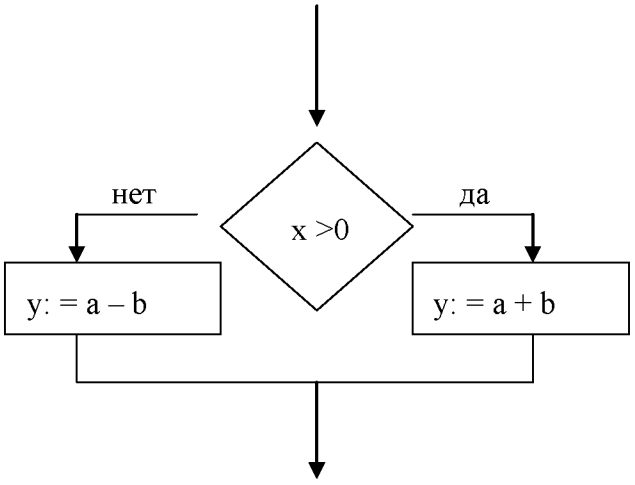


Схема выбора с двумя альтернативами

Пример 2: (неполный)

If x >0 Then y: = a + b;

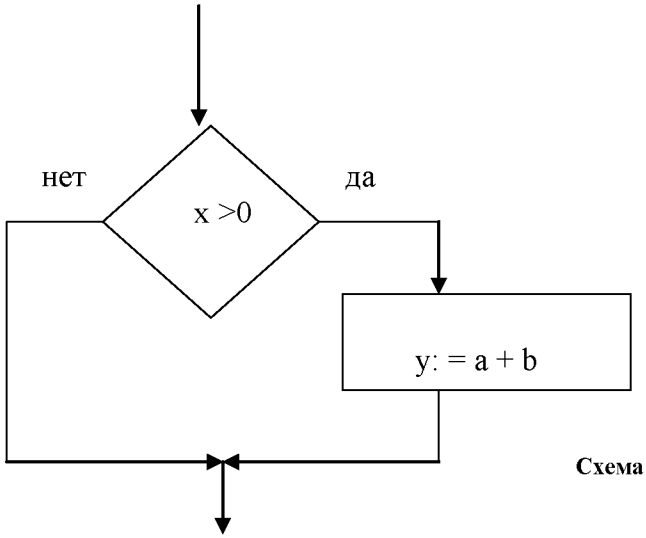


Схема выбора с одной альтернативой

If < условие > Then Begin
<оператор 1>;
<оператор 2>;
.....
<оператор N >;

End

Else Begin
<оператор 1>;
<оператор 2>;
.....
<оператор N >

End;

Пример квадратного уравнения

$$AX^2 + BX + C = 0$$

$$X_{1,2} = \frac{-b + \sqrt{b2 - 4ac}}{2a}$$

Program Prim;

Label 1;

Var a, b, c , x1, x2, d: r e a l;

Begin

```
        WriteLn(`вв. a, b, c`);

        ReadLn(a, b, c);

If    a  =  0   Then  begin

                      WriteLn(`ошибка ввода`);

                      GoTo 1

                      End;

D: = b * b – 4 * a * c;

If    d > = 0    Then  begin
                        WriteLn(`корни действ.`);
                        X1: = (-b + SQRT (d)) / (a + a);
                        X2: = (-b - SQRT (d)) / (a + a);
                        WriteLn(`x1=`, x1:8:3,`  x2=`, x2:8:3)
                        End

                  Else
                        WriteLn(`корни компл.`);

1: ReadLn;

End.
```

ЛЕКЦИЯ №6.

Циклы программ

Ветвление в программе служит организацией логической структуры.
Цикл – фрагмент программы, который выполняется последовательно несколько раз.
В паскале для организации цикла используется три специализированных оператора:

1. While
2. Repeat
3. For

Для определения количества повторов цикла (итерации цикла) используются счетчики циклов.
Счетчики цикла – это переменная, значение которой при каждом повторе меняется по определенному закону. Проверка счетчика, то есть сравнение значений счетчика с заданной величиной определяет момент завершения цикла.

В зависимости от места проверки счетчика (в начале или конце цикла) различают:

- Циклы с предусловием (While, For)
- Циклы с постусловием (Repeat)

Досрочный выход из цикла возможен из любой точки цикла и в любой момент, а вход в цикл только через его заголовок (While, For, Repeat), так как в противном случае значения счетчика могут быть неопределенными.
Счетчиком является переменная, которая проверяется в заголовке цикла.

Цикл While

Общий вид:

```
While   <условие> Do                {до тех пор, пока условие истинно , выполнить}
          Begin
             <оператор 1>;
             <оператор 2>;
             <оператор N>;
           End;
```

В качестве условия это операция сравнения или логического выражения.
В цикле While счетчик организует сам программист. Если счетчик неисправен, может возникнуть бесконечный цикл. Выйти можно с помощью: **CTRL + break.**
В этом цикле условие определяет продолжение цикла.

Например: табуляция функции (построение значений некой функции)

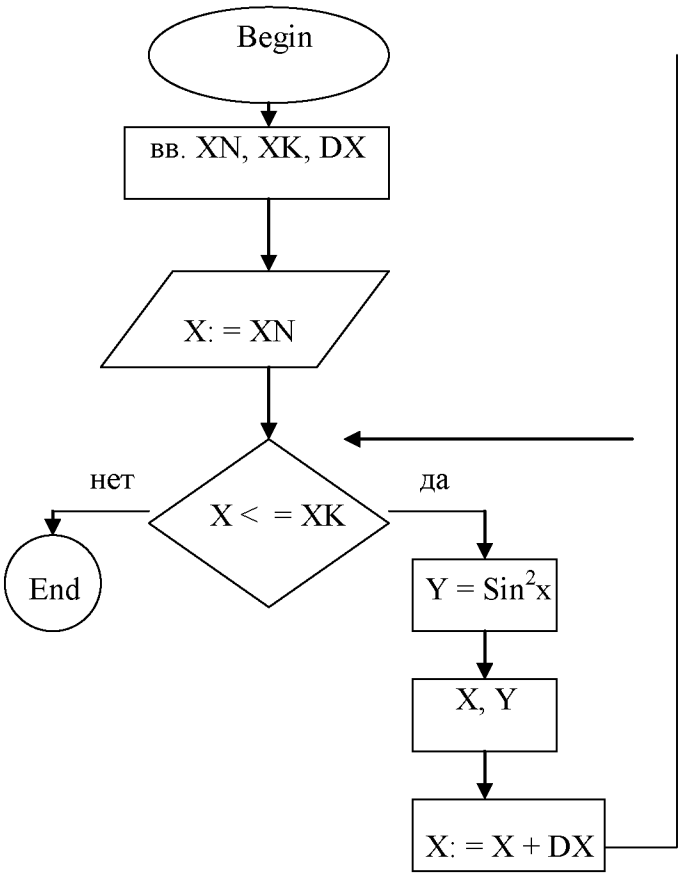
$$Y = Sin^2x$$

Xn- вводим с клавиатуры;
Xk – вводим с клавиатуры;

△ X – вводим с клавиатуры.

```
Program  Prim;
Uses  CRT;
Var   XN, XK, DX, X, Y: Real;
Begin
   Clrscr;
      WriteLn('вв. XN, XK, DX');          } установка
      ReadLn(XN, XK, DX);                 } начальных
         X: = XN;                           значений
            While   X <  = XK   do
                     begin
                          Y: = SQR(Sin(x));
                          WriteLn('X=', X:7:2,'«пробел» «пробел» y=', y:6:2);
                          X: = X + DX                    {счетчик цикла}
                     End;
ReadLn;
End.
```

Блок – схема этой программы



Оператор Repeat

```
Repeat
        <оператор 1>;
        <оператор 2>;
        <оператор N>;
Until  <условие выхода>;
```

Так как условие проверяется в конце цикла, цикл будет выполнен, обязательно, хотя бы один раз.

