Министерство образования Российской Федерации
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Кафедра ИСУ

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

Методические указания к выполнению лабораторных работ
в системе программирования PascalABC.Net

для студентов направлений подготовки
09.03.01 «Информатика и вычислительная техника», профиль «Интеллектуальные системы обработки информации и управления»
09.03.02 «Информационные системы и технологии», профиль «Безопасность информационных систем»

Нижний Новгород
2020

Составитель Э.С.Соколова

УДК 681

Программирование на языке высокого уровня: Метод. указания к выполнению лаб. работ в системе программирования PascalABC.Net /НГТУ; Н.Новгород, 2020. – 18 с.

Приведены методические рекомендации для выполнения лабораторных работ по курсу “ Программирование на языке высокого уровня ” в соответствии с учебным планом направлений подготовки
09.03.01 «Информатика и вычислительная техника», профиль «Интеллектуальные системы обработки информации и управления»
09.03.02 «Информационные системы и технологии», профиль «Безопасность информационных систем»
Могут быть использованы для проведения лабораторных работ по аналогичным курсам других специальностей.

ВВЕДЕНИЕ

Цель данных лабораторных работ состоит в изучении и закреплении основ программирования в системе программирования PascalABC.Net, позволяющей бучить основам программирования (алгоритмы, структуры данных, функции, динамика), получить практические навыки разработки, отладки и тестирования программ.

Это простая и одновременно мощная среда программирования, хорошо русифицированная и с интерактивными подсказками по коду, нацеленная на написание компактных интуитивно понятных программ.

После обучения основам программирования выбор языка для развития является техническим моментом в соответствии условиям решаемой задачи. Студент легко переходит на более высокий уровень языка программирования.

PascalABC.Net представляет собой гибридный язык Pascal, C#, Java, с понятным для обучения синтаксисом Pascal, но коды приближены к C# и Java, позволяет программировать в классическом процедурном стиле, в объектно-ориентированном стиле и содержит множество элементов для программирования в функциональном стиле.

Система программирования PascalABC.NET

PascalABC.NET разрабатывается под свободной лицензией LGPLv3 как язык программирования для сферы образования и научных исследований и вбирает в себя лучшее, что предлагают современные языки, такие как C#, Kotlin, Python, Haskell и другие:

– использует огромные возможности платформы Microsoft.NET;

– включает бесплатную, простую и мощную среду разработки с подсказками, автоформатированием и образцами кода для начинающих;

– мощный язык с простым и логичным синтаксисом для написания компактных, эффективных, понятных программ, идеален для обучения современному программированию:

– PascalABC.NET является мультипарадигменным языком: на нём можно программировать в структурном, объектно-ориентированном и функциональном стилях

– содержит все современные языковые средства: классы, перегрузку операций, интерфейсы, обработку исключений, кортежи, обобщенные классы и подпрограммы, сборку мусора, лямбда-выражения, средства параллельного программирования.

Перед выполнением лабораторных работ студенты должны пройти инструктаж по технике безопасности и расписаться в соответствующем контрольном листе инструктажа.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

1. Получить у преподавателя вариант задания к лабораторной работе.
2. Составить блок-схему алгоритма к 1 или 2 лабораторной работе, если в коде есть управляющие операторы. В 3 и 4 лабораторных работах использовать комментарии для пояснения кода.
3. Отладить и протестировать программу. На этапе отладки исправить все синтаксические ошибки – несоответствие между исполняемой программой и правилами построения программ с помощью алфавита и операторов языка. Протестировать программу – убедиться в правильности ее работы для различных входных наборов данных. Часто после тестирования программа вновь требует отладки.
4. Оформить отчет по лабораторной работе.

При разработке программы особое внимание следует уделить следующим моментам:

- программа должна обладать свойством массовости (универсальности), т.е. быть применимой не к единственному входному данному, а к целому классу данных. Например, исходное данное – любое вещественное число или любая последовательность чисел, размерность которой ограничена некоторым значением. Если программа зависит от конкретного набора данных – она не универсальна;

- **программа должна быть структурирована**, т.е. разбита на небольшие, легко управляемые части – блоки. Каждый блок должен иметь один вход и один выход. Это может быть

подпрограмма (процедура или функция), часть программы – оператор цикла или условный оператор. **Следует избегать использования операторов безусловного перехода** *goto***,** т.к. это нарушает логику вычислительного процесса и противоречит принципу структурированности кода;

- создать **удобный интерфейс пользователя**, защитить программу от неправильного ввода исходных данных. Вводу данных с клавиатуры должны предшествовать текстовые приглашения. Следует организовать контроль правильности ввода данных, включив в программу блок вывода введенных значений на экран. Рекомендуется использовать защиту от неправильности ввода данных с выдачей соответствующих диагностических сообщений. Результаты работы программы следует выводить в удобном для восприятия виде с соответствующими текстовыми сообщениями;

- **программа должна быть понятна и удобочитаема**. Используйте длинные наглядные имена, например, *NumberOfIterations (Number_Of_Iterations), LineLength, Number_Columns,* отражающие суть и назначение переменной;

- **следует выделять структуру программы**, делая отступы от начала строки при записи условных операторов, операторов цикла, а также вложенных операторов (каждый уровень вложенности должен начинаться пропуском определенного числа позиций). Метод улучшения наглядности программы - пропуск строк для разделения отдельных фрагментов программы;

- в программу следует **включать комментарии**, указывающие цель действия или объясняющие логику программы.

