МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ЛЕСОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра интеллектуальных систем

Мельник Л.Ю.

Информатика

Методические указания по выполнению
лабораторно-практического цикла по ***PascalABC***
для студентов всех направлений и
специальностей

ЕКАТЕРИНБУРГ 2023

ОБЩИЕ ПОЛОЖЕНИЯ
Среда / Язык программирования PascalABC

Для того чтобы запустить /вызвать среду **PascalABC** следует:

- *или* выбрать **Пуск**, **Программы**, **PascalABC;**
- *или* воспользоваться ярлыком **PascalABC.**

Для завершения работы выполните команду **Файл/Выход** или закройте окно редактора щелчком на кнопке «×».

Окно PascalABC
Меню

Меню **PascalABC** русифицировано и интуитивно понятно.

- **Файл.** Пункт содержит команды, задающие действия над файлами. С помощью этой команды можно создать, открыть, сохранить, распечатать программу, закончить работу с программой.
- **Правка.** Позволяет редактировать документ: отмечать, копировать, удалять.
- **Программа.** Этот пункт меню позволяет использовать команду для компиляции, запуска программы и дальнейшего выполнения ее.

ПЛАН РАБОТЫ

Выполните следующие действия:

1. Запустите программу **PascalABC** .
2. Создайте проект или воспользуйтесь открытым окном **PascalABC.**
3. Наберите текст программы.
4. Сохраните текст программы в файл с именем ***labNN.pas*** где ***NN*** – номер лабораторной работы.
5. Проверьте программу на синтаксические ошибки (пункт меню ***Программа/Компилировать*** или воспользуйтесь горячими клавишами.
6. Если нет ошибок, запустите программу на выполнение**,** в противном случае повторите пункты с 4 по 6. с ошибками.
7. Результаты выполнения программы можно посмотреть в окне вывода.
8. Проверить программу на правильность выполнения с контрольным примером, если результат не верен, исправить ошибки и повторить пункты с 4 по 7.

ЛАБОРАТОРНАЯ РАБОТА № 1

ТЕМА. Программирование линейных алгоритмов.

Основные типы данных

К основным типам данных языка Pascal относятся:

- целые числа (`integer` и др.);
- действительные числа (`real` и др.);

- символы (`char`);
- строки (`string`);
- логический (`boolean`).

Целые числа и числа с плавающей точкой могут быть представлены в различных форматах.

Идентификатор	Длина (байт)	Диапазон значений	Операции
Целые типы			
`integer`	2	-32768..32767	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
`byte`	1	0..255	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
`word`	2	0..65535	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
`shortint`	1	-128..127	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
`longint`	4	-2147483648..2147483647	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
Вещественные типы			
`real`	6	$2{,}9x10^{-39}$ - $1{,}7x10^{38}$	+, -, /, *, >=, <=, =, <>, <, >
`single`	4	$1{,}5x10^{-45}$ - $3{,}4x10^{38}$	+, -, /, *, >=, <=, =, <>, <, >
`double`	8	$5x10^{-324}$ - $1{,}7x10^{308}$	+, -, /, *, >=, <=, =, <>, <, >
`extended`	10	$3{,}4x10^{-4932}$ - $1{,}1x10^{4932}$	+, -, /, *, >=, <=, =, <>, <, >
Логический тип			
`boolean`	1	true, false	Not, And, Or, Xor, >=, <=, =, <>, <, >
Символьный тип			
`char`	1	все символы кода ASCII	+, >=, <=, =, <>, <, >

Примеры описания типов данных

Пример объявления одной переменной целочисленного типа:

```
var
  i: integer;
```

Пример объявления стразу 3 переменных (a, b, c) типа integer.

```
begin
  // Тут код программы
end.
var
  a, b, c: integer;
```

Пример присваивание значений переменным: Сначала объявляем переменные, затем присваиваем значения.

```
Var   {объявление переменных}
  i: integer;
  r: real;
  s: string;
  b: boolean;

begin
 {присваиваем значения переменным}
  i := 10;
  r := 1.2;
  s := 'Hello World';
  b := True;
end.
```

Как видно из этого примера каждой переменной можно присвоить определённый тип данных.

Можно присвоить значение переменной, сразу после объявления:

```
var
  i: integer := 10; // Сразу присвоили значение
  s := 'Hello World'; // Можно присвоить значение, без
объявления типа.

begin
  WriteLn(i);
  WriteLn(s);
end.
```

В этом примере я использовал процедуру `WriteLn` для вывода переменной на экран.

Структура программы на Паскале:

```
Program <Имя программы>;
  Label <раздел описания меток>;
  Const <раздел описания констант>;
  Uses  <раздел описания стандартных модулей>;
  Type  <раздел описания типов>;
  Var   <раздел описания переменных>;
Procedure (Function) <раздел описания подпрограмм>;
   Begin
   <раздел операторов>
   End.
```

Заключенный текст в {..} является комментариями к программе.

Для любой программы обязательным является лишь раздел операторов. Все программные объекты (константы, переменные, типы и пр.) должны быть описаны в соответствующих разделах описаний.

Здесь и в дальнейшем служебные слова Паскаль будут выделяться полужирным шрифтом. Служебными называются слова, значения которых в языке однозначно определены.

Программы с линейной структурой составляются из процедур присваивания, ввода, обращения к процедурам. Оператор присваивания можно назвать основным в любом языке программирования. Оператор присваивания:

<переменная> := <выражение>

Оператор выполняется следующим образом. Вычисляется значение <выражения>, после чего <переменная> получает вычисленное значение. При этом тип выражения должен быть совместим с типом переменной.

Примеры оператора присваивания:

```
X:=(Y+Z)/(2+Z*10)-1/3;
LogPer :=(A>B) and (C<=D).
```

Выражение может включать в себя константы, переменные, знаки операций, функции, скобки. В результате вычисления выражения получается значение определенного типа.

Тип выражения определяется типом полученного значения.

Арифметическое выражение — выражение числового типа (целого или вещественного). Идентификатор целого типа: integer, вещественного типа: real.

Арифметические операции бывают унарными и бинарными. К унарным относится операция изменения знака. Ее формат: — <*величина*>.

В следующей таблице представлены бинарные арифметические операции Паскаль. A и B обозначают операнды, для типов величин использованы обозначения: I — целый, R — вещественный.

Выражение	**Типы операндов**	**Тип рез-та**	**Операция**
A + B	R, R	R	Сложение
	I, I	I	
	I, R R, I	R	
A - B	R, R	R	Вычитание
	I, I	I	
	I, R R, I	R	
A * B	R, R	R	Умножение
	I, I	I	
	I, R R, I	R	
A/B	R, R	R	Вещественное
	I, I	R	деление
	I, R R, I	R	
A div B	I, I	I	Целое деление
A mod B	I, I	I	Остаток от целого деления

Стандартные математические функции Паскаль представлены в следующей таблице:

Обращение	**Тип аргумента**	**Тип рез-та**	**Функция**
abs (x)	I, R	I, R	Модуль аргумента
arctan(x)	I, R	R	Арктангенс(радианы)
cos (x)	I, R	R	Косинус (x в радианах)
exp (x)	I, R	R	e^x— экспонента
frac(x)	I, R	R	Дробная часть x
int(x)	I, R	R	Целая часть x
1n**(x)**	I, R	R	Натуральный логарифм
random		R	Псевдослучайное число в интервале [0,1]
random (x)	I	I	Псевдослучайное число в интервале [0,x]
round(x)	R	I	Округление до ближайшего целого
sin(x)	I, R	R	Синус (x — в радианах)
sqr (x)	I, R	R	Квадрат x
sqrt(x)	I, R	R	Корень квадратный
trunk(x)	R	I	Ближайшее целое, не превышающее x по модулю

Старшинство операций (в порядке убывания приоритета):
=> вычисление функции;
=> унарный минус;
=> *, /, `div, mod;`
=> +, -

Возведение положительного числа в вещественную степень следует производить, используя следующее математическое тождество: $x^y=e^{y \ln x}$ На Паскале это записывается так: `exp(y*ln(x))`

Пример 1. Записать математические выражения в виде арифметических выражений на Паскале.

Математическое выражение	Выражение на Паскале
$x^2 - 7x + 6$	`Sqr(x) - 7*x+6`

Ввод данных с клавиатуры производится путем обращения к стандартным процедурам:

read(<список ввода>) **readln** (<список ввода>)

Элементы списка ввода — идентификаторы переменных. Вводимые значения отражаются на экране. При выполнении оператора пользователь

набирает на клавиатуре соответствующую последовательность значений, разделяя их пробелами.

Вывод данных на экран производится путем обращения к стандартным процедурам:

write(<список вывода>) **writeln**(<список вывода>)

Элементы списка вывода — константы, переменные, выражения, форматы вывода.

Пример 2. Скорость первого автомобиля v_1 км/ч, второго — v_2 км/ч, расстояние между ними s км. Какое расстояние будет между ними через t ч, если автомобили движутся в разные стороны?

Решение. Согласно условию задачи искомое расстояние $s_1 = s + (v_1 + v_2)t$ (если автомобили изначально двигались в противоположные стороны) или $s_2 = |(v_1 + v_2)t - s|$ (если автомобили первоначально двигались навстречу друг другу).

Программа организует ввод исходных данных, вычисление искомых величин по формулам и вывод их на экран. Все величины в программе — вещественного типа.

```
Program Car;
Var VI, V2, T, S, SI, S2: Real;
Begin
Write('Введите скорости автомобилей, расстояние
между ними и время движения: ');
ReadLn (VI, V2 , S, T);
SI:= S+(V1+V2)*T;
S2:=Abs( (VI + V2)*T-S);
WriteLn('Расстояние будет равно ',S1:7:4,'км или
',S2:7:4, ' км')
End.
```

Тестовый пример. VI=50 км/ч, V2=70 км/ч , S=1000 км, T=1час
S1=1120 км
S2=880 км

Логические выражения в результате вычисления принимают логические значения **true** или **false**. Операндами логического выражения могут быть логические константы, переменные логического типа, отношения. Идентификатор логического типа в Паскале: ***boolean***.

Логические операции. В Паскале имеются 4 логические операции: отрицание — **NOT**, логическое умножение — **AND**, логическое сложение — **OR**, исключающее «или» — **XOR**. Результаты логических операций для различных значений операндов приведены в таблице. Использованы обозначения: T — true, F — false.

A	B	not A	A and B	A or B	A xor B
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

Приоритеты логических операций:

1) not; 2) and; 3) or; 4) xor.

Примеры логических выражений:

1) True; 2) False; 3) A>B; 4) (A=B) and (C<=D) .

Операции отношений (= , <>, <=, <=, <, >) имеют более низкий приоритет, чем логические операции, поэтому их следует заключать в скобки при использовании по отношению к ним логических операций.

Задание к лабораторной работе №1

1. Создайте свою папку на своем носителе, где будут находиться тексты программ на языке **PascalABC**.
2. Получите задание у преподавателя по теме лабораторной работы *Линейные алгоритмы*.
3. Подготовьте текст программы и контрольный пример по своему заданию.
4. Используя план работы в **PascalABC**, описанный выше, выполните свой вариант.
5. Результаты выполнения программы должны совпадать с результатами контрольного примера.
6. Если нет ошибок, контрольный пример совпадает с результатом выполнения программы, результаты лабораторной работы покажите преподавателю.

ЛАБОРАТОРНАЯ РАБОТА № 2

ТЕМА. Программирование ветвящихся алгоритмов

Для программирования ветвящихся алгоритмов применяются условный оператор (оператор ветвления) и оператор выбора.

Условный оператор имеет следующий формат:

```
if <логическое выражение> then <оператор 1> else
                    <оператор 2>;
```

Оператор 1 и 2 могут быть простыми или составными.

Если логическое выражение, выступающее в качестве условия ветвления, принимает значение False, то выполняется **оператор 2**, если True — **оператор 1**.

Неполная форма условного оператора:

```
if <логическое выражение> then <оператор>;
```

Пример 1. Из трех данных вещественных чисел X, Y, Z выбрать наибольшее.

Решение 1. Используем алгоритм с вложенными полными ветвлениями.

```
Program Max3_1;
Var X,Y,Z,MAX: real;
begin
write ('Введите X, Y, Z' );
readln(X, Y, Z);
  if X>=Y then
  if X>=Z then MAX: =X
          else MAX: =Z
          else
  if Y>=Z then MAX : =Y
          else MAX : =Z ;
writeln ('Максимальное значение=' , MAX:4:2)
end.
```

Решение 2. Используем алгоритм с последовательными неполными ветвлениями и сложными логическими выражениями.

```
Program Max3_2;
Var X, Y, Z: real;
begin
write (' Введите X, Y, Z' ) ;
readln(X, Y, Z) ;
    if (X>=Y) and (X>=Z) then MAX: =X;
    if (Y>=X) and (Y>=Z) then MAX :=Y;
    if (Z>=X) and (Z>=Y) then MAX :=Z;
writeln ('Максимальное значение=' , MAX);  end.
```

Пример 2. Дано действительное число а. Вычислить ***f(x)***, если

$$f(x) = \begin{cases} 0 \text{ при } x \le 0, \\ x^2 - \text{x при } 0 < x \le 1, \\ x^2 - \sin\pi x^2 \text{ при других } x. \end{cases}$$

Решение. Алгоритм имеет вложенную ветвящуюся структуру-

```
  Program Formula;
  Var X, F: Real;
  Begin
  writeln (' Введите действительное число: ');
  readln (X);
if X<=0 then F:=0
        else if X<=1 then F:=sqr (X) -X
             else F:=sqr (X) -sin (Pi*X*X) ;
writeln (' Значение функции F (x) при x = ', X,'
```

Задание к лабораторной работе №2

Получите задание у преподавателя по теме лабораторной работы *Разветвленные алгоритмы*.

1. Подготовьте текст программы и контрольный пример по своему заданию.
2. Используя план работы на языке **PascalABC**, описанный выше, выполните свой вариант.
3. Результаты выполнения программы должны совпадать с результатами контрольного примера.
4. Если нет ошибок, контрольный пример совпадает с результатом выполнения программы, результаты лабораторной работы покажите преподавателю.

ЛАБОРАТОРНАЯ РАБОТА № 3
ТЕМА. Программирование циклических алгоритмов

Цикл — многократное повторение последовательности действий по некоторому условию. Известны три типа циклических алгоритмических структур: цикл с предусловием, цикл с постусловием и цикл с параметром. В Паскале существуют операторы, реализующие все три типа циклов.

Цикл с предусловием (цикл-пока) — наиболее универсальная циклическая структура. Реализуется оператором **While.** Формат оператора:

while <логическое выражение> **do** <тело цикла>

Пока значение логического выражения — **true,** выполняется тело цикла. Тело цикла может быть простым или составным оператором.

Цикл с постусловием (цикл-до) имеет формат:

repeat <тело цикла>
until <логическое выражение>

Повторяется выполнение тела цикла. Цикл заканчивается, когда логическое выражение принимает значение **true.** Тело цикла с постусловием выполняется хотя бы один раз. Использования **begin и end** для ограничения составного тела цикла не требуется.

Цикл с параметром имеет два варианта записи:

1) for I: = In to Ik do <тело цикла>;

2) for I: = In downto I k do <тело цикла>.

Здесь I — параметр цикла - простая переменная порядкового типа;

In — выражение того же типа, определяющее начальное значение параметра;

Ik — выражение того же типа, определяющее конечное значение параметра;

<тело цикла> может быть простым или составным оператором.

Цикл повторяется, пока значение параметра лежит в интервале между `In` и `Ik`. Причем эти выражения вычисляются, только один раз в начале выполнения цикла.

В первом варианте при каждом повторении цикла значение параметра изменяется на следующее значение в данном типе (для целого типа - увеличивается на 1).

Во втором варианте при каждом повторении цикла значение параметра изменяется на предыдущее значение в данном типе (для целого типа — уменьшается на 1).

Пример 1. Вычислить сумму натурального ряда чисел от 1 до N.

Решение. Программа будет состоять из трех частей, в которых повторяется решение этой задачи с использованием операторов цикла **`while, repeat и for.`**

```
 Program Natur;
Var a, Summa, n : integer;
  Begin
  write('N=');
  readln(N);
  {Цикл с предусловием}
    a:=1;
    Summa:=0;
    while a<=N do
      begin
      Summa:= Summa + a;
      a := a + 1
      end;
  Writeln (' Результат первого суммирования:' , Summa) ;
    {Цикл с постусловием}
    a:=1;
    Summa:=0;
     repeat
     Summa:=Summa+ a;
     a:=a+1
     until a>N;
  Writeln (' Результат второго суммирования:' , Summa) ;
    {Цикл с параметром}
    Summa:=0;
    for a := 1 to N do
    Summa : =Summa + a;
  Writeln (' Результат третьего суммирования:', Summa);
end.
```

Очевидно, что все три результата будут одинаковыми.

Пример 2. Функцию $y := \sqrt{x}$ можно вычислить как предельное значение последовательности, определяемой рекуррентной формулой:

$$У_k = (У_{k-1} + x/y_{k-1})/2 \qquad \text{для } k= 1,2...$$

Начальное значение y_0 задается произвольно. За приближенное с точностью **ε** значение корня берется первое y_k , для которого выполняется условие: $|y_k - y_{k-1}| < \varepsilon$

Решение. Для вычисления значений числовой последовательности достаточно двух простых переменных, в которых на каждом шаге будут храниться последнее и предпоследнее значения: y_k и y_{k-1}. Обозначим эти переменные Anew и Aold. При программировании этой задачи нельзя использовать цикл с параметром, т.к. неизвестно заранее число повторений цикла. Воспользуемся циклом с предусловием.

```
Program Posled;
Var X, eps, Aold, Anew : Real;
    k: integer;
begin
Write (' Введите число Epsilon ' ) ;
ReadLn (eps) ;
Write('Введите значение X ');
ReadLn(X);
  Aold :=X;
  Anew :=( Aold +X / Aold )/2;
  while abs( Anew - Aold)>= eps do
   begin
    Aold : = Anew;
    Anew: = (Aold + X/Aold)/2;
   end;
WriteLn('Корень квадратный(', X,')=', Anew);
end.
```

Пример 3. На интервале `[2; n]` найти натуральное число с максимальной суммой делителей.

Решение. Идея алгоритма состоит в том, что все делители числа ***X***, меньшие *X*, лежат в интервале от `1 до X div 2 + 1`. Наибольшим делителем является само число *X*. Следовательно, для каждого из чисел *`[2...n]`* нужно отобрать и просуммировать все делители из указанного множества. По ходу вычислений производить отбор наибольшего значения.

Алгоритм будет содержать два вложенных цикла. Исполнение вложенных циклов происходит так: для каждого значения параметра внешнего цикла происходит полная «прокрутка» внутреннего цикла.

```
Program Sum_Del;
Var N, I, Sum_Max, Sum, K, Ch : Integer;
begin
Write (' Введите число N : ');
ReadLn(N);
Sum_Max := 1; Ch := 1; { Начальные значения величин}
for I: =2 to N do      {Внешний цикл: перебор чисел}
begin
Sum:=0;
{Внутренний цикл: поиск делителей}
for  K: =1 to  I div 2 + 1 do
if  I  mod  K = 0
then Sum: = Sum + K; {Суммирование делителей}
Sum := Sum + I; {Включение в сумму максимального делителя}
{Выбор максимальной суммы делителей} if Sum >Sum_Max then
  begin
  Sum_Max : = Sum;
  Ch : = I
  end;
end;
WriteLn('Максимальную сумму делителей ' , Sum_Max, ' имеет число ' ,Ch);
end.
```

Задание к лабораторной работе №3

Получите задание у преподавателя по теме лабораторной работы *Циклические алгоритмы*.

1. Подготовьте текст программы и контрольный пример по своему заданию.
2. Используя план работы на языке **PascalABC**, описанный выше, выполните свой вариант.
3. Результаты выполнения программы должны совпадать с результатами контрольного примера.
4. Если нет ошибок, контрольный пример совпадает с результатом выполнения программы, результаты лабораторной работы покажите преподавателю.

ЛАБОРАТОРНАЯ РАБОТА № 4

ТЕМА. Работа с массивами

Массив — упорядоченный набор однотипных значений — компонент массива. Тип компонент называется базовым типом массива.

В Паскале массив рассматривается как переменная структурированного типа. Массиву присваивается имя, посредством которого можно ссылаться на него, как на единое целое, так и на любую из его компонент.

Переменная с индексом — идентификатор компоненты массива.

Формат записи:

```
<имя массива>[<индекс>],
```

где индекс может быть выражением порядкового типа.

Описание массива определяет имя, размер массива и базовый тип. Формат описания в разделе переменных:

```
Var <имя массива> : array [<тип индекса>] of <базовый тип>
```

Чаще всего в качестве типа индекса используется интервальный целый тип.

Линейный (одномерный) массив — массив, у которого элементы — простые переменные. В одномерных массивах хранятся значения линейных таблиц.

Примеры описания одномерных массивов:

```
Var B : array [0. .5] of real;
    R : array [1. . 34] of char;
    N : array ['A' . . ' Z' ] of integer;
```

Ввод и вывод массива производится поэлементно. Обычно для этого используется цикл с параметром, где в качестве параметра применяется индексная переменная.

Пример 1. В программе вводится десять значений целочисленного массива A и выводятся значения вещественного массива B, содержащего 50 элементов. Соответствующие фрагменты программы:

```
Var A : array [1. .10] of integer;
    B : array [1. .50] of real;
    i : integer;
begin
for  i : = 1 to 10 do
begin
write ('A[', i, ']=') ;
readln (A[i] )
end;
for  i: = 1 to 50 do
begin
writeln('B[' ,i,']=', B[i])
end;
end.
```

Пример 2. Заполнить случайными числами из диапазона [0,1] вещественный линейный массив из `N` чисел. Найти максимальное значение и его индекс (первый, если таких значений несколько).

Решение. Поскольку размер массива в программе должен быть однозначно задан, определим `N` в разделе констант, например, `N=20`. При изменении размера массива достаточно будет отредактировать в программе лишь описание константы `N`.

```
Const N=20;
Var  X : array [1:N] of real;
     k: integer;
     max: real;
     Kmax: integer;
  Begin
   {Заполнение случайными .числами}
    for k:=1 to N do X[k]: =random;
 max := X [1] ;
 Kmax : =1;          {Инициализация вычисляемых переменных}
 for k: =2 to N do  {Поиск максимального значения}
 if X [k] > max then
 begin
 max: =X [k] ;
 Kmax := k;
 end;
 writeln ('Первое максимальное значение: X[',Kmax,'] =',max)
end.
```

Пример 3. Дан целочисленный линейный массив. Отсортировать его элементы в порядке возрастания значений.

Решение. Воспользуемся алгоритмом, известным под названием «метод пузырька». Идея состоит в последовательном перемещении путем попарных перестановок наибольшего значения сначала на место N-го элемента, затем N-1 -го и т.д.

Опишем массив на максимальный размер (например, 100), а фактический размер N определим вводом.

```
Program Sortirovka;
Var N, I, J, P : integer;
A array [ 1. . 100] of integer;
begin
write('Введите число элементов: ' ) ;
readln(N);
for I: = 1 to N do
begin
write ('Введите A[' , I, '] ');
readln (A [I]);
end;
     for I:=1 to N-1 do
     begin for J: =1 to N-I do
           if A[J] <=A[J+1] then
           begin
            P:=A[J];
            A[J] :=A[J+1];
            A[J+1]:=P;
           end
end ;
for I :=1 to N do
write (A[I] , ' ');
end.
```

Тестовый пример: N = 10; {количество элементов в массиве A_i}

элементы массива A_i: [1, 2, 2, 2, -1, 1, 0, 34, 3, 3]

результаты выводятся в массив с именем A_i: [-1, 0, 1, 1, 2, 2, 2, 3, 3, 34]

Пример вывода результатов в файловую переменную: Для этого необходимо в блоке описания описать файловую переменную:

```
f:Text;
```

В теле программы, которая начинается с операторных скобок Begin…End

```
....
begin
  assign(F,d:\mus\alg10.txt); {путь к файлу, где будут
  находиться результирующие данные}
  rewrite(f); {оператор открытия файловой переменной}
  writeln(f,'aaa');  {оператор вывода в файловую
переменную с именем f  }
  Close(f) {оператор закрытия файловой переменной};
End.
```

Пример вывода результатов программы `Sortirovka` в файловую переменную:

```
for I :=1 to N do
writeln (f,A[I]);
```

Задание к лабораторной работе №4

Получите задание у преподавателя по теме лабораторной работы *Работа с массивами*.

1. Подготовьте текст программы и контрольный пример по своему заданию.
2. Используя план работы на языке **PascalABC**, описанный выше, выполните свой вариант.
3. Результаты выполнения программы должны совпадать с результатами контрольного примера.
4. Если нет ошибок, контрольный пример совпадает с результатом выполнения программы, результаты лабораторной работы покажите преподавателю.