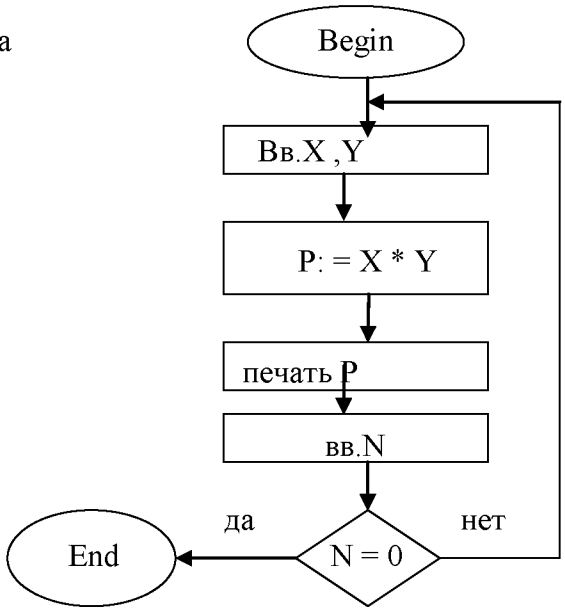
Например:

Программа умножения двух чисел. Завершение программы определяет оператор. Условие задает сам оператор в процессе.

```
Program  Prim;
Uses  CRT;
Var      X, Y, P: Real;
              N,I: Integer;
Begin
Repeat
  Clrscr;
     WriteLn('вв.2 числа');
     ReadLn(X, Y);
            P: = X * Y;
              WriteLn('P = ', P:10:2);
      WriteLn('1 – прод.', 0 – выход');
      ReadLn(N);
Until    N = 0;
End.
```

Блок – схема этой программы

- точка возврата



Оператор цикла For

```
For   I: = IN  to  IK    Do
              Begin
                   <оператор 1>;
                   <оператор 2>;
                   <оператор N>;
               End;
```

Цикл **For** условно считается четным циклом. Счетчик встроен конструктивно. Здесь:

- **I –** счетчик цикла
- **IK –** начальное значение счетчика
- **IN** – начальные значения

Ими могут быть:

1. целые числа
2. целые переменные
3. целые арифметические выражения

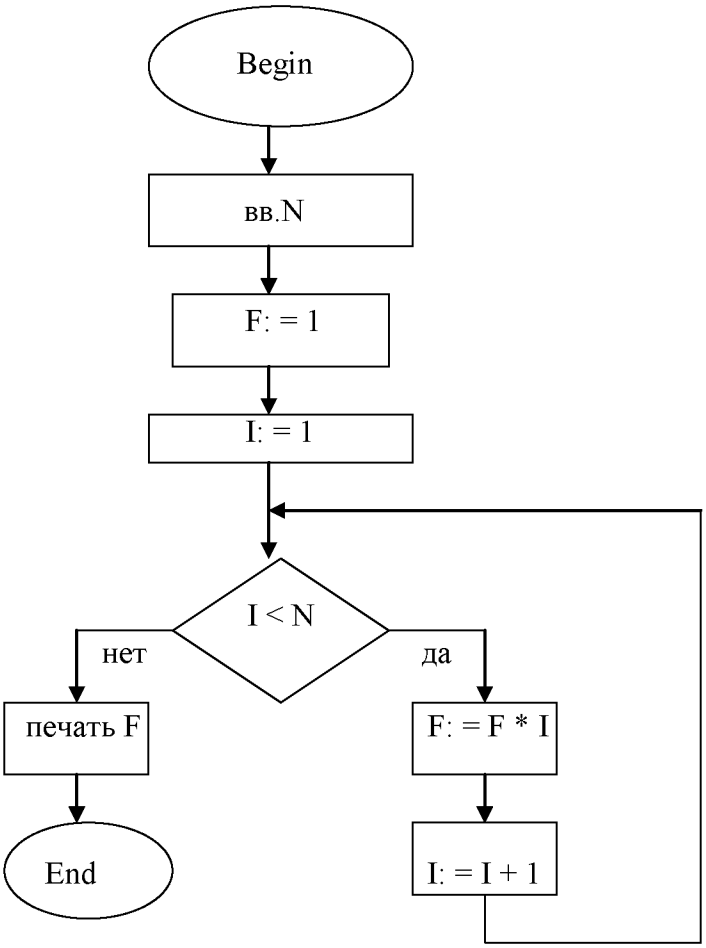
При каждом повторе, счетчик увеличивается строго на 1, поэтому **IK > IN.**

For I: = IN Do Wn To **IK**…
В этом выражении счетчик уменьшается на 1.

Пример: Вычислите **N** факториала

```
Program Prim;
Uses CRT;                                                F: = 1 * 2 * 3 * 4*..*N
Var  F,N,I: integer;
Begin
 Clrscr;
        WriteLn('вв.N');
        ReadLn(N);
            F: = 1;
             For  I: = 1 To  N  Do
               Begin
                F: = F * I;
                End;
         WriteLn('N: = ', F: = 10);
ReadLn;
End.
```

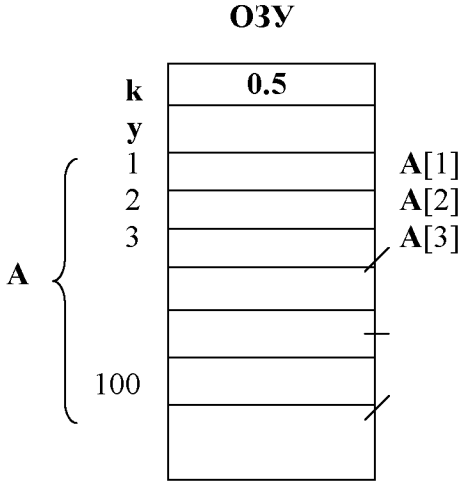
Блок – схема этой программы



ЛЕКЦИЯ №7.

Массивы

Массив – это упорядоченная последовательность однотипных элементов, имеющих общее имя**.**
По сути, **массив** – множество обычных переменных, у которых имена образуются по общей схеме – *общее имя массивы + индекс (порядковый номер)*. По этой особенности **элементы массива** называются **индексированными переменными**.



Массивы могут быть **одномерными** (линейными) и **многомерными**. Все массивы должны объявляться в разделе Var. Массивы могут объявляться непосредственно или с предварительным объявлением типа в разделе Type.