ОФОРМЛЕНИЕ ОТЧЕТОВ

Отчет должен содержать титульный лист, на котором необходимо указать номер выполненной лабораторной работы, Ф.И.О. студента и номер учебной группы

Каждый отчет должен содержать:

1) формулировку задачи;
2) блок-схему алгоритма;
3) распечатку текста программы и результатов ее работы.

ЛАБОРАТОРНАЯ РАБОТА №1
Основные типы данных, операции и выражения, операторы ввода-вывода. Линейные и разветвляющиеся алгоритмы

Методические указания

Для первой лабораторной работы преподаватель не дает индивидуальное задание каждому студенту. Студент должен составить программу, обрабатывающую переменные простых стандартных типов:

Char, Integer, Real, Boolean, String. Требуется:

1) организовать ввод с клавиатуры значений указанных типов в переменные программы;
2) составить выражения с использованием переменных программы, арифметических, логических, **побитовых** операций, операций отношения, основных встроенных процедур и функций;
3) вывести значения выражений на экран;
4) для организации разветвляющихся вычислительных процессов включить в программу условный оператор *if***.** Можно использовать **операторы циклов**.

При вводе данных следует использовать встроенные процедуры ввода *read* (список переменных); *readln* (список переменных); при выводе *write* (список элементов вывода); *writeln* (список элементов вывода).

Выражение в программировании служит для определения действий, которые в математике обычно описываются формулами. Выражения состоят из операндов (констант, переменных, функций), знаков операций и круглых скобок. Операции по количеству операндов делятся на унарные и бинарные. Унарные операции имеют только один операнд, перед которым стоит символ операции (операция смена знака `-`, логическое отрицание *not*).

По приоритету все операции делятся на группы. Операции с равным приоритетом выполняются слева направо, хотя иногда компилятор для генерации более оптимального кода может переупорядочить операнды. Скобки служат для изменения обычного порядка обработки операций. Подвыражение, заключенное в скобки, выполняется как отдельный операнд.

Используйте в программах средства **модуля** *Crt*, содержащим набор процедур и функций управления текстовым выводом на экран дисплея, звуковым генератором и чтением символов с клавиатуры без отображения их на экране, а также переменных и констант режимов работы и цветов.

Примеры приоритета операций

Приоритет	Операции	Категория операций	Для классов
1 (высший)	+, -, not, @,	Унарные операции	new
2	* / div mod and shl shr	Бинарные операции типа умножения	as, is
3	+ - or xor	Бинарные операции типа сложения	
4	= <> > < <= >= in	Бинарные операции отношения	
5 (низший)	?:	Имеет 3 операнда	

Примеры фрагментов «простых» программ Лабораторная работа №1

```
{1. работа с простыми типами данных}

program lab1;

var
  n, s, i,w,m,d,k,v:integer;
  y,z:integer;
  x: char;
  c: char;
  b: char;
  kod:byte;
  r1,r2,r3,r4:boolean;
begin
  readln(n);
  s := 0; k := 0;
  for i := 1 to n do
  begin
    if (i mod 3 = 0) then
    begin
      s := s + i;
      k := k + 1;
    end;
  end;
  writeln(s:3, k:3);
  if k = 0 then writeln('чисел, кратных 3 нет');
  readln(x);
  writeln(pred(x)); //Вывод предыдущего символа в таблице ASCII
  writeln(succ(x)); //Вывод следующего символа в таблице ASCII
  readln(c);
  readln(b);
  if c > b then writeln(c) else writeln(b);
     readln(m,d);
  w:=m and d;
  v:=m or d;
  writeln(w); writeln(not(w));
  writeln(v);
```

```
  readln(y,z);
  r1:=z<y;
  r2:=z>y;
  r3:=z=y;
  r4:=z<>y;
  writeln('первое меньше второго',r1);
  writeln('первое больше второго',r2);
  writeln('первое равно второму',r3);
  writeln('первое неравно второму',r4);
    for kod:=33 to 255 do
  begin
    if (kod mod 10=0) then writeln;
    write(chr(kod):3, kod:4) {вывод символа и его кода}
  end;

end.


{2. округлить число до ближайшей степени 2}

program lab1;
var n,m,a: integer;
k:real;
begin
  m:=0;
  writeln('Введите число');
  readln(n);
  n := abs(n);
  a:=n;
  if (n <> 0) and (n and (n - 1) = 0)
    then writeln('Число является степенью двойки')
      else begin
        repeat
        n:=n div 2;
        m+=1;
        until n=1;
        k:=((exp(ln(2)*(m+1))-exp(ln(2)*m))/2)+exp(ln(2)*m);
        if a>k then k:=exp(ln(2)*(m+1))
          else k:=exp(ln(2)*m);
        writeln('Число не является степенью двойки, ближайшей степенью двойки к ', a,
' является ', k);
      end;
end.


{3. поменять местами значения двух переменных без дополнительной}

program lab2;
var a,b: integer;
begin
  writeln('введите значения a и b');
  read(a,b);
  a:=a xor b;
  b:=a xor b;
  a:= a xor b;
  writeln('a=', a);
  write( 'b=', b);
end.


{4. вывести четвёртый бит числа}

program lab3;

var
  x: integer;
```

```
begin
  writeln('введите x');
  read(x);
  if x > 6 then
    if ((1 shl 3) and x) > 0 then
       writeln('четвертый бит = 1')
    else
       writeln('четвертый бит = 0')
  else
    writeln('Если число меньше 7, то четвертый бит=0')
end.
```

Пример №2 «продвинутой» программы , представленной студентом на защиту

```
uses crt;

// Перегрузка функций перевода

// Функция перевода десятичкых чисел в двоичные для типа byte
function DecToBin(x:byte;y:string): string;
 var i:byte;
 begin
   for i:=0 to 7 do
    begin
     if (x and 128) = 128 then
      begin
       y:=y + '1';
       x:=x shl 1;
      end
     else
      begin
       y:=y + '0';
       x:=x shl 1;
      end;
     end;
   DecToBin:=y;
   end;

// Функция перевода десятичкых чисел в двоичные для типа shortint
function DecToBin(x:shortint;y:string): string;
 var i:byte;
 begin
   for i:=0 to 7 do
    begin
     if (x and 128) = 128 then
      begin
       y:=y + '1';
       x:=x shl 1;
      end
     else
      begin
       y:=y + '0';
       x:=x shl 1;
      end;
     end;
   DecToBin:=y;
   end;

// Функция перевода десятичкых чисел в двоичные для типа integer
function DecToBin(x:integer;y:string): string;
 var i:byte;
 begin
```

```
   for i:=0 to 15 do
    begin
     if (x and 32768) = 32768 then
      begin
       y:=y + '1';
       x:=x shl 1;
      end
     else
      begin
       y:=y + '0';
       x:=x shl 1;
      end;
     end;
   DecToBin:=y;
   end;

// Функция перевода десятичкых чисел в двоичные для типа word
function DecToBin(x:word;y:string): string;
 var i:byte;
 begin
   for i:=0 to 15 do
    begin
     if (x and 32768) = 32768 then
      begin
       y:=y + '1';
       x:=x shl 1;
      end
     else
      begin
       y:=y + '0';
       x:=x shl 1;
      end;
     end;
   DecToBin:=y;
   end;

// Функция удаления незначащих нулей
Function NezNol(x:string): string;
var i,n: byte;
 begin
   for i:=1 to length(x) do
   begin
    if x[i]='0' then n:=n+1
    else break;
   end;
   Delete(x,1,n);
   if x='' then x:='0';
   NezNol:=x;
 end;

// Объявление глобальных переменных
var a: byte; b: shortint; c: word; d: integer;
 abcde:string; n,nol: byte;

begin

// Работа с главным меню
var menu1:byte;
menu1:=1;
while(menu1 = 1) do
begin
 clrscr();
 write('С каким типом чисел вы хотите работать?',#13#10,#13#10);
 writeln('1 byte',#13#10,'2 shortint',#13#10,'3 word',#13#10,'4 integer',#13#10);
 Write('при вводе номера, которого нет, вы по умолчанию будуте работать с типом
integer    Ваш ответ ');
```

```
 readln(n);
 if (n < 1) or (n > 4) then n:=4;
 writeln(#13#10);
 writeln('Желаете ли удалять незначащие нули при работе?',#13#10,#13#10,'1
Yes',#13#10,'2 No',#13#10);
 Write('при вводе номера, которого нет, вы по умолчанию будете работать с нулями
Ваш ответ ');
 readln(nol);
 if (nol < 1) or (nol > 4) then nol:=2;
 writeln(#13#10);



// Работа с типом byte
 if n=1 then
  begin
   var menu3:byte;
   var     help:longint;
   menu3:=0;
   help:=0;
   while (menu3 = 0) do
    begin
     clrscr();
     writeln('Введите число, ДЛЯ НАГЛЯДНОЙ РАБОТЫ ОТЛИЧНО ПОДХОДИТ ЧИСЛО 170');
     writeln(#13#10,'Пожалуйста, вводите числа из промежутка 0..255',#13#10);
     Write('Ваш ответ ');
     readln(help);
     if (help < 0) or (help > 255) then writeln('Неверный ввод, попробуйте еще раз')
     else
       begin
        menu3 :=1;
        a:=help;
       end;
     end;
   abcde:=DecToBin(a,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln(abcde,#13#10);
   writeln('Сейчас вы выполните несколько побитовых операций с выбраным числом');
   writeln(#13#10,'Результат сразу будет выводится на экран',#13#10,#13#10);
   abcde:=DecToBin(a,abcde);
   if nol=1 then abcde:=NezNol(abcde);

   writeln('Первая операция это NOT, применим ее к вашему числу');
   a:=not(a);
   abcde:='';
   abcde:=DecToBin(a,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',a);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая Операция это AND , рассмотрим ее работу на примере обнуления
битов');
   writeln(#13#10,'ОБНУЛЯТЬ БУДЕМ БИТЫ С НОМЕРАМИ 0.2.4.6 !!!',#13#10);
   writeln('Теперь работаем с числом    ',a,'    ',abcde,#13#10#13#10);
   a:=a and(255-85);
   abcde:='';
   abcde:=DecToBin(a,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',a);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая операция OR, рассмотрим ее работу на примере установки единиц в
отдельные биты');
   writeln(#13#10,'СТАВИТЬЕДИНИЦЫ БУДЕМ В БИТЫ С НОМЕРАМИ 0.2.4.6 !!!',#13#10);
```

```
   writeln('Теперь работаем с числом    ',a,'    ',abcde,#13#10#13#10);
   a:=a or 85;
   abcde:='';
   abcde:=DecToBin(a,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',a);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая операция XOR, рассмотрим ее работу на примере перемены значения
КАЖДОГО БИТА');
   writeln(#13#10,'Теперь работаем с числом    ',a,'    ',abcde,#13#10#13#10);
   a:=a xor 255;
   abcde:='';
   abcde:=DecToBin(a,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',a);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Также существуют операции SHL и SHR, продемонстрируем их работу на СДВИГЕ
НА 5 БИТОВ');
   writeln(#13#10,'Теперь работаем с числом    ',a,'    ',abcde,#13#10#13#10);
