

Учебный комплект уроков по теме: «Программирование на языке Pascal ABC»

Учитель информатики
Буркина Л.И.

Для того чтобы выпускники, ориентированные на получение высшего образования в области информационных и компьютерных технологий были более подготовлены к сдаче ЕГЭ и соответствовали требованиям высшей школы, необходимо уделить большее внимание в преподавании школьного курса «Информатика и ИКТ» разделу «Алгоритмизация и программирование». Данный комплект уроков разработан для освоения обучающихся основной и средней школы методам программирования на языке Pascal ABC.

Урок №1. Тема: «Языки программирования».

Цели урока:

- познакомить учащихся с историей развития языков программирования;
- помочь учащимся усвоить основные понятия о языках программирования;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (1 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (22 мин)
- IV. Практическая часть. (10 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (3 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На этом уроке мы познакомимся с краткой историей развития ЯП. Узнаем о том, какие из ЯП являются наиболее популярными в наше время. Рассмотрим, что такое ЯП и что включает в себя алфавит ЯП.

III. Теоретическая часть.

ВВЕДЕНИЕ.

Каждый из нас, так или иначе, по крайней мере, на бытовом уровне занимался программированием. Самый простой пример такого программирования - поставить будильник на нужное время, чтобы вовремя проснуться. Есть ещё мобильные телефоны, автоматические стиральные машины, микроволновые печи, регуляторы на холодильниках, таймеры на телевизорах и т.д.

Любая машина, в том числе и компьютер, в своей работе выполняет те команды, которые специально составлены человеком. Чем сложнее техника, тем большее количество операций она может выполнять. На данный момент компьютер является одним из самых сложных технических устройств. Он может решать самые сложные задачи. Однако же, для того, чтобы компьютер мог решать такие задачи, человек должен написать для него специальную программу на одном из языков программирования.

Языки программирования (ЯП) для ЭВМ начали разрабатывать с середины 50-х годов XX в. В настоящее время в мире имеется более 2500 различных языков программирования и их разновидностей. Для решения большинства задач можно использовать любой из них.

Всё множество языков программирования можно разделить на две группы: языки низкого уровня и языки высокого уровня. Языки низкого уровня (типа ассемблеров) понятны лишь компьютеру и узкому кругу программистов высокой квалификации, поэтому их и называют «машинными языками». Написание программ на этих языках - процесс сложный и трудоёмкий. Большинство программистов пользуются для составления программ языками высокого уровня. Языки высокого уровня более понятны человеку и играют роль посредника между человеком и компьютером, позволяя общаться с компьютером более привычным для человека способом. Для таких языков нужен «переводчик» на машинный язык - транслятор, но процесс программирования упрощается. Наиболее известными высокоуровневыми языками программирования являются языки Бейсик, Си, JAVA, HTML и Паскаль. Каждый из них имеет множество версий. Ява и HTML применяются в основном в Интернете. Бейсик считается одним из самых простых ЯП. Си считается высокопрофессиональным языком, соответственно гораздо сложнее.

По эффективности и простоте программирования, Паскаль занимает промежуточное положение между Бейсиком и Си. Поэтому он наиболее подходит для освоения его учащимися в школе. Существует много разновидностей языка Pascal (Object Pascal, Turbo Pascal, Delphi, PascalABC и др.). Синтаксис во всех версиях Pascal практически одинаков.

Для обучения школьников наиболее подходит свободно распространяемая версия **Pascal ABC**. Система **Pascal ABC** предназначена для обучения

программированию на языке Паскаль и ориентирована на школьников и студентов младших курсов.

По мнению автора программы **Pascal ABC** С.С.Михалковича первоначальное обучение программированию должно проходить в достаточно простых и дружественных средах, в то же время эти среды должны быть близки к стандартным по возможностям языка программирования и иметь достаточно богатые и современные библиотеки стандартных подпрограмм.

Язык Паскаль признан многими российскими преподавателями, как один из лучших, именно для начального обучения. Система **Pascal ABC** основана на языке Delphi Pascal и призвана осуществить постепенный переход от простейших программ к модульному, объектно-ориентированному, событийному и компонентному программированию. В свободно распространяемую версию **Pascal ABC & Programming Taskbook Mini Edition** входит мини-версия (автор М.Э.Абрамян) электронного задачника (200 задач) и комплект задач для исполнителей Робот и Чертежник. Программа, написанная на языке PascalABC, после небольшой корректировки, будет работать и в других версиях Pascal. Освоив один из простых ЯП, можно переходить к более сложным. Изучив приёмы программирования на Pascal, вы сможете без особых усилий перейти на другие языки программирования, и дальнейшее изучение профессиональных языков программирования будет значительно легче.

Раздел 1. Языки программирования.

Язык программирования – это формальная знаковая система (набор команд), которую понимает компьютер. Язык программирования определяет набор лексических, синтаксических и семантических правил, используемых при написании алгоритмов компьютерных программ. Алфавит языка - множество символов, используемых в этом ЯП.

Язык программирования предназначен для того, чтобы компьютер понимал инструкции по выполнению той или иной программы, написанной на соответствующем ЯП. Языки программирования - искусственные языки. Они отличаются от естественных языков тем, что предназначены для передачи команд и данных от человека к компьютеру, в то время как естественные языки используются лишь для общения людей между собой. В ЯП имеется ограниченное число "ключевых слов", значение которых понятно транслятору, и строгие правила записи команд .

Перед тем как писать программу на языке высокого уровня, программист должен составить *алгоритм* решения задачи, то есть пошаговый план действий, который нужно выполнить для решения этой задачи. Поэтому языки, требующие предварительного составления алгоритма, часто называют *алгоритмическими языками*. Для написания текста программы можно использовать обычный текстовый редактор (например, Блокнот), а затем с помощью компилятора перевести её в машинный код, т.е. получить исполняемую программу. Но проще и удобнее пользоваться специально разработанными системами программирования.

В начале 70-х годов XX века швейцарский учёный Никлаус Вирт разработал язык программ-мирования, и дал ему название Паскаль, в честь знаменитого французского математика XVII века, изобретателя первой счётной машины Блеза Паскаля. С помощью ЯП Паскаль можно разрабатывать программы самого разного назначения. Синтаксис этого языка интуитивно понятен даже тем, кто только начинает осваивать азы программирования.

Язык Паскаль удобен для начального обучения программированию, не только потому, что учит как правильно написать программу, но и тому, как правильно разрабатывать методы решения задач программирования.

Раздел 2. Элементы языка. Алфавит ЯП Pascal.

Алфавитом языка называют совокупность всех допустимых символов, которые можно использовать в этом языке.

Алфавит языка Паскаль включает в себя следующие символы:

- прописные и строчные буквы латинского алфавита от **A** до **z**, а также символ подчеркивания (**_**), который тоже считается буквой. Прописные и строчные буквы взаимозаменяемы (рАвНоЗнАчНы);
- арабские цифры **0 1 2 3 4 5 6 7 8 9**;
- специальные одиночные знаки: **+ - * / = < > . , : ; ^ \$ # @**;
- специальные парные знаки: **[] () { } ;**;
- составные знаки : **< = > = < > .. (* *) (..)**.

В состав ЯП Паскаль входят также буквы русского алфавита, но они могут использоваться только при вводе и выводе данных строкового типа (т.е. при вводе и выводе текста заключённого в апострофы (**' '**), или в комментариях к программе).

Вопросы:

- Для чего нужны ЯП?
- Назовите наиболее известные ЯП.
- Что такое ЯП низкого уровня и высокого уровня. В чём их отличия?
- Что такое алфавит ЯП?
- Назовите состав ЯП Паскаль.

III. Практическая часть.

Сегодня на практической части мы научимся запускать систему Pascal ABC, открывать проект и сохранять свои работы.

IV. Д/з

Подготовить сообщения о истории развития ЯП.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На этом уроке мы познакомились с краткой историей развития ЯП. Узнали о том, какие из ЯП являются наиболее популярными в наше время.

Рассмотрели, что такое ЯП и что включает в себя алфавит ЯП.

Урок №2. Тема: «Структура программы. Идентификаторы и зарезервированные слова. Константы».

Цели урока:

- помочь учащимся принцип построения программ на ЯП Pascal ABC, усвоить структуру программы, дать понятия о служебных словах ЯП Pascal ABC, необходимые для начала работы с программой.
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, навыков работы с мышью и клавиатурой, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (1 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (10 мин)
- IV. Практическая часть. (20 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке вы познакомились с краткой историей, и с основными

понятиями о ЯП. Для понимания того, что необходимо делать, чтобы научиться писать программы на языке Pascal ABC, необходимо рассмотреть структуру программы и основные термины ЯП Pascal ABC.

III. Теоретическая часть.

Структура программы.

Программа на языке Паскаль состоит из "заголовка" и "тела" программы, называемого блоком. В "заголовке" программы даётся имя и перечисляются её параметры (если это необходимо). В последних версиях языка, заголовок не является обязательной частью программы. Далее следует раздел подключения модулей, за которым следует список имен модулей, перечисляемых через запятую.

После него идёт описательная часть программы (блок описаний), состоящая из пяти разделов, причем любой из них, кроме описания переменных, может отсутствовать. В блоке описаний разделы обычно следуют в таком порядке:

1. *описание меток;*
2. *определение констант;*
3. *определение типов;*
4. *описание переменных;*
5. *описание процедур и функций.*

Далее следует блок `begin ... end` (*раздел операторов*), внутри которого находятся операторы, отделяемые один от другого символом "точка с запятой".

```
Program имя программы;           {заголовок программы}
uses                             {раздел подключения модулей}
Label ...; {раздел описания меток}
Const ...; {раздел описания констант}
Type ...; {раздел определения типов}
Var ...; {раздел описания переменных}
Function ...; Procedure ...; {раздел описания функций и процедур}
BEGIN ... } {раздел операторов}
END.
```

блок описаний

Раздел подключения модулей (`uses`) и раздел описаний могут отсутствовать. Например:

```

program MyFirstProgram;
var a,b: integer; c: real;
begin
    readln(a,b);
    c := a/b;
    writeln(c);
end.

```

В Паскале блок операторов начинается со служебного слова `begin`. Конструкция `begin ... end` называется операторными скобками. Операторы, находящиеся внутри конструкции `begin ... end`, считаются одним составным оператором. Каждый блок завершает зарезервированное слово `End`. Вся программа завершается словом `End` с точкой.

Идентификаторы и зарезервированные слова.

Имена переменных, констант, меток, типов, модулей, процедур и функций, используемых в программе, называются - идентификаторами. Имена задаёт разработчик программы. На имена (идентификаторы) накладываются некоторые ограничения, такие как невозможность использования ключевых (служебных) слов, например `integer` или `var`. Идентификатор должен начинаться с буквы и может содержать буквы латинского алфавита, цифры и знаки подчеркивания. Символ подчеркивания "_" также считается буквой. Желательно выбирать *мнемонические* имена, т.е. несущие смысловую нагрузку, как, например, `result`, `summa`, `cena`. Использование осмысленных имен предпочтительнее, так как это делает программу более простой для понимания.

Например: `a1`, `b_2`, `k123`, `_d` - идентификаторы,
`1a`, `и2` , `@ru` – не идентификаторы.

Служебные слова являются зарезервированными и не могут быть использованы в качестве идентификаторов. Список всех служебных слов языка **Pascal ABC** приведен ниже:

<code>and</code>	<code>array</code>	<code>as</code>	<code>begin</code>
<code>break</code>	<code>case</code>	<code>class</code>	<code>const</code>
<code>constructor</code>	<code>continue</code>	<code>destructor</code>	<code>div</code>
<code>do</code>	<code>downto</code>	<code>else</code>	<code>end</code>
<code>exit</code>	<code>external</code>	<code>externalsync</code>	<code>file</code>
<code>finalization</code>	<code>for</code>	<code>forward</code>	<code>function</code>
<code>if</code>	<code>in</code>	<code>inherited</code>	<code>initialization</code>
<code>is</code>	<code>mod</code>	<code>not</code>	<code>of</code>
<code>or</code>	<code>private</code>	<code>procedure</code>	<code>program</code>
<code>property</code>	<code>protected</code>	<code>public</code>	<code>record</code>
<code>repeat</code>	<code>set</code>	<code>shl</code>	<code>with</code>

shr	sizeof	string	xor
then	to	type	unit
until	uses	var	while

Константы.

Постоянной (константой) называется величина, значение которой не изменяется в процессе исполнения алгоритма.

Раздел описания именованных констант начинается со служебного слова `const`, после которого следуют строки вида:

имя константы = значение;

или

имя константы : тип = значение;

Например:

const

Pi = 3.14;

Number = 10;

Name = 'Victor';

Cifra = ['0'..'9'];

Mass: **array** [1..5] **of integer** = (1,3,5,7,9);

Spisok: **record** name: **string**; age: **integer** **end** = (name: 'Иван'; age: 17);

Компьютер "знает", чему равны константы **e** и **π**.

Вопросы:

1. Из чего состоит программа на языке Паскаль?
2. Опишите структуру программы?
3. Что называется операторными скобками
4. Что такое служебные слова ЯП?
5. Что такое идентификаторы? В чём их отличия?

III. Практическая часть.

На прошлом уроке вы пробовали открыть программу Pascal ABC. Сегодня на практической части мы научимся составлять структурную схему программы на ЯП Pascal ABC .

Откройте программу Pascal ABC и в открывшемся окне напишите текст:

Program Zdorovye; {заголовок программы}

var t:real; (*раздел описания переменных*)

begin //начало раздела операторов

writeln('Измерь свою температуру и введи её значение!');

```
read(t);      {занесение в ячейку t её значение}
              {вычисление значения s}
if t>36.6 then {вычисление значения s}
writeln('Ты заболел, надо идти к врачу!') {вывод на
экран текста}
else
writeln('Ты здоров, можешь идти в школу!'); //вывод
на экран текста
end.
```

Проверьте работу программы.

IV. Д/з.

Знать структурную схему программы на ЯП Pascal ABC. Выучить описание всех разделов программы и определения терминов: идентификатор, служебное слово, константа.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы разобрались с понятиями: структура программы, идентификатор, служебное слово, константа.

Урок №3. Тема: «Ввод и вывод данных».

Цели урока:

- помочь учащимся усвоить правила ввода и вывода данных в языке Pascal ABC, продолжить изучение основ программирования;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости;
- развитие познавательных интересов, навыков работы с мышью и клавиатурой, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (2 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы познакомились со структурой программы ЯП Pascal ABC. Сегодня мы узнаем как вводятся и выводятся данные в программу на ЯП Pascal ABC.

III. Теоретическая часть.

Ввод и вывод данных.

Рассмотрим схему, по которой происходит исполнение программы на языке Pascal ABC:

1. **ВВОД** исходных данных с клавиатуры, из файла или с носителя информации;
2. **ОБРАБОТКА** данных с помощью операторов языка Pascal ABC;
3. **ВЫВОД** результатов обработки на экран, принтер, в файл или на носитель информации.

Для того, чтобы ввести или вывести данные, необходимо выполнить определённые команды (*процедуры*). Процедура, которая в режиме диалога с клавиатуры присваивает значение для переменной величины, называется *процедурой ввода*.

В языке Pascal эта команда выглядит следующим образом:

```
read(список переменных) ;
```

Например,

```

Var
X : real; Y: integer; Z : char;
Begin
read(X, Y, Z)
End.

```

При запуске программы на исполнение, встретив строку `read(X, Y, Z)`, программа останавливает свою работу, и в нижней части окна программы появляется поле для ввода данных. В него необходимо ввести с клавиатуры значения переменных `(X, Y, Z)`, в соответствии с их типом. Если вводимое значение не будет соответствовать объявленному типу переменной, то программа завершится сообщением об ошибке. Ввод каждого значения переменной завершается нажатием клавиши `Enter`.

Процедура, которая выводит содержимое переменных на экран, называется процедурой вывода на экран.

В языке Pascal эта команда выглядит следующим образом:

```
write (список переменных);
```

Например:

```

Var X : real; Y: integer; Z : char;
Begin
    read(X, Y, Z)
    write (X, Y, Z)
End.

```

Строка `(write)` интерпретируется: “Вывести на экран через запятую, значения переменных соответственно, как `X, Y, Z`”.

В ЯП Паскаль для ввода и вывода данных, также имеются процедуры `readln` и `writeln`, отличающиеся от описанных выше, только тем, что при своём выполнении осуществляют перевод курсора на следующую строку.

Формат вывода данных можно задать в скобках, указав после двоеточия `(:)` количество символов выделяемых для выводимого значения переменных.

Например, если `x, y` – целые переменные (типа `integer`), то при выполнении программы:

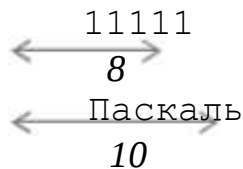
```

var x,y: integer;
begin
    x:=-9999; y:=11111;
    writeln(x:6, 'Здравствуй!':12);
    writeln(y:8);
    writeln('Паскаль':10);
end.

```

будет выведен следующий текст:

6	12	6	- количество
выделяемых символов;			
		(ширина поля вывода)	



Для вещественных чисел можно также использовать формат с двумя указателями `writeln(x:m:n);`, где `x` - значение переменной типа `real`, `m` - ширина всего поля вывода, а `n` - количество знаков после десятичной точки (имеет приоритет). Например: `writeln(-11.789:10:3);` // `_____ -11.789` `writeln(-11.789:10:5);` // `_____ -11. 8900` `writeln(-11.789:10:2);` // `_____ -11.79` `writeln(-11.789:10:0);` // `_____ -12` `writeln(-11.789:10:7);` // `_____ -11.7890000`
`writeln((0,151):10:1);` // `_____ 0.2`

(символом `_` изображены пробелы).

Разработка программ.

Примерная схема разработки программ:

1. Анализ условия задачи, чёткая формулировка вопроса задачи.
2. Выделение исходных данных, определение формы выходных данных.
3. Детальное изучение задачи.
4. Определение необходимых переменных и их типа.
5. Составление алгоритма.
6. Написание программы.
7. Ввод программы и запуск её на исполнение.
8. Отладка программы (поиск ошибок).
9. Анализ работы и доработка программы.

Вопросы:

- Что такое данные?
- Какие команды существуют для ввода данных?
- Какие существуют команды для вывода данных?
- Как называются эти команды?
- Опишите примерную схему разработки программ.

III. Практическая часть.

На прошлом уроке мы познакомились со структурой программы ЯП Pascal ABC. Сегодня мы практически научимся вводить и выводить данные.

IV. Д/з.

Знать, как объявлять переменные, как вводятся и выводятся данные, примерную схему разработки программ.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы узнали, каким образом производится ввод и вывод данных в программах.

Урок №4.

Тема: «Переменные. Типы переменных. Типы данных. Комментарии к программе».

Цели урока:

- помочь учащимся усвоить такие понятия как - переменные, типы переменных, типы данных, комментарии к программе, необходимые для осваивания основ программирования.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

I. Орг. момент. (2 мин)

II. Актуализация знаний. (5 мин)

III. Теоретическая часть. (12 мин)

IV. Практическая часть. (17 мин)

V. Д/з (2 мин)

VI. Вопросы учеников. (5 мин)

VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы научились объявлять переменные, и написали простую программу.

Получили понятие о константах, идентификаторах и служебных словах. Для понимания того, как работает программа, необходимо познакомиться с понятием - переменные.

III. Теоретическая часть. Переменные.

Любая программа обрабатывает некоторые данные. Данные могут быть представлены только как *константы* или *переменные*, причём имеющие

собственные идентификаторы (имена). Как уже говорилось, рекомендуется давать имена отражающие смысл *константы* или *переменной*.

Переменные – одно из главных понятий в программировании. Для того, чтобы разбираться в программировании необходимо иметь чёткое представление о том, что такое переменная, как и где она хранится, и что с ней происходит в процессе выполнения программы.

Переменной называется величина, значение которой меняется в процессе исполнения алгоритма.

Переменные – это некоторые данные, обрабатываемые в программе и имеющие имя.

Как вы знаете, данные хранятся и обрабатываются в памяти компьютера. При работе программы – в оперативной памяти, а при выключении сохраняются в постоянной памяти. При создании программ используются разные типы данных, т.е. переменные различного типа. Это могут быть числа, символы, текст, логические переменные, процедуры, даты и др., которые, в свою очередь, могут подразделяться на определённые виды. Например, числовые данные могут быть целого типа, с дробной частью и т.д. В зависимости от типа данных, программа после объявления переменных, выделяет определённое количество ячеек в памяти, для хранения этих переменных. То есть, этим ячейкам присваиваются имена переменных и в этих ячейках, затем хранятся значения этих переменных.

Храниться они могут или до конца выполнения программы, или до тех пор, пока переменной не присвоится другое значение. Имя переменной остается неизменным до конца программы, а значение самой переменной может меняться. В ЯП Паскаль обязательное объявление переменных, с описанием их имён, защищает программы от ошибок и повышает их надежность.

Раздел описания переменных начинается со служебного слова `var`, после которого следуют элементы описания. Переменные могут описываться как в начале программы, так и непосредственно внутри любого блока `begin ... end`. Внутриблочные описания переменных имеют тот же вид, что и в разделе описаний.

```
begin
  var a1,a2,a3: integer;
end.
```

Кроме того, переменные-параметры цикла могут описываться в заголовке оператора `for`.

Имена переменных в списке перечисляются через запятую. Например:

```
var
  a,b: integer;
  c,d: real;
  m,n: byte;
```

```
s,s1: string;  
ch,ch1: char;  
f: boolean;
```

Типы переменных.

В зависимости от версии языка программирования Pascal типы переменных могут немного различаться. В программах написанных на ЯП **PascalABC** используются данные следующих типов:

```
integer (целый)  
byte (байтовый)  
real (вещественный)  
complex (комплексный)  
string (строковый)  
char (символьный)  
boolean (логический)  
тип "массив"  
процедурный  
файловый  
классовый и некоторые другие.
```

Типы в **PascalABC** подразделяются на простые, строковые, структурированные, типы указателей, процедурные и классовые.

К **простым** относятся числовые (целые и вещественные) типы, логический, символьный, перечислимый и диапазонный тип. Перечислимый тип данных задается перечислением всех значений, которые может принимать переменная данного типа. При описании отдельные значения указываются через запятую, а весь список заключается в круглые скобки. Например:

```
Var Mesyac: (May, June, July, August );
```

Структурированные типы (т.е. имеющие какую-то структуру), образуются массивами, записями, множествами и файлами.

Все простые типы, кроме вещественного, являются **порядковыми**. Значения только этих типов могут быть *индексами* переменных и массивов и параметрами цикла **for**.

Индекс – это порядковый номер в последовательности. Обычно обозначается символом *i*. Нумерация начинается с единицы. Например: В последовательности A, B, C...Z, индексы символов соответственно 1, 2, 3...26.

Если индекс *i* выходит за пределы длины строки, то при выполнении программы появляется сообщение об ошибке.

Типы данных:

1. Порядковые целые.

Имя типа	значение	Размер, байт	тип
byte	0..255	1	числовой беззнаковый целый
word	0..65535	2	числовой беззнаковый целый
integer	-2147483648.. -2147483647	4	числовой знаковый целый
char	все символы в кодировке ASCII	1	символьный

К порядковым относятся также *перечислимый* и *интервальный* тип.

Перечислимый тип определяется упорядоченным набором идентификаторов. Например:

```
type
    Season = (Winter, Spring, Summer, Autumn) ;
    DayOfWeek = (Mon, Tue, Wed, Thi, Thr, Sat, Sun) ;
```

Значения перечислимого типа занимают 4 байта.

Интервальный тип представляет собой подмножество значений целого, символьного или перечислимого типа и описывается в виде $a..b$, где a - нижняя, b - верхняя граница интервального типа:

```
var
    a: 0..10;
    c: 'a'..'n';
    d: Mon..Sat;
```

Тип, на основе которого строится интервальный тип, называется базовым для этого интервального типа. Значения интервального типа занимают 4 байта.

2. Вещественный тип.

Тип `real` (числовой вещественный). Значения вещественного типа занимают 8 байт, содержат 15-16 значащих цифр и находятся в диапазоне $-1.8 \cdot 10^{308} .. 1.8 \cdot 10^{308}$. Константы типа `real` можно записывать как в форме с плавающей точкой, так и в экспоненциальной форме:

```
1.7;
0.013;
```

```
2.5e3 (2500);  
1.4e-1 (0.14).
```

3. Логический тип.

Тип `boolean` (логический). Переменные и константы логического типа занимают 1 байт и могут иметь одно из двух значений, задаваемых константами `True` (истина - 1) и `False` (ложь - 0).

4. Строковый тип.

Тип `string` (строковый). Применяется при использовании текстовых данных в программе, состоит из набора последовательно расположенных символов `char`. По умолчанию под переменную типа `string` отводится 256 байт, при этом в нулевом байте хранится длина строки. Т.е. строки состоят, не более чем, из 255 символов. Пример описания:

```
var s: string;
```

Можно явно указать количество символов для переменной в `[ ]`. Например:

```
var s: string[50];
```

В данном случае под переменную выделяется 50 символов.

Допускается при записи выражений строкового типа применять данные, символьного типа (`char`). В этом случае эти данные воспринимаются как `string`. К отдельным символам строкового типа можно обратиться по номеру этого символа в строке, аналогично индексу в массивах (см. раздел 4 Массивы).

2.8. Комментарии к программе.

В программе может присутствовать текст написанный разработчиком для пояснения к программе. Этот текст называется “комментарием к программе”. Даже опытные программисты считают необходимым присутствие комментариев в программах.

Комментарии заключаются между *скобками* `{ ... }`, `(*...*)` или пишутся после символов `//` (слеш). Комментарии не воспринимаются компьютером и не обрабатываются программой:

```
{ это - комментарий }  
(* это - тоже комментарий *)  
// это - тоже комментарий
```

Вопросы:

- Что такое переменная?
- Какие типы переменных вы знаете?
- Какие типы данных вы знаете?

- В чём разница между простыми и структурированными типами?
- Что называется индексом переменной?

III. Практическая часть.

На прошлых уроках вы разобрались со структурной схемой программы на ЯП Pascal ABC. Сегодня на практической части мы научимся объявлять переменные, и напишем простую программу.

```
Program StObed;
var a,b,c,s:real;
begin
  writeln('стоимость первого блюда');
  writeln('стоимость второго блюда');
  writeln('стоимость третьего блюда');
  readln(a,b,c);
  s:=a+b+c;
  write('стоимость      обеда      ',s,'
  руб. '); readln;
end.
```

Исправьте программу так, чтобы в результате выполнения программы на экран выводилось ровно 4 строки вместе с ответом.

IV. Д/з

Выучить определение переменной, типы переменных и как записываются комментарии к программе.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы узнали, что такое переменные, типы переменных и как записываются комментарии к программе.

Урок №5. Тема: «Действия над данными. Выражения, операнды и операции».

Цели урока:

- помочь учащимся усвоить какие действия возможно выполнять над данными в ЯП Pascal ABC, а также с понятиями «Выражения, операнды и операции»;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости;

- развитие познавательных интересов, навыков работы с мышью и клавиатурой, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (1 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (10 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы узнали, что такое переменные, типы переменных и как записываются комментарии к программе. Сегодня нам предстоит узнать какие действия над данными можно выполнять при разработке программ на ЯП Pascal ABC.

III. Теоретическая часть.

Раздел 3. Действия над данными. Работа с программой.

Раздел 3.1. Выражения, операнды и операции.

В алгоритмах программ участвуют *выражения*.

Простыми выражениями являются переменные и константы.

Сложные выражения строятся из более простых с использованием операций, скобок, вызовов функций, процедур, индексов и приведений типов. Данные, к которым применяются операции, называются *операндами*. *Операциями* в ЯП называются действия над данными (*операндами*).

В Pascal ABC имеются следующие операции: @, not, ^, *, /, div, mod, and, shl, shr, +, -, or, xor, =, >, <, <>, <=, >=.

Арифметические операции.

К арифметическим относятся бинарные (применяемые к двум операндам) операции + - * / для вещественных и целых чисел, бинарные операции div и mod для целых чисел и унарные (применяемые к одному операнду) операции + и - для вещественных и целых чисел.

Выражение, имеющее числовой тип, называется *арифметическим*. Тип арифметического выражения определяется по следующему правилу: если все операнды целые и в выражении отсутствует операция деления /, то выражение имеет тип `integer`, в противном случае выражение имеет тип `real`. Например, если `b` имеет тип `byte`, а `c` имеет тип `word`, то выражения `b+c`, `b-c` или `b*c` имеют тип `integer`, а выражение `b/c` - тип `real`.

Операции DIV и MOD.

В Pascal есть операции целочисленного деления и нахождения остатка от деления. Применяются они для данных типа `integer`. При выполнении целочисленного деления (операция DIV) остаток от деления отбрасывается. Например, $12 \text{ div } 4 = 3$; $19 \text{ div } 5 = 3$; $136 \text{ div } 10 = 13$, $27 \text{ div } 10 = 2$. С помощью операции MOD можно найти остаток от деления одного целого числа на другое. Например: $12 \text{ mod } 3 = 0$; $19 \text{ mod } 5 = 4$; $136 \text{ mod } 10 = 6$, $27 \text{ mod } 10 = 7$.

При работе с различными типами данных в ЯП Паскаль имеются стандартные процедуры и функции, представленные ниже.

1. Математические процедуры и функции.

Имя и параметры	Процедура или функция	Типы параметров	Тип возвращаемого значения	Действие
<code>Abs (x)</code>	функция	<code>x - integer, real</code>	совпадает с типом параметра	возвращает абсолютное значение (модуль) <code>x</code>
<code>Sqr (x)</code>	функция	<code>x - integer, real</code>	совпадает с типом параметра	возвращает квадрат <code>x</code>
<code>Sqrt (x)</code>	функция	<code>x - real,</code>	совпадает с типом параметра	возвращает квадратный корень из <code>x</code>
<code>Sin (x)</code>	функция	<code>x - real,</code>	совпадает с типом параметра	возвращает синус <code>x</code>
<code>Cos (x)</code>	функция	<code>x - real,</code>	совпадает с типом параметра	возвращает косинус <code>x</code>
<code>Arctan (x)</code>	функция	<code>x - real,</code>	совпадает с типом параметра	возвращает арктангенс <code>x</code>
<code>Power (x, y)</code>	функция	<code>x, y - real</code>	<code>real</code>	возвращает <code>x</code> в степени <code>y</code>
<code>Round (x)</code>	функция	<code>x - real</code>	<code>integer</code>	возвращает результат округления <code>x</code> до ближайшего целого
<code>Int (x)</code>	функция	<code>x - real</code>	<code>real</code>	возвращает целую часть <code>x</code>
<code>Frac (x)</code>	функция	<code>x - real</code>	<code>real</code>	возвращает дробную часть <code>x</code>
<code>Ord (x)</code>	функция	<code>x - порядкового типа</code>	<code>integer</code>	возвращает номер значения порядкового типа (символа)
<code>Chr (x)</code>	функция	<code>x - integer</code>	<code>char</code>	возвращает символ с кодом <code>x</code>
<code>Odd (x)</code>	функция	<code>x - integer</code>	<code>boolean</code>	возвращает <code>True</code> , если <code>x</code> - нечетное, и <code>False</code> в

				противном случае
Inc (x)	процедура	x - порядкового типа		Увеличивает x на 1
Dec (x)	процедура	x - порядкового типа		Уменьшает x на 1
Inc (x, n)	процедура	x - порядкового типа, n - целого типа		Увеличивает x на n
Dec (x, n)	процедура	x - порядкового типа, n - целого типа		Уменьшает x на n
Pred (x)	функция	x - порядкового типа	совпадает с типом параметра	возвращает предыдущее значение порядкового типа
Succ (x)	функция	x - порядкового типа	совпадает с типом параметра	возвращает следующее значение порядкового типа
Random (x)	функция	x - integer	integer	возвращает случайное целое в диапазоне от 0 до x-1
Random	функция		real	возвращает случайное вещественное в диапазоне [0..1)

Вопросы:

- что такое - выражения в ЯП Pascal ABC;
- что называют простыми выражениями в ЯП Pascal ABC ?;
- что называют сложными выражениями в ЯП Pascal ABC ?;
- что называют операндами в ЯП Pascal ABC ?;
- что называют операциями в ЯП Pascal ABC ?;

III. Практическая часть.

Сегодня нам предстоит узнать какие действия над данными можно выполнять при разработке программ на ЯП Pascal ABC.

Программа вычисления площади S прямоугольного треугольника, по двум катетам

a и b:

```

Program PloshadTreugl;
Var a,b: integer, S:real; //объявление переменных
begin
  writeln ('введите целые значения длины катетов a,b');
  {ВЫВОД пояснительного текста на экран}
  readln (a,b); //ввод данных (a и b) с клавиатуры
  S:=a*b/2; //вычисление площади треугольника
  writeln('площадь треугольника = ',S);
end.

```

Программа вычисления квадрата и квадратного корня числа n:

```

Program Kvad_Kor;
var kv,n:integer; kor:real; //объявляем переменные
begin
writeln('Введите число n');
readln(n);
kv:=sqr(n);
kor:=sqrt(n);
writeln('Квадрат числа ',n,' = ',kv,' Корень числа ',n,'
= ',kor);
end.

```

Программа вычисления длины отрезка по введённым координатам:

```

Program DlinaOtrezka;
var x1,y1,x2,y2,d:real; //объявляем переменные begin
// Вычисляем длину отрезка по теореме Пифагора
    writeln(' Введите координаты точек A(X1,Y1)
и B(X2,Y2) ');
    readln( x1,y1,x2,y2 );
    d:=sqrt(sqr(y2-y1)+sqr(x2-x1));
    writeln(' Длина отрезка |AB|=',d);
end.

```

IV. Д/з

Выучить какие все термины урока.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

Сегодня вы познакомились с тем, какие арифметические действия над данными числового типа, можно выполнять при разработке программ на ЯП Pascal ABC.

Урок №6. Тема: «Логические (булевы) операции. Операции отношения (сравнения)».

Цели урока:

- дать учащимся понятие о логических операциях, а также об операциях сравнения;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, навыков работы с мышью и клавиатурой, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (2 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке вы познакомились с тем, какие арифметические действия над данными числового типа, можно выполнять при разработке программ на ЯП Pascal ABC.

III. Теоретическая часть.

Логические (булевы) операции.

К *логическим* относятся бинарные операции **and**, **or** и **xor**, а также унарная операция **not**. Эти операции выполняются с использованием операндов типа **boolean** и возвращают значение типа **boolean**. Выражение, имеющее тип **boolean**, называется *логическим (булевым)*. Выражения в программах могут конструироваться с помощью булевых операций. Эти операции используют понятия алгебры логики, разработанной британским математиком Джорджем Булем. Операция **and** – конъюнкция (логическое умножение, пересечение, **&**, **^**, **"и"**).

Выражение

a & b дает значение **true** только в том случае, если **a** и **b** имеют значения **true**, в остальных случаях – **false**:

```
true and true = true
```

```
true & false = false
```

```
false ^ false = false
```

Операция **or** – дизъюнкция (логическое сложение, объединение, +, v, "или").
Выражение **a + b** дает значение **false** в том и только в том случае, если **a** и **b** имеют значения **false**, в остальных случаях – результат **true**:

```
true or true = true
```

```
true + false = true
```

```
false v false = false
```

Операция **not** – инверсия (логическое отрицание, $\neg$, $\bar{}$, операция "не").
Выражение **not a** имеет значение, противоположное значению **a**:

```
not true = false
```

```
 $\neg$  false = true
```

Эти операции полезны, если нужно проверить сложное условие.

Операции отношения (сравнения).

Операции отношения также являются логическими. Их можно использовать для проверки отношений между переменными: **a < b**, **c >= d**, **x = y** и т.д. Над данными типа **real**, **integer**, **char**, **string** можно выполнять следующие операции отношения (сравнения):

```
=      равно;  
<>     не равно;  
<      меньше;  
> больше;  
<=     меньше или равно,  
>=     больше или равно.
```

В операциях сравнения должны участвовать операнды одного типа. Исключение сделано только в отношении данных числовых типов **real** и **integer**,

которые могут сравниваться друг с другом. Результат применения операции отношения к любым операндам имеет тип `boolean`.

Вопросы:

- какие логические операции вы знаете?
- какие операции отношения вы знаете?
- операнды какого типа можно сравнивать?
- что является результатом сравнения логической операции?

III. Практическая часть.

Напишем программу, в которой сравниваются различные операнды.

Программа сравнения введенных чисел (полная форма условного оператора `if`):

```
Program SravChisel;  
var a,b: integer ; //объявляем переменные  
begin  
  writeln('Введите a и b');  
  read(a,b);  
  if a<b then  
    writeln(a, '<', b)  
  else  
    if a>b then  
      writeln(a, '>', b)  
    else writeln(a, '=', b);  
end.
```

Программа простой сортировки 3-х чисел:

```
Program ProstSort;  
var a,b,c: integer;  
begin  
  readln(a,b,c);  
  if (c>b) and (b>a) then write(a, ' ', b, ' ', c);  
  if (b>c) and (c>a) then write(a, ' ', c, ' ', b);  
  if (c>a) and (a>b) then write(b, ' ', a, ' ', c);  
  if (a>c) and (c>b) then write(b, ' ', c, ' ', a);  
  if (b>a) and (a>c) then write(c, ' ', a, ' ', b);  
  if (a>b) and (b>c) then write(c, ' ', b, ' ', a);  
end.
```

Программа определяющая, встречаются ли символы «, –» в строке.

```

Program Simvol'y;
var s:string;
i:integer;  f:boolean;
begin
  writeln('введите строку');
  readln(s);
  f:=false;
  for i:=1 to length(s)-1 do
    begin
      if copy(s,i,2)=',-' then f:=true;
    end;
    begin
      if f=true then writeln('символы ",- " встречаются')
      else

        writeln('символы ",- " не встречаются');
    end;
  end.

```

IV. Д/з

Знать какие операции относятся к логическому типу, а также все операции сравнения и правила их применения.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы узнали, какие операции относятся к логическому типу, а также правила применения операций сравнения в ЯП Pascal ABC.

Урок №7. Тема: «Строковые операции».

Цели урока:

- помочь учащимся получить сведения о том, какие операции можно производить со строковыми данными в ЯП Pascal ABC, а также каков приоритет выполнения операций;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости;
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (2 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлых уроках мы узнали, что какие операции относятся к логическому типу, а также правила применения операций сравнения в ЯП Pascal ABC.

III. Теоретическая часть.

Строковые операции.

Основной операцией, применяемой к строковым и символьным операндам, помимо операций отношения $<$, $>$, $<=$, $>=$, $=$, $<>$, является операция конкатенации (" $+$ ", слияния). Ее результат имеет строковый тип. Например, $'a'+'b'='ab'$ ('пол'+'нота'='полнота'). Поскольку строки могут содержать максимум 255 символов, и если сливаются строки суммарной длины больше 255 символов, то программа завершается сообщением об ошибке:

» Ошибка: произошло переполнение строки при выполнении операции "+" .

Стандартные процедуры и функции для работы со строками

Имя и параметры	Процедура или функция	Типы параметров	Тип возвращаемого значения	Действие
Length(s)	функция	s - string	integer	возвращает длину строки s

Copy(s, index, count)	функция	s - string , index и count - integer	string	возвращает подстроку строки s длины count, начиная с позиции index
Delete(s, index, count)	процед.	s - string , index и count - integer		удаляет в строке s count символов начиная с позиции index
Insert(pods, s, index)	процед.	s, pods - string , index - integer		вставляет подстроку pods в строку s с позиции index
SetLength(s, n)	процед.	s - string , n - integer		устанавливает длину строки s равной n
Str(x, s) Str(x:n, s) Str(x:n:m, s)	процед.	s - string , x - integer, real и n, m - integer		преобразует x к строковому представлению (во втором и третьем случаях согласно формату вывода, устанавливаемому n и m) и записывает результат в строку s
Val(s, v, code)	процед.	s - string , v - integer, real, и code - integer		преобразует строку s к числовому представлению и записывает результат в переменную v. Если преобразование возможно, то в переменной code возвращается 0, если невозможно, то в code возвращается ненулевое значение
Concat(s1,...,sn)	функция	s1,..., sn - string	string	возвращает строку, являющуюся результатом слияния строк s1,..., sn. Результат тот же, что у выражения s1+s2+...+sn
IntToStr(i)	функция	i - integer	string	преобразует целое число к строке
StrToInt(s)	функция	s - string	integer	преобразует строку в целое число. Если преобразование невозможно, то возникает ошибка времени выполнения

Приоритет выполнения операций.

При записи программ используются выражения разного типа (логические, строковые, алгебраические).

В алгебраических выражениях используют арифметические операции (сложение, умножение, вычитание, деление), функции Pascal, круглые скобки. Приоритет определяет порядок выполнения операций в выражении. Первыми выполняются операции, имеющие высший приоритет. Операции, имеющие одинаковый приоритет, выполняются слева направо. Изменить порядок операций можно круглыми скобками, тогда в первую очередь выполняются действия в них.

@, not, &

1 (наивысший)

<code>*, /, div, mod, and, shl,</code>	2
<code>shr</code>	
<code>+, -, or, xor</code>	3
<code>=, <>, <, >, <=, >=, in</code>	4 (низший)

Порядок действий при вычислении значения выражения:

- 1) вычисляются значения в скобках;
- 2) вычисляются значения функций;
- 3) выполняется унарные операции (унарный минус — смена знака);
- 4) выполняются операции умножения и деления (в том числе целочисленного деления и нахождения остатка от деления);
- 5) выполняются операции сложения и вычитания.

Вопросы:

- Что такое строковые данные?
- Назовите основные операции, которые выполняются со строками?
- С помощью, какой функции можно преобразовать числовые данные в строковые?
- Назовите операции в порядке приоритетности их выполнения.

III. Практическая часть.

На прошлых уроках мы узнали, какие операции относятся к логическому типу, а также правила применения операций сравнения в ЯП Pascal ABC. Сегодня на практической части мы напишем программы, в которых будут производиться действия над переменными строкового типа.

Программы работы со строками.

1. Программа конкатенации (сложения) строк:

```
Program Concatenacia;
var s,s1,s2: string; //объявляем переменные
begin

    Writeln('Введите слова');
    readln(s1,s2);
    s:=Concat(s1,s2);
    Writeln('Полученное слово =
',s); end.
```

2. Программа вычисления длины строки (количество символов в строке):

```
Program DlinaStroki;
var s: string; L:integer; //объявляем переменные
```

```

begin
  writeln('введите строку');
  readln(s);
  L:= Length(s);
  Writeln('Длина строки = ',L);
end.

```

3. Программа вырезания n символов из строки s , начиная с позиции i , с помощью функции $\text{Copy}(s, i, n)$:

```

Program VyrezSimvol;
  var s,s1: string; i,n:integer; //объявляем
переменные
  begin
    Writeln('введите слово');
    readln(s1);
    Writeln('введите с какого символа и сколько их
вырезать');
    Write('i = '); read(i);
    Write('n = '); read(n);
    s1:=Copy(s,i,n); // смотри раздел Стандартные
процедуры и функции
    Writeln('полученное слово = ',L);
  end.

```

4. Программа удаления из введённой строки s , первых n символов.

```

Program UдалenieSimv;
uses crt;
  var s:string;n:integer;
  begin
    writeln('Введите строку и количество удаляемых
символов');
    readln(s,n);
    delete(s,1,n);
    writeln('после удаления получилось - ',
s); end.

```

5. Программа поиска символа в строке:

```

Program PoiskSimvola;

  var s,si:string; i:integer; f:boolean; //объявляем
переменные

```

```

begin
  writeln('Введите строку');
  readln(s); //считываем строку
  writeln(' Введите символ');
  readln(si); //обозначаем искомый символ si
  f:=false; //изначально считаем, что символа в
строке нет
  for i:=1 to length(s) do
    begin //функцией copy проверяем
совпадают ли
      if copy(s,i,1)= si then f:=true //вырезаемые
символы с искомым
    end; //тогда логическая функция f примет
значение true
  writeln(f); //выводим результат
end.

```

6. Программа вычисления количества слов в строке;

```

Program
KolichSlov; var s:
string; n,k:
integer; begin
  writeln(введите строку ');
  readln(s);
  for n:=1 to length(s)
do begin
  if (s[n]=' ')
    then k:=k+1;
  end;
writeln('В строке',k+1,' слов'
); end.

```

7. Программа записи слова в обратном порядке (перевёртыш):

```

Program Perevorot;
var c,cl:string;
i:integer; //объявляем переменные

```

```

begin
  writeln('Введите слово');
  readln(c);
  for i:=length(c) downto 1 do
    c1:=c1+copy(c,i,1);
    writeln('перевёртыш = ',c1);
  end.

```

IV. Д/з

Выучить определения этого урока, записать в тетрадь процедуры и функции для работы со строками, запомнить приоритет выполнения операций над данными.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы получили сведения о том, какие операции можно производить со строковыми данными в ЯП Pascal ABC, а также каков приоритет выполнения операций. Так же мы написали программу, в которой выполнялись действия над переменными строкового типа.

Урок №8. Тема: «Операторы ЯП Pascal ABC».

Цели урока:

- помочь учащимся усвоить понятие об операторах в ЯП Pascal ABC;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (2 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлых уроках мы получили сведения о том, какие операции можно производить со строковыми данными в ЯП Pascal ABC, а также каков приоритет выполнения операций. Так же мы написали программу, в которой выполнялись действия над переменными строкового типа.

III. Теоретическая часть. Операторы ЯП Pascal ABC.

Оператор присваивания.

Как и во всех языках программирования, в **Pascal ABC** имеется *оператор присваивания*.

В некоторых языках, символом присваивания является знак равенства, однако для того, чтобы не путаться, оператор присваивания в Паскале выглядит так “:=”. Он служит для определения или переопределения значения переменной. В случае переопределения, новое значение переменной, записывается в ячейки с именем переопределяемой переменной, а прежнее значение стирается. Например:

```
a:=1000;  
b:=5;  
a:=a+b
```

Исходное значение переменной a – 1000, конечное 1005.

Условный оператор.

Ход выполнения программы может быть различным. Если в задаче, в зависимости от какого-либо условия, можно будет двигаться по разным путям, то алгоритм такой решения такой задачи называется ветвящимся. Как раз таким оператором и является *условный оператор* (if – если, else – иначе) .

Условный оператор позволяет проверить некоторое условие, и в зависимости от результатов проверки выполнить то или иное действие. Таким образом, условный оператор - это средство ветвления вычислительного процесса.

Условный оператор имеет *полную* и *краткую* формы.

Полная форма условного оператора выглядит следующим образом:

```
if условие then оператор1  
else оператор2
```

В качестве условия указывается какое-либо логическое выражение. Если условие оказывается истинным, то выполняется *оператор1*, в противном случае выполняется *оператор2*.

Краткая форма условного оператора имеет вид:

```
if условие then оператор
```

Если условие оказывается истинным, то выполняется **оператор**, в противном случае происходит переход к следующему оператору программы.

В случае конструкции вида

```
if условие1 then
    if условие2 then оператор1
    else оператор2
```

else всегда относится к ближайшему предыдущему оператору **if**, для которого ветка **else** еще не указана. Если в предыдущем примере требуется, чтобы **else** относилась к первому оператору **if**, то необходимо использовать составной оператор:

```
if условие1 then оператор
begin
    if условие2 then оператор1
end
else оператор2
```

Составной и пустой операторы

Составной оператор предназначен для объединения нескольких операторов в один. Он имеет вид:

```
begin
    операторы
end
```

Операторы отделяются один от другого символом ";".

Как уже говорилось выше, служебные слова **begin (начало)** и **end (конец)**, окаймляющие операторы, называются *операторными скобками*.

Например:

```
s:=0; p:=1;
for i:=1 to 10 do
begin //открывающая операторная скобка
    p:=p*i;

    s:=s+p
end //закрывающая операторная скобка
```

Так, как служебное слово **end** является закрывающей операторной скобкой, оно одновременно указывает и конец предыдущего оператора, поэтому ставить перед ним символ ";" необязательно. Перед **end** может ставиться ";". В этом

случае считается, что последним оператором перед **end** является *пустой* оператор, не выполняющий никаких действий. Пустой оператор не содержит никаких действий, просто в программу добавляется лишняя точка с запятой. В основном пустой оператор используется для передачи управления в конец составного оператора.

Оператор выбора.

Иногда требуется осуществить выбор более чем из двух условий. В этом случае применяется оператор множественного выбора, позволяющий выбрать из списка одно из условий.

Оператор выбора выполняет одно действие из нескольких в зависимости от значения некоторого выражения, называемого *переключателем*. Он имеет следующий вид:

```
case переключатель of
    список выбора 1: оператор1;
    ...
    список выбора N: операторN;
else оператор0
end;
```

Переключатель представляет собой выражение порядкового типа (целого, символьного, перечислимого или интервального), а списки выбора содержат константы совместимого типа. Как и в операторе **if**, ветвь **else** может отсутствовать.

Оператор **case** (**в случае**) работает следующим образом. В списке выбора проверяется на совпадение текущее значение переключателя и меток списка, и если найдено совпадение, то выполняется оператор, соответствующий данной строке списка. Если же значение переключателя не найдено ни в одной строке, то выполняется оператор **else**. В **случае**, если ветвь **else** отсутствует, оператор **case** не выполняет никаких действий, а управление передается внешнему оператору, следующему за конструкцией CASE.

Список выбора состоит либо из одной константы, либо из диапазона значений вида **a..b** (константа **a** должна быть меньше константы **b**). Можно также перечислить несколько констант или диапазонов через запятую:

```
case DayOfWeek of
    1..5: writeln('Рабочий день');
    6,7: writeln('Выходной');
end;
```

В списках выбора диапазоны меток не должны пересекаться. Например, в результате выполнения следующего фрагмента:

```

case i of
  1,5: write(1);
  4..9: write(2);
end;

```

программа завершится сообщением об ошибке:

" Ошибка: Пересечение диапазонов меток в операторе case".

Оператор безусловного перехода goto.

Оператор безусловного перехода **goto** используется для изменения естественной последовательности выполнения программы, и имеет следующую форму:

goto метка Содержимое строки можно интерпретировать как:
перейти к <метке>

Он передаёт выполнение программы к оператору, помеченному меткой.

Метка представляет собой идентификатор. Для совместимости со стандартным Паскалем в качестве метки можно использовать целое число без знака. Чтобы пометить оператор меткой, необходимо перед оператором указать метку с последующим двоеточием. Формат описания метки:

label1: оператор

Метки должны быть описаны в разделе меток с использованием служебного слова **label**:

```
label 1,2,3;
```

Например, в результате выполнения программы:

```

label 1,2; //метки
begin
  i:=6;
2: if i<0 then goto 1; //условие перехода к метке 1
   write(i);           //выводим значение i
   Dec(i);             //увеличиваем i на 1
   goto 2;             //переходим к метке
1:                       2 //метка 1
end.

```

будет выведено 654321.

Метка должна помечать оператор в том же блоке, в котором описана. Нельзя помечать одной меткой несколько операторов. Запрещается из внешней стороны цикла переходить на метку внутри цикла.

Оператор безусловного перехода (goto) надо использовать только в крайнем случае. По количеству применений этого оператора судят о квалификации программиста, хотя всё же, иногда его применение бывает оправдано. Например, при использовании меню в программе или при проверке данных при вводе их с клавиатуры. Применение циклов в программах позволяет обходиться без применения оператора goto вообще.

Вопросы:

- Что называют оператором в языках программирования?
- Перечислите операторы ЯП Pascal ABC?
- Что такое пустой оператор?
- Что такое составной оператор?
- В чём разница между оператором присваивания и знаком «равно»?

III. Практическая часть.

На прошлом уроке вы получили сведения о том, какие операции можно производить со строковыми данными в ЯП Pascal ABC, а также каков приоритет выполнения операций. Так же мы написали программу, в которой выполнялись действия над переменными строкового типа. Сегодня на практической части мы напишем программу, в которой будут участвовать различные операторы ЯП Pascal ABC.

1. Программа определения дня недели, по введённому номеру:

```
Program OperatorCase;  
var den: integer; //объявляем переменные  
begin  
  
    writeln('Введите номер дня недели (1..7):  
' ); readln(den);  
    case den of // Оператор выбора 1..5:  
        1..5: writeln('Рабочий день');  
        6,7: writeln('Выходной');  
        else writeln('Введите правильно номер  
            дня!'); end;  
end.
```

2. Программа сравнения введенных чисел (полная форма условного оператора if):

```
Program SravChisel;  
var a,b: integer ; //объявляем переменные  
begin  
  writeln('Введите a и b');  
  read(a,b);  
  if a<b then  
    writeln(a,'<',b)  
  else  
    if a>b then  
      writeln(a,'>',b)  
    else writeln(a,'=',b);  
end.
```

Программа, вычисляющая сезон года по введённому № месяца.

```
Program SezonGoda;  
var g:integer;  
begin  
  writeln('введите № месяца');  
  readln(g);  
  case g of  
    1,2,12: writeln('Зима');  
    3..5: writeln('Весна');  
    6..8: writeln('Лето');  
    9..11: writeln('Осень');  
  else  
    writeln('Нет такого месяца');  
  end;  
end.
```

Программа вычисляющая результат с помощью оператора безусловного перехода goto:

```
Program Sravnit;  
label m1,m2;  
var a,k,s :integer;  
begin  
  writeln('введите a и k');
```

```

    read(a,k);
    if a>=k then
        goto m1;
    if a<k then
        goto m2;
m1: s:=a-k;
    writeln('s = ',s);
m2: s:=a+k;
    writeln('s = ',s);
end.

```

IV. Д/з

Выучить действия всех операторов ЯП Pascal ABC.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы усвоили понятие об операторах в ЯП Pascal ABC, а также написали программу, в которой участвовали различные операторы.

Урок №9. Тема: «Циклы. Итерация.».

Цели урока:

- помочь учащимся усвоить понятие о циклах в ЯП Pascal ABC;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (2 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке вы усвоили понятие об операторах в ЯП Pascal ABC, а также написали программу, в которой участвовали различные операторы.

III. Теоретическая часть. Циклы. Итерация.

В большинстве программ требуется многократное повторение каких-то операций. Если известно количество этих повторений, то можно написать линейный алгоритм, обрабатывающий последовательно все эти операции. Но при этом программа будет неоправданно большой. В случае же, когда количество этих повторений неизвестно, то линейный алгоритм применить невозможно.

В таком случае необходимо применять циклы.

Цикл предназначен для повторения одной и той же последовательности команд неоднократно (до тех пор, пока не выполнится какое-либо условие).

Каждое повторение цикла называется – *итерация*.

Цикл с известным количеством повторений называют *циклом с параметром*. Для того, чтобы организовать циклы, используют специальные операторы. Тело цикла в **Pascal ABC** не может быть пустым. Это сделано для защиты от часто встречающейся у начинающих ошибки, ставить после **do** точку с запятой.

Цикл с известным количеством повторений FOR.

Оператор цикла **for** (для) имеет одну из двух форм:

for переменная:=начальное значение **to** конечное значение **do** оператор **или**

for переменная:=начальное значение **downto** конечное значение **do** оператор

для обеих форм содержимое строк можно интерпретировать как:

для <параметр цикла>:=<нач.знач> до <конеч.знач>
выполнять <оператор>

Содержимое текста от слова **for** до слова **do** включительно называется *заголовком цикла*, а оператор после **do** - *телом цикла*. Переменная после слова **for** называется *параметром цикла*. Параметр цикла, его начальное и конечное значения должны принадлежать к одному и тому же, порядковому (целому,

символьному, перечислимому или интервальному) типу данных, т.е. параметр выступает в качестве счётчика цикла.

Если в цикле изменяется простая переменная, то она является параметром цикла, если в цикле изменяется переменная с индексом, то параметром цикла является индекс этой переменной. В теле цикла параметр цикла не должен меняться.

Для цикла с ключевым словом **to** значение параметра цикла последовательно увеличивается на единицу при каждом прохождении цикла (повторе). Для цикла с ключевым словом **downto** значение параметра цикла при каждом повторе последовательно уменьшается на единицу.

Если для цикла **for ... to** начальное значение переменной цикла больше конечного значения или для цикла **for ... downto** начальное значение переменной цикла меньше конечного значения, то тело цикла не выполнится ни разу.

Изменение переменной-параметра цикла внутри цикла является логической ошибкой. Например, следующий фрагмент со вложенным оператором **for** является ошибочным:

```
for i:=1 to 10 do
  for i:=1 to 5 do
    write(i);
```

Цикл с неизвестным количеством повторений WHILE.

В цикле с неизвестным количеством повторений вместо счётчика цикла используется условие выхода из цикла.

Оператор цикла **while** имеет следующую форму:

```
while условие do оператор // эту строку можно
интерпретировать как:
пока выполняется <условие> выполнять <оператор>
```

Условие представляет собой выражение логического типа и может быть простым, с использованием операций отношения (<, >, <=, >=, =, <>), или составным с использованием логических операций ('и', 'или').

Оператор после **do** называется *телом цикла*.

Перед каждой итерацией цикла условие проверяется, и если оно истинно, то выполняется тело цикла, в противном случае происходит выход из цикла.

Если *условие* всегда оказывается истинным, то может произойти *заикливание*:

```
while 2>1 do
  write(1);
```

Для выхода из заикливающейся программы, можно использовать комбинацию клавиш **Ctrl-F2** или кнопку Stop.

Цикл WHILE является циклом с предусловием, т.е. сначала проверяется условие цикла и только в том случае, если оно истинно, выполняется тело цикла.

Цикл с неизвестным количеством повторений REPEAT.

Оператор цикла **repeat** имеет следующую форму:

```
repeat операторы until условие // эту строку можно
понимать как:
повторять < операторы > до тех пор, пока не выполнится
< условие >
```

В отличие от цикла **while**, условие вычисляется после очередной итерации цикла, и если оно истинно, то происходит выход из цикла. Таким образом, операторы, образующие тело цикла оператора **repeat**, выполняются по крайней мере один раз.

Если *условие* всегда оказывается ложным, то может произойти *заикливание*:

```
repeat
  write(1);
until 2=1;
```

Для выхода из заикливающейся программы, можно использовать комбинацию клавиш **Ctrl-F2** или кнопку Stop.

Цикл REPEAT является циклом с постусловием, т.е. сначала выполняется тело цикла, затем проверяется условие цикла и до тех пор пока оно не выполнится, (т.е. пока условие ложно) цикл будет повторяться. Из-за того, что в циклах этого типа выполнение цикла происходит раньше проверки условия, в них бывает трудно найти ошибки. Поэтому предпочтительнее использовать циклы с предусловием, тем более, что ими всегда можно заменить циклы с постусловием.

Вопросы:

- Что называют циклом? С постусловием? С предусловием?
- В каком случае оправдано применение циклов?
- Какие виды циклов вы знаете?
- Что такое заикливание и когда оно может возникнуть?

III. Практическая часть.

На прошлом уроке вы усвоили понятие об операторах в ЯП Pascal ABC, а также написали программу, в которой участвовали различные операторы. Сегодня мы рассмотрим на практике применение циклов в ЯП Pascal ABC.

1. Программа вычисления суммы чисел кратных 3, в диапазоне 0...100, с помощью цикла *for*:

```
Program Summ3;
var s,i:integer; //объявляем переменные
begin
  for i:=1 to 100 do //задаём диапазон от 0 до
100 if i mod 3 = 0 then //находим числа кратные
3 write(i:3);
  writeln();
  begin
    s:=0;
    for i:=1 to 100 do
      if i mod 3 = 0 then
        s:=s+i; //находим сумму чисел кратных 3
      end;
    writeln(' s = ',s); // выводим результат
  end.
```

2. Программа вычисления суммы нечетных чисел последовательности от 0 до 100, с помощью цикла *repeat*.

```
Program SumChet;
var i,s:integer; // i-индекс,s-накопитель суммы
begin i:=0; s:=0; //обнуление исходных значений i,s
  repeat //оператор цикла повтора
    i:=i+2; //нахождение чётных индексов.
    s:=s+i; {т.к. индексы последовательности
соответствуют значениям чисел, то к предыдущему значению
's' прибавляем значения чётных индексов}
  until i>99; //условие выхода из цикла
  writeln('s=',s); //вывод результата
end.
```

3. Напишем программу нахождения суммы нечетных чисел последовательности от 0 до 100, с помощью цикла *while*.

```
Program SumChet1;
var i,s:integer; // i-индекс,s-накопитель суммы
begin i:=0; s:=0; //обнуление исходных значений i,s
  while i<100 do //оператор цикла повтора с условием
выхода из него
```

```

begin
  i:=i+2; //нахождение чётных индексов.
  s:=s+i; //вычисление суммы
end;
writeln('s=',s); //вывод результата
end.

```

4. Напишем ту же программу нахождения суммы нечетных чисел последовательности от 0 до 100, с помощью цикла *for*.

```

Program SumChet3;
var i,s:integer;      // i-индекс,s-накопитель суммы
begin
  s:=0;
  for i:=0 to 100 do
    begin
      s:=s+i;      //вычисление суммы
      i:=i+1;
    end;
    writeln('s=',s); //вывод результата
  end.

```

IV. Д/з

Иметь понятие о циклах. Выучить формат организации всех видов циклов ЯП Pascal ABC.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы рассмотрели на практике применение циклов в ЯП Pascal ABC.

Урок №10. Тема: «Вложенные циклы».

Цели урока:

- помочь учащимся усвоить понятие о вложенных циклах в ЯП Pascal ABC;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

I. Орг. момент. (2 мин)

- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы рассмотрели на практике применение циклов в ЯП Pascal ABC.

III. Теоретическая часть.

Вложенные циклы.

Циклы могут быть *простыми* или *вложенными* (цикл в цикле).

Например:

```
Program VCicl;  
    var          i,j:integer;  
    begin  
        for i:=1 to 5 do  
            begin  
                writeln;  
                for j:=10 to 13 do  
                    write('i=',i, ' j=',j);  
            end;  
            readln;  
        end.
```

Для цикла for i:=1 to 5 do телом цикла является:

```
begin for j:=10 to 13 do  
write(' i= ', i, ' , j = ',  
j); writeln;  
end;
```

Этот цикл является внешним, по отношению к нему внутренним будет цикл:

```
for j:=10 to 13 do с телом write (' i = ', i , j =', j);
```

Разберём работу программы, с *вложенным* циклом.

Сначала программа и начинает выполнять внешний цикл: присваивает i=1,

затем переходит к его телу, а здесь встречает внутренний цикл и присваивает j значение 10, после чего выполняет тело внутреннего цикла, т.е. выводит на экран $i=1, j=10$. Так как внутренний цикл еще не окончен, то машина продолжает его выполнять, т.е. присваивает j значение 11 и добавляет к уже выведенной строке $i=1, j=11$.

Заметим, что оператор `write` отличается от оператора `writeln` тем, что он не начинает вывод с новой строки, а продолжает писать в той же строке, т.е. после второго выполнения внутреннего цикла на экране появится
 $i=1, j=10 \quad i=1, j=11$.

Программа продолжит выполнение внутреннего цикла, и, когда он закончится (выполнится для $j = 10, 11, 12, 13$), на экране будет строка
 $i=1 \quad j=10 \quad i=1 \quad j=11 \quad i=1 \quad j=12 \quad i=1 \quad j=13$.

Внутренний цикл закончится, однако тело внешнего цикла еще не закончилось, поэтому выполняется оператор `writeln`, который переводит курсор на новую строку. После этого тело внешнего цикла закончится, но сам цикл отработал только для $i = 1$. Поэтому внешний цикл продолжит работу, присвоив $i := 2$ и вновь начав выполнение своего тела. Встретив внутренний цикл j , на экран с новой строки выведется: $i=2, j=10$, затем к этой строке добавится $i=2, j=11$ и т.д., пока не закончится внутренний цикл.

Таким образом, внешний цикл, изменяя индекс i от 1 до 5, заставит каждый раз выполняться полностью внутренний цикл, и в результате работы программы на экране появится:

```
i=1, j=10    i=1, j=11 i=1, j=12 i=1, j=13
i=2, j=10    i=1, j=11 i=1, j=12 i=1, j=13
i=3, j=10    i=1, j=11 i=1, j=12 i=1, j=13
i=4, j=10    i=1, j=11 i=1, j=12 i=1, j=13
i=5, j=10    i=1, j=11 i=1, j=12 i=1, j=13
```

Вкладывать циклы друг в друга можно сколько угодно раз, необходимо лишь помнить, что количество выполнений самого внутреннего тела цикла при этом будет расти в геометрической прогрессии. Например:

```
for i:=1 to 9 do
  for j:=1 to 9 do
    for k:=1 to 9 do
      writeln (i, j, k);
```

дает столбик цифр:

```
111 112 113 114 ... 119
- - - - -
121 122 123 124 ... 129
- - - - -
211 212 213 214 ... 219
- - - - -
991 992 993 994 ... 999
```

что составляет 1000 строчек.

Вопросы:

- Какие виды циклов вы знаете?
- Что называют вложенным циклом?
- Сколько может быть уровней вложения циклов?
- В чём отличие простых циклов от вложенных?

III. Практическая часть.

На прошлом уроке мы рассмотрели на практике применение циклов в ЯП Pascal ABC.

Сегодня мы рассмотрим на практике применение вложенных циклов в ЯП Pascal ABC.

1. Программа наглядно показывающая работу вложенных циклов:

```
Program VlozCicly;
var x,y:integer;
begin
    for x:=1 to 2 do
        for y:=1 to 4 do
            begin
                writeln('x=',x,' y=',y);
            end;
        readln;
    end.
```

2. Программа вычисляющая сумму всех чисел последовательности от 0 до 100, с помощью вложенных циклов:

```
Program SummDvuznach; // Сумма всех чисел
последовательности от 0 до 100
var s: real; i,j:integer;
begin
    s := 0;
    for i:=1 to 99 do
        s := s + i;
        for j:=1 to 99 do
            write(j:3);
            writeln;
        writeln(' Сумма всех чисел от 0 до 100 =
',s); end.
```

IV. Д/з.

Разобраться с пониманием работы программы с вложенными циклами.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы рассмотрим на практике применение вложенных циклов в ЯП Pascal ABC.

Урок №11. Тема: «Процедуры и функции в ЯП Паскаль. Рекурсия».

Цели урока:

- помочь учащимся усвоить понятия о процедурах и функциях в ЯП Pascal ABC;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (1 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (10 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы рассмотрели на практике применение вложенных циклов в ЯП Pascal ABC.

III. Теоретическая часть.

Процедуры и функции в ЯП Паскаль. Рекурсия.

Практически во всех больших программах, встречаются группы одинаковых команд, которые выполняются много раз. Для того, чтобы упростить вид

программы, наборам таких команд присваивают имена и когда требуется, выполнить эти группы команд, указывают только их имена (вызывают по имени). Такие группы команд называют *функциями* или *процедурами* (*подпрограммами*).

Функциями называют такие группы команд, которые при своём выполнении производят какие-либо вычисления и соответственно возвращают какое-то значение (вычисляет *синус*, *корень*, *модуль числа*, *длину строки* и т.д.).

Процедурами называют какие-либо действия, которые выполняет программа (*очищает экран*, *считывает данные с клавиатуры*, *выводит данные на экран*, *удаляет символы из строки* и т.п.).

Все процедуры и функции делятся на две группы:

стандартные и *пользовательские* (создаваемые разработчиком программы).

Стандартные, входят в состав языка и вызываются для выполнения по своему имени. Процедура или функция представляет собой последовательность операторов, которая имеет имя, список параметров и может быть вызвана из различных частей программы.

И процедура, и функция должна иметь собственное имя и может содержать произвольное число операторов и даже внутренних процедур и функций. Любая используемая в программе процедура или функция должна быть предварительно описана в разделе описаний.

Процедуры и функции, создаваемые разработчиком программы, должны соответствовать следующему виду.

Описание процедуры:

procedure имя (список формальных параметров) ; раздел описаний

begin

операторы

end;

Описание функции:

function имя (список формальных параметров) : тип возвращаемого значения ;

раздел описаний

begin

операторы

end;

Операторы подпрограммы, окаймленные операторными скобками **begin** ... **end**, называются *телом* этой подпрограммы.

Рекурсия — это такой метод организации работы подпрограммы, при котором эта подпрограмма (процедура или функция) в ходе ее выполнения обращается *сама к себе* (т.е. вызов метода из тела самого метода). Рекурсивный стиль программирования довольно эффектен, но не очень эффективен, с точки

зрения расходования ресурсов компьютера. Применение рекурсии увеличивает время исполнения программы и требует значительного объема памяти. В связи с этим, предпочтительнее использовать не рекурсию, а итерацию.

В рекурсивных алгоритмах необходимо предусматривать условие завершения процесса, т.е. когда вызова больше не происходит.

Примером рекурсивного алгоритма, является программа вычисляющая факториал числа N.

```
Program Factorial;
var n:integer;
function f(x:integer):real; //определяем функцию f
обращающуюся,
begin //к предыдущему значению самой
себя,
    if x = 1 then f:= 1 else f:= x * f(x-1); //для
вычисления своего значения
end;
begin
    writeln('Введите число N (N=1..170) '); //при N>170
возникает
    readln(n); // ошибка « Вещественное переполнение »
    writeln('Факториал N!=', f(n) );
end.
```

Вопросы:

- Что называют процедурой в ЯП?
- Что называют функцией в ЯП?
- В чём разница между процедурой и функцией в ЯП?
- Что называют рекурсией в ЯП?

III. Практическая часть.

На прошлом уроке мы рассмотрели на практике применение вложенных циклов в ЯП Pascal ABC. Сегодня на практической части мы напишем программу, с использованием процедуры.

Программа выполняющая процедуры с числами:

```
Procedure Operations(a,b: integer);

begin
```

```

writeln(a, ' + ', b, ' = ', a+b);
writeln(a, ' - ', b, ' = ', a-b);
writeln(a, ' * ', b, ' = ', a*b);
writeln(a, ' / ', b, ' = ', a/b);
writeln(a, ' div ', b, ' = ', a div b);
writeln(a, ' mod ', b, ' = ', a mod b);
end;
begin
    Operations(5,3);
    writeln;
    Operations(7,4);
end.

```

Программа определяющая N-ое число последовательности Фибоначчи, с помощью рекурсивной функции:

```

Program FibonacciRekursia;
var n:byte;
function F(k:byte):integer;
begin
    if k<2 then F:=1 else F:=F(k-1)+F(k-2); {рекурсивный
    вызов}
end;
begin
    write('введите номер числа Фибоначчи
    '); readln(N);
    writeln(N, '-ое число Фибоначчи = ', F(N));
    readln
end.

```

IV. Д/з

Разобраться с понятием процедура, функция и рекурсия.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы усвоили понятия о процедурах и функциях в ЯП Pascal ABC.

Урок №12. Тема: «Стандартные процедуры и функции».

Цели урока:

- помочь учащимся усвоить понятие о стандартных процедурах и функциях в ЯП Pascal ABC;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (2 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы усвоили понятия о процедурах и функциях в ЯП Pascal ABC.

III. Теоретическая часть. Стандартные процедуры и функции.

При работе с различными типами данных в ЯП Паскаль имеются стандартные процедуры и функции для разных типов данных, представленные ниже.

1. Общие стандартные процедуры и функции.

Имя и параметры	Процедура или функция	Типы параметров	Тип возвращаемого значения	Действие
<code>Read(a,b,...)</code>		процедура переменные числового или типа <code>string</code>	<code>a,b</code> -	вводит значения с клавиатуры в переменные <code>a, b ...</code>
<code>Write(a,b,...)</code>	процедура	<code>a,b,c</code> - выражения числового типа или <code>string</code>		выводит значения <code>a, b ...</code> в окно вывода
<code>Readln(a,b,...)</code>	<code>a,b,c</code> -	процедура переменные простого типа или типа <code>string</code>		вводит значения с клавиатуры в переменные <code>a, b ...</code> , и переводит курсор на следующую строку.
<code>Writeln(a,b,...)</code>	процедура	<code>a,b,c</code> - выражения простого типа, типа <code>string</code> или указатели		выводит значения <code>a, b ...</code> в окно вывода и осуществляет переход на следующую строку. Если параметры процедуры не указаны, то выполняет только переход на следующую строку.
<code>Abs(x)</code>	функция	<code>x</code> - <code>integer</code> , <code>real</code>	совпадает с типом параметра	возвращает абсолютное значение (модуль) <code>x</code>
<code>Sqr(x)</code>	функция	<code>x</code> - <code>integer</code> , <code>real</code>	совпадает с типом параметра	возвращает квадрат <code>x</code>
<code>Sqrt(x)</code>	функция	<code>x</code> - <code>real</code> ,	совпадает с типом параметра	возвращает квадратный корень из <code>x</code>
<code>Sin(x)</code>	функция	<code>x</code> - <code>real</code> ,	совпадает с типом параметра	возвращает синус <code>x</code>
<code>Cos(x)</code>	функция	<code>x</code> - <code>real</code> ,	совпадает с типом параметра	возвращает косинус <code>x</code>
<code>Arctan(x)</code>	функция	<code>x</code> - <code>real</code> ,	совпадает с типом параметра	возвращает арктангенс <code>x</code>
<code>Power(x,y)</code>	функция	<code>x, y</code> - <code>real</code>	<code>integer</code>	возвращает результат округления <code>x</code> до ближайшего целого
<code>Round(x)</code>	функция	<code>x</code> - <code>real</code>	<code>real</code>	возвращает целую часть <code>x</code>
<code>Int(x)</code>	функция	<code>x</code> - <code>real</code>	<code>real</code>	возвращает дробную часть <code>x</code>
<code>Frac(x)</code>	функция	<code>x</code> - <code>real</code>	<code>integer</code>	возвращает номер значения порядкового типа (символа)
<code>Ord(x)</code>	функция	<code>x</code> - порядкового типа		
<code>Chr(x)</code>	функция	<code>x</code> - <code>integer</code>	<code>char</code>	возвращает символ с кодом <code>x</code>
<code>Odd(x)</code>	функция	<code>x</code> - <code>integer</code>	<code>boolean</code>	возвращает <code>True</code> , если <code>x</code> - нечетное, и <code>False</code> в противном случае
<code>Inc(x)</code>	процедура	<code>x</code> - порядкового типа		Увеличивает <code>x</code> на 1

Dec (x)	процедура	x - порядкового типа		Уменьшает x на 1
Inc (x, n)	процедура	x - порядкового типа, n - целого типа		Увеличивает x на n
Dec (x, n)	процедура	x - порядкового типа, n - целого типа		Уменьшает x на n
Pred (x)	функция	x - порядкового типа	совпадает с типом параметра	возвращает предыдущее значение порядкового типа
Succ (x)	функция	x - порядкового типа	совпадает с типом параметра	возвращает следующее значение порядкового типа
Random (x)	функция	x - integer	integer	возвращает случайное целое в диапазоне от 0 до x-1
Random	функция		real	возвращает случайное вещественное в диапазоне [0..1)

2. Стандартные процедуры и функции для работы со строками

Имя и параметры	Процедура или функция	Типы параметров	Тип возвращаемого значения	Действие
Length (s)	функция	s - string	integer	возвращает длину строки s
Copy (s, index, count)	функция	s - string , index и count - integer	string	возвращает подстроку строки s длины count, начиная с позиции index
Delete (s, index, count)	процед.	s - string , index и count - integer		удаляет в строке s count символов начиная с позиции index
Insert (pods, s, index)	процед.	s, pods - string , index - integer		вставляет подстроку pods в строку s с позиции index
SetLength (s, n)	процед.	s - string , n - integer		устанавливает длину строки s равной n
Str (x, s) Str (x:n, s) Str (x:n:m, s)	процед.	s - string , x - integer, real и n, m - integer		преобразует x к строковому представлению (во втором и третьем случаях согласно формату вывода, устанавливаемому n и m) и записывает результат в строку s
Val (s, v, code)	процед.	s - string , v - integer, real, и code - integer		преобразует строку s к числовому представлению и записывает результат в переменную v. Если преобразование возможно, то в переменной code возвращается 0, если невозможно, то в code возвращается ненулевое значение

<code>Concat(s1,..,sn)</code>	функция	<code>s1,..., sn - string</code>	<code>string</code>	возвращает строку, являющуюся результатом слияния строк <code>s1,..., sn</code> . Результат тот же, что у выражения <code>s1+s2+...+sn</code>
<code>IntToStr(i)</code>	функция	<code>i - integer</code>	<code>string</code>	преобразует целое число к строке
<code>StrToInt(s)</code>	функция	<code>s - string</code>	<code>integer</code>	преобразует строку в целое число . Если преобразование невозможно, то возникает ошибка времени выполнения

Вопросы:

- Что называют процедурой в ЯП?
- Что называют функцией в ЯП?
- В чём разница между процедурой и функцией в ЯП?
- В чём разница между процедурой `read` и `readln`?

III. Практическая часть.

На прошлых уроках вы получили общие понятия о процедурах и функциях в ЯП Pascal ABC.

Сегодня мы познакомимся на практике с работой программы, в которой используются стандартные процедуры и функции.

Программа возведения в квадрат, с использованием функции `power`:

```

Program stepen2;
var n: integer; a:real; //объявляем переменные
begin
for n:=1 to 10 do
begin
a:=power(2,n);
writeln('2^',n,'=',a)
end;
end.

```

IV. Д/з

Выписать в тетрадь основные стандартные процедуры и функции ЯП Pascal ABC.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы познакомились на практике с работой программы, в которой используются стандартные процедуры и функции ЯП Pascal ABC.

Урок №13. Тема: «Массивы».

Цели урока:

- помочь учащимся усвоить общие понятия о массивах в ЯП Pascal ABC;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости;
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (1 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (10 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы изучили стандартные процедуры и функции, а также познакомились на практике с работой программы, в которой используются стандартные процедуры и функции ЯП Pascal ABC.

III. Теоретическая

часть. Массивы.

Основные понятия.

Если в программе используются множества, содержащие однотипные элементы, то можно использовать понятие *массив*.

Массив – это упорядоченная последовательность данных одного типа, рассматриваемых как единое целое. Доступ к элементам массива осуществляется по индексу (порядковому номеру). В качестве данных в массивах могут

храниться переменные *числового, строкового* и других типов, кроме *файлового*. При описании массива, ему присваивается имя, пишется служебное слово `array`, указывается число входящих в массив элементов `[ ... ]` и тип этих элементов. Например:

```
var //если массивы одного типа и одинакового диапазона, их можно
    объявить списком
a,b : array [1..15] of integer; } {объявлены два массива a и
b}
mass1 : array [20..50] of real;      пример A;
massiv : array [0..255] of char;
```

Диапазон массива задается левой и правой границами изменения индекса массива, так что, массив `a` состоит из 15 элементов, массив `mass1` - из 31, а массив `massiv` - из 256 элементов.

Доступ к каждому элементу массива в программе осуществляется при помощи индекса. В случае, когда левая граница диапазона равна 1, индекс элемента совпадает с его порядковым номером. В программе имя любого элемента массива состоит из имени массива и индекса элемента в квадратных скобках. Например:

```
a[5];
mass1[30];
massiv[256].
```

Во избежание ошибок, индекс не должен выходить за пределы, определенные диапазоном.

В *примере A*, нельзя использовать элементы:

```
a[0];
mass1[19];
massiv[257].
```

В этом случае при выполнении программы появится сообщение об ошибке:

" Ошибка: выход за границы диапазона".

Ввод массивов можно производить вручную (с клавиатуры), или автоматически, с помощью функции генерирования случайных чисел `random`, из файла или вычислить по формуле.

Массивы бывают *одномерные* и *многомерные*.

Вопросы:

- Дайте определение массива?
- Какие виды массивов вы знаете?
- Какие типы массивов вы знаете?

- Опишите формат объявления массива?
- Что такое диапазон массива?
- Как осуществляется доступ к элементам массива?

III. Практическая часть.

На прошлом уроке мы изучили стандартные процедуры и функции, а также познакомились на практике с работой программы, в которой используются стандартные процедуры и функции ЯП Pascal ABC. Сегодня на практической части мы напишем программу, в которой будет вводиться массив переменных числового типа.

1. Программа ввода массива из 20-ти последовательных элементов:

```

Program VvodMass;
const n=20;      //объявляем константу
var a: array [1..n] of integer; //объявляем массив
i:integer; //объявляем переменные
begin
a[1]:=1; //задаем значение первому элементу
последовательности
write(a[1], ' ');
    For i:=2 to n do
        begin
            a[i]:=a[i-1]+1;
            write(a[i], ' ');
        end;
end.

```

2. Программа ввода массива из 20-ти чётных элементов:

```

Program Posledovatelnost;
const n=20;
var a: array [1..n] of integer; //Объявление массива
i,k:integer; //объявляем переменные
begin
a[1]:=0;
write(a[1], ' ');
    for i:=2 to n do
        begin
            a[i]:=a[i-1]+2;
            write(a[i], ' ');
        end;
end.

```

3. Программа ввода массива случайным образом, с помощью функции Random:

```
Program MassSluchayno;  
Var a : Array[1..12] Of Integer; //объявляем массив из  
12 чисел  
i : integer;  
begin  
    Randomize; //процедура генерации случайных  
чисел  
    for i := 1 to 12 do  
        begin  
            a[i] :=random(18)-5;//генерируем числа в  
диапазоне -5..12  
            Write('a',I,'=',a[i]:2,' ');//результат  
end;  
            writeln();  
        end.  
end.
```

IV. Д/з.

Разобраться с понятием «массив». Выучить определения урока.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы познакомились с понятием «массив» теоретически и на практике.

Урок №14. Тема: «Одномерные массивы».

Цели урока:

- помочь учащимся усвоить понятие об одномерных массивах в ЯП Pascal ABC;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (1 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)

IV. Практическая часть. (20 мин)

V. Д/з (2 мин)

VI. Вопросы учеников. (5 мин)

VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы познакомились с общими понятиями о массиве.

III. Теоретическая часть.

Одномерные массивы.

Массив называется одномерным (линейным), если у каждого из его элементов имеется только один индекс.

К примеру, если мы будем в течение месяца ежедневно записывать среднесуточную температуру, и заносить эти данные в таблицу,

	t[1]	t[2]	t[3]	t[...]	t[30]
Моздок	15	17	14	...	21

то у нас получится одномерный массив, в котором будет храниться переменная $t[j]$, с одним индексом (номером столбца).

$t[1], t[2], t[3], \dots, t[n]$.

Пример описания одномерного массива:

```
Var A: array [1 .. 15] of real;
```

Для указания количества элементов массива можно применять константы:

```
Const a=10;
```

```
Var A: array [1 .. a] of integer;
```

Вопросы:

- Дайте определение линейного массива?
- Приведите примеры линейного массива?
- Опишите формат ввода линейного массива?
- Как осуществляется доступ к элементам линейного массива?

III. Практическая часть.

На прошлом уроке мы познакомились с общими понятиями о массивах. Сегодня на практической части мы напишем программу, в которой введём одномерный массив переменных числового типа и вычислим его среднее арифметическое.

1. Программа вычисляющая среднее арифметическое массива:

```
Program SRedMass;
const N=10;
var SR,S:real;
    I,Kol:integer;
    A:array[1..N] of real;
begin S:=0; Kol:=0;
    for I:=1 to N do
    begin
        read(A[I]);
        if A[I]>0 then
            S:=S+A[I]; Kol:=Kol+1;
            SR:=S/Kol;
        end;
    Write('ср.арифм.= ',SR);
end.
```

2. Программа сортировки элементов массива пузырьковым методом:

```
Program SortPuzzyrok;
const n=20;
var a : array [1..n] of integer;
    i,j,k,prom : integer;
begin
    for i:=1 to n do
    begin
        a[i]:=random(101);
        write(a[i], ' ');
    end;
    writeln;
    for i:=2 to n do
        for j:=1 to n do
            if a[i]<a[j] then
                begin
                    prom:=a[j];
                    a[j]:=a[i];
                    a[i]:=prom;
                end;
        end;
    for i:=1 to n do
        begin
            write(a[i], ' ');
        end;
    end.
```

IV. Д/з

Разобраться со всеми понятиями одномерных массивов. Знать, как производится ввод и обработка одномерных массивов.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы узнали, что такое линейный (одномерный) массив. Так же мы написали программу, в которой вводится случайным образом одномерный массив переменных числового типа, и вычислили его среднее арифметическое.

Урок №15. Тема: «Двумерные массивы».

Цели урока:

- помочь учащимся усвоить понятие об одномерных массивах в ЯП Pascal ABC;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости.
- развитие познавательных интересов, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (1 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (10 мин)
- IV. Практическая часть. (20 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На прошлом уроке мы узнали, что такое линейный (одномерный) массив. Так же мы написали программу, в которой ввели случайным образом одномерный массив переменных числового типа, и вычислили его среднее арифметическое.

III. Теоретическая часть.

Двумерные массивы.

В двумерном массиве у каждого элемента имеются два индекса.

В математике такие массивы называют матрицами.

Если мы будем заносить в таблицу данные о температуре в нескольких городах,

		[1]	[2]	[3]	[...]	[30]
1	Моздок	15	17	14	...	21
2	Ставрополь	16	15	17	...	20
3	Пятигорск	14	12	16	...	18

то получится двумерный массив, где в каждой ячейке будет храниться переменная с двумя индексами $t[i, j]$, где i – номер строки, а j – номер столбца.

Матрицы могут быть прямоугольными или квадратичными (в которых количество строк и столбцов равны). Именно квадратные матрицы наиболее интересны для решения задач на программирование, в связи с тем, что у них имеются главная и вспомогательная диагонали.

X	X	X	X
11	12	13	14
X	X	X	X
21	22	23	24
X	X	X	X
31	32	33	34
X	X	X	X
41	42	43	44

главная диагональ

X	X	X	X
11	12	13	14
X	X	X	X
21	22	23	24
X	X	X	X
31	32	33	34
X	X	X	X
41	42	43	44

вспомогательная диагональ

На главной диагонали лежат элементы, у которых $i = j$.

На вспомогательной диагонали - элементы, у которых $i = n - j + 1$,

где n – количество строк (столбцов).

Пример описания двумерного массива (4 – строки, 6 - столбцов):

```
Var A: array [1 .. 4, 1 .. 6] of real;
```

Для указания границ массива можно применять константы:

```
Const a=4; b=6;
```

```
Var A: array [1 .. a, 1 .. b] of integer;
```

Обработку массивов производят с помощью циклов.

Вопросы:

- В чём отличие двумерных массивов от одномерных?
- Почему именно квадратные матрицы наиболее часто используются для решения задач на программирование?
- Опишите условие принадлежности элемента квадратной матрицы главной диагонали?
- Опишите условие принадлежности элемента квадратной матрицы вспомогательной диагонали?
- Как осуществляется доступ к элементам двумерного массива?
- Как производят обработку массивов?

III. Практическая часть.

На прошлом уроке рассматривали понятие одномерного массива. Так же мы написали программу вычисления среднего арифметического линейного массива. Сегодня на практической части мы напишем программу, в которой производится ввод элементов массива и вычисляется сумма элементов главной диагонали квадратичной матрицы.

1. Программа ввода матрицы вручную:

```
Program VvodVruch;  
const n=4; m=3; var a: array [1..n,1..m] of integer;  
i,j:integer;  
begin  
  for i:=1 to n do  
    begin  
      for j:=1 to m do  
        begin  
          readln(a[i,j]);  
        end;  
      end;  
    end;  
  for i:=1 to n do
```

```

begin
    for j:=1 to m do
        begin
            write(a[i,j]:3);
        end;
        writeln;
    end;
end.

```

2. Программа вычисления суммы элементов главной диагонали квадратной матрицы:

```

Program SummGlavDiag;
const n=3;
var a: array [1..n,1..n] of integer;
s,i,j:integer;
begin
s:=0;
for i:=1 to n do
begin
for j:=1 to n do
begin
a[i,j]:=random(101);
write(a[i,j]:5);
if i=j then s:=s+a[i,j];
end;
writeln;
end;
writeln('Сумма эл. главной диаг.',s);
end.

```

IV. Д/з

Разобраться со всеми понятиями двумерных массивов. Знать, как производится ввод и обработка двумерного массива.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы ознакомились с понятием двумерный массив. Так же мы написали

программу, в которой производится ввод элементов массива и вычисляется сумма элементов главной диагонали квадратичной матрицы.

Урок №16. Тема: «Работа с графикой».

Цели урока:

- помочь учащимся усвоить принципы работы с графикой в программе PascalABC, дать основные понятия, необходимые для начала написания программ, выводящих графические изображения на компьютере;
- воспитание информационной культуры учащихся, внимательности, аккуратности, дисциплинированности, усидчивости;
- развитие познавательных интересов, навыков работы с мышью и клавиатурой, самоконтроля, умения конспектировать.

Оборудование:

доска, компьютер, компьютерная презентация.

План урока:

- I. Орг. момент. (2 мин)
- II. Актуализация знаний. (5 мин)
- III. Теоретическая часть. (12 мин)
- IV. Практическая часть. (17 мин)
- V. Д/з (2 мин)
- VI. Вопросы учеников. (5 мин)
- VII. Итог урока. (2 мин)

Ход урока:

I. Орг. момент.

Приветствие, проверка присутствующих. Объяснение хода урока.

II. Актуализация знаний.

На уроке мы познакомились с понятием двумерный массив. Так же мы написали программу, в которой производится ввод элементов массива и вычисляется сумма элементов главной диагонали квадратичной матрицы.

III. Теоретическая часть.

Работа с графикой.

После запуска PascalABC, по умолчанию, запускается текстовый режим. Для работы с графикой служит отдельное графическое окно. Чтобы его открыть, необходимо подключить модуль GraphABC. В этом модуле содержится обширный набор процедур и функций, предназначенных для работы с

графическим экраном, а также некоторые встроенные константы и переменные, которые могут быть использованы в программах с графикой. С их помощью можно создавать разнообразные графические изображения и сопровождать их текстовыми надписями. Подключение осуществляется в разделе описаний.

Формат подключения модуля GraphABC:

```
uses GraphABC;
```

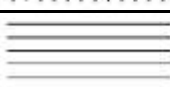
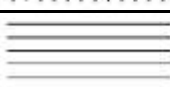
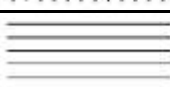
Начало координат графического окна модуля находится в верхнем левом углу.

Ось *x* направлена вправо, а ось *y* - вниз. Координаты исчисляются в пикселях.

Все команды библиотеки GraphABC являются подпрограммами и описаны в виде процедур и функций. Для того, что бы команда выполнялась необходимо указать команду и задать значения параметров.

Некоторые команды библиотеки GraphABC.

Имя и параметры команды	Действие команды																					
Line(x1,y1,x2,y2)	Рисует отрезок из точки с координатами (x1,y1) в точку с координатами (x2,y2)																					
LineTo(x, y)	Рисует отрезок из текущей точки в точку с координатами (x, y) (соответствует команде сместиться_в с опущенным пером для исполнителя Чертежник)																					
Rectangle(x1,y1,x2,y2)	Рисует контур прямоугольника со сторонами параллельными сторонам экрана. Точки с координатами (x1, y1) и (x2, y2) определяют диагональные вершины прямоугольника.																					
RoundRect(x1,y1,x2,y2,w,h)	Рисует прямоугольник со скругленными краями; (x1,y1) и (x2,y2) задают пару противоположных вершин, а w и h – ширину и высоту эллипса, используемого для скругления краев.																					
FillRect(x1,y1,x2,y2);	Заливает прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2), цветом текущей кисти.																					
Circle(x,y, radius);	Рисует окружность с центром в точке с координатами (x,y) и радиусом radius.																					
Ellipse(x1,y1,x2,y2);	Рисует эллипс, заданный своим описанным прямоугольником с координатами противоположных вершин (x1, y1) и (x2, y2).																					
MoveTo(x, y)	Переводит текущую точку в положение новое положение, в точку с координатами (x, y) (соответствует команде сместиться_в с поднятым пером для исполнителя Чертежник)																					
TextOut (x, y, text)	Выводит Text на экран. Начало текста в точке с координатами (x, y)																					
SetPixel(x,y,color)	Закрашивает цветом color точку с координатами (x, y)																					
SetPenColor(color)	Задаёт цвет рисования линий. Параметр color – число в промежутке от 0 до 256 ³ -1.																					
SetPenStyle(ps)	Устанавливает стиль пера, задаваемый параметром ps. Стили пера задаются следующими именованными константами: <table><tr><td>Значение</td><td>Описание</td><td>Вид линии</td></tr><tr><td>psSolid</td><td>Сплошная линия</td><td>_____</td></tr><tr><td>psDash</td><td>Штриховая линия</td><td>-----</td></tr><tr><td>psDot</td><td>Пунктирная линия</td><td>.....</td></tr><tr><td>psDashDot</td><td>Штрихпунктирная линия</td><td>-----</td></tr><tr><td>psDashDotDot</td><td>Линия - штрих и два пунктира</td><td>-----</td></tr><tr><td>psClear</td><td>Отсутствие линии</td><td></td></tr></table>	Значение	Описание	Вид линии	psSolid	Сплошная линия	_____	psDash	Штриховая линия	-----	psDot	Пунктирная линия		psDashDot	Штрихпунктирная линия	-----	psDashDotDot	Линия - штрих и два пунктира	-----	psClear	Отсутствие линии	
Значение	Описание	Вид линии																				
psSolid	Сплошная линия	_____																				
psDash	Штриховая линия	-----																				
psDot	Пунктирная линия																					
psDashDot	Штрихпунктирная линия	-----																				
psDashDotDot	Линия - штрих и два пунктира	-----																				
psClear	Отсутствие линии																					
Arc(x,y,r,a1,a2)	Рисует дугу окружности с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки).																					
Pie(x,y,r,a1,a2)	Рисует сектор окружности, ограниченный дугой (параметры процедуры имеют тот же смысл, что и в процедуре Arc).																					
Chord(x,y,r,a1,a2)	Рисует фигуру, ограниченную дугой окружности и отрезком, соединяющим ее концы (параметры процедуры имеют тот же смысл, что и в процедуре Arc).																					

FloodFill(x,y,color);	Заливает область одного цвета цветом color, начиная с точки (x,y).																				
SetBrushColor(color)	Устанавливает цвет кисти, задаваемый параметром color																				
SetBrushStyle(bs);	Устанавливает стиль кисти, задаваемый параметром bs. Стили кисти задаются следующими именованными константами: <table><tr><td>Константа</td><td>Стиль</td><td>Константа</td><td>Стиль</td></tr><tr><td>bsSolid</td><td></td><td>bsClear</td><td></td></tr><tr><td>bsCross</td><td></td><td>dsDiagCross</td><td></td></tr><tr><td>bsHorizontal</td><td></td><td>bsVertical</td><td></td></tr><tr><td>bsBDiagonal</td><td></td><td>bsFDiagonal</td><td></td></tr></table>	Константа	Стиль	Константа	Стиль	bsSolid		bsClear		bsCross		dsDiagCross		bsHorizontal		bsVertical		bsBDiagonal		bsFDiagonal	
Константа	Стиль	Константа	Стиль																		
bsSolid		bsClear																			
bsCross		dsDiagCross																			
bsHorizontal		bsVertical																			
bsBDiagonal		bsFDiagonal																			
SetFontColor(color);	Устанавливает цвет шрифта																				
SetFontSize(sz);	Устанавливает размер шрифта в пунктах.																				
SetFontName(name: string);	Устанавливает наименование шрифта По умолчанию установлен шрифт, имеющий наименование MS Sans Serif. Наиболее распространенные шрифты – это Times, Arial и Courier New. Наименование шрифта можно набирать без учета регистра.																				
SetFontStyle(fs);	Устанавливает текущий стиль шрифта. Стили шрифта задаются следующими именованными константами: fsNormal – обычный; fsBold – жирный; fsItalic – наклонный; fsBoldItalic – жирный наклонный; fsUnderline – подчеркнутый; fsBoldUnderline – жирный подчеркнутый; fsItalicUnderline – наклонный подчеркнутый; fsBoldItalicUnderline – жирный наклонный подчеркнутый.																				
ClearWindow;	Очищает графическое окно белым цветом																				
ClearWindow(c);	Очищает графическое окно цветом (c).																				
SetWindowSize(w,h)	Устанавливает ширину и высоту графического окна.																				

Модуль GraphABC содержит константы и функции для работы с цветами. Тип ColorType, описывающий цвет, определен следующим образом:

```
type ColorType=integer;
```

Стандартные цвета задаются символическими константами:

clBlack	–	черный	clAqua	–	бирюзовый
clPurple	–	фиолетовый	clOlive	–	оливковый
clWhite	–	белый	clFuchsia	–	сиреневый
clMaroon	–	темно-красный	clTeal	–	сине-зеленый
clRed	–	красный	clGray	–	темно-серый
clNavy	–	темно-синий	clLime	–	ярко-зеленый
clGreen	–	зеленый	clMoneyGreen	–	цвет зеленых денег
clBrown	–	коричневый	clLtGray	–	светло-серый
clBlue	–	синий	clDkGray	–	темно-серый
clSkyBlue	–	голубой	clMedGray	–	серый
clYellow	–	желтый	clSilver	–	серебристый
clCream	–	кремовый			

Для работы с цветами используются следующие функции.

```
function RGB(r,g,b: integer): ColorType;
```

Возвращает целое значение, являющееся кодом цвета, который содержит красную, зеленую и синюю составляющие с интенсивностями R, G и B соответственно (R, G и B – целые в диапазоне от 0 до 255, причем, 0 соответствует минимальной интенсивности, 255 – максимальной).

```
function GetRed(color: ColorType): integer;
```

Выделяет красный цвет интенсивностью (целое число от 0 до 255);

```
function GetGreen(color: ColorType): integer;
```

Выделяет зеленый цвет интенсивностью (целое число от 0 до 255);

```
function GetBlue(color: ColorType): integer;
```

Выделяет синий цвет интенсивностью (целое число от 0 до 255).

Программа рисующая фигурку:

```
Program Figurka;
uses GraphABC;
var w,r,c: integer;
begin
SetPenWidth(3);
SetBrushColor(clFuchsia);
SetWindowSize(500,500);
Circle(225,160,50);
Line(225,160,225,180);
Line(210,190,240,190);
Line(225,210,225,250);
Line(100,100,200,260);
Line(200,260,400,260);
Line(210,350,200,480);
Line(240,350,250,480);
Rectangle(200,230,250,350);
SetBrushColor(clLime);
FillRect(0,480,500,500);
SetBrushColor(clWhite);
Circle(205,150,10);
Circle(245,150,10);
end.
```

Вопросы:

1. Что необходимо сделать при написании программы для того, чтобы вывести графическое изображение?
2. Каков формат подключения модуля GraphABC?

3. Какие алгоритмические структуры использовались при написании данной программы?
4. Какая команда необходима для рисования линии? Прямоугольника? Закрашенного прямоугольника?

III. Практическая часть.

На прошлом уроке мы ознакомились с понятием двумерный массив и написали программу, вычисления суммы элементов главной диагонали квадратичной матрицы. Сегодня мы на практике попробуем с помощью программы написанной на ЯП Pascal ABC нарисовать рисунок дома.

Задачи на построение графики.

Написать программу, которая рисует на экране компьютера домик, используя процедуры `line`, `rectangle`, `circle`. Подписать рисунок.

Образец выполнения задания.



Этапы выполнения задания.

- I Определение исходных данных: результат работы программы не зависит от переменных.
- II Определение результатов: результатом является готовый рисунок.
- III Алгоритм решения задачи.
 - 1 Инициализация графического режима.
 - 2 Выполнение рисунка. Рисунок состоит из следующих элементов: прямоугольники, линии, окружность. Для правильного расчета координат рекомендуется предварительно сделать рисунок на листе бумаги в клеточку.
- IV Описание переменных:

Для работы программы понадобится описать целые переменные `gd, gm` необходимые для инициализации графики.
- V Пример графической программы, рисующей изображение дома:

```

Program Domik;
uses graphABC;           //подключение модуля graphABC
begin
  SetWindowWidth(800);    //ширина окна программы
  SetWindowHeight(600);   //высота окна программы
  SetFontStyle(fsBold);   //жирный стиль шрифта
  SetFontSize(18);        //размер шрифта
  SetFontColor(clRed);    //цвет шрифта
  TextOut(100,100, 'Домик'); //текст
  Rectangle(200,300,600,600); //дом
  Circle(400,225,40);     //круг
  SetBrushColor(clAqua);  //цвет заливки окна
  FillRect(300,400,500,500); //процедура заливки окна
  Rectangle(300,400,500,500); //окно
  Line(400,400,400,500);  //окно
  Line(300,450,500,450);  //окно
  Line(200,300,400,150);  //крыша
  Line(400,150,600,300);  //крыша
  Line(480,210,480,160);  //труба
  Line(480,160,520,160);  //труба
  Line(520,160,520,240);  //труба
end.

```

IV. Д/З.

Выучить команды рисования графических примитивов и функций для работы с цветом.

V. Вопросы учеников.

Ответы на вопросы учащихся.

VI. Итог урока.

Подведение итога урока. Выставление оценок.

На уроке мы узнали, что в системе программирования Pascal ABC можно создавать графические изображения. Так же мы познакомились с основными процедурами и функциями графического модуля GraphABC и наглядно проверили работу графической программы.

Раздел 8. Решение задач.

Задачи на математические вычисления:

1. Напишем самую простую программу вычисления результата сложения двух переменных $s = a+b$.

```

Program Slozenie;      {заголовок программы}
var  a,b,s:integer;    (*раздел описания переменных*)
begin                  //начало раздела операторов
  read(a,b);           {занесение в ячейки a и b их значений}
  s:=a+b;               {вычисление значения s}
  writeln('сумма = ',s); {вывод на экран текста 'сумма = ' и
значения s}
end.                   {конец программы}.

```

Напоминаю, что тексты, заключённые в скобки {}, (* *) и после // являются комментариями.

2. Программа вычисления площади S прямоугольного треугольника, по двум катетам

a и b:

```

Program PloshadTreugl;
Var a,b: integer, S:real; //объявление переменных
begin
  writeln ('введите целые значения длины катетов a,b');
                                {вывод пояснительного текста на экран}
  readln (a,b); //ввод данных (a и b) с клавиатуры
  S:=a*b/2; //вычисление площади треугольника
  writeln('площадь треугольника = ',S);
end.

```

3. Эту же программу можно записать без объявления переменной S.

Прямо в операторе вывода writeln можно вычислить значение площади по формуле, и вывести его значение на экран в виде числа, с пояснением.

```

Program PloshadTreugl;
Var a,b: integer; //объявление переменных begin

  writeln ('введите целые значения длины катетов a,b');
                                {вывод пояснительного текста на экран}
  readln (a,b); //ввод данных (a и b) с клавиатуры
  S:=; //вычисление площади треугольника
  writeln('площадь треугольника = ', a*b/2);

```

```
    {Вывод пояснительного текста на экран и результата  
    вычисления площади тр.}
```

```
end.
```

4. Программа вычисления площади трапеции по высоте и двум основаниям:

```
Program PloshTrap;  
    var a,b,h,s: real; //объявляем переменные  
begin  
    writeln(введите значения оснований и высоты');  
    readln(a,b,h); //вводим значения оснований и  
    ВЫСОТЫ  
    s:=0.5*(a+b)*h;      //вычисляем площадь  
    writeln('Площадь трапеции = ',s); //выводим на экран  
end.
```

3. Программа, определяющая вид треугольника по его сторонам:

```
Program SravStoronTreug;  
var a,b,c: real; //объявляем переменные  
begin  
    writeln('Введите стороны a= b= c=');  
    readln(a,b,c);  
    if (a+b<=c) or (b+c<=a) or (b+c<=a) then  
        writeln('Треугольник не существует')  
    else  
        begin  
            if (a=b) and (b=c) then writeln('Треугольник -  
            равносторонний');  
            if (a<>b) and (b<>c) and (a<>c) then writeln  
            ('Треугольник - разносторонний');  
            if (sqr(a)=sqr(b)+sqr(c)) or (sqr(b)=sqr(a)+sqr(c)) or (sqr(c)=sq  
            r(b)+sqr(a)) then writeln('Треугольник -  
            прямоугольный');  
            if ((a=b) and (c<>a)) or ((b=c) and (a<>b)) or  
            ((c=a) and (c<>b)) then writeln('Треугольник -  
            равнобедренный');  
        end  
    end.
```

4. Программа определения дня недели, по введённому номеру:

```

Program OperatorCase;
var den: integer; //объявляем переменные
begin

  writeln('Введите номер дня недели (1..7):
  '); readln(den);
  case den of // Оператор выбора 1..5:
    writeln('Рабочий день');
    6,7: writeln('Выходной');
    else writeln('Введите правильно номер
    дня!'); end;
end.

```

7. Программа сравнения введенных чисел (полная форма условного оператора if):

```

Program SravChisel;
var a,b: integer ; //объявляем переменные
begin
writeln('Введите a и b');
read(a,b);
  if a<b then
    writeln(a, '<', b)
  else
    if a>b then
      writeln(a, '>', b)
    else writeln(a, '=', b);
end.

```

8. Программа, выводящая на экран таблицу умножения на 3:

```

Program Umnoz3;
var n: integer; a: real; //объявляем переменные
begin
for n:=1 to 10 do
  begin
    a:=n*3;
    writeln('3*', n, '=', a)
  end;
end.

```

9. Программа нахождения всех делителей числа a:

В цикле `for` последовательно делим число `a` на параметр цикла, начиная от 1 до значения числа `a`, и находим все числа, остаток от деления которых (`mod`) равен нулю. Если такие числа есть, то выводим их на экран их с помощью оператора `write(i, ' ');`

```
Program Deliteli;
var a,i: integer; //объявляем переменные
begin
    write('Введите число a=');
    readln(a);
    write('Делители числа a: ');
    for i:=1 to a do
        begin
            if a mod i =0 then write(i, ' ');
        end;
    end.
```

11. Программа вычисления квадрата и квадратного корня числа `n`:

```
Program Kvad_Kor;
var kv,n:integer; kor:real; //объявляем переменные
begin
    writeln('Введите число n');
    readln(n);
    kv:=sqr(n);
    kor:=sqrt(n);
    writeln('Квадрат числа ',n,' = ',kv,' Корень числа ',n,' '
    = ',kor);
end.
```

12. Программа возведения в квадрат, с использованием функции `power`:

```
Program stepen2;
var n: integer; a:real; //объявляем переменные
begin
    for n:=1 to 10 do begin
        a:=power(2,n);
        writeln('2^',n,' =',a)
    end;
end.
```

13. Программа вычисления длины отрезка по введенным координатам:

```

Program DlinaOtrezka;
var x1,y1,x2,y2,d:real; //объявляем переменные begin
// Вычисляем длину отрезка по теореме Пифагора
    writeln(' Введите координаты точек A(X1,Y1)
и B(X2,Y2) ');
    readln( x1,y1,x2,y2 );
    d:=sqrt(sqr(y2-y1)+sqr(x2-x1));
    writeln(' Длина отрезка |AB|=',d);
end.

```

14. Программа возведения в степень любого числа:

```

Program StepenChisla;
var p,a: real; n,i: integer; //объявляем переменные
begin
    writeln('Введите a,n: ');
    read(a,n);
    p:=1; //задаём начальное значение отличное
от 0
    for i:=1 to n do //организуем цикл
        p := p * a; // Вычисляем степень числа
    writeln(a,' В степени ',n,' = ',p); // выводим результат
end.

```

Задачи с числовыми последовательностями.

5. Программа вычисления суммы чисел кратных 3, в диапазоне 0...100.

```

Program Summ3;
var s,i:integer; //объявляем переменные
begin
for i:=1 to 100 do //задаём диапазон от 0 до
100 if i mod 3 = 0 then //находим числа кратные
3 write(i:3);
    writeln();
    begin
        s:=0;
        for i:=1 to 100 do
            if i mod 3 = 0 then
                s:=s+i; //находим сумму чисел кратных 3
            end;
        writeln(' s = ',s); // выводим результат
    end.
end.

```

6. Программа вычисления суммы нечетных чисел последовательности от 0 до 100, с помощью цикла *repeat*.

```
Program SumChet;  
var i,s:integer;      // i-индекс,s-накопитель суммы  
begin  i:=0; s:=0; //обнуление исходных значений i,s  
      repeat //оператор цикла повтора  
        i:=i+2; //нахождение чётных индексов.  
        s:=s+i; {т.к. индексы последовательности  
соответствуют значениям чисел, то к предыдущему значению  
's' прибавляем значения чётных индексов}  
      until i>99; //условие выхода из цикла  
      writeln('s=',s); //вывод результата  
end.
```

7. Напишем программу нахождения суммы нечетных чисел последовательности от 0 до 100, с помощью цикла *while*.

```
Program SumChet1;  
var i,s:integer;      // i-индекс,s-накопитель суммы  
begin i:=0; s:=0; //обнуление исходных значений i,s  
      while i<100 do //оператор цикла повтора с условием  
выхода из него  
      begin  
        i:=i+2; //нахождение чётных индексов.  
        s:=s+i; //вычисление суммы  
      end;  
      writeln('s=',s); //вывод результата  
end.
```

8. Напишем ту же программу нахождения суммы нечетных чисел последовательности от 0 до 100, с помощью цикла *for*.

```
Program SumChet3;  
var i,s:integer;      // i-индекс,s-накопитель суммы  
begin  s:=0;  
      for i:=0 to 100 do  
        begin  
          s:=s+i; //вычисление суммы  
          i:=i+1;  
        end;  
      writeln('s=',s); //вывод результата
```

end.

9. Та же программа, с помощью цикла *for* и операции *mod*..

Для поиска чётных чисел используется проверка чётности с помощью операции целочисленного деления *mod* . (Если остаток от деления числа на 2 равен 1 то, следовательно, число нечётное).

```
Program SumChet4;
var i,s:integer;      // i-индекс,s-накопитель суммы
begin
  for i:=0 to 100 do
    begin
      if i mod 2=1 then      //нахождение нечётных
индексов
        s:=s+i;      //вычисление суммы
      end;
      writeln('s=',s); //вывод результата
    end.
```

10. Простая сортировка 3-х чисел:

```
Program ProstSort;
var a,b,c: integer;
begin
  readln(a,b,c);
  if (c>b)and(b>a) then write(a,' ',b,' ',c);
  if (b>c)and(c>a) then write(a,' ',c,' ',b);
  if (c>a)and(a>b) then write(b,' ',a,' ',c);
  if (a>c)and(c>b) then write(b,' ',c,' ',a);
  if (b>a)and(a>c) then write(c,' ',a,' ',b);
  if (a>b)and(b>c) then write(c,' ',b,' ',a);
end.
```

11. Программа нахождения Max элемента массива и его порядкового номера:

```
Program MaxElement;
const n=100;
var a : array [1..n] of integer;
    i,j,p,max : integer;
begin
  for i:=1 to n do
```

```

begin
    a[i]:=random(101);
    write(a[i], ' ');
end;
writeln;
max:=a[1];
p:=1;
for j:=2 to n do
    begin
        if a[j]>max then
            begin
                max:=a[j];
                p:=j;
            end;
        end;
    writeln('Max элемент = ',max, 'Его № = ',p);
end.

```

12. Программа вычисления N-го числа последовательности (ряда)

Фибоначчи, с использованием метода рекурсии.

Последовательность Фибоначчи – это ряд чисел, в котором, начиная с третьего элемента ряда, каждое следующее число равняется сумме двух предыдущих.

```

Program FibonachiRekursia;
var n:byte;      //объявляем переменные
function F(k:byte):word;      //определяем функцию F,
    осуществляющую
begin            //рекурсивный вызов, т.е. обращение к двум
    предыдущим своим
    if k<2 then F:=1 else F:=F(k-1)+F(k-2);      //значениям F(k-
1)и F(k-2)
end;
begin
    write('введите номер числа Фибоначчи
    '); readln(N);
    writeln(N, '-ое число Фибоначчи
    =', F(N)); readln
end.

```

13. Та же задача, выполненная с помощью цикла, причём выводящая весь ряд чисел Фибоначчи, от 1 до N:

```
Program FibonachCicl;  
var  
i,N,k1,k2,kn:integer; //объявляем переменные  
begin  
    writeln('Введите число N');  
    readln(N);  
    k1:=1;           //первые два члена ряда равны = 1  
    k2:=1;  
    writeln('f1 = ',k1);  
    writeln('f2 = ',k2);  
    for i:=1 to n-2 do  
    begin  
        kn:=k1+k2; // kn - это число Фибоначчи и индексом i+2,  
т.к. вычисление  
        writeln('f',i+2,' = ',kn); // начинаем с 3-го элемента  
последовательности  
        k1:=k2; //переопределяем значения переменных, т.е.  
присваиваем  
        k2:=kn; //двум последним элементам значение двух  
предыдущих  
    end;  
end.
```

Программы работы со строками.

8. Программа конкатенации строк:

```
Program Concatenacia;  
var s,s1,s2: string; //объявляем переменные  
begin  
  
    Writeln('Введите слова');  
    readln(s1,s2);  
    s:=Concat(s1,s2);  
    Writeln('Полученное слово =  
' ,s); end.
```

9. Программа вычисления длины строки (количество символов в строке):

```

Program DlinaStroki;
var s: string; L:integer; //объявляем переменные
begin

  writeln('введите строку');
  readln(s);
  L:= Length(s);
  Writeln('Длина строки =
',L); end.

```

10. Программа вырезания n символов из строки s , начиная с позиции i , с помощью функции $\text{Copy}(s, i, n)$:

```

Program VyrezSimvol;
  var s,s1: string; i,n:integer; //объявляем
переменные
begin
  Writeln('введите слово');
  readln(s1);
  Writeln('введите с какого символа и сколько
их вырезать');
  Write('i = '); read(i);
  Write('n = '); read(n);
  s1:=Copy(s,i,n); // смотри раздел Стандартные
процедуры и функции
  Writeln('полученное слово =
',L); end.

```

11. Программа удаления из введенной строки s , первых n символов.

```

Program UдалenieSimv;
uses crt;
var s:string;n:integer;
begin
  writeln('Введите строку и количество
удаляемых символов');
  readln(s,n);
  delete(s,1,n);
  writeln('после удаления получилось - ',
s); end.

```

12. Программа поиска символа в строке:

```

Program PoiskSimvola;

```

```

var s,si:string; i:integer; f:boolean; //объявляем
переменные
begin
  writeln('Введите строку');
  readln(s); //считываем строку
  writeln(' Введите символ');
  readln(si);      //обозначаем искомый символ si
  f:=false; //изначально считаем, что символа в
строке нет
  for i:=1 to length(s) do
  begin
    //функцией copy проверяем
совпадают ли
    if copy(s,i,1)= si then f:=true //вырезаемые
символы с
искомым
  end;      //тогда логическая функция f примет
значение true
  writeln(f); //выводим результат
end.

```

13. Программа вычисления количества слов в строке;

```

Program KolichSlov;
var s: string;
n,k: integer;
begin
  writeln(введите строку ');
  readln(s);
  for n:=1 to length(s) do
  begin
    if (s[n]=' ') then
      k:=k+1;
  end;
  writeln('В строке',k+1,' слов'
); end.

```

14. Программа записи слова в обратном порядке (перевёртыш):

```

Program Perevorot;

```

```

var c,c1:string;
i:integer; //объявляем переменные
begin
  writeln('Введите слово');
  readln(c);
  for i:=length(c) downto 1 do
    c1:=c1+copy(c,i,1);
    writeln('перевёртыш = ',c1);
  end.

```

Задачи с массивами.

4. Программа ввода массива из 20-ти последовательных элементов:

```

Program VvodMass;
const n=20;      //объявляем константу
var a: array [1..n] of integer; //объявляем массив
i:integer; //объявляем переменные
begin
  a[1]:=1;      //задаем значение первому элементу
последовательности
  write(a[1], ' ');
  For i:=2 to n do
    begin
      a[i]:=a[i-1]+1;
      write(a[i], ' ');
    end;
end.

```

5. Программа ввода массива из 20-ти чётных элементов:

```

Program Posledovatelnost;
const n=20;
var a: array [1..n] of integer; //Объявление массива
i,k:integer; //объявляем переменные
begin
  a[1]:=0;
  write(a[1], ' ');
  for i:=2 to n do
    begin
      a[i]:=a[i-1]+2;
      write(a[i], ' ');
    end;
end;

```

end.

6. Программа ввода массива случайным образом, с помощью функции Random:

```
Program MassSluchayno;
Var a : Array[1..12] Of Integer;  //объявляем массив из
12 чисел
i : integer;
begin
    Randomize; //процедура генерации случайных
чисел
    for i := 1 to 12 do
        begin
            a[i] := random(18)-5; //генерируем числа в
диапазоне -5..12
            Write('a',I, '=',a[i]:2, ' '); //результат
            end;
            writeln();
        end.
end.
```

7. Программа вычисления сумм отдельных строк матрицы:

```
Program SummStrok;
const n=3; m=3;
var a: array [1..n,1..m] of
integer; i,j,s,k,t:integer; begin

    for j:=1 to n do
        begin
            for i:=1 to m do
                begin
                    a[i,j]:= random(101)-50;
                    write(a[i,j]:4);
                    s:= s+a[i,j];
                end;
                writeln(' s=',s);
                s:=0;
            end;
        end.
end.
```

8. Программа ввода матрицы вручную:

```
Program VvodