Федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный аграрный университет имени П.А.Столыпина»

Университетский колледж агробизнеса

ООП по специальности 09.02.07 Информационные системы и программирование

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТОВ
по дисциплине
ОП.04 Основы алгоритмизации и программирования

Специальность: **09.02.07 Информационные системы и программирование**

Ведущий преподаватель (руководитель) дисциплины	А.В.Кортусов

Омск 2023

Пояснительная записка

Методические рекомендации по учебной дисциплине **ОП.04 Основы алгоритмизации и программирования** предназначены для выполнения самостоятельной работы обучающимися по специальности 09.02.07 Информационные системы и программирование.

Самостоятельная работа выполняется по заданию и при методическом руководстве преподавателя, но без его непосредственного участия.

Целью самостоятельной работы является овладение обучающимся умениями работать с источниками, обобщения и анализа юридической практики, аргументации собственной точки зрения.

Методические рекомендации по самостоятельной работе студентов содержат материалы для подготовки к лекционным, практическим занятиям, к формам текущего и промежуточного контроля. Наряду с методическими рекомендациями по подготовке и написанию курсовых работ, защите квалификационных работ составляют единый комплекс методического обеспечения студента по профессиональному модулю.

Предложенные в рекомендациях задания позволят успешно овладеть профессиональными знаниями, умениями и навыками, и направлены на формирование общих и профессиональных компетенций:

ОК. 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОП 9 Пользоваться профессиональной документацией на государственном и иностранном языках

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

При выполнении самостоятельной работы обучающийся самостоятельно осуществляет сбор, изучение, систематизацию и анализ информации, а затем оформляет информацию и представляет на оценку преподавателя или группы.

Виды самостоятельной работы

№	Вид самостоятельной работы	Форма контроля	Максимал кол-во ов
1.	Работа с источниками	Устный ответ на занятии Составление аннотации	5
2.	Составление опорного конспекта	Опорный конспект	5
3.	Составление сравнительной ицы	Сравнительная таблица	5
4.	Решение ситуационных задач	Письменный ответ	5
5.	Анализ судебной практики	Письменный отчет	5
6.	Участие в научно-едовательской деятельности*	Выступление на конференции	5

Методические рекомендации по работе с источниками

Работа с источниками осуществляется с целью приобретения обучающимся навыков самостоятельного изучения учебного материала. Работа с источниками является важной составляющей при подготовке к занятиям.

Для подготовки к устному опросу необходимо прочитать текст источника, выделить главное, составить план ответа, повторить текст несколько раз. На учебном занятии полно, точно, доступно, правильно, взаимосвязано и логично изложить материал, иллюстрируя при необходимости примерами.

Работа с источником может быть предложена в форме аннотирования. Аннотация позволяет составить обобщенное представление об источнике. Для составления аннотации необходимо ответить на следующие вопросы:

1. Фамилия автора, полное наименование работы, место и год издания.
2. Вид издания (статья, учебник, и пр.).
3. Цели и задачи издания.
4. Структура издания и краткий обзор содержания работы.
5. Основные проблемы, затронутые автором.
6. Выводы и предложения автора по решению выделенных проблем.

Источник аннотирования определяет преподаватель, он же оценивает аннотацию, сданную в письменной форме.

Методические рекомендации по составлению опорного конспекта

Опорный конспект составляется с целью обобщения, систематизации и краткого изложения информации. Составление опорного конспекта способствует более быстрому запоминанию учебного материала.

Составление опорного конспекта включает следующие действия:

1. Изучение текста учебного материала.
2. Определение главного и второстепенного в анализируемом тексте.
3. Установление логической последовательности между элементами.
4. Составление характеристики элементов учебного материала в краткой форме.
5. Выбор опорных сигналов для расстановки акцентов.
6. Оформление опорного конспекта.

Опорный конспект может быть представлен в виде схемы с использованием

стрелок для определения связи между элементами; системы геометрических фигур; логической лестницы и т.д.

Оценкой опорного конспекта может служить качество ответа, как самого студента, так и других студентов его использовавших. Преподаватель также может проверить опорные конспекты, сданные в письменной форме. Допускается проведение конкурса на самый лучший конспект по следующим критериям: краткость формы; логичность изложения; наглядность выполнения; универсальность содержания.

Методические рекомендации по составлению сравнительной таблицы

Сравнительная таблица составляется с целью выявления сходств, отличий, преимуществ и недостатков анализируемых объектов.

Критерии для составления сравнительной таблицы предлагает преподаватель. Студент, самостоятельно сформулировавший критерии для сравнения, получает дополнительные баллы.

Проверка и оценка сравнительной таблицы осуществляется преподавателем в письменной форме.

Методические рекомендации по решению ситуационных задач

Ситуационные задачи решаются с целью приобретения обучающимся навыков самостоятельной работы с источниками, обобщения и анализа юридической практики, а также умений аргументировать собственную точку зрения и делать выводы.

При решении задач студентам можно рекомендовать такую основную схему:

1) проанализировать приведенную в задаче ситуацию и поставленный вопрос;

2) найти оптимальный способ решения задачи;

4) составить в письменной форме мотивированный вывод по задаче.

Практическая работа № 1

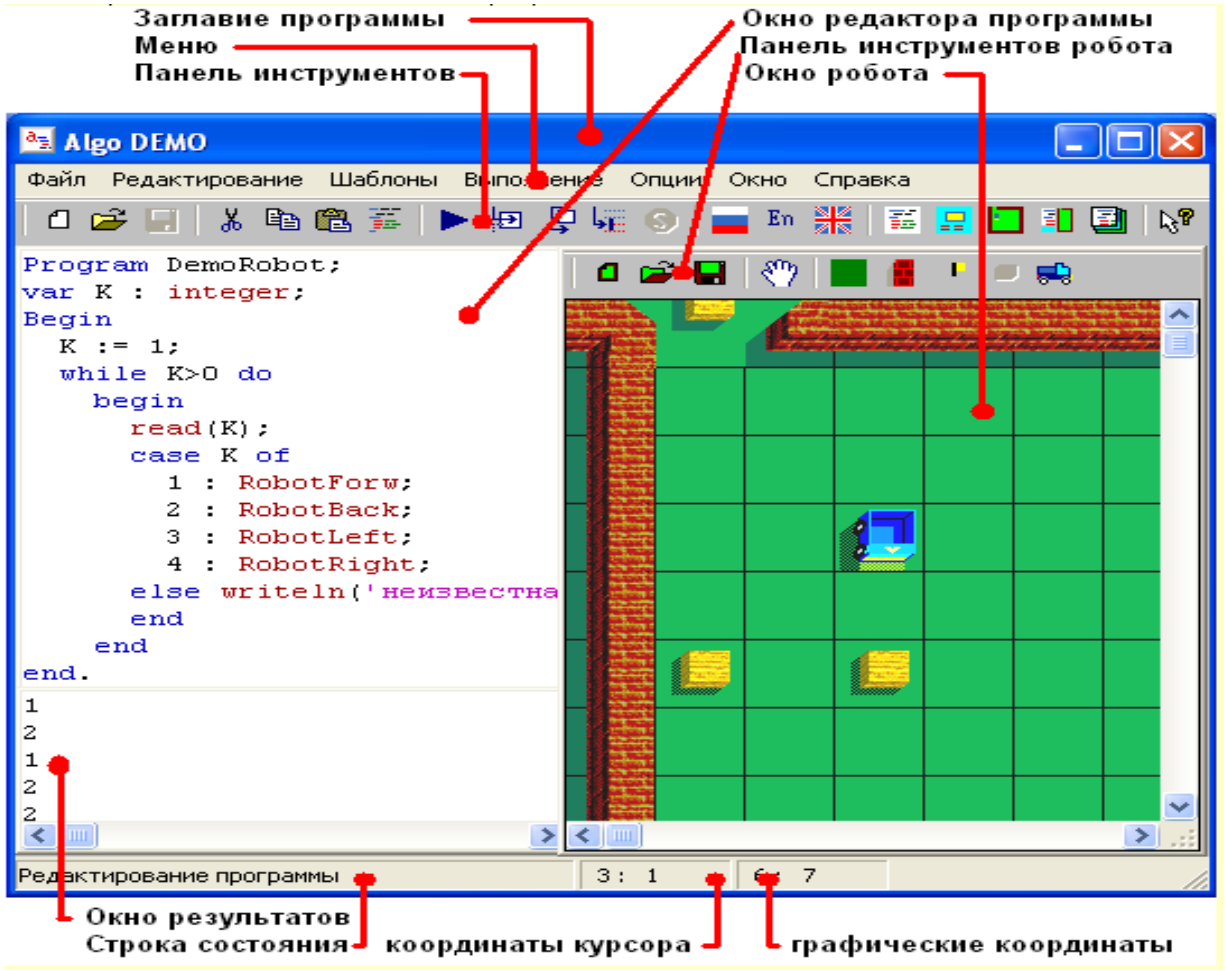
Тема: Знакомство с системой АВС Паскаль. Ввод программ.
Основные приемы работы.
Простейший ввод и вывод данных различных типов.

Цель работы:

- Формирование и отработка умений и навыков работы в среде АВС Паскаль.
- Научиться вводить, редактировать и запускать простейшие программы.
- Развивать познавательный интерес к программированию, предмету.

Краткая теория

АВС Паскаль, как и любая программа, имеет свой интерфейс:



Оболочка АВС Паскаля (АЛГО) выполнена двумя языками - английским и русским. Для перехода на английский язык надо нажать кнопку с английским флажком, а на русский -с российским (Ctrl+E; Ctrl+U) или через меню. Следствием переключения является перевод всех ключевых слов подготовленной программы на выбранный язык.

Аналогично можно выбрать язык среды (меню, сообщений об ошибках, диалоговых окон).

Язык оболочки и язык записи программы никак не связаны с режимом работы клавиа-туры. Для переключения клавиатуры из латинских букв на кириллические и наоборот, как правило используют варианты Ctrl + Shift или Alt + Shift.

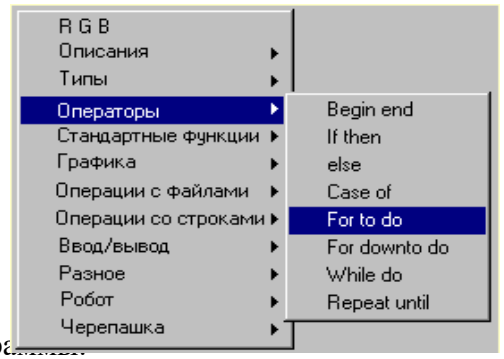
Рекомендуется записывать программу после набора каждых 10-20 строк текста и пе- ред первым выполнением. Иначе вследствие ошибочных действий или сбоя программы под-готовленный текст программы может быть утрачен.

Группа "Шаблоны"

Шаблоны используют для вставки в текст программы операторов, описаний и обращений к стандартным функциям и процедурам. При вставке оператора в текст программы,

имеющийся текст раздвигается.

Во время вставки шаблонов список фактических параметров содержит соответствующее количество запятых, которые указывают на количество параметров. Исключение представляют шаблоны группы Графика, использующие цвета, составляющиеR,G,B цвета. Для ввода чисел, которые задают нужный цвет, можно воспользоваться командой Шаблоны / RGB, которая позволяет с помощью системного диалога выбора цвета установить



нужный цвет и записывает R,G,B этого цвета в текст програ[illegible].

Операторы в Паскале отделяются друг от друга точкой с запятой и могут располагаться в одну строчку или начинаться с новой строки.

Наберем простейшую программу на Паскале с соблюдением правил.