Пример 1:

```
Program Prim;
Uses  CRT;
Var  A, B, C: array [1..100]  of   Real;
```

Объявляются три одномерных массива с именами A, B, C, вещественного типа из 100 элементов с индексами 1 – 100. Значение индексов выбираем сами. Две точки (..) обозначают интервал.

Пример 2:

```
Program Prim;
Uses  CRT;
Type  Dim = array [1..100]  of   Real;
Var  A, B, C: Dim;
```

Этот вариант используются при работе с подпрограммами.

Работа с массивами.
Индексы.

В программе с элементами массивов можно работать как с обычными переменными, но значения индексов должны быть определены. Индексом может быть целое число, целая переменная, целое арифметическое выражение.

Пример:

1) A [6]: = 0,5;
2) n = 7; A[n]: = A[6];
3) A[n + 2] = A[n]

Значение индексов не должно выходить за объявленный интервал.

Пример:

В массиве B из 20 символов найти сумму всех элементов.

```
Program  Prim;
Uses  CRT;
Var  B: array [1..20]  of  Real;
       I: Integer;
       S: Real;
Begin
Clrscr;
For  I: = 1 To 20 do
 Begin
   WriteLn('A[',I:2,'] =');
   ReadLn(A[I]);
 End;

S: = 0;
For   I: = 1 To 20 Do
      S: = S + A[I];
Clrscr;
For   I: = 1 To 20 Do
     WriteLn('A[',I:2,'] =', A[I]:10:2);
     WriteLn('сумм.=', S:10:2);
ReadLn;
End.
```

Вв.масс.из 20 элем.

A[1] = …

A[2] = …

……

A[20] =…

пример ввода массива

пример распечатки массива столбиком

Дан массив А из 15 элементов. Подсчитать количество положительных и отрицательных элементов в массиве.

```
Program  Prim;
Uses  CRT;
Var  A: array [1..15]  of  Real;
      P, N, I: Integer;
Begin
P: = 0;
N: = 0
For  I: = 1 To 15 do
If  A [I] < 0 Then  N: = N + 1
              Else  P: = P + 1;
For  I:=1 To 15 do
   Writeln('A[',I,']=',A[I]:5);
Writeln('Кол-во отрицаиельных в массиве=',N);
Writeln('Количество положительных в массиве=',P);
Readln;
End.
```

Массив А и В из 20 элементов. Сформировать массив.

Ci = Ai + Bi

```
Program Prim;
Uses  CRT;
Var  A, B, C: array[1..10]  of  Real;
                          S1,S2: Real;
                             I: Integer;
Begin
S1: = 0;
S2: = 0;
For  I: = 1 To 100 do
If    I  mod  2 = 0 Then   S2: = S2 + B[I]
                     Else   S1: = S1 + B[I];
```

или

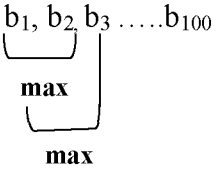
```
S1: = 0;
S2: = 0;
For   I: = 1 To 100 do
    begin
        S1: = S1 + B[2 + I  - 1];
        S2: = S2 + B[2 + I];
    End;
```

Пример определения наибольшего элемента массива

Вариант №2

```
Program  Prim;
Uses  CRT;
Var  B:array[1..100]  of   Real;
            Max, min: Real;
                    I: Integer;
Begin
……..
Max:=B[1];
Min:=B[1];
For  I: = 1 To 100 do
Begin
If    B[I]>=Max Then Max:= B[I];
If    B[I]<=Min Then Min:=B[i];
End;
```

Задача:
Выделить значение массива с наибольшими и наименьшими значениями с индексами.




```
Program Prim;
Uses  CRT;
Var  B: = array[1..100] of  Real;
              Max: Real;                    ⟶
                N: Integer;                          порядковый номер наиб.числа
                 I: Integer;
Begin
……..
Max: = B [1];
N: = 1;
For  I: = 1 To 100 do
If   B[I] >= Max   Then    Begin
                            Max: = B[I];
                            N: = I;
                            End;
Clrscr;
WriteLn('наиб.эл. – B[',N:3,'] =', Max:10:2);
Readln;
End.
```

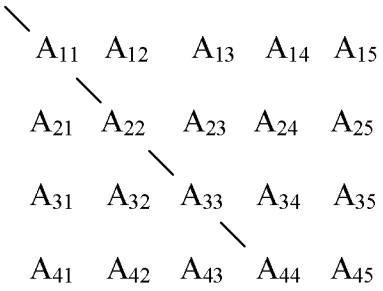
ЛЕКЦИЯ №8.

Многомерные массивы

Паскаль разрешает работу с семи - и девятимерными массивами.
Кроме одномерных массивов в Паскале используются многомерные массивы, в которых положение элемента задается несколькими индексами. Простейший вариант – **матрица** – двумерный массив. Здесь положение элемента определяется номером строки и столбца.

Пример:

Матрица 4 * 5



Для обработки матриц обычно требуется два вложенных цикла (циклы, когда один цикл внешний, другой внутренний) для перебора строк и столбцов.

В матрице 4*5 найти сумму <u>всех</u> элементов и сумму главной диагонали.

```
Program Prim;
Uses CRT;
Var A: array [1..4, 1..5] of Real;
                    S,SD: Real;
                   I, Y: Integer;
 Begin
Clrscr;
WriteLn('вв.матр.A[4*5]по стр.'];
For «I: = 1 To 4 Do     перебор строк  ------>
  For Y: = 1 To 5 Do                    ------> перебор столбцов
    ReadLn (A[I,Y];
 S: = 0;
For I: = 1 To 4 Do
  For Y: = 1 To 5 Do
    Begin
          S: = S + A[I,Y];
    If I = Y Then SD: = SD + A[I,Y];
    End;
 Clrscr;
 WriteLn ('Матр.A');
 For I: = 1 To 4 Do
  Begin
    For Y: = 1 To 5 Do
      Write(A[I,Y]: 7:2);
    WriteLn;  ------> перевод курсоры в нач.след.строки
```

внеш. цикл

внутр. цикл

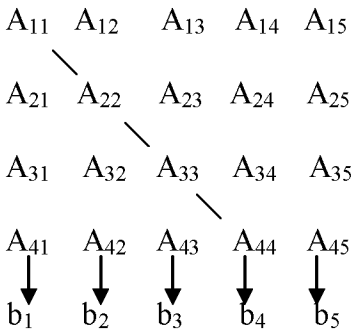
```
End;

WriteLn('Сумма главной диагонали =', SD:10:2);
WriteLn('Сумма элементов всей матрицы=',S:10:2);
ReadLn;
End.
```

Пример суммы для каждой строки отдельно:

```
Program  Prim;
Uses  CRT;
Var  A: array [1..4, 1..5]  of  Real;
                S: Real; I, Y: Integer;
Begin
Clrscr;
WriteLn('вв.матр.A[4*5]по стр.'];
 For  I: = 1 To 4 Do
    For  Y: = 1 To 5 Do
                  ReadLn (A[I,Y];
 For  I: = 1 To 4 Do
Begin
 S: = 0;
    For  Y: = 1 To 5 Do
       S: = S + A[I,Y];
  WriteLn('Сумм.стр.', I: 1,'=', S:10:2);
End;
Readln;
End.
```

Пример: в матрице А найти произведение в столбцах и сохранить их в одномерном массиве.




```
Program  Prim;
Uses  CRT;
Var  A: array [1..4, 1..5]  of  Real;
                I, Y: Integer;
                B: array [1..5]  of  Real;
Begin
```

```
Clrscr;
WriteLn(‘вв.матр.A[4*5]по стр.'];
For  I: = 1 To 4 Do
   For  Y: = 1 To 5 Do
                ReadLn (A[I,Y];
For  I: = 1 To 5 Do
Begin
  B[I]: = 1;
            For  Y: = 1 To 4 Do
               B[I]: = B[I] * A[I,Y];
End;
WriteLn;                          }   распечатка
For  I: = 1 To 5 Do               }   одномерного
WriteLn(B[I]:10:2);               }   массива
```

ЛЕКЦИЯ №9.