   help:=a;
   a:=a shl 5;
   abcde:='';
   abcde:=DecToBin(a,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат SHL 5, ваше новое число:',#13#10,#13#10'В десятичной
системе ',a);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);
   a:=help;
   a:=a shr 5;
   abcde:='';
   abcde:=DecToBin(a,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат SHR 5, ваше новое число:',#13#10,#13#10'В десятичной
системе ',a);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);
   abcde:='';
  end



 // Работа с типом shortint
 else if n=2 then
  begin
   var menu3:byte;
   var     help:longint;
   menu3:=0;
   help:=0;
   while (menu3 = 0) do
    begin
     clrscr();
     writeln('Введите число, ДЛЯ НАГЛЯДНОЙ РАБОТЫ ОТЛИЧНО ПОДХОДИТ ЧИСЛО -85');
     writeln(#13#10,'Пожалуйста, вводите числа из промежутка -128..127',#13#10);
     Write('Ваш ответ ');
     readln(help);
     if (help < -128) or (help > 127) then writeln('Неверный ввод, попробуйте еще
раз')
     else
       begin
        menu3 :=1;
        b := help;
       end;
     end;
```

```
   abcde:=DecToBin(b,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln(abcde,#13#10);
   writeln('Сейчас вы выполните несколько побитовых операций с выбраным числом');
   writeln(#13#10,'Результат сразу будет выводится на экран',#13#10,#13#10);
   abcde:=DecToBin(b,abcde);
   if nol=1 then abcde:=NezNol(abcde);

   writeln('Первая операция это NOT, применим ее к вашему числу');
   b:=not(b);
   abcde:='';
   abcde:=DecToBin(b,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',b);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая Операция это AND , рассмотрим ее работу на примере обнуления
битов');
   writeln(#13#10,'ОБНУЛЯТЬ БУДЕМ БИТЫ С НОМЕРАМИ 0.2.4.6 !!!',#13#10);
   writeln('Теперь работаем с числом    ',b,'    ',abcde,#13#10#13#10);
   b:=b and(170);
   abcde:='';
   abcde:=DecToBin(b,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',b);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая операция OR, рассмотрим ее работу на примере установки единиц в
отдельные биты');
   writeln(#13#10,'СТАВИТЬЕДИНИЦЫ БУДЕМ В БИТЫ С НОМЕРАМИ 0.2.4.6 !!!',#13#10);
   writeln('Теперь работаем с числом    ',b,'    ',abcde,#13#10#13#10);
   b:=b or 85;
   abcde:='';
   abcde:=DecToBin(b,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',b);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая операция XOR, рассмотрим ее работу на примере перемены значения
КАЖДОГО БИТА');
   writeln(#13#10,'Теперь работаем с числом    ',b,'    ',abcde,#13#10#13#10);
   b:=b xor 255;
   abcde:='';
   abcde:=DecToBin(b,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',b);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Также существуют операции SHL и SHR, продемонстрируем их работу на СДВИГЕ
НА 5 БИТОВ');
   writeln(#13#10,'Теперь работаем с числом    ',b,'    ',abcde,#13#10#13#10);
   help:=b;
   b:=b shl 5;
   abcde:='';
   abcde:=DecToBin(b,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат SHL 5, ваше новое число:',#13#10,#13#10'В десятичной
системе ',b);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);
   b:=help;
   b:=b shr 5;
   abcde:='';
```

```
   abcde:=DecToBin(b,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат SHR 5, ваше новое число:',#13#10,#13#10'В десятичной
системе ',b);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);
   abcde:='';
  end


 // Работа с типом word
 else if n=3 then
  begin
   var menu3:byte;
   var     help:longint;
   menu3:=0;
   help:=0;
   while (menu3 = 0) do
    begin
     clrscr();
     writeln('Введите число, ДЛЯ НАГЛЯДНОЙ РАБОТЫ ОТЛИЧНО ПОДХОДИТ ЧИСЛО 21845');
     writeln(#13#10,'Пожалуйста, вводите числа из промежутка 0..65535',#13#10);
     Write('Ваш ответ ');
     readln(help);
     if (help < 0) or (help > 65535) then writeln('Неверный ввод, попробуйте еще
раз')
     else
       begin
        menu3 :=1;
        c := help;
       end;
     end;
   abcde:=DecToBin(c,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln(abcde,#13#10);
   writeln('Сейчас вы выполните несколько побитовых операций с выбраным числом');
   writeln(#13#10,'Результат сразу будет выводится на экран',#13#10,#13#10);
   abcde:=DecToBin(c,abcde);
   if nol=1 then abcde:=NezNol(abcde);

   writeln('Первая операция это NOT, применим ее к вашему числу');
   c:=not(c);
   abcde:='';
   abcde:=DecToBin(c,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',c);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая Операция это AND , рассмотрим ее работу на примере обнуления
битов');
   writeln(#13#10,'ОБНУЛЯТЬ БУДЕМ БИТЫ С НОМЕРАМИ 0.2.4.6.8.10.12.14 !!!',#13#10);
   writeln('Теперь работаем с числом    ',c,'    ',abcde,#13#10#13#10);
   c:=c and(65535-43690);
   abcde:='';
   abcde:=DecToBin(c,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',c);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая операция OR, рассмотрим ее работу на примере установки единиц в