Текст задачи: Ввести три целых числа (**A,B,C**) и вывести их среднее арифметическое (**D**)

Program Primer; Заголовок программы с именем
Var A,B,C : Integer; Объявлены три переменных целого типа
D : Real ; Объявлена одна переменная дробного типа
Begin Начало твоей программы
Write(' Укажи целые A, B, C - '); Вывод произвольного текста на экран. **ReadLn(A,B,C);** Ввод пользователем значений переменных **A,B,C**
WriteLn('----------'); Вывод черты (текста) на экран.
D := (A+B+C)/3; Вычисление значения переменной **D** по формуле
Writeln(' D = ',D:6:2); Вывод текста и значения переменной **D**
End. Конец программы

Ваши программы могут содержать следующие символы:
Латинские буквы - A,B,C, … a, b, … x, y, z.
Русские буквы - А,Б,В, … а, б, … э, ю, я.
Цифры - 0, 1, 2, …9.
Специальные символы - +, -, /, = , >=, <=, <>, [,], (,), ;, :, #, …

В качестве **имен переменных** и других объектов можно использовать **латинские** бу-квы, буквы с цифрой, латинские слова без пробелов (высота букв значения не имеют).

Для решения задач в любой программе выполняется обработка **данных.**

Данные могут быть самых различных типов: целые и дробные (вещественные) числа,символы (отдельные буквы), строки (текст), массивы (таблицы).

Все данные в языке Паскаль должны быть обязательно описаны в начале про-граммы в разделе Var.

Различные типы данных занимают различные объемы памяти. Переменные могут ме-нять свои значения в процессе выполнения программы.

Пример описания переменных для чисел:

Var
A, B : Real; - Дробное число
C : Integer; - Целое число от -2147483648 до 2147483647

Пример описания переменных для текста:

Var
: Char; - Текст длиной в один любой символ .
: String; - Длинный текст (до 255 любых символов)
: String[15]; - Текст с фиксированным количеством символов (15)

Пример описания переменных логического (булевского) типа:

Var
A : Boolean; - Логический тип данных принимает одно из двух значений – **True(** Истина) или **False**(Ложь).

Наберем простейшую программу на Паскале с соблюдением правил.

Текст задачи: Ввести целое число (**X**) и вывести таблицу умножения на это числоиз 3 строк, начиная от 1 до 3, по образцу:

Укажи твое число - 8
----------1 x 8 = 8
2 x 8 = 16
3 x 8 = 24

Программу набирал – Иванов В.

```
Program ab;
Var      X,Y : Integer;                  Объявлены две переменных целого типа
Z            : String;                   Объявлена переменная типа текст
Begin                                    Начало программы
Write(' Укажи число X - ');              Вывод текста на экран.
ReadLn(X);                               Ввод значения переменной X (Целого числа –X)
Write(' Укажи свое имя - ');             Вывод текста на экран.
ReadLn(Z);                               Ввод значения Z ( Текста твоего имени)
WriteLn('------------------------------');  Вывод черты (текста) на экран.
Y := 1;                                  Переменной Y присвоили значение 1
WriteLn(' ',Y,' x ',X,' = ',X*Y);        Вывели на экран 1 x 8 = 8
Y := Y + 1;                              Переменой Y присвоили значение на 1 больше
WriteLn(' ',Y,' x ',X,' = ',X*Y);        Вывели на экран 2 x 8 = 16
Y := Y + 1;                              Переменой Y присвоили значение на 1 больше
WriteLn(' ',Y+1,' x ',X,' = ',X*Y); Вывели на экран 3 x 8 = 24 WriteLn('-
------------------------------');        Вывод черты (текста) на экран.
WriteLn(' Программу набирал -',Z); Вывели на экран текст и твое имяEnd.
                                         Конец программы
```

Задания к работе

1. Загрузите АВС Паскаль.

1. Наберите предыдущую программу-код.
2. Запустите набранную программу несколько раз с различными значениями переменных A, B, C, поменяйте цифры в предпоследней строке на **6:3**,
проанализируйте результат.
3. Переключите программу на русский вариант написания команд, запустите код.
4. Верните английский вариант написания команд.
5. Сохраните свою программу в **D:/Паскаль** с именем **Primer_Иванов**
6. Закройте программу Паскаль
7. Загрузите АВС Паскаль снова.
8. Откройте свою программу.
9. Введите первоначальное целое число и выведите числа по схеме:

Укажите число - 9

9
10
11
12

10. Введите 2 целых числа и выведите числа по схеме:

Укажите 2 целых числа - 15 27

15 27
16 26
17 25
18 24

11. Введите первоначальное целое число и выведите числа по схеме:

Укажите число - 15

15 15
14 14
13 13
12 12

12. Наберите программу формирования таблицы Пифагора для 3-4 строк, по образцу:

Практическая работа № 2

Тема: Стандартные операции и функции Паскаля.
Форматированный вывод информации.

Цель работы:

- Формирование и отработка умений и навыков работы по вводу и выводу информации разного типа в среде ABC Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

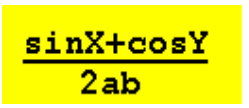
Краткая теория

В Паскале пользователь может вычислять значения сложных выражений, состоящихиз многих переменных и функций, с использованием скобок. Любые выражения записываются в одну строку с использованием скобок, что позволяет менять очередность вычислений в сложных выражениях.

Пример выражения:

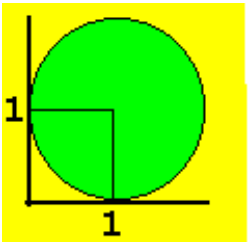
Y := ((A+B)/(C+D*Cos(X))*2)/4 - Переменной **Y** присвоено значение выражения

Пример выражения.




```
(sin(x)+cos(y))/(2*a*b)
```

Пример из математики




```
(sqr(x-1)+sqr(y-1))<1
```

Выражение принимает значение TRUE, когда точка (X,Y) находится внутри круга

В Паскале в распоряжении пользователя имеются множество операций и функций.
Вот некоторые из них:

Операции целочисленной арифметики:

Целочисленное деление **Div** - (при целочисленном делении операция **Div** возвращает целую часть частного (дробная часть отбрасывается))

11 div 4 = 27 div 3 = 2
26 div 5 = 5

Остаток от деления **Mod** - (остаток от деления (целая часть отбрасывается))

11 mod 4 = 3
7 mod 3 = 1
26 mod 5 = 1

Обратите внимание: если X Mod Y = 0, то X кратно Y
если X Mod 2 = 0, то X четное число
если X Mod 2 = 1, то X нечетное число

Стандартные функции Паскаля: Abs(X) - Модуль числа
Sin(X) - Функция Синус
Cos(X) - Функция Косинус
Sqrt(X) - Корень квадратный
Round(X) - Округление числа
Trunc(X) - Целая часть дроби
Random - Случайное дробное число от 0 до 1
Random(N) - Случайное целое число от 0 до N

При использовании операций и функций пользователю необходимо помнить о типахданных исходных и конечных значений, например частное от деления двух целых чисел может быть дробным числом, что естественно соответствующим образом должно быть объявлено в разделе переменных **Var**.

Форматированный вывод информации:

Операторы **Write** и **Writeln** выводят значения переменных и тексты на экран. Ве-щественные (дробные) числа выводятся в формате с плавающей точкой, например: **1.7480000000E+02,** что не совсем удобно и наглядно.

Решаем проблему с помощью форматированного вывода информации, где имеется возможность указать ширину поля вывода для переменной или выражения, например: **Var**

```
      A      :
    string[15];
    B,C :
Begin integer;
    D,Y :
    Real;

     ReadL
    n(B,C);
    ReadLn(
    A);
     D := B
    Mod C;
     Y :=
    25.178;
     Y := Int(Y);
     WriteLn(Y:6:1);
     WriteLn(‘ A = ‘, A:22);  Вывод значения текста A шириной в 22
    символа WriteLn(B:6, C:4); Вывод значений чисел B и C шириной 6 и 4
    символаWriteLn(D:8:2);    Вывод дробного числа D шириной 8 с 2
    знаками
End.                                  после запятой
```

Форматированный вывод удобно использовать при выводе **чисел** и **текста** для выравнивания столбцов выводимой информации.

Задания к работе

1. Загрузите ABC Паскаль.
2. Наберите программу ввода одного или двух чисел и вывода в отформатированном виде всех вышеперечисленных значений функций Паскаля для введенных чисел. Сохраните программу **D:/Паскаль/R_02_Иванов**
3. Введите дробное число. Выведите отдельно на своей строке дробную и целую части.
4. Запиши программу вычисления выражения. Значения **X** и **Y** вводятся.

$$a = \frac{|x-y|}{10x + |xy + x + y|}$$

5. Запиши программу вычисления выражения. Значения **S, A**, вводятся.

$$f = s^2 + \frac{a}{a+s}\sqrt{s^4 - s^3} - 176$$

6. Запиши программу вычисления выражения. Значения **X, Y, Z** вводятся.

$$a = \frac{\sqrt{|x-1|+y} - \sqrt{|y+z|}}{1 + \frac{x^2}{2+z} + \frac{y^2}{4+z}} + zxy$$

7. Запиши программу вычисления суммы квадратов Синуса и Косинуса для двух-трех значений аргументов. $(\mathrm{Sin}(X))^2 + (\mathrm{Cos}(X))^2$
8. Напиши программу вычисления площади треугольника, если известны его стороны

9. Напиши программу ввода A, B. Увеличить A в 20 раз, уменьшить B в 30 раз. Во сколько раз отличается первоначальная сумма A + B от конечной суммы A + B ?

10. Найдите площадь прямоугольной рамки, внутренняя сторона которой равна 5, а внешняя заданному числу R >= 5.

Практическая работа № 3

Тема: Условный оператор и оператор выбора Паскаля.

Цель работы:

- Формирование и отработка умений и навыков работы по использованию ветвлений и выбора в среде программирования АВС Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

Решение большинства задач **редко** сводится к простому последовательному расчету (линейным алгоритмам). Чаще порядок вычислений зависит от определенных условий, на- пример от исходных данных или от промежуточных результатов, полученных несколько ра-нее. Для организации вычислений в зависимости от какого-либо условия в Паскале исполь- зуется условный оператор.

Для формулировки условия используются следующие логические операции:

>	- больше	>=	- больше либо равно
<	- меньше	<=	- меньше либо равно
=	- равно	<>	- не равно

Кроме того, для записи сложного условия, когда необходимо сразу проверить не-сколько условий, используются операции их объединения:

And (оба условия должны выполняться - **И**)

Or (хотя бы одно из условий должно выполняться - **Или**)

Not (Логическое отрицание - **Не**)

При выполнении программы компьютером или человеком при проверке, любое условие может иметь только два значения: либо **истина (True**), когда оно выполняется,либо **ложь (False),** когда оно не выполняется.

Пример условного оператора на Паскале:

```
A := 2;  B := 4;
If  A > B  Then
WriteLn(‘  A > B  ‘)
Else
WriteLn(‘ A <= B ‘);
```

Если условие **A > B** истинно, то выполняются команды между **Then** и **Else.** Но если условие ложно, то будут выполняться команды после **Else.**

По ветке **Then**, и по ветке **Else** должен выполняться единственный оператор.

Если по смыслу задачи необходимо выполнить несколько команд, тогда следует исполь-зовать скобки (составной оператор), например:

```
A := 2;  B := 4;  C := 0;
If A > B ThenBegin
WriteLn(‘ A > B ‘);C := A-B;
End
Else
WriteLn(‘ A <= B ‘);
```

```
If A > B ThenBegin
WriteLn(‘ A > B ‘);End
                C := A-B;
```

Альтернативная часть **Else** может вообще отсутствовать, если в ней нет необходимости**.**
Это так называемый неполный вариант ветвления.

```
A := 2;  B := 4;  C := 0;
If A > B ThenBegin
If A = C Thenbegin
WriteLn(‘ A = C ‘);C := A-B;
endelse
                    begin
                    C := A + B;
                    WriteLn(‘ A <>
                C’);end
EndElse
WriteLn(‘ A <= B ‘);
```

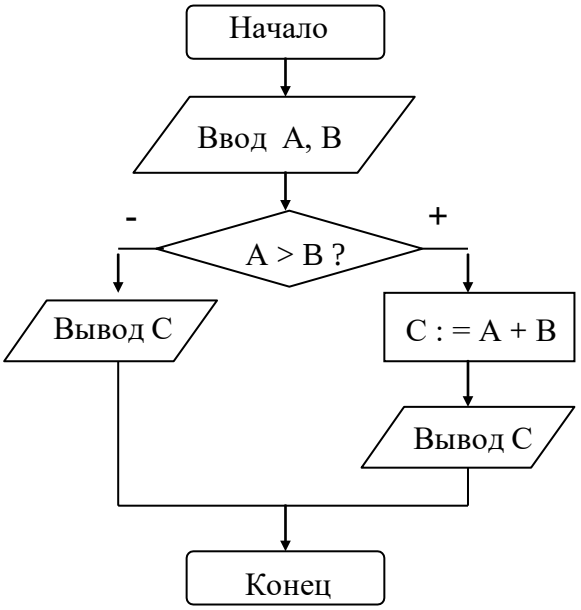
Условные операторы могут быть вложены друг в друга. При использовании вложенных ветвлений во избежание недоразумений нужно также использовать скобки - **Begin** и **End**

Условие может состоять из нескольких простых условий. В этих случаях каждое простое условие заключается в круглые скобки и используются логические связки (логические операции) - **And, Or, Not**

Пример

If (X>Y) And (Y>=0) Or (X=0) Then

Конструкцию ветвления (условный операто
пример простой блок-схемы, содержащей ветв



<u>Использование оператора варианта:</u>

Оператор варианта **Case** необходим в тех случаях, когда в зависимости от значенийкакой-либо переменной надо выполнить те или иные операторы.

```
Case управляющая переменная of Набор
значений 1 : оператор 1;
Набор значений 2 : оператор 2;
Набор значений 3 : оператор 3;
……
Набор значений N : оператор N;
Else
Альтернативный оператор;
End;
```

Если управляющая переменная принимает значение из набора значений 1, то выполняется оператор 1.

Если управляющая переменная принимает значение из набора значений 2, то выполняется оператор 2 и т.д.

Если управляющая переменная не принимает ни одно значение из имеющихся наборов, то выполняется альтернативный оператор.

Примеры:

```
Program ab;
Var D : Integer;Begin
WriteLn(‘ Укажи число от 1 до 31 – ‘); ReadLn(D);Case  D
mod  7  of
1 : WriteLn(‘ Понедельник ‘);2 :  WriteLn(‘
Вторник ‘);
3                    :  WriteLn(‘ Среда ‘);
4                    : WriteLn(‘ Понедельник ‘);
5 :  WriteLn(‘ Вторник ‘);
6 :  WriteLn(‘ Среда ‘);
                     E
                l        WriteLn(‘ Воскресенье ‘);
        End;    s
                e
```

```
Program ab;
Var D : Integer;Begin
WriteLn(‘ Укажи число от 1 до 12 – ‘); ReadLn(D);If (D >= 1)
and  (D <= 12)  Then
Case  D  of
12, 1, 2 :  WriteLn(‘ Зима ‘);
3.. 5 :  WriteLn(‘ Весна ‘);
6.. 8 :  WriteLn(‘ Лето ‘);
9.. 11 : WriteLn(‘ Осень ‘);End
Else
WriteLn(‘ Введено ошибочное число ! ‘);
End.
```

Задания к работе

1. Введите три числа. Найдите Минимальное из них. Покажи его.
2. Введите три числа. Найдите разность между Максимальным и Минимальным из них. Покажи разность.
3. Введите четыре числа. Найдите разность между Максимальным и Минимальным из них. Покажи разность.
4. Введите натуральное число и вопрос о характере вывода чисел. Выведите 4-5 четных или нечетных числа через запятую между ними.
5. Получи несколько чисел (4-5 чисел) через запятую, если известно, что каждое очередное число получается суммой двух предыдущих. Первые два числа ввести.
6. Ввести целое двузначное число. Проверить кратно ли оно - 7.
7. Ввести четыре различных числа.

   Программа не должна разрешать ввод одинаковых чисел.
8. Ввести число. Проверить делится ли оно на 3 и 11 нацело одновременно.
9. Ввести целое однозначное число. Проверить является ли оно простым. (Число считается простым, если делится само на себя и единицу).

Практическая работа № 4

Тема: Циклические структуры Паскаля.

Цель работы:

- Формирование и отработка умений и навыков работы по использованию циклических структур в среде программирования Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

Циклом в программировании называют повторение одних и тех же действий (команд).

Последовательность действий, которые повторяются в цикле, называют - **телом цикла**.

Существует несколько типов циклических структур:

1. цикл со счетчиком
2. цикл с предусловием
3. цикл с постусловием.

Цикл со счетчиком:

Данный цикл обычно используют тогда, когда тело цикла должно быть выполнено заданное количество раз. Примеры:

```
----------------------------------------
Var I, C : integer;Begin
For I:= 3 to 50 doBegin
WriteLn(‘ I = ‘, I);C:= I*I;
End
----------------------------------------
For I:= 30 downto - 5 do
WriteLn(‘ I = ‘, I);
----------------------------------------
For N:= ‘A’ to ‘R’ do
WriteLn(NI);
----------------------------------------
```

Выполнение цикла начинается с присвоения параметру стартового значения. Затем следует проверка, не превосходит ли параметр конечного значения. Если результат проверки утвердительный, то цикл не выполняется ни разу и управление передается к первой команде после цикла. В противном случае выполняется тело цикла и параметр автоматически меняет свое значение на следующее (на 1-у больше или на 1-у меньше). Далее снова происходит проверка значения параметра (счетчика) цикла, и алгоритм повторяется.

Оператор цикла с предусловием - (W

Пример:

```
X :=  - 120;
While X < 0 doBegin
WriteLn(X);X := X +4.5;
End;
```

Условие выполнения тела цикла While проверяется **до начала** работы, при входе в цикл. Поэтому, если условие сразу не выполняется, то тело цикла игнорируется и управление переда ется первому оператору после цикла.

Тело цикла будет выполняться до тех пор, пока логическое условие истинно.

В цикле данного типа (с предусловием) предварительной проверкой определяется, выполнять тело цикла или нет.

В этом цикле пользователь предусматривает изменение всех переменных, здесь ничего автоматически не меняется.

Оператор цикла с постусловие

Пример:

```
X :=  - 120;
Repeat
WriteLn(X);X := X
+4.5;
UntiL  X > 0
```

Цикл с постусловием всегда выполняется хотя бы один раз.

Тело цикла Repeat выполняется до тех пор, пока не станет истинным условие после UntiL .

В данном цикле пользователь сам предусматривает изменение всех переменных, здесь также автоматически ничего не меняется.

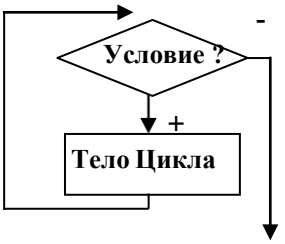
Скобки в этом типе цикла не нужны.

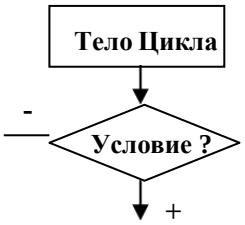
Для всех циклов по команде **Exit** можно покинуть цикл. Пример:

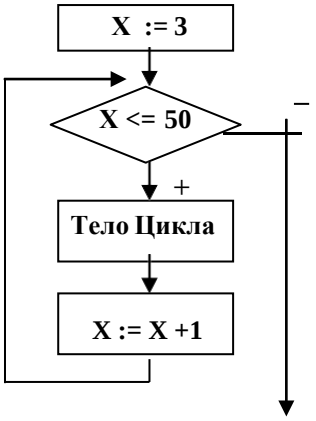
```
X :=  - 120;
Repeat
WriteLn(X);X := X
+4.5;
If X = - 20 then Exit;UntiL  X >
0;
```

При использовании циклов следует помнить следующее:
- Чтобы цикл гарантированно заканчивался, содержимое цикла должно обязательно влиять на условие цикла.
- Условие должно состоять из корректных выражений и значений, определенных до входа в цикл.

В заключении нарисуем блок-схемы циклических структур Паскаля.







Задания к работе

1. Введите два натуральных числа. Выведите все числа через запятую между ними, используя различные виды циклов.
2. Введите два натуральных числа и вопрос о характере выводимых чисел. Выведите все четные или нечетные числа через запятую между ними, используя все типы циклов.
3. Запиши программу вычисления Факториала указанного числа. (5**!** = 1*2*3*4*5)
4. Получи 5-6 чисел через запятую, используя цикл, если известно, что каждое очередное число получается суммой двух предыдущих. Первые два числа ввести.
5. Ввести 2 произвольных целых числа. На сколько отличаются суммы четных чисел от нечетных чисел от первого до второго числа.
6. Ввести натуральное двухзначное число. Определить, является ли оно простым. (Число – простое, если оно делится без остатка на себя и единицу).
7. Запиши программу нахождения НОД (Наибольшего Общего Делителя двух чисел)
8. При возведении в квадрат иногда последние цифры повторяют эти числа. (5 х 5 = 25). Между 25 и 1000 есть такие числа. Покажи их.
9. Найди все целые двузначные числа кратные 7.
10. Найди все целые двузначные числа, сумма цифр которых равна 8.
11. Найди все трехзначные числа, сумма первых двух цифр которых меньше суммы второй и третьей цифр.
12. Среди четырехзначных целых чисел найди те, в которых все цифры различны.
13. Найди все целые трехзначные числа, которые делятся на 3 и 11.
14. Число 66 можно представить в виде суммы четырех последовательных чисел. Найдите их.
15. Число 1190 можно представить в виде произведения двух последовательных чисел. Найдите их.
16. Сколько 0 в произведении всех натуральных чисел от 10 до 20.
17. Чтобы пронумеровать страницы в книге, понадобилось 1164 цифры. Сколько в ней страниц.
18. Сколько раз встречается цифра 1 в числах от 1 до 1000 включительно.

Практическая работа № 5

Тема: Обработка массивов.

Цель работы:

- Формирование и отработка умений и навыков работы по использованию массивов в среде программирования Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

Часто для работы с множеством однотипных данных (числами, датами, текстами и т.п.) оказывается удобным использовать массивы. **Массив** – структурированный тип дан- ных, состоящий из конечного числа однотипных элементов. Для доступа к данным, храня- щимся в определенном элементе массива, необходимо указать имя массива и порядковый номер этого элемента, называемый **индексом**. Массивы бывают одномерные (линейные), двумерные (прямоугольные) и т.д.

При работе с массивами их необходимо, как и все данные, описать в разделе перемен- ных **Var**. Это можно сделать так:

```
Var A : array[1..50] of integer;B :
array[1..12] of string;
C :  array[1..10,1..15]  of real;
```

После объявления массива его элементы **пусты.**

1. Объявлен линейный массив с именем A из целых чисел, состоящий из 50 элементов. Нумерация элементов с 1.
2. Объявлен линейный массив с именем B из тек- стов, состоящий из 12 элементов.
3. Объявлен двумерный массив с именем C из дроб- ных чисел, состоящий из 10 строк по 15 столбцов.

Для обращения к отдельному элементу массива, присвоения значения отдельному элементу используют имя массива и порядковый номер элемента (индекс).

При заполнении массива данными необходимо последовательно вводить 1-й, 2-й, 3-й и т.д. его элементы и аналогичным образом поступать при его выводе на экран или принтер. Следовательно, для ввода и вывода необходимо **организовывать цикл**, в котором практиче- ски все операции с массивом необходимо проводить поэлементно.

Пример присваивания значения конкретным элементам:

A[1] := 2; - Первому элементу присвоено значение 2
A[3] := X+1; - Третьему элементу присвоено значение X+1
A[N] := A[N-1]; - Элементу с номером N присвоено значение предыдущего элемента
A[N] := N; - Элементу с номером N присвоено значение N

При заполнении массива элементы могут в цикле заполняться пользователем склавиатуры или в цикле программно (автоматически).

Пример ручного ввода: Var A :

```
array[1..12] of  Real;
J, N : integer;Begin
Write(‘ Укажи кол-во элементов массива не более 12 – ‘);ReadLn(N);
For J := 1 to N doBegin
                    Write(‘ A [ ‘,J,’] = ‘);
                    ReadLn(A[J]);
                E
End.        n
            d
            ;
```

Пример программного ввода массива с параллельным показом:

```
Var A : array[1..120] of Integer;J, N :
integer;
Begin
Write(‘ Укажи кол-во элементов массива не более 120 – ‘);ReadLn(N);
For J := 1 to N doBegin
                                A[J] := J +
                            2; Write( A [J] ,
                        E   ‘,              ‘);
            End.        n
                        d
                        ;
```

Пример вычисления суммы только четных элементов массива кратных 5

```
Program
ab;
Var A : array[1..10] of  Integer;J, N :
integer;
Begin              N  := 0;  Randomize;        При каждом пуске – новая череда случайных чисел
For J := 1 to 10 doBegin
A[J] := Random(100);                           клетке с номером  J  присваивается случайное
Write( A [J] , ‘,                       ‘);    число от 0 до 100 и показывается на экране.
End;
WriteLn;                                       Перевели курсор на следующую строку
WriteLn(‘ ---------------------------------------------------------------------------------------- ‘);
J := 1;
While J <= 10 doBegin
If (A[J] mod 5  = 0)  and  (A[J]  mod  2  = 0)  ThenN := N + A[J];
J := J + 1;
End;
WriteLn(‘ Сумма  =  ‘, N:4);
End.
```

Пример поиска максимального элемента массива и его номера:

```
Var  A :
array[1..20]  of Integer;
J, Nomer, Max : integer;Begin
                    Randomize;
For J := 1 to 20 doBegin
A[J] := Random(100);  Write( A [J] , ‘,                    ‘); WriteLn; WriteLn;
End;
Max := A[1];  Nomer := 1;For J := 2  to  20
do
Begin
If A[J] > Max ThenBegin
                                    Max := A[J];   Nomer := J;
                                E
                    End;        n
                                d
                                ;
WriteLn(‘ Максим. элемент =  ‘, Max:4,’  Его номер –  ‘, Nomer:3);
End.
```

Обработка определенной части элементов прямоугольного массива:Элементы главной диагонали:

```
for I := 1 to N dobegin
{ цикл с элементами a[i,i] }
end;
```

Выше главной диагонали:

```
for  I := 1 to N do
begin
for J := i+1 to N dobegin
                                { цикл с элементами a[i,J] }
                            e
            end;        n
                        d
                        ;
```

Ниже главной диагонали:

```
for  I := 1 to N do
begin
for J := 1 to I - 1 dobegin
                                { цикл с элементами a[i,J] }
                            e
            end;        n
                        d
                        ;
```

Элементы обратной диагонали:

```
for  I := 1 to N do
begin
{ цикл с элементами a[i,N+i-1] }
end;
```

Выше обратной диагонали:

```
For I := 1 to N-1 dobegin
for J := 1 to N-i dobegin
{ цикл с элементами a[i,J] }end;
end;
```

Ниже обратной диагонала:

```
for  I := 2 to N do
begin
for j := N-i+2 to N dobegin
{ цикл с элементами a[i,J] }end;
end;
```

Задания к работе

1. Найди разность между максимальным и минимальным элементами массива.
2. Подсчитайте кол-во и сумму всех положительных элементов числового массива.
3. Введите произвольное число. Заполните массив. Подсчитайте количество элементов, которые меньше указанного числа.
4. Заполни массив целыми случайными числами. Подсчитай сумму всех четных чисел.
5. Заполни массив целыми случайными числами. Подсчитай сумму всех простых чисел
6. Заполни массив целыми случайными числами, покажи его. Поменяй местами числа всех четных и нечетных ячеек. Покажи все элементы после перестановки.
7. Заполни массив случайными дробными числами. Первую половину массива заполни дробной частью числа а вторую половину массива целой частью.
8. Заполни массив целыми случайными числами. Покажи числа. Помести элементы в обратном порядке.
9. Заполни массив случайными целыми числами. Сдвинь все элементы налево на 1 число а последний элемент перенеси в 1 клетку массива.
10. Заполни двумерный массив случайными целыми числами так, чтобы на его обоих диагоналях были только 0.
11. Запиши программу, реализующую сортировку числового массива по убыванию.
12. Заполните массив A(N,N) целыми случайными числами. Покажите индекс самого первого положительного элемента, кратного заданному числу.
13. Заполни массив случайными числами. Найти наибольшее кол-во равных элементов.
14. Заполни массив случайными числами, включая отрицательные. Вычисли максимальное число подряд идущих положительных чисел.
15. Заполни массив случайными числами. Найди сумму уникальных элементов.
16. Заполни массив случайными, не повторяющимися числами.
17. Заполните массив A(N,N) целыми случайными числами. Покажите все числа выше главной диагонали, а все остальные замени 0.
18. Заполните массив A(N,N) целыми случайными числами. Покажите все числа ниже главной диагонали, а все остальные замени 0.
19. Заполните массив A(N,N) целыми случайными числами. Замените самое первое отрицательное число, его индексом.
20. Заполните массив A(N,N) целыми случайными числами. Найдите кол-во строк, не содержащих отрицательных элементов. Покажите их номера.
21. Заполни массив. Задай произвольный числовой диапазон, например 3-8. Подсчитай кол-во строк, содержащих хотя бы один элемент из Вашего диапазона. Выведи их но-мера.
22. Заполни массив случайными числами. Подсчитай количество различных (не повторяющихся) чисел. Покажи их.
23. Заполни массив. Покажи его элементы. Покажи его после удаления каждого третьего элемента.
24. Заполните массив A(N,N) целыми случайными числами. Покажи все его элементы, расположенные выше главной диагонали, остальные замени пробелом.
25. Заполните массив A(N,N) целыми случайными числами. Покажи все его элементы, расположенные ниже главной диагонали, остальные замени пробелом.
26. Создай проект «Проверка знаний», так чтобы ваш проект проверял знание таблицы умножения у первоклассника.

Практическая работа № 6

Тема: Графика АВС Паскаля - АЛГО.

Цель работы:

- Формирование и отработка умений и навыков по использованию графики в среде программирования Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

Для графики в среде ALGO выделено рабочее поле – так называемый лист, просмат- риваемый ползунками. Началом координат считается верхний левый угол листа.

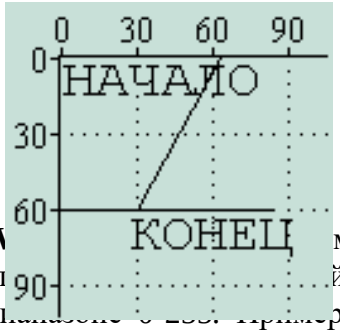
Ось X направлена слева направо, Y - сверху вниз. Для всей графики координаты за- дают в форме (X,Y). Если координаты при построении изображения выходят за пределы лис-та, то информация за пределами листа игнорируется. Исключение составляют операторы ReadLn и WriteLn. При попытке вывода за нижнюю границу листа все изображение двигает- ся вверх на величину высоты символа.

Система запоминает последний вывод и параметры инструментов. В момент запуска программы устанавливаются нулевые координаты (верхний левый угол экрана), черный ка- рандаш единичной толщины и черный цвет текста. Цвет заполнения замкнутых фигур (цвет кисти) выбирается прозрачный, т.е. фигуры не закрашиваются.

В процессе работы активные координаты меняются так, чтобы следующий вывод осуществлялся там, где закончился предыдущий.

Увидеть, как меняются текущие координаты, можно на примере такой программы:

```
Program Координаты;Begin
Write( 'НАЧАЛО' );LineTo(
30, 60 ); Write( 'КОНЕЦ' );
LineTo( 0, 100 )End.
```



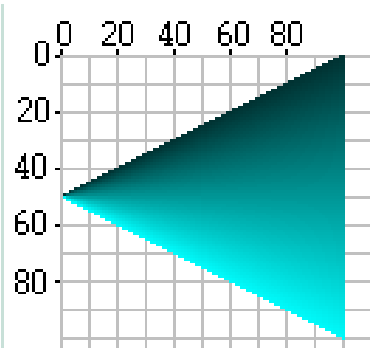
Цвет текста для **Read** и **W**мандой ЦветТекста:

По умолчанию установлен черный цвет**й**, Зеленый, Синий), пользователь регулирует каждый цвет в диапазоне 0-255. Пример:

```
Program ЦветТекста;Begin
WriteLn( ' Информатика ' );  -  Текст Информатика выведется черным цветом.
TextColor( 0, 0, 255 );                    - Включили для будущего вывода Синий цвет.
WriteLn( ' Майна ' )
end.
```

Переместиться в точку: MoveTo (x, y)Program

```
Перемещение;
Var
i: integer;Begin
For i: =0 to 100 doBegin
MoveTo( 100, i );
Pen( 1, 0, 2*i+50, 2*i+50 );
LineTo( 0, 50)
endend.
```

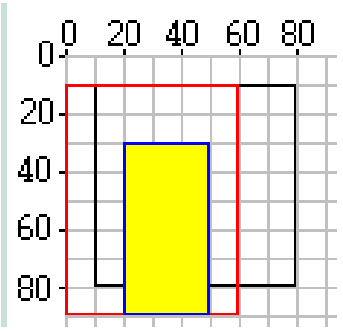


Построение Прямоугольника: Rectangle (x1, y1, x2, y2)

Где **x1,y1** и **x2,y2** - координаты двух диаметрально противоположных вершин пря-моугольника. Прямоугольник будет построен активным карандашом и закрашен активным цветом кисти, или не закрашен, если установлен прозрачный цвет кисти.

После выполнения процедуры активная графическая позиция устанавливается в точку с координатами **(x2, y2).**

```
Program Прямоугольник;Begin
Rectangle( 10, 10, 80, 80 );
Pen( 1, 255, 0, 0 );
Rectangle( 60, 10, 0, 90 );
Pen( 1, 0, 0, 255 );
Brush( 1, 255, 255, 0 );
Rectangle( 20, 30, 50, 90)
end.
```

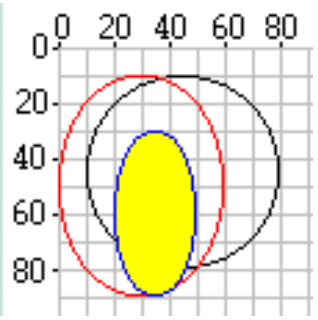


Построение Эллипса: Ellipse (x1, y1, x2, y2)

где x1, y1 и x2, y2 - координаты двух диаметрально противоположных вершин пря-моугольника, в который вписывается эллипс. Эллипс будет построен активным карандашом и закрашен активной кисточкой, или не закрашен, если установлен прозрачный цвет кисти.

После выполнения процедуры активная графическая позиция не изменяется Program Еллипс;

```
Begin
Ellipse( 10, 10, 80, 80 );
Pen( 1, 255, 0, 0 );
Ellipse( 60, 10, 0, 90 );
Pen( 1, 0, 0, 255 );
Brush( 1, 255, 255, 0 );
Ellipse( 20, 30, 50, 90)
end.
```

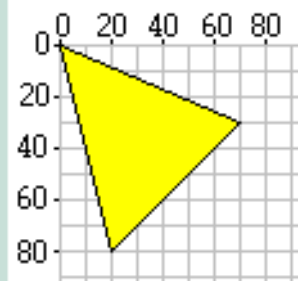


Закрасить замкнутую фигуру: Fill (x, y)

Начиная из точки с координатами x,y, процедура заполняет экран во всех направлени-ях активным цветом кисти до тех пор, пока не встретится граница другого цвета, чем цвет указанной точки (x,y), т.е. процедура меняет цвет запертой области, внутри которой лежит точка (x,y) на активный цвет кисти.

Процедура не выполняет никаких действий, если задан прозрачный цвет заполнения или цвет точки (x,y) совпадает с цветом заполнения.

```
Program Закраска;Begin
LineTo( 70, 30 );
LineTo( 20, 80 );
LineTo( 0, 0 );
Brush( 1, 255, 255, 0 );
Fill( 10, 10)
end.
```



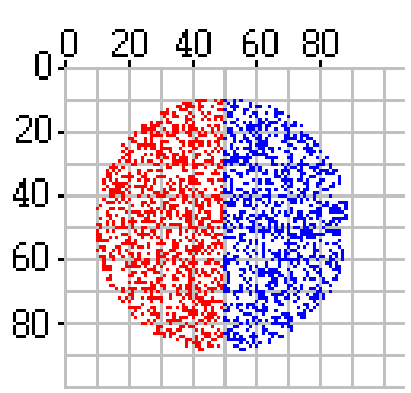
Очистить окна результатов: Clear

Очищает окно результатов и устанавливает начальные параметры графики (цвет,толщину, координаты …)

Нарисовать Точку: Point (x, y)

Цвет точки определяется активным цветом карандаша. После выполнения процеду-ры активная графическая позиция устанавливается в точку с координатами (x, y).

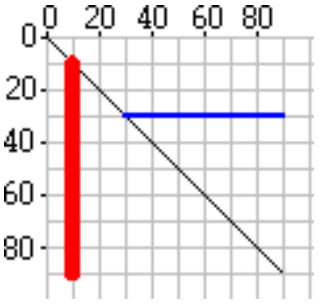
```
Program Точка;Var
i, x, y: integer;Begin
For i:=1 to 5000 doBegin
x := random( 100 );y := random( 100 );
If sqr( x-50 )+ sqr( y-50 ) < 1600 thenBegin
If  x > 50  then
Pen( 1, 0, 0, 255 )
                                    e
                                                Pen( 1, 255, 0, 0 );
                                                Point( x, y )
                                lse
                            e
                end.        n
                            d       end
```



Нарисовать Линию : Line (x1, y1, x2, y2)

где **(x1, y1)** и **(x2, y2)** - координаты концов линии. Линия будет проведена активным карандашом. После выполнения процедуры активная графическая позиция устанавливается в точку с координатами **(x2,y2).**

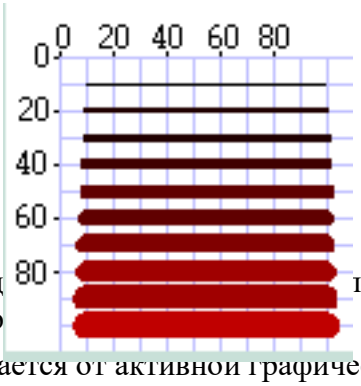
```
Program Линия;Begin
Line( 0,  0, 90, 90 );
Pen( 2,  0,  0, 255 );
Line( 30, 30, 90, 30 );
Pen( 6, 255,  0,  0 );
Line( 10, 10, 10, 90)
end.
```



Установить цвет и толщину линий : Pen (n, r, g, b)

где **n** задает толщину линии в пикселях, **r, g, b** - задают части красного, зеленого исинего в цвете линий. Установки действуют на команды **LineTo, Line, Rectangle, Ellipse, Point**. По умолчанию установлен черный цвет линий **(0,0,0)** и единичная толщина.

```
Program Цвет_Толщина;Var
i: integer;Begin
For i:= 1 to 10 doBegin
Pen( i, i*20, 0, 0 );
Line( 10, i*10, 100, i*10 )
end
end.
```

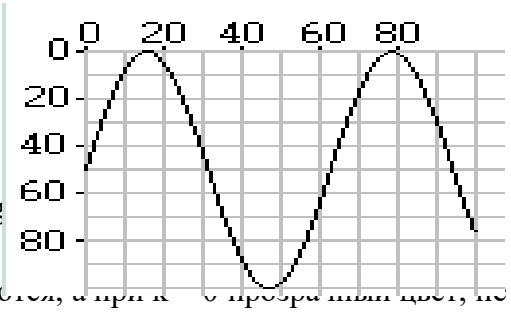


Когда следующая линия начинается из конца пред ыдущей, например, при построении ло-манных линий или графиков, удобно пользоваться пр оцедурой **LineTo (x, y**)

где (x, y) - координаты конца линии. Линия начинается от активной графической позиции. Линия будет проведена выбранным карандашом.

После ЛинияДо графическая позиция устанавливается в точку с координатами (x, y).

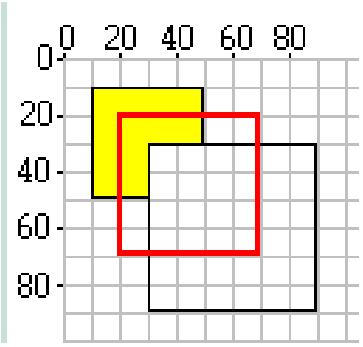
```
Program График;Var
i: integer;Begin
MoveTo( 0, 50 ); For i:=1 to 100 do
LineTo( i, round( 50-50*sin( i/10 ) ) );
end.
```



Установка цвета заполнения: Brush (k, r, g

где **r, g, b** - задают части красного, зеленого и синего в цвете **стиль** заполнения. Если k = 1 то замкнутые фигуры закрашиваются, а при k = 0 прозрачный цвет, не закрашиваются. Установки действуют на **Rectangle, Ellipse** и **Fill**.

```
Program Кисть;Begin
Brush( 1, 255, 255, 0 );
Rectangle( 10, 10, 50, 50 );
Brush( 1, 255, 255, 255 );
Rectangle( 30, 30, 90, 90 );
Pen( 2, 255, 0, 0 );
Brush( 0, 0, 0, 0 );
Rectangle( 20, 20, 70, 70 );end.
```

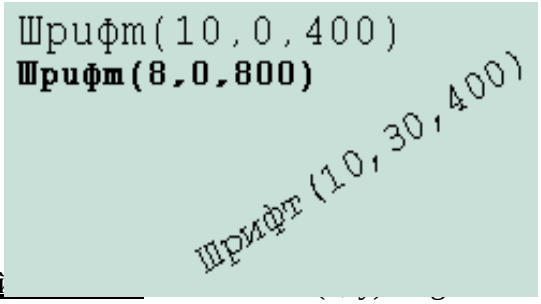


Установка Шрифта: Font (h, a, b)

где **h** - высота в пикселях, a -угол наклона в градусах, b - насыщенность шрифта. Насыщенность число из диапазона 0 - 1000. Обычный шрифт имеет насыщенность 400, а жир- ный - 600. Установки действуют на **Read** и **Write**.

По умолчанию установлен шрифт (8,0,400). Замечание: не все шрифты подвергаются наклону.

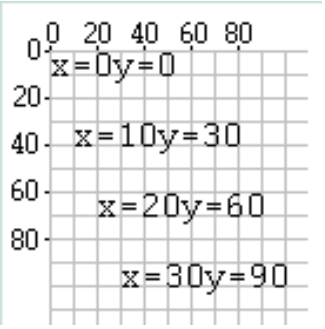
```
Program Шрифты;Begin
Font( 10, 0, 400 );
WriteLn( 'Шрифт(10,0,400)' );Font(8,0,800);
WriteLn( 'Шрифт(8,0,800)' );MoveTo( 80,
100 );
Font( 10,30,400 );
WriteLn( 'Шрифт(10,30,400)' );end.
```



Определение координат активной графической

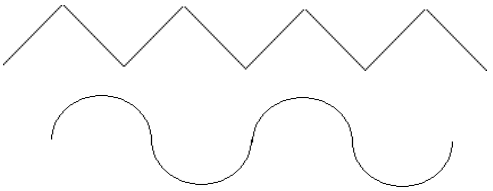
Координаты;

```
Var
i, x, y: integer;Begin
For i:=0 to 3 doBegin
                MoveTo( 10*i,
              30*i );
              Coordinates( x, y
        e     ); Write( 'x=', x,
end.    n     'y=', y )
        d
```



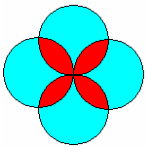
Задания к работе

1. Используя цикл, нарисуйте рисунок.

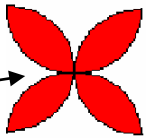


2. Используя цикл, нарисуйте рисунок.

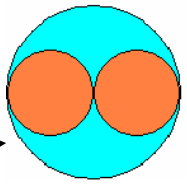
3. Создайте рисунок.



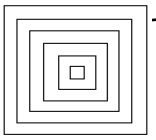
4. Создайте рисунок.

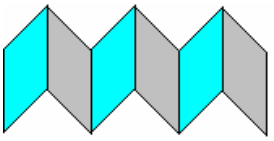


5. Создайте рисунок.



6. Создайте рисунок, используя цикл.



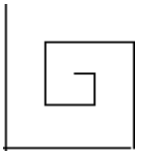


7. Создайте рисунок, используя цикл.

8. Нарисуй забор, используя цикл. Покрась его.

9. Нарисуй орнамент Меандр, используя цикл.
10. Создай проект, выводящий 100 точек внутри квадрата, используя случайные числа.
11. Создай проект, выводящий 100 разноцветных точек в левой половине окна.
12. Нарисуй серию горизонтальных линий, используя цикл.
13. Нарисуй серию вертикальных линий, используя цикл, запрашивая интервал.
14. Нарисуй треугольник, а внутри закрашенный квадрат другого цвета.
15. Нарисуй серию окружностей с общим центром, цвета окружностей меняются.
16. Нарисуй серию квадратов с общим нижним левым углом, цвета квадратов меняются.
17. Создай проект рисования объемного куба с разноцветными гранями.
18. Нарисуй сетку – серию вертикальных и горизонтальных линий.
19. Нарисуй сетку – серию взаимно перпендикулярных наклонных линий.
20. Нарисуй серию соприкасающихся окрашенных внутри окружностей, центры которых расположены на диагонали.
21. Нарисуй серию горизонтальных не соприкасающихся квадратов, закрашенных внутри в разные цвета.
22. Нарисуй шахматную доску.
23. Нарисуй квадрат и вписанный в него круг, используя различные цвета.
24. Нарисуй окружность и впиши в него треугольник, используя различные цвета.
25. Нарисуй окружность и впиши внутрь правильный многоугольник. Количество углов запрашивается.
26. Создай проект рисунка, радиус запрашивается.

27. Создай проект рисунка спирали. Каждая очередная сторона больше на запрашиваемую в начале величину.



28. Создай проект, в котором две вертикальные линии разного цвета двигаются навстречу друг другу.
29. Создай проект, управления движущимся кругом.

Практическая работа № 7

Тема: Использование процедур и функций в Паскале.

Цель работы:

- Формирование и отработка умений и навыков работы по использованию процедур и функций в среде программирования Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

В программировании часто возникают ситуации, когда одну и ту же группу операто- ров, реализующих определенную цель, требуется повторить без изменений в нескольких местах программы. Чтобы не писать заново несколько раз один и тот же фрагмент програм- мы в разных местах используют **подпрограммы**.

Подпрограмма – именованная, логически законченная группа операторов языка, ко- торую можно вызывать, по мере необходимости, для выполнения любое количество раз из различных мест программы.

В Паскале существуют два вида подпрограмм: **Процедуры** и **Функции.**

Каждая процедура или функция должна быть предварительно описана в разделе опи- сания процедур и функций. Для использования процедуры необходимо написать оператор вызова.

Описание процедуры состоит из заголовка процедуры и тела процедуры. Заголовок включает служебное слово **Procedure**, имя процедуры и заключенный в круглые скобки список параметров (переменных) с указанием их типа. Например:

```
Program ab;
Var  A, B, C :  integer;
Procedure  KUB(X : integer; Var Y, Z : integer);      ← Заголовок процедуры
Begin
Y : = X*X;                                            ← Тело процедуры
Z : =  X+X;
End;
Begin                                                 ← Начало программы
Write(‘ Укажи число –  ‘);  ReadLn(A);
KUB(A, B, C);                                         ← Вызов процедуры
WriteLn(‘ Квадрат =  ‘,B:6, ‘          Сумма = ‘,C:4);
Write(‘ Укажи число –  ‘);             ReadLn(A);
KUB(A, B, C);                          ← Вызов процедуры с фактическими параметрами.
WriteLn(‘ Квадрат =  ‘,B:6, ‘          Сумма =  ‘,C:4);
End.
```

При использовании процедур количество, типы данных и их порядок следования формальных и фактических параметров должны совпадать. Иногда раздел описания вход-ных и выходных параметров процедуры может отсутствовать. Например, в случае, если процедура просто должна выдавать какое-то сообщение на экран:

```
procedure Print;begin
writeln(' Процедура выполнена ! ');
WrieLn(‘  Привет  !  ‘);
end;
```

Отметим, что функций и процедур в программе может быть несколько.

Рассмотрим пример на использование процедуры при вычислении Факториала произвольного числа, например 6 (**6 ! = 1*2*3*4*5*6**)

```
Program ab;
Var  A, J, F : integer;
Procedure  Fakt(X : integer; Var Y : integer);        ← Заголовок процедуры
Var K : integer;
Begin
Y : = 1;                                              ← Тело процедуры
For  K :=2  to  X do
Y := Y*K;
End;
Begin                                                 ← Начало программы
For J :=1 to 4 doBegin
Write(‘ Укажи число –  ‘);          ReadLn(A);
Fakt(A, F);                                           ← Вызов процедуры
WriteLn(‘ Факториал ‘,A:2,’ = ‘,F:6);End;
End.
```

Функции в языке Паскаль:

Функции описываются в том же разделе программы, где и процедуры, и могут чере- доваться с ними. Для этого используется служебное слово **function** Все замечания относи- тельно входных и выходных параметров процедур верны и для функций. Различие между процедурой и функцией заключается в том, что Имя_функции принимает какое-либо значе-ние, тогда как Имя_процедуры - нет. **В теле функции всегда должен быть один опера- тор, присваивающий значение имени Функции**. Примеры описаний Функций:

```
Function Max(X, Y: Real) : Real; Function
Z(X : Real; Y : Real)) : Real;
```

Рассмотрим пример на использование Функции при вычислении Факториала числа.

```
Program ab;
Var  A, J, X, F : integer;
Function  Fakt(Var X : integer) : integer;            ← Заголовок Функции
Var K, Y : integer;
Begin
Y : = 1;                                              ← Тело Функции
For  K :=2  to  X do
Y := Y*K;
Fakt := Y;
End;
Begin                                                 ← Начало программы
For J :=1 to 4 doBegin
Write(‘ Укажи число –  ‘);          ReadLn(A);
WriteLn(‘ Факториал  ‘,A:2,’  =  ‘,Fakt(A):6);        ← Печать результата
End;
End.
```

Рассмотрим использование Функции при вычислении значения математического выра-жения.
Пример: дана функция $y(x)=5\cos(x)+3\sin(4x)-2x^2$.

Запишем для нее функцию на Паскале.

```
Program ab; Var J :
integer;
T : Real;
Function Y(Var X : Real) : Real;          ← Заголовок Функции
Begin
Y:=5*Cos(X)+3*Sin(4*X)-2*Sqr(X);End;
Begin                                     ← Начало программы
For J :=1 to 3 doBegin
Write(‘ Укажи X – ‘);          ReadLn(T);
WriteLn(‘ Y = ‘,Y(T):8:2);                ← Печать результата
End;
    End.
```

Задания к работе

4. Запиши программу нахождения наибольшего числа из трех указанных, используя процедуру или функцию.
5. Запиши программу вычисления по формуле, $a=\dfrac{|x-y|}{10x+|xy+x+y|}$ используя процедуру или функцию модуля числа.
6. Запиши программу нахождения **НОД** (наибольший общий делитель) для двух указанных чисел, используя функцию или процедуру.
7. Найди разность между максимальным и минимальным элементами массива, с использованием процедуры или функции.
8. Заполни массив случайными целыми положительными и отрицательными числами с использованием процедуры или функции.
9. Подсчитайте кол-во и сумму всех положительных элементов числового массива с использованием процедуры или функции.
10. Введите произвольное число. Заполните массив. Подсчитайте количество элементов, которые меньше указанного числа с использованием процедуры или функции.
11. Заполни массив целыми случайными числами. Подсчитай сумму всех четных чисел с использованием процедуры или функции.
12. Заполни массив целыми случайными числами. Подсчитай сумму всех простых чисел с использованием процедуры или функции.
13. Заполни массив целыми случайными числами, покажи его. Поменяй местами числа всех четных и нечетных ячеек с использованием процедуры или функции. Покажи всеэлементы после перестановки.
14. Заполни массив случайными дробными числами. Первую половину массива заполни дробной частью числа а вторую половину массива целой частью с использованием процедуры или функции.
15. Заполни массив целыми случайными числами. Покажи числа. Помести элементы в обратном порядке с использованием процедуры или функции.
16. Заполни массив случайными целыми числами. Сдвинь все элементы направо на 1 число а последний элемент перенеси в 1-ю клетку массива с использованием процедуры или функции.
17. Заполни массив случайными, не повторяющимися числами. Используй функцию или процедуру.

Практическая работа № 8

Тема: Обработка текстовых величин в Паскале.
Преобразование типов данных.

Цель работы:

- Формирование и отработка умений и навыков работы по использованию текстовых данных и преобразованию типов данных в среде программирования Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

Практически любая программа содержит обработку текста (строки) или преобразова- ние типов данных. Любой символ занимает в памяти один байт. Значение символьной пере- менной – это символ (символы), заключенные в апострофы. Значение (длина текста) ограни- чена 255 любыми символами, включая пробел, цифры и русский алфавит.

Символы, как и любые однотипные данные, могут быть объединены в массивы. Обра-ботка символьных массивов практически ничем не отличается от обработки чисел. Вот при- меры описания строковых (текстовых) переменных в разделе **Var**:

X : String; - Объявлен длинный текст **X**, до 255 любых символов.
Y : String[15]; - Объявлен текст **Y** длиной 15 любых символов.
Z : Char; - Объявлен текст **Z** длиной в один любой символ.
A : Array[1..20] of Char; - Объявлен массив **A** из 20 элементов из символов.
B : Array[1..30] of String; - Объявлен массив **B** из 30 элементов из слов и предложенийДля строк определены операции **конкатенации (+) и сравнение (=, <>, <, >, <=, >=)**

```
Begin
A := ‘Турбо‘;
B := ‘Паскаль’;
C := ‘Windows’;
D := A+ ‘ ‘ + B + ‘ ‘ + C;
Write(D);
If A = B Then
Write(C);
End;
```

Сравнение строк возможно, т.к. каждый символ в таблице кодов имеет свой номер и символы расположены по возрастанию.

Сравнение строк производится слева направо до первого несовпадающего символа: длиннее считается та строка, в которой первый несовпадающий символ имеет больший номер в таблице кодов: ‘ABD’ > ‘ABC’

Обращение к элементу строки аналогично обращению к элементу массива: A[4], B[J].

Процедуры и функции обработки строк:

- Функция **POS(S, ST)** – функция поиска. Она определяет, с какой позиции строка **S** входит в строку **ST**. Если вхождение имеет место, то результатом работы функции бу-дет номер символа (число) в строке **ST**, с которого начинается строка **S**.

Если вхождения нет, то результат – ноль. Например:

```
ST := ‘р.п. Майна’S := ‘Май’
Write(Pos(S,ST));
```

- Процедура **DELETE(ST, P, N)** удаляет из строки **ST** (тип String) **N** (тип Integer) символов начиная с позиции **P** (тип Integer). Например:

```
ST := ‘Майна’
Delete(ST,4,2);
```

- Процедура **INSERT(S, ST, N)** вставляет в строку **S** (тип string) подстроку **ST** (тип string) начиная с позиции **N** (тип integer).
- Функция **LENGTH(S)** возвращает длину (количество символов) текста **S**
- Процедура **STR(X, S)** преобразует числовое значение **X** (тип Real) в текст **S** (Возможно задание формата числа - **X**)
- Процедура **VAL(S, X, err)** преобразует текст **S** (тип String) в число **X** (тип Real)

Разберем два конкретных примера на обработку текста:

Пример 1:

Ввести слово. Определить, является ли оно «перевертышем» (слова, читаемые одинаково в обоих направлениях называются «Палиндромами») Примеры палиндромов - «ПоП», «потоп», «БоБ».

```
   Var                  text :
string;
   K, J, N : integer;
   Flag : Boolean;
   Begin
   Write(‘ Укажи текст – ‘); ReadLn(text);N :=
Length(text);
   For  J := 1 to N  div 2 do
   If text[J] <> text[N-J+1] ThenFlag :=
False;
   If  Flag  Then
   Write(‘ Перевертыш !‘);
                    E
                l          Write(‘ Нет ! ‘);
        End.        s
                    e
```

Пример 2:

Ввести предложение. Группа символов между пробелами считается словом. Подсчитайте количество слов в тексте. Считаем пробелы.

```
   Var                  text :
string;
   K, J : integer;
   Begin
   Write(‘ Укажи текст – ‘); ReadLn(text);For J :=
1 to  Length(text) do
   If text[J] = ‘ ‘  Then K := K+1;
   Write(‘ В  Вашем тексте  слов -  ‘, K:2);
   End.
```

Пример 3:

Ввести текст из цифр. Встречается ли в тексте цифра «7» ? Сложи все цифры текста. Например ввели текст - 2376. Ответ - «Да». Сумма = 18.

```
   Var            text : string;
   J,  zifra,  summa, kod  : integer;
   a : char; NaL_7 : Boolean;
   begin
   writeln(' Введите Текст из цифр'); read(text);for  J:=1
to  length(text)  do
   begin
   a:=text[J]; if a='7' then nal_7 := true; val(a,zifra,kod);
summa := summa + zifra;
   end;
   if  nal_7 = true  then
   writeln(' Текст  содержит 7')
   else
   writeln(' Текст  не  содержит 7');
   writeln(' Сумма всех цифр = ',Summa:10);end.
```

Задания к работе

1. Создай проект, подсчитывающий количество запятых в указанном тексте.
2. Подсчитайте сколько раз встречается в вашем тексте указанный символ.
3. Подсчитайте сколько раз встречается в вашем тексте указанное слово.
4. В произвольно введенном тексте вместо одного пробела вставьте везде два пробела.
5. Среди четырехзначных целых чисел найди те, в которых все цифры различны.
6. Сколько раз встречается цифра 1 в числах от 1 до 1000 включительно.
7. Подсчитайте количество цифр в числе. Ввели - 345623. Кол-во цифр - 6.
8. Переверните введенное число. Ввели 12345, получили 54321.
9. Выведи все натуральные 2-е числа, в которых нет одинаковых цифр.
10. Вычисли сумму только четных чисел введенного числа. Ввели 19352, сумма = 14.
11. Введите число и уберите ближние повторы. Ввели 23444552 получили 23452.
12. Преобразуйте введенное число в максимально число. Ввели 528, получили 852.
13. Преобразуйте введенное число в минимальное. Ввели 528, получили 258.
14. Вводите число и удвойте каждое число. Ввели 233451 получили 223333445511.
15. Замените все цифры в произвольном тексте на символ «+».
16. Замените текст «Мин» на «Мах» в произвольном тексте.
17. Подсчитайте общее количество букв «а» и «б» в вашем тексте.
18. Замените в произвольном тексте все буквы «а» на «б», а все буквы «б» на «а».
19. Подсчитайте сколько раз в произвольном тексте встречается слово «Майна».
20. Выясните, есть ли в предложении все буквы, входящие в указанное слово.
21. Удали слова, содержащие хотя бы одну цифру.
22. Поменяй местами самое длинное и короткое слова.
23. Из заданного текста удалите те его части, которые заключены в скобки.
24. Найдите самое длинное и самое короткое слова в произвольном тексте.
25. В заданном тексте подсчитай наибольшее количество подряд идущих пробелов.
26. Введи произвольное слово. Удали из него все повторяющиеся символы.
27. Из заданного текста выбери символы, встречающиеся только один раз.
28. Введите предложение и удалите из него все лишние пробелы. Покажите результат.
29. Подсчитайте сколько раз встречается каждая буква вашего текста.
30. Введите текст. На какую букву начинается больше всего слов этого текста.
31. Введите слово. Получи новое слово, где буквы расположены в алфавитном порядке.

Практическая работа № 9

Тема: Работа с файлами в языке АВС Паскаль.

Цель работы:

- Формирование и отработка умений и навыков по работе с файлами в среде Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

Существуют три вида файлов – типизированные, бестиповые, и текстовые, позво- ляющие считывать большие объемы данных непосредственно с диска, не вводя их с клавиа-туры.

Текстовые файлы состоят из любых символов. Они организуются по строкам, каж- дая из которых заканчивается символом «конец строки». Конец файла обозначается симво- лом «конец файла». При записи информации в текстовый файл, просмотреть который можнолюбым текстовым редактором, все данные преобразуются в символьный тип и хранятся в этом виде.

Типизированные файлы состоят из компонентов одного типа данных, число кото- рых заранее не определено и может быть любым. Они заканчиваются символом «конец фай-ла», хранятся в двоичных кодах и не просматриваются с текстовыми редакторами.

Бестиповые файлы записываются и считываются блоками определенного размера. Вних могут хранится данные любого вида и структуры.

Текстовый файл описывается так:

Var F : text;

Для начала работы с файлом любого типа необходимо связать файловую переменнуюв программе с файлом на диске. Для этого используют процедуру **assign(f,d),** где **f** - имя файловой переменной, а **d** - полное имя файла на диске (файл должен находится в теку- щем каталоге при условии, что к нему специально не указывается путь). Например:

```
Var f : text;
Begin
Assign(f, ‘d:\turboPascal\tmp\abc.dat’);
```

Рассмотрим несколько примеров работы с файлами:Пример 1:

Записать N действительных чисел в файл

```
Program ab;Var  f
: Text;
  A :  real;
  J, N : Integer;
  Begin
  Assign(f, ’d:\abc.txt’);          - Связываем файловую переменную с файлом
  Rewrite(f);                       -  Открываем пустой файл для записи.
  Read(N);                          - Указываем кол-во элементов в файле
  For J := 1 to N doBegin
  Write(‘ a = ‘); Read(a);          - В цикле вводим  очередной элемент
  Write(f, a);                      -  и записываем  его  в  файл abc.Dat
  End;
  Close (f);                        - Закрываем файл. Делаем это обязательно !
  End.
```

Пример 2:

В корне диска D: находится файл ABC.DAT дробных чисел. Нужно вывести его содержимое и подсчитать кол-во элементов

```
Program ab;Var  f
: Text;
  A :  real;
  N : Integer;
  Begin
  Assign(f, 'd:\abc.txt');          - Связываем файловую переменную с файлом
                  Reset(f);         - Открываем файл для чтения.
                  Read(N);          - Указываем кол-во элементов в файле
                  N :=0
                  While Not eof(f) do   - Работаем внутри цикла пока не достигнут
                  Begin                   конец открытого файла.
                  ReadLn(f, a);     - Считываем очередной элемент из файла
                  N := N + 1;
                  WriteLn(N,'-й  = ',a:10:4);   - Выводим на экран элемент a с номером N
                End;
                  WriteLn;
                  WriteLn(' В файле элементов -', N); - Просто
                выводим NClose (f);       - Закрываем файл.
        End.
```

При программировании бывают весьма полезными следующие функции Паскаля для работы с файлами:

Функция **Filesize(f)** - возвращает число реальных элементов открытого файла **f**.

Процедура **Seek(f, n)** – устанавливает указатель в открытом файле **f** на компонент сномером n (Нумерация с нуля). Затем значение может быть считано.

Процедура **Seek(f, n)** не работает с текстовыми файлами.

Для текстовых файлов можно пользоваться процедурой **Append(f).** Применяется для открытия уже существующих текстовых файлов для будущего добавления в них тек- стовой информации в конец файла. В текстовых файлах операторы **Write** записывают ин-формацию целого, дробного, символьного типа преобразовав их предварительно в тип **String**. Оператор **WriteLn** в добавок добавляет символ конца строки.

Задания к работе

Во всех заданиях составить две программы. Первая формирует файл исходных данных.Вторая – считывает данные из созданного файла, выполняет необходимые действия.

1. Запиши список своей группы по информатике, с указанием пола.

Запиши вторую программу по подсчету количества лиц мужского и женского пола.

2. Запиши программу записи в файл 20 случайных целых чисел, покажи числа.

Создай вторую программу по удвоению всех четных записанных чисел с показом.

3. Запиши программу записи в файл 30 дробных случайных чисел.

Создай программу, записывающую округленные числа в файл с показом чисел.

4. Запиши программу записи в файл 40 дробных случайных чисел.

Создай программу, записывающую целую часть чисел в файл с показом чисел.

5. Запиши программу записи в файл 40 целых случайных чисел.

Создай программу нахождения среднего арифметического всех чисел. Замени все чис-ла меньше среднего арифметического самим средним арифметическим.

6. Запиши список своей группы по информатике.

Запиши вторую программу по подсчету количества имени «Оля».

Практическая работа № 10

Тема: Работа с записями в языке ABC Паскаль.

Цель работы:

- Формирование и отработка умений и навыков по обработке таблиц с помощью записей в среде программирования Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

Запись – это структурированный тип данных, состоящий из определенного числа компонентов, которые называются **полями записи**. Они могут иметь разный тип. Описаниепеременной типа запись начинается ключевым словом **Record**, за которым следует список полей с указанием их типов. Заканчивается описание служебным словом **End**.

Примеры:

```
Var
Name :  Record
X : string;
Y : Real;
Z :  Integer;
End;
Var
SKL : ABC
```

или

```
Type
Abc  =  Record;
A, B  : String;
C, D  : Real;
E : array[1..30] of integer;
End;
```

Для того чтобы обратится к полю записи, необходимо указать имя переменной и че-рез точку – имя поля. Например: **SKL .C :=SKL . D + 122;**

Записи часто используют при работе с таблицами, где каждая запись – это одна строкатаблицы. Следовательно, для обработки всей таблицы, необходимо использовать массивы записей. Например:

```
Type
   Ved =
Record; Fio
: String[25];
data :
End;Var   Integer;
Zarplata :
Real;

A  :  array[1..30]  of  Ved;
```

Для обращения к некоторому полю например к полю Fio в пятой строке таблицы достаточно указать соответствующий элемент массива - **A[5] . Fio**

Рассмотрим пример :

О каждом учащемся есть информация:

- ✓ Фамилия, Имя
- ✓ Класс
- ✓ Оценка за четверть по Информатике

Заполним таблицу, отсортируем по алфавиту, выведем ее на экран, подсчитаем средний балл по предмету.

```
Program  ab;
Type          TabL = Record
Fio : String[24]; Klass : String[4];
Info : Integer;
End;
Var           I, J : Integer; A:TabL;
Sr_BaLL : Real;
Mas : array[1..3] of TabL;
Begin
Sr_BaLL := 0;
For i:=1 to 3 doBegin
Write(i,'  Уч-ся -  '); ReadLn(Mas[i].Fio);
Write(' Класс  ');ReadLn(Mas[i].Klass);
Write(' Оценка ');ReadLn(Mas[i].Info);Sr_BaLL :=
Sr_Ball + Mas[i].Info;
End;
```

```
   { Упорядочение массива записей в алфавитном порядке фамилий }For i :=1 to 2
do
   For  J :=1 to 2  do
   If Mas[J].Fio > Mas[J+1].Fio ThenBegin
   A := Mas[J];
   Mas[J] := Mas[J+1];Mas[J+1] := A;
   End;
   { Вывод  результатов }
   WriteLn(' ---------------------------------------------------- ');
   WriteLn('  Уч-ся:        Класс: Оценка:'); WriteLn('  -- ');
   For i := 1 to 3 dobegin
   Write(Mas[i].Fio:15);
   Write(Mas[i].Klass:9);
   WriteLn(Mas[i].Info:9);
   End;
   WriteLn(' ---------------------------------------------------- ');
   Write(' Средний балл = ',Sr_BaLL/3:6:2);End.
```

Задания к работе

1. Сформируй записи пропусков уроков учащимися разных классов. Отсортируй записи по болезням или кол-ву пропущенных занятий. Подсчитай среднее кол-во пропусков по школе и по классу.

2. О сотрудниках некоторой фирмы известна информация: Фамилия, год рождения, стаж работы, оклад. Сформируйте таблицу записей со всей информацией, причем добавьте 10 % к окладу, работающим более 10 лет и 15 % к окладу работающим более 20 лет.

3. Известна информация о клиентах банка: Фамилия, первоначальная сумма вклада, срок вклада в месяцах. За каждый месяц банк начисляет 1 % от суммы в начале месяца. Подсчитай сумму вклада клиентов, подлежащую выплате каждому клиенту вконце срока. Сформируй таблицу записей.

4. Известен список проданных товаров с ценами и количеством. Сформируйте таблицу записей стоимости всех проданных товаров. Отсортируйте товары по стоимости.

Практическая работа № 11

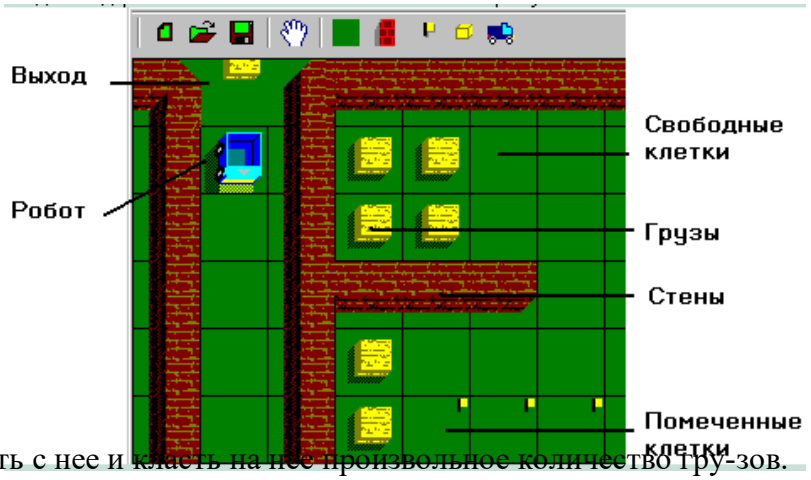
Тема: АВС Паскаль АЛГО. Управление Роботом.

Цель работы:

- Формирование и отработка умений и навыков по управлению и моделированию поведения Робота в среде АВС Паскаль.
- Развивать познавательный интерес к предмету, к изучаемой теме.

Краткая теория

Робот-грузчик изображен в виде маленького грузовика. Он двигается только по вертикали или по горизонтали. Шаг перемещения - одна клетка склада. Вид робота и типыклеточек показаны на рисунке.



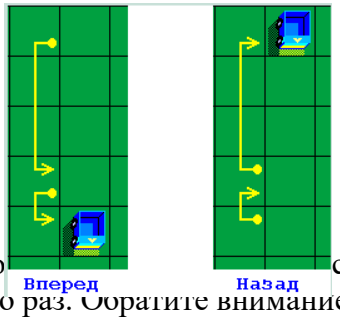
Робот может перевозить один груз, положить груз на свободнуюклеточку, а также пометить клеточку флажком или стереть метку. Исключением является клеточка выход. Робот не может заходить на эту клеточку, но может брать с нее и класть на нее произвольное количество гру-зов.

Перемещения робота : РоботВперед(n) и РоботНазад(n)

Где **n** задает количество клеток перемещения робота. Если количество клеточек не указано, то робот перемещается на одну клеточку. При этом скобки употреблять нельзя. Движение осуществляется в том направлении, в котором ориентирован робот в момент об-ращения к процедуре.

Если на пути встречается стена или груз, то выдается сообщение об ошибке.

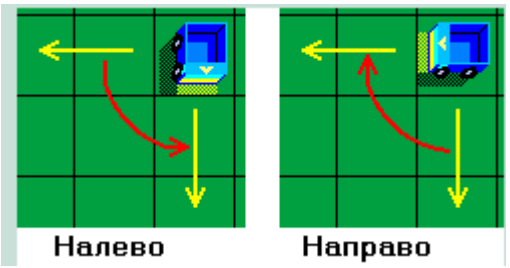
Program Движение;Begin
RobotForw(3) ;RobotForw; RobotBack; RobotBack(3)
end.



Повороты робота : РоботНалево(n)

Если аргумент не указан, то робот по сли указан целый пара- метр n, то робот повторяет поворот заданное число раз. Обратите внимание, что направлениеповорота задается относительно робота, а не относительно наблюдателя.

Program Поворот;Begin
RobotLeft; ReadLn;
RobotRight
end.



Проверка клеток:

Робот может двигаться только по свободным клеточкам. Для проверки соседних клеток используют логические функции СвободноВпереди, СвободноПозади, СвободноСле-ва, СвободноСправа:

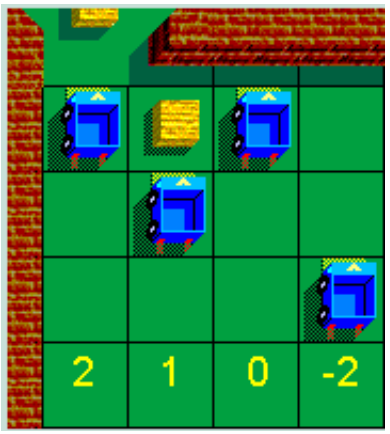
Функции не имеют параметров и возвращают истину (true), если клеточка свободная или помечена, и значение (false), если - груз или стена. Направление опрашивания задается относительно робота, а не относительно наблюдателя.


```
Program Проверка;Begin
While FreeForw do
RobotForw
end.
```

В результате выполнения программы робот двигает

Локатор - детальное опрашивания обстановки перед роботом.

В зависимости от ситуации в направлении движения робота функция возвращает одноиз таких целых значений:



0	- перед роботом стена;
+1	- перед роботом груз;
+2	- перед роботом выход;
-N	- перед роботом N свободных клеточек;

```
Program Локатор;
Begin
Write( 'Перед роботом ' );Case InFront of
0: Write( 'стена' );
1: Write( 'груз' );
2: Write( 'выход' );
else
Write( - InFront, ' свободных клеточек' );end;
end.
```

Пометить Клетку

Робот может пометить клетку, на которой находится, флажком. Для этого надо обра-титься к процедуре Обозначить: Если клеточка уже обозначена, то команда игнорируется роботом. Если клетка, в которой робот, обозначена, то функция возвращает истину (true), если не обозначена - значение (false). Убирают метку процедурой СтеретьМетку:

```
Program Пометить;Var
i: integer;
Begin
For i:=0 to 10 doBegin
RobotForw;
If i mod 2 = 0  then  Select
end;
RobotLeft( 2 );
ReadLn;
For i:=0 to 10 doBegin
If Selected then Unselect;RobotForw
endend.
```

Взять груз: Take

По команде **Взять** робот берет груз, который находится в клетке непосредственноперед ним. Если груза нет или робот уже загружен, то выдается сообщение об ошибке.

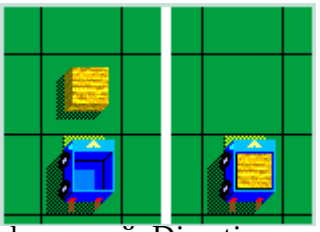
Положить груз: Put

Робот кладет груз на клетку непосредственно перед собой. Если робот пустой или клетка перед ним занята, то выдается сообщение об ошибке. Если клетка, на которую кладутгруз, помечена, то метка удаляется.

Робот может взять груз если он не загружен, и положить груз, если загружен.

Для проверки, пустой ли робот, обращаются к функции **Empty**

Если робот пустой, то возвращается значение (true), если нет значение (false)



```
Program Проверка;Begin
RobotLeft( 2 );Take;
RobotLeft( 2 );
If FreeForw and not Empty then Put;end.
```

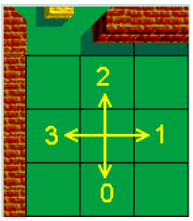
Направление Робота (Ориентация): Direction

Можно проверить ориентацию робота относительно наблюдателя, функцией Direction

В зависимости от ориентации робота функция возвращает одно из таких значений:0 - робот направлен вниз (на юг);

1 - робот направлен вправо (восточнее);2 - робот направлен вверх (на север);

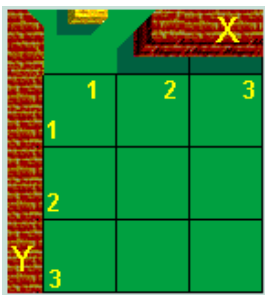
3 - робот направлен влево (на мероприятие);



Текущие координаты робота: RobotCoord

Процедура возвращает переменной **X** значение номера столбца, а **У** - номера строки,на пересечении которых находится робот. Нумерация строк ведется сверху вниз, а нумера- ция столбцов - слева направо.

```
Program Координаты;var
x, y: integer;
Begin
RobotCoord(x,y);
WriteLn( 'Робот на пересечении ', x, '-го столбца и ', y, '-й строки' );Write( 'и напрвлен на ' );
Case Direction of 0:  Write( 'юг' );
1:  Write( 'восток' );
2:  Write( 'север' );
3:  Write( 'запад' );
end
end.
```



Задания к работе

1. Создайте свой лабиринт, используя встроенные инструменты робота.

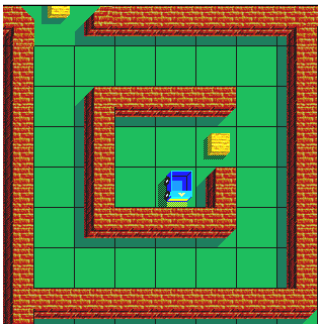
2. Выберите ручное управление роботом и выполните 5-6 различных команд.

3. Напишите программу обхода роботом всего поля по периметру.

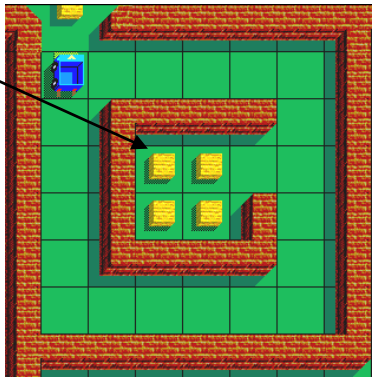
4. Положите грузы во все клетки первого столбца.

5. Положите грузы через клетку первой строки.

6. Есть некий лабиринт. Верните Робота в начало.



7. Заполни склад грузами по схеме.



8. На поле расположены произвольно несколько грузов, соберите их.

9. На поле расположены произвольно несколько грузов, соберите их, а клетки пометьте.

10. На поле расположены произвольно несколько грузов. Напишите программу подсчета роботом их количества.

11. На поле есть одинаковое количество грузов и помеченных клеток. Перенеси грузы в помеченные клетки.

12. Положите грузы ниже главной диагонали.

13. Положите грузы выше главной диагонали.

14. Положите грузы в шахматном порядке.

Критерии оценки внеаудиторной (самостоятельной) работы

Процент результативности	Балл (оценка)	Критерии оценивания
90-100%	5	— глубокое изучение учебного материала, литературы и нормативных актов по вопросу; — правильность формулировок, точность определения понятий; — последовательность изложения материала; — обоснованность и аргументированность выводов; — правильность ответов на дополнительные вопросы; — своевременность выполнения задания.
70-89%	4	— полнота и правильность изложения материала; — незначительные нарушения последовательности изложения; — неточности в определении понятий; — обоснованность выводов приводимыми примерами; — правильность ответов на дополнительные вопросы; — своевременность выполнения задания.
50-69%	3	— знание и понимание основных положений учебного материала; — наличие ошибок при изложении материала; — непоследовательность изложения материала; — наличие ошибок в определении понятий, неискажающих их смысл; — несвоевременность выполнения задания.
0-49%	2	— незнание, невыполнение или неправильное выполнение большей части учебного материала; — ошибки в формулировке определений, искажающие их смысл; — беспорядочное и неуверенное изложение материала; — отсутствие ответов на дополнительные вопросы; — отсутствие выводов и неспособность их сформулировать; — невыполнение задания.