Текстовые данные.
Символьный тип Char.

- Объявляет, что значением переменной или элемента массива является символ (один); можно сказать изображение символа.

```
Var  A, B, C: Char;
     Dim: array [1..40]  of  Char;
Begin
A: = 'x';
B: ='y';
Dim [1]: = '+';
WriteLn(A, dim[1],B);
```

На экране появится информация - x+y

Каждый символ в компьютере имеет свой код. Символы размещаются в так называемых кодировочных таблицах. Набор символов в этих таблицах определяется ASCII – стандарт. Этот стандарт был изначально разрешен для телеграфа, включал те символ, которые были на клавиатуре телеграфа.
Включал в себя 128 символов, и кодировались они семью двоичными разрядами. Этот стандарт включал в себя: цифры, знаки препинания, латинские буквы. С переходом на вычислительную технику стандарт был расширен до 256 символов и символы стали кодироваться восемью двоичными разрядами (байт).

Содержание таблиц

0 ÷ 31	→	управляющие
32 ÷ 126	→	лат.алфавит, цифры, знаки препинания
127	→	служебный код
128 ÷ 255	→	национальные алфавиты + специальные символы

Первые 32 символа имеют свои изображения, которые мы практически не видим.

Работа с символьными данными

1. Символы можно сравнивать друг с другом, при этом сравниваются их коды.

X код {88}
Y код {89}

Пример:

```
Если X < Y
If 'Y' > 'X' Then  WriteLn('Y > X')
               Else   WriteLn('X < Y');
```

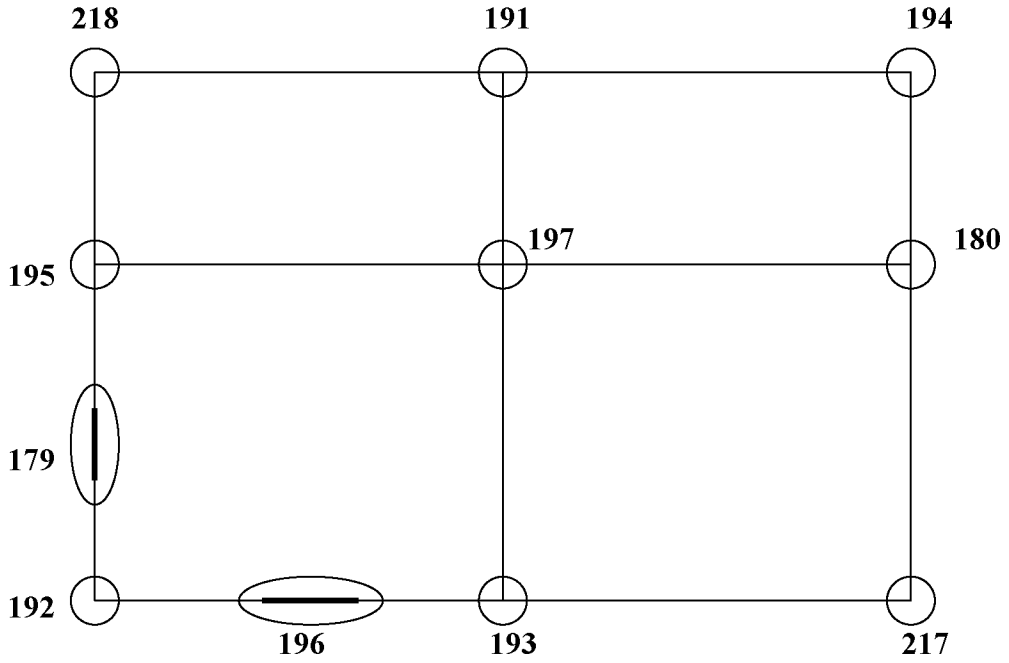
2. Функция Ord (w) – возвращает код символа или символьные переменные W.

```
N: = Ord('X');
{N=88}
```

3. Функция Chr (n) – возвращает символ (изображение символа) по его коду n.

```
B:  = Chr(88);
{B='X'}
```

Элементы псевдографики
для построения таблицы



Отдельные элементы граф., из которых можно собрать изображения любой таблицы.

Они позволяют создавать непрерывные линии в таблицах, код **179** и **196**.

< Alt > + код – вызвать символ.

!Только на маленькой клавиатуре должен набираться **сам код символа!**

Строковой тип данных String.

- Объявляет, что значения переменной или элемента массива является строка символов (текстовая строка).

```
Var  S: String[40];                    ——————→ здесь мы явно указываем длину текста
     S1, S2, S3: String;
Begin
```

Во втором случае длина – по умолчанию, первые S1, S2, S3 – резерв строки до 255 символов длинной (максимальная длина строки – 255 символов)

Работа с символьными строками

1. строки можно объединять (складывать)

```
Var  S: String[40];
     S1, S2, S3: String;
 Begin
 S1: = 'Turbo «пробел»';
 S2: = 'Pascal';
 S3: = S1 + S2;
 WriteLn(S3);  ——————————→ Turbo «пробел» Pascal;
```

2. строки можно сравнивать, при этом коды символов сравниваются последовательно слева направо.
   Первый отсутствующий символ имеет уникальный (наименьший) код.

   'ABC' > 'AB'

3. элементы строки можно выделить по их порядковым номерам (индексам).

   WriteLn(S3[4]); ——————→ b (на экране)

   В произвольной строке определить наличие символа единица ('1')

```
Var «пробел» S: String;
            I, K: Integer;
Begin
WriteLn('вв.строку');
ReadLn(S);                          {S = 'страница «пробел» 112'}
K: = 0;
For  I: = 1 To 12 Do
   If  S[I] = '1] Then Begin
                       WriteLn('1 в позиции «пробел»', I:2);
                        K: = 1;
                        End;
   If  K = 0 Then WriteLn('симв.1 нет');
```

Процедуры и функции обработки строк

1. процедура **Delete** (S, N, L);
   удаляет из строки ***S*** с позиции ***N*** символов ***L***

2. процедура **Insert** (P, S, N);
   помещает (вставляет) строку ***P***, в строку ***S*** с позиции ***N***

3. процедура **Str** (X:m:n, S);
   преобразует число ***X*** в тексте ***S***. ***m*** и ***n*** – форматы числа. (***m – общая длина числа; n – длина общей части)***

   Str(3.14:4:2, S)
   S: = ‘3.14’ ⟶ получившийся текст

4. процедура **Val** (S, X, K);
   обратное действие: преобразует текстовую строку ***S*** в число ***X***, а ***K*** - служебное целое переменное.

   S: = ‘100.2’; ⟶ это не число!
   Val(S, X, K);
   {K = 2}
   X: = 100.2

5. функция **Copy** (S, N. L);
   возвращает (копирует) часть символа ***S*** с позиции ***N*** длинной ***L.***

6. функция **Length** (S);
   возвращает длину строки ***S***

7. функция **Pos** (P, S).
   определяет наличие строки ***P*** в строке ***S,*** возвращает порядковый номер первого совпавшего символа

Процедуры оформляются как отдельные операторы и закрываются « **;** ».
Функции входят в состав выражений.

Операторы Break, Continue

Эти операторы используются в циклах. Оператор **Break** выполняет досрочный выход из цикла. Управление передается на следующий за циклом оператор. Значение параметров цикла сохраняются на момент выхода (такими, какими были). Действие оператора **Break** аналогично действию оператора **Continue.**

Оператор **Continue**

- Выполняет досрочное завершение текущей итерации цикла и начинает новую итерацию. В циклах **While** и **Repeat** оператор **Continue** значение счетчиков сам не меняет. В цикле **For** счетчик меняется автоматически.

Пример:

Дан некий большой массив [1..1000] of Real;

```
Program Prim;
Var  A: array [1..1000] of Real;
              N, I: Integer;
Begin
……..
N: = 0;
For  I: = 1 To 1000 Do
 Begin
    If  A [I] < 0 Then
                Begin
                   N: = 1;
                   Break
                End;
 End;
If  N = 1  Then  WriteLn('Отрицательные числа есть')
           Else   Writeln('Отрицательных чисел нет');
ReadLn;
……
```

Операторов выбора **Case**

Оператор работает как оператор **If**, но с большим числом вариантов.

Общий вид оператора:

Case <КЛЮЧ> of
C1: < оператор 1 >;
C2: < оператор 2 >;
……
Cn: < оператор n >
[Else «пробел» < оператор S >] – необязательный элемент оператора

End;
Ключ – целая переменная или целое арифметическое выражение.
Эта переменная или выражение может принимать любые значения (Ci). Этот оператор может использоваться для организации меню в программе. Его действия аналогичны действию нескольких операторов If.

Пример:
Ввести с клавиатуры цифру 0-9. вывести на экран ее название.

```
Program  Prim;
Var  N: Integer;
Begin
WriteLn('вв.цифру 0-9');
ReadLn(N)
Case  N  of
  0: WriteLn('ноль');
   ……..
  6: WriteLn('шесть');
   ……...
  9: WriteLn('девять')
  Else writeLn ('неверный ввод');
End;
ReadLn;
End.
```

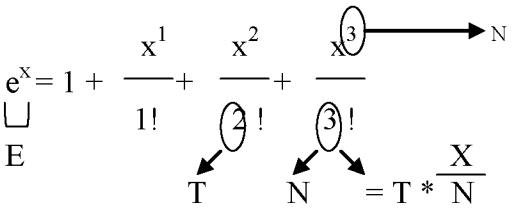
В качестве значения можно указывать либо отдельное число (**1:**), либо список, через запятую (**2,4,6:**), либо диапазон значений (**7..9:**).

ЛЕКЦИЯ №10.
Итерационные циклы

- Циклы, для которых условием завершения является достижение заданной точности вычисление результата.

Примером является вычисление бесконечного рода.

Рассчитаем ряд, который образует значение функции E^X



Точность вычисление такого рода соответствует значению последнего рассчитанного элемента ряда.

```
Program «пробел» Prim;
Var  N: Integer;
       E, T, X, D: Real;
Begin
WriteLn('вв. степень X');
ReadLn(X)
WriteLn('вв.погр.D');
ReadLn(D);
  N: = 1;
  T: = 1;
  E: = 1;
Repeat
T: = T * X / N;
E: = E + T;
N: = N + 1;
Until  D > T;
WriteLn('Exp(', X:8:3,') = ', E:12:3);
ReadLn;
End.
```

текст **текст**

Exp(…) = ….

число **число**

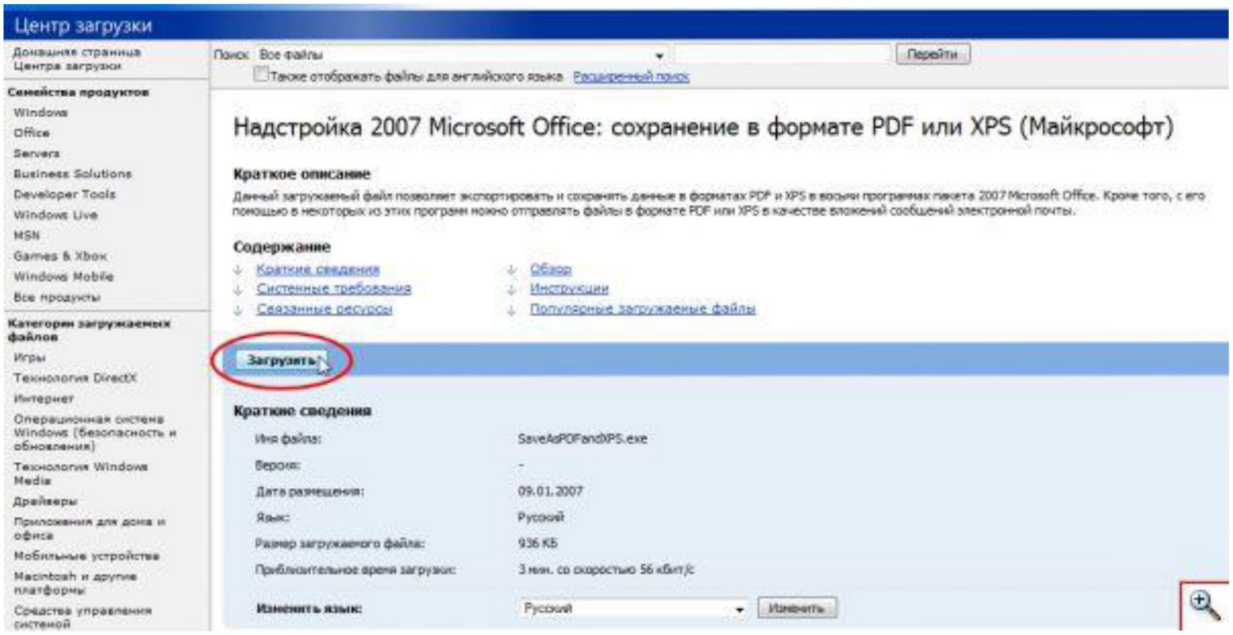
X E

Чтобы в Microsoft Office 2007 сохранять документы в формате PDF необходимо установить на компьютер бесплатную надстройку (дополнение) к офисному пакету компании Microsoft.

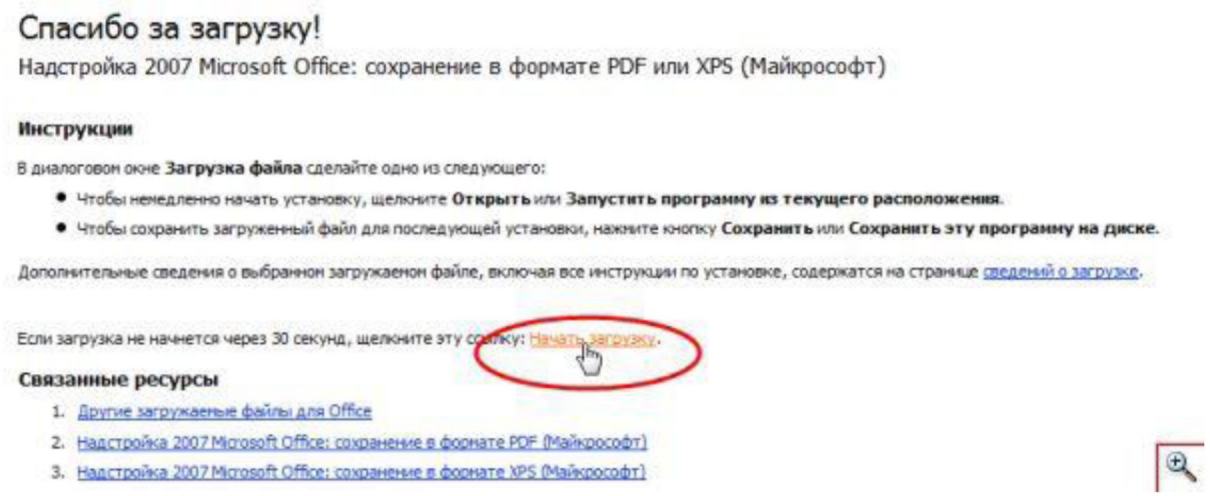
Дополнение от Microsoft встраивается в Office 2007 и позволяет сохранять документы в PDF формате, воспользовавшись стандартной командой «Сохранить как».

Если быть точнее, у Microsoft имеется две надстройки для Office 2007: первая позволяет сохранять документы только в формате PDF, вторая – в PDF и XPS. Как известно, лишних возможностей не бывает, поэтому мы скачаем и установим второе дополнение.

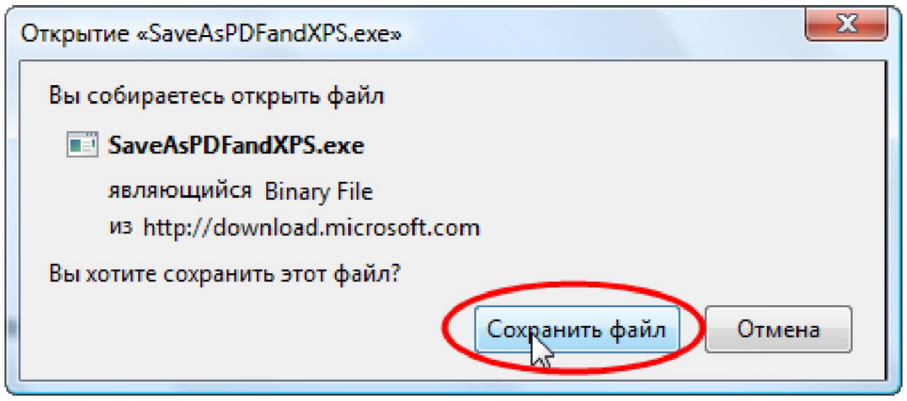
Для загрузки дополнения перейдите на следующую страницу:
http://www.microsoft.com/downloads/details.aspx?FamilyId=4D951911-3E7E-4AE6-B059-A2E79ED87041&displaylang=ru
Ознакомьтесь с информацией о надстройке и нажмите кнопочку «Загрузить», чтобы скачать ее на свой компьютер.



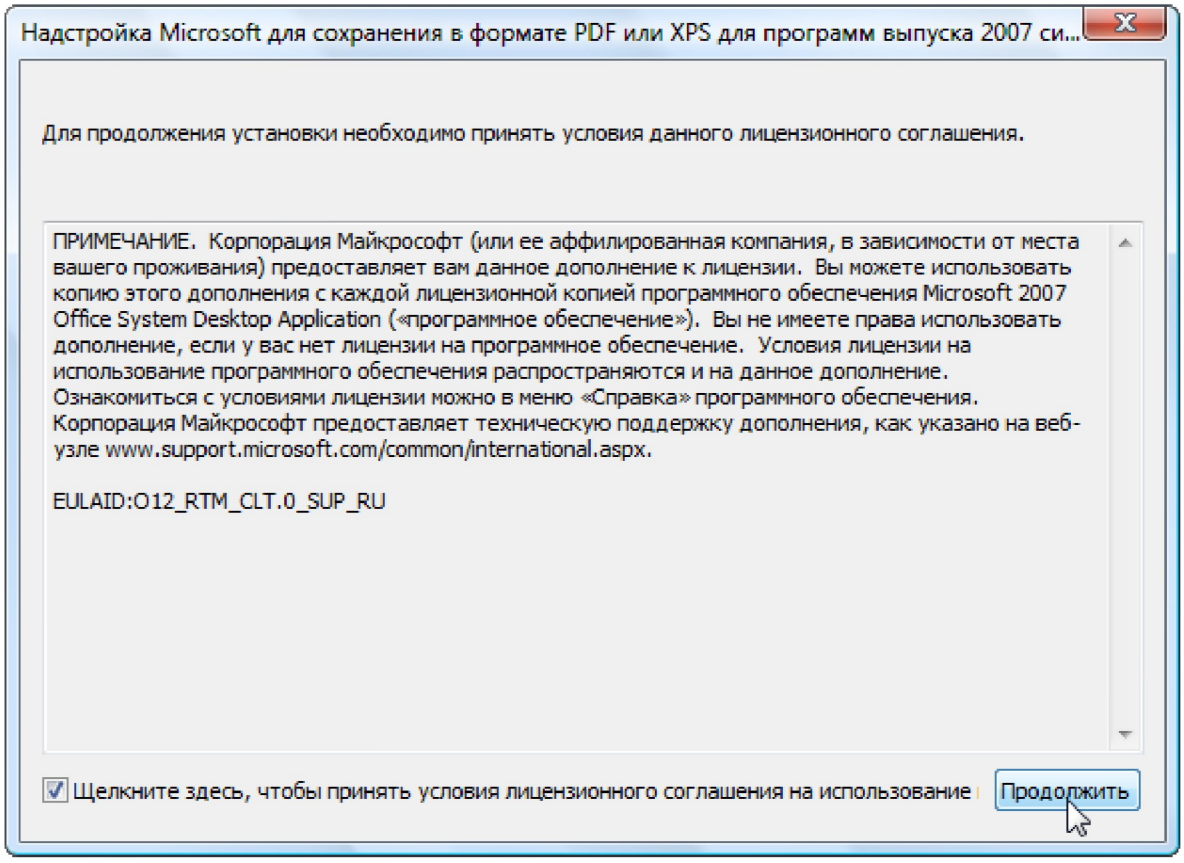
Если в вашем браузере отключен JavaScript или загрузка дополнения автоматически не началась через 30 секунд, щелкните ссылку «Начать загрузку».



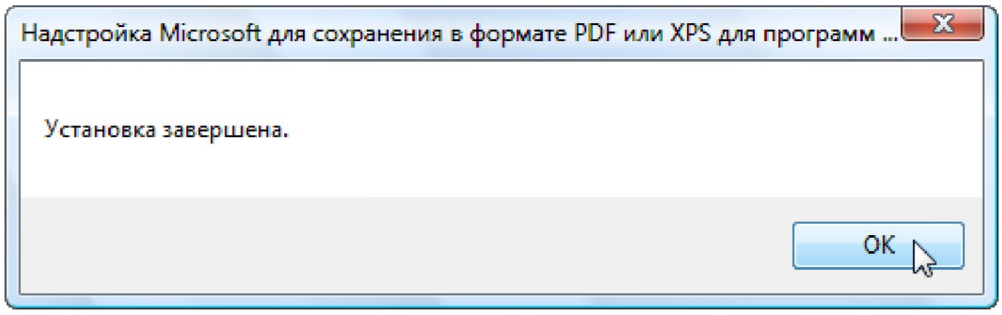
В появившемся окошке, нажмите кнопку «Сохранить» или «Сохранить файл» (в зависимости от используемого браузера) и выберите на компьютере папку для сохранения файла. Размер надстройки PDF – 935 Кб.



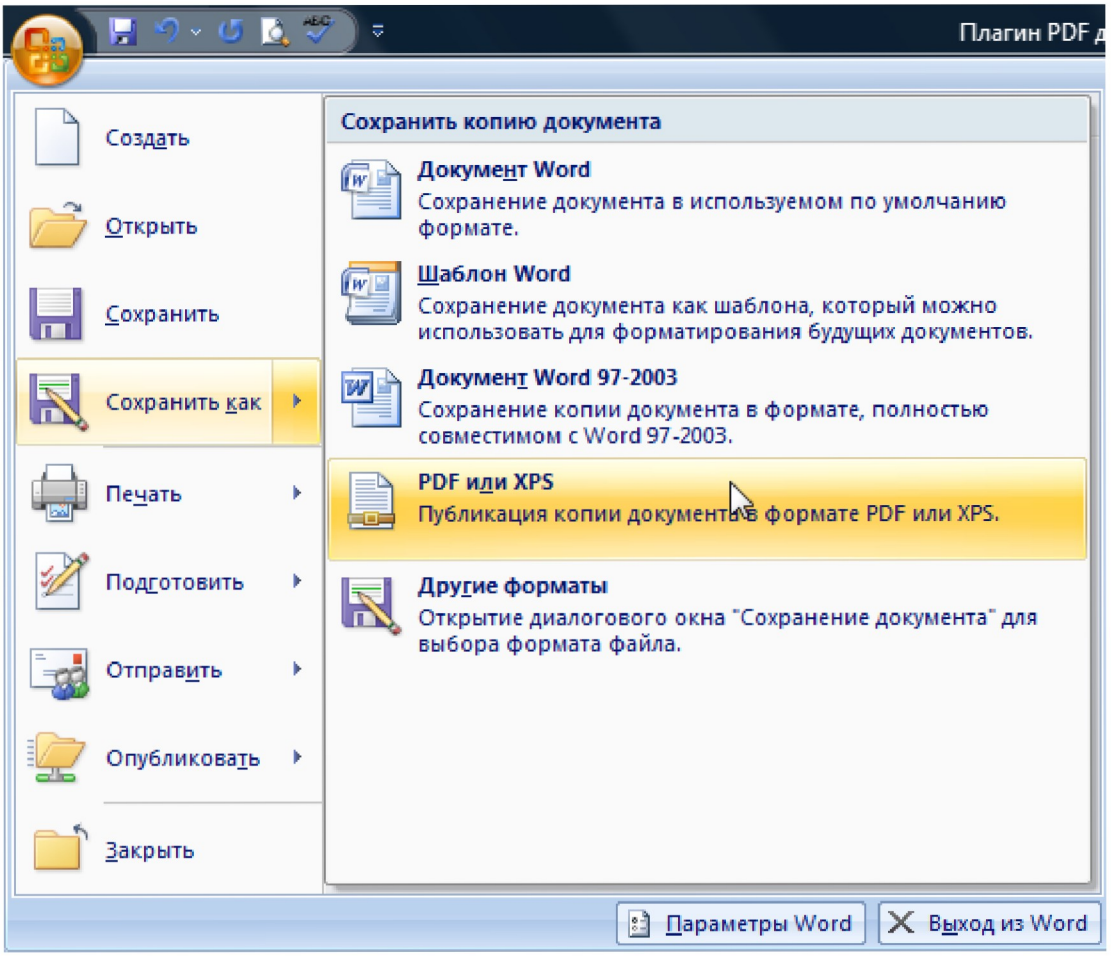
Для начала установки запустите сохраненный EXE файл. При этом все программы из пакета Microsoft Office 2007 должны быть закрыты. Вначале инсталляции надстройки вам необходимо будет прочитать и принять лицензионное соглашение. Для продолжения нажмите кнопку «Продолжить».



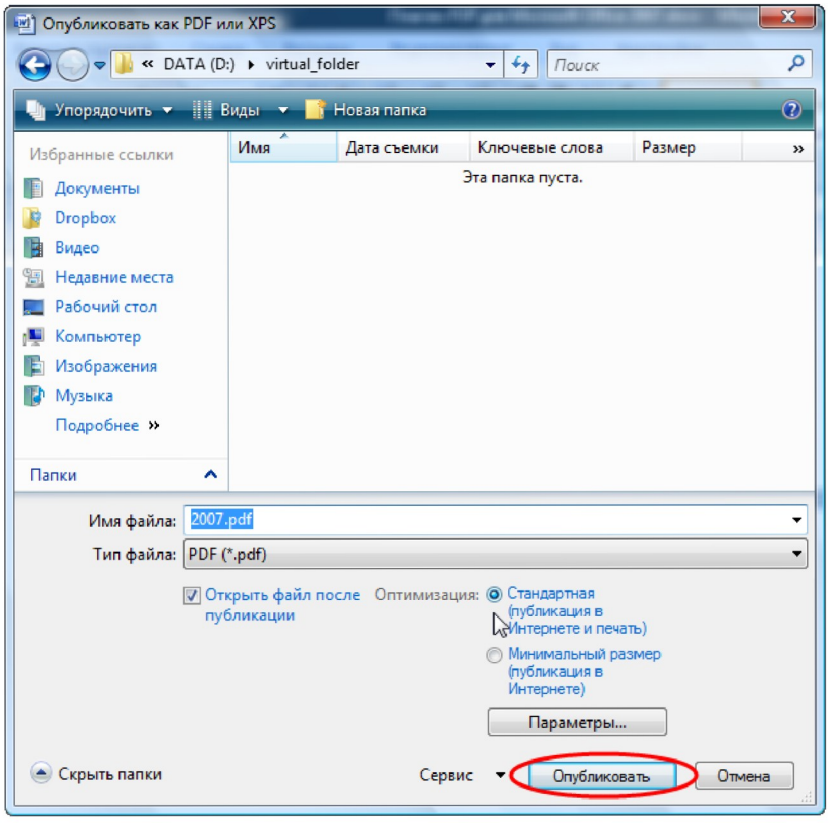
Установка дополнения занимает 2-5 секунд. По окончании оного процесса нажмите кнопку «ОК».



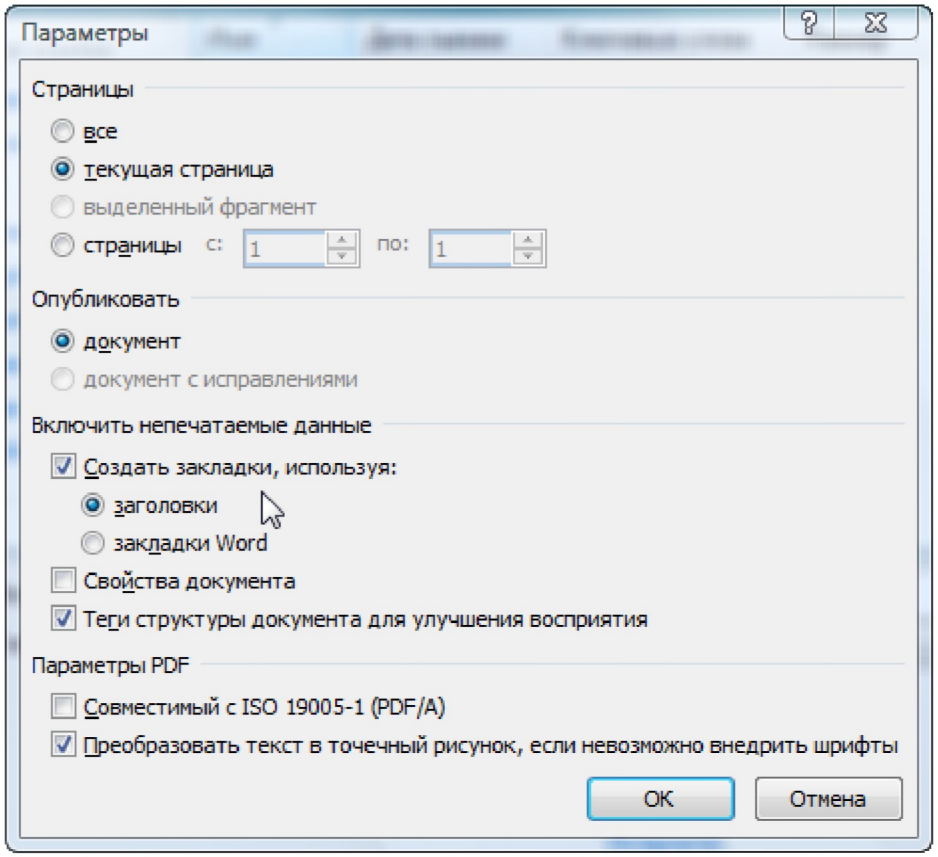
Теперь откройте в Microsoft Word нужный документ и выберите в главном меню «Сохранить как» - «PDF или XPS».

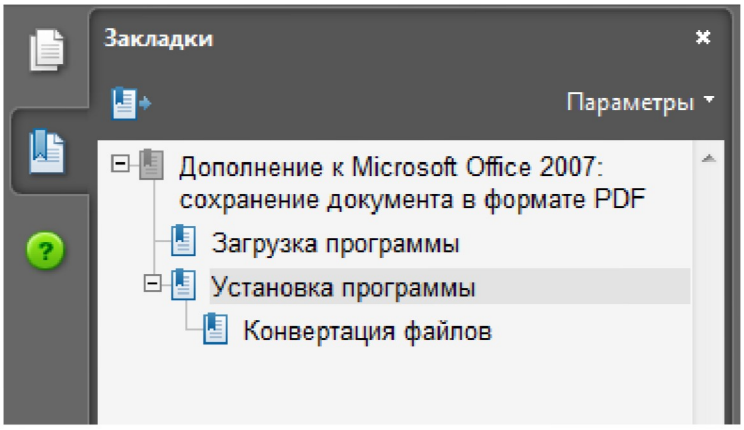


В открывшемся окне выберите место сохранения PDF файла и введите его название. Если в документе нет цветных изображений, и он предназначен для публикации в интернете, отметьте в разделе «Оптимизация» опцию «Минимальный размер». В противном случае используйте стандартную оптимизацию. Для сохранения документа в PDF формате нажмите кнопочку «Опубликовать».

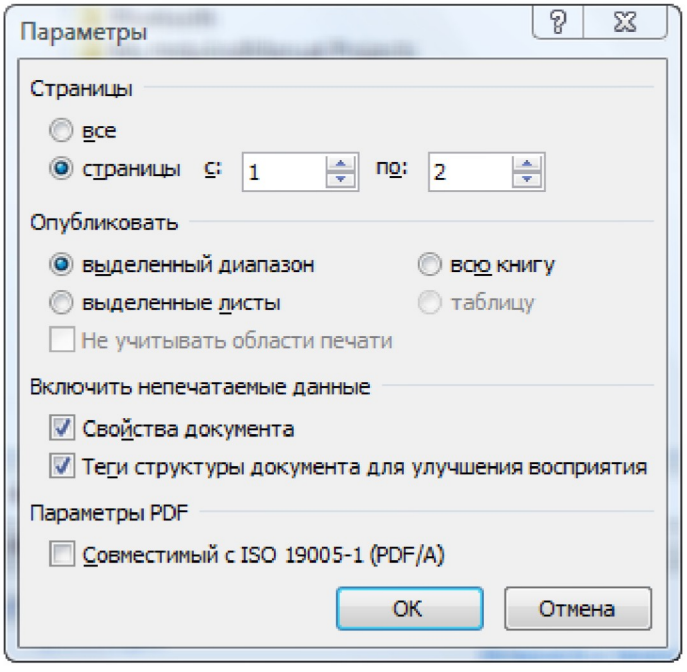


В этом окошке также есть кнопка «Параметры», щелкнув по которой вы сможете настроить некоторые опции сохранения документа в PDF формате: страницы, закладки, включить в файл свойства документа и т.д. Если документ содержит заголовки и подзаголовки, можно отметить опцию «Создать закладки, используя» - «заголовки».





В таком случае заголовки и подзаголовки будут играть роль оглавления документа и отобразятся на Панели закладок программы для просмотра PDF файлов. Выбрав нужную закладку, пользователь сможет быстро перейти к соответствующему ей содержанию.



Сохранение таблицы Microsoft Excel в формате PDF происходит аналогично. Просто откройте нужную книгу Excel и выберите в главном меню программы «Сохранить как» - «PDF или XPS», а затем нажмите кнопку «Опубликовать».
Параметры сохранения файла в формате PDF, которые доступны из Microsoft Excel, несколько отличаются от тех, которые вы видели ранее в Word. Поскольку в Excel вы работаете с книгами и листами, то здесь имеется возможность выбора книги, листов, страниц и диапазона ячеек, которые будут сохранены в формате PDF.