отдельные биты');
   writeln(#13#10,'СТАВИТЬЕДИНИЦЫ БУДЕМ В БИТЫ С НОМЕРАМИ 0.2.4.6.8.10.12.14
!!!',#13#10);
   writeln('Теперь работаем с числом    ',c,'    ',abcde,#13#10#13#10);
   c:=c or 43690;
```

```
   abcde:='';
   abcde:=DecToBin(c,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',c);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая операция XOR, рассмотрим ее работу на примере перемены значения
КАЖДОГО БИТА');
   writeln(#13#10,'Теперь работаем с числом    ',c,'    ',abcde,#13#10#13#10);
   c:=c xor 65535;
   abcde:='';
   abcde:=DecToBin(c,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',c);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Также существуют операции SHL и SHR, продемонстрируем их работу на СДВИГЕ
НА 5 БИТОВ');
   writeln(#13#10,'Теперь работаем с числом    ',c,'    ',abcde,#13#10#13#10);
   help:=c;
   c:=c shl 5;
   abcde:='';
   abcde:=DecToBin(c,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат SHL 5, ваше новое число:',#13#10,#13#10'В десятичной
системе ',c);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);
   c:=help;
   c:=c shr 5;
   abcde:='';
   abcde:=DecToBin(c,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат SHR 5, ваше новое число:',#13#10,#13#10'В десятичной
системе ',c);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);
   abcde:='';
  end


// Работа с типом integer
 else
   begin
   var menu3:byte;
   var     help:longint;
   menu3:=0;
   help:=0;
   while (menu3 = 0) do
    begin
     clrscr();
     writeln('Введите число, ДЛЯ НАГЛЯДНОЙ РАБОТЫ ОТЛИЧНО ПОДХОДИТ ЧИСЛО -10922');
     writeln(#13#10,'Пожалуйста, вводите числа из промежутка -32768..32767',#13#10);
     Write('Ваш ответ ');
     readln(help);
     if (help < -32768) or (help > 32767) then writeln('Неверный ввод, попробуйте еще
раз')
     else
       begin
        menu3 :=1;
        d := help;
       end;
     end;
    abcde:=DecToBin(d,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln(abcde,#13#10);
```

```
   writeln('Сейчас вы выполните несколько побитовых операций с выбраным числом');
   writeln(#13#10,'Результат сразу будет выводится на экран',#13#10,#13#10);
   abcde:=DecToBin(d,abcde);
   if nol=1 then abcde:=NezNol(abcde);

   writeln('Первая операция это NOT, применим ее к вашему числу');
   d:=not(d);
   abcde:='';
   abcde:=DecToBin(d,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',d);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая Операция это AND , рассмотрим ее работу на примере обнуления
битов');
   writeln(#13#10,'ОБНУЛЯТЬ БУДЕМ БИТЫ С НОМЕРАМИ 0.2.4.6.8.10.12.14 !!!',#13#10);
   writeln('Теперь работаем с числом    ',d,'    ',abcde,#13#10#13#10);
   d:=d and(43690);
   abcde:='';
   abcde:=DecToBin(d,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',d);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая операция OR, рассмотрим ее работу на примере установки единиц в
отдельные биты');
   writeln(#13#10,'СТАВИТЬЕДИНИЦЫ БУДЕМ В БИТЫ С НОМЕРАМИ 0.2.4.6.8.10.12.14
!!!',#13#10);
   writeln('Теперь работаем с числом    ',d,'    ',abcde,#13#10#13#10);
   d:=d or 10921;
   abcde:='';
   abcde:=DecToBin(d,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',d);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Следующая операция XOR, рассмотрим ее работу на примере перемены значения
КАЖДОГО БИТА');
   writeln(#13#10,'Теперь работаем с числом    ',d,'    ',abcde,#13#10#13#10);
   d:=d xor 65535;
   abcde:='';
   abcde:=DecToBin(d,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат, ваше новое число:',#13#10,#13#10'В десятичной системе
',d);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);

   writeln('Также существуют операции SHL и SHR, продемонстрируем их работу на СДВИГЕ
НА 5 БИТОВ');
   writeln(#13#10,'Теперь работаем с числом    ',d,'    ',abcde,#13#10#13#10);
   help:=d;
   d:=d shl 5;
   abcde:='';
   abcde:=DecToBin(d,abcde);
   if nol=1 then abcde:=NezNol(abcde);
   writeln('Вот результат SHL 5, ваше новое число:',#13#10,#13#10'В десятичной
системе ',d);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);
   d:=help;
   d:=d shr 5;
   abcde:='';
   abcde:=DecToBin(d,abcde);
   if nol=1 then abcde:=NezNol(abcde);
```

```
   writeln('Вот результат SHR 5, ваше новое число:',#13#10,#13#10'В десятичной
системе ',d);
   writeln('В двоичной системе   ',abcde,#13#10,#13#10);
   abcde:='';
  end;


// Меню с предлагающее продолжить работу или выйти из программы
  var menu2:integer;
  menu2:=1;
   while(menu2 = 1) do
    begin
     writeln('Желаете продолжить работу?',#13#10,#13#10,'1 Yes',#13#10,'2
No',#13#10);
     Write('Ваш ответ ');
     readln(menu1);
     clrscr();
     if (menu1 < 1) or (menu1>2) then
      begin
       clrscr();
       writeln('неверный ввод,нажми клавишу,чтобы попробовать еще раз');
       readkey();
       clrscr();
      end
     else if menu1 = 1 then menu2 := 2
    else menu2 := 2;
   end;
  end;
 end.
```

ЛАБОРАТОРНАЯ РАБОТА №2
Решение задач с использованием циклических алгоритмов. Обработка числовых и символьных массивов

Методические указания

В соответствии с вариантом задания (2 задания – массив чисел и массив строк):

1. Составить блок-схему решения задачи.
2. Написать программу по разработанному алгоритму с выполнением требований:

- максимальный размер массива, с которым может работать данная программа, определить в разделе описания констант *Const*, фактический размер массива пользователь вводит в переменные в процессе работы программы;

- описать тип массива в разделе *Type*. Недопустимо явное описание типа массива в разделе описания переменных *Var*;

- ввод массива осуществляется с клавиатуры пользователем или читается из заранее подготовленного файла исходных данных. При этом массив вводится в виде матрицы. Недопустимо вводить двухмерный массив в одну строку или по одному элементу в каждой строке;

- после ввода массива реализовать его вывод для контроля за правильностью ввода данных;

- для повышения эффективности программы текстовые массивы следует читать и записывать строками, используя переменные типа *String;*

- отдельные блоки обработки массивов можно оформлять в виде подпрограмм – процедур и функций;

- вывод результатов работы программы должны сопровождать текстовые пояснения;

- программа должна работать циклически. Выход из программы или возврат в ее начало должны происходить по выбору пользователя.

3. Протестировать программу, получив результаты ее работы для разных вариантов входных данных. Убедиться в правильности работы программы.

При работе с текстовой матрицей следует использовать стандартные процедуры и функции работы со строковыми типами, такие, как *Length, Copy, Delete, Insert, Pos.*

ВНИМАНИЕ!

В разрабатываемой программе студент должен организовать ввод исходных данных с клавиатуры в обязательном порядке, чтобы можно было протестировать работу программы. Для предварительной демонстрации работы кода следует организовать ввод данных из файла.

Пример программы

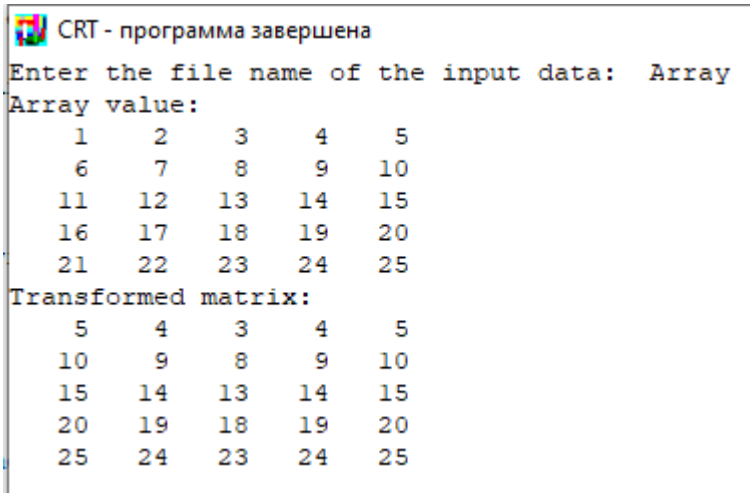
Дан целочисленный двухмерный массив размером до n×m элементов. Выполнить «зеркальное отображение» элементов матрицы относительно вертикальной оси симметрии (поменять местами элементы первого столбца с последним, второго с предпоследним и т.д.).

```
Program Mirroring_Matrix;
Uses Crt;
const  max_line = 50;
       max_col = 60;
type  Arr_int = array[1..max_line, 1..max_col]  of integer;
var matr : Arr_Int;
  file_matr : text;
  x , line, col, i , j : integer;
  MyFile : string[30];
procedure PrintMatr( mas : Arr_Int; n,m : integer); {вывод матрицы}
var  i , j : integer;
begin
  for  i := 1  to  n  do
      begin
    for  j := 1  to  m do  write ( mas [i,j] : 5);
    writeln
      end;
end;            {end PrintMatr}
begin
  clrscr;
  write ('Enter the file name of the input data:  ');
  readln (MyFile);

  Assign (file_matr, MyFile);    {связывание файловой переменной file_matr}
                                 {с физическим файлом  MyFile}
    Reset (file_matr);            { открытие файла для чтения }
    readln (file_matr, line, col);{чтение из файла фактического размера массива}
for  i := 1  to  line  do
        begin
    for  j := 1  to  col  do
       read (file_matr, matr[i,j]); {ввод данных в переменную matr из файла}
    readln (file_matr);
        end;
writeln ('Array value:');
PrintMatr( matr,line,col);
for i:=1 to line do
    for j:=1 to col div 2 do
      matr[i,j]:=matr[i,col-j+1];
writeln ('Transformed matrix:');
PrintMatr( matr,line,col);
Close(file_matr);
end.
```

В данном примере ввод исходных данных в массив *matr* и в переменные для хранения фактического размера массива *line* и *col* организован из файла исходных данных, подготовленного заранее программным путем или в блокноте.

Результаты работы программы имтют следующий вид:



Пользователь с клавиатуры вводит имя файла исходных данных, которое читается в переменную *MyFile*. Процедура Assign связывает файловую переменную *file_matr* с физическим файлом *MyFile*. Процедура Reset открывает существующий физический файл, который связан с файловой переменной *file_matr*. Так как *file_matr* - текстовый файл, то после работы процедуры Reset он доступен только для чтения с последовательным доступом к его элементам. Процедура read, у которой первый аргумент - файловая переменная, осуществляет ввод значений из этого файла в список аргументов. Таким образом, процедура *read(fil_matr, line, col*) вводит в область памяти переменных *line* и *col* первые два значения из файла (фактический размер массива – количество строк и столбцов), а процедура *readln (file_matr, matr [i, j])* вводит из файла последовательность чисел и заполняет массив.

ЛАБОРАТОРНАЯ РАБОТА № 3
Программирование вычислительных процессов с использованием комбинированных и файловых типов данных.
Использование процедур и функций

Методические указания

При выполнении лабораторной работы приобретаются навыки работы с типизированными файлами. Исходные данные программы вводить в переменные типа запись - record. Для хранения и работы с информацией создается файл, тип компонент которого запись. Результаты работы программы, кроме вывода на экран, следует сохранить в текстовом файле с последующим выводом его содержимого на принтер.

В соответствии с полученным вариантом задания:

1. Составить блок-схему решения задачи на уровне процедур и функций. Блок-схема должна отражать логическую структуру алгоритма.
2. По разработанному алгоритму написать программу, удовлетворяющую следующим требованиям:

   - в программу включить меню работы с программой, которое должно содержать пункты:

   а) создание файла исходных данных;

   б) просмотр содержимого файла;

   в) поиск заданной информации в файле;

   г) корректировка информации в файле (дозапись информации в файл, удаление информации из файла, изменение значения компонент файла);

   д) выход из программы;

   - алгоритм, соответствующий каждому пункту меню, должен быть реализован в виде подпрограммы – процедуры или функции. Входные данные для работы подпрограмм и результаты их работы следует передавать через параметры подпрограмм. Не следует все переменные программы описывать как глобальные объекты;

- перемещение по пунктам меню можно организовать с помощью клавиш <стрелка вверх> и <стрелка вниз> или вводом номера выбранного пункта меню;

- предусмотреть ввод информации в файл из заранее подготовленного текстового файла. Для этого в текстовом редакторе системы заранее набрать входную информацию и сохранить ее в файле исходных данных;

- все пользовательские типы описать в разделе type;

- если при корректировке файлов создавались вспомогательные файлы, их необходимо удалить после работы программы.

Пример задания

Описать в виде записи зачетную книжку студента. Записать в файл содержимое зачетных книжек некоторой группы студентов. После сессии добавить в файл результаты сдачи экзаменов студентами. Исключить из файла сведения о студентах, не сдавших сессию.

Пример описания типов и переменных для решения задачи:

```
type  Exam = record
             disciplines: string[30];
             exam_grade : 2 . . 5;
             end;
      Session = array [1 . . 5] of exam;
      Set_of_Session = array [1 . . 10] of Sesson;
      Stud_Record  = record
                      Number     : longint;
                      Name       : string[20];
                      SurName    : string[20];
                      GroupCode  : string[8];
                      grades     : Set_of_Session;
                     end;
      File_Stud_Record = file of Stud_Record;

var  inf : StudCard;
       f : File_Stud_Record;
```

Запись *record* объединяет в единое целое любое число структур данных разных типов. Чтобы присвоить значение переменной типа “запись”, нужно присвоить значения всем полям записи. Ввод и вывод информации в файл записей осуществляется только записями.

Для связи файловой переменной с дисковым файлом служит процедура *Assign (f, name).*

Файлы открываются для обмена с помощью одной из двух системных процедур – Reset и *Rewrite,* единственным параметром которых является файловая переменная. Процедура *Reset* предполагает, что открываемый дисковый файл уже существует, в противном случае возникает ошибка. Процедура *Rewrite* допускает, что открываемый файл может не существовать, в этом случае она создаст заданный файл. Если файл существует, то *Rewrite* очищает его. В обоих случаях текущий указатель файла устанавливается на его нулевой элемент. Обе процедуры допускают в дальнейшем как чтение из файла, так и запись в него.

При открытии с помощью процедуры *Reset* отсутствующего файла генерируется динамическая ошибка, которую следует “перехватить” и которая служит индикатором отсутствия файла. Функция *FileExists(name1)* осуществляет проверку наличия на диске файла с данным именем.

Процедура *Close(f)* завершает действия с файлом, который указывается в качестве ее параметра. При этом буфер, образованный при открытии файла, ликвидируется. После этого файловую переменную можно связать посредством процедуры *Assign* с каким-либо другим дисковым файлом.

Переименовать дисковый файл можно с помощью системной процедуры *Rename(f, newname).* Для уничтожения файла используется *процедура Erase(f).*

ЛАБОРАТОРНАЯ РАБОТА №4
Указатели, работа с динамическими структурами данных. Динамическое управление памятью

Методические указания

Варианты заданий для лабораторной работы включают обработку динамических массивов и связанных динамических данных линейной, кольцевой, древовидной и разветвленных структур.

В соответствии с полученным вариантом задания:

1. Составить блок-схему решения задачи на уровне процедур и функций. Блок-схема должна отражать логическую структуру алгоритма.
2. По разработанному алгоритму написать программу, удовлетворяющую следующим требованиям:

- все пользовательские типы, включая динамические, должны быть определены в разделе описания типов type;
- в программу включить меню работы с программой, которое должно содержать пункты:

а) создание исходной динамической структуры данных;

б) вывод на экран информации, хранящейся в динамической структуре данных;

в) поиск заданной информации в динамической структуре;

г) вставка и удаление информации из связанных динамических структур, корректировка значений;

д) выход из программы;

- алгоритм, соответствующий каждому пункту меню, должен быть реализован в виде подпрограммы – процедуры или функции.

Данные динамической структуры создаются и уничтожаются во время работы программы, связи и взаиморасположение элементов связанных динамических структур также изменяются в процессе выполнения программы.

Указатели, объявленные в разделе *var*, являются статическими переменными. В начале работы программы значение указателя не определено. Указателю можно присвоить значение адреса выделенной под переменную области памяти или значение *nil* – пустой адрес. После освобождения области памяти, с которой был связан указатель, его значение не определено.

Выделение и освобождение памяти под динамические переменные определенного типа выполняется стандартными процедурами *New* и *Dispose*. Пример создания и удаленя динамической записи:

```
type data=record
            i : byte;
            s : string
          end;
var p : ^data;
begin
   new(p);
   p^.i := 5;
   p^.s := 'string';
   write(p^.i:5,  p^.s:15);
   Dispose (p);
end.
```

При создании связанной динамической структуры каждый элемент данных должен состоять из двух полей – информационного и указательного. Тип такого элемента объявляется следующим образом:

```
point = ^ struct;
struct = record
           str : string;
           p : point
         end;
```

При этом тип указателя на элемент динамической структуры должен быть описан перед описанием типа этого элемента.

Возможно создание линейных (списков, очередей, стеков) и нелинейных (деревьев) динамических структур. В первом случае элемент структуры должен содержать одно поле указателя на элемент такого же типа, во втором случае – несколько полей указателей.

Линейный список (динамическая цепочка, строка) – совокупность линейно связанных однородных элементов, для которых разрешается добавление элемента в любое место и удаление любого элемента.

Кольцевой список имеет в последнем элементе значение поля указателя, равное адресу первого элемента списка.

Очередь – частный случай линейного списка, для которого разрешено добавление элемента только в конец очереди и удаление элемента из ее начала.

Стек – частный случай линейного списка, для которого разрешено добавлять или удалять элементы только с одного конца, называемого вершиной стека.

Пример программы

Сохранить текст, вводимый с клавиатуры, в динамическом списке. Признак окончания ввода текста – точка.

```
type point=^node;
            node=record
                   ch:char;
                    p:point
                   end;
var  r, cur:point;
     ch,sym:char;
procedure Creat(var r:point);
var  cur:point;
     sym:char;
begin
read(sym);
if sym='.' then
     begin r:=nil; exit end;
new(r); r^.ch:=sym; r^.p:=nil;
cur:=r; read(sym);
while sym <>'.' do
     begin
         new(cur^.p); cur:=cur^.p;
        cur^.ch:=sym; cur^.p:=nil;
       read(sym);
     end; readln;
end;

procedure Print(r:point );
 begin
   while r<>nil  do
    begin
      write(r^.ch);
      r:=r^.p;
    end;
end;

begin
 Creat(r);
Writeln('List created:'); Print(r);
 end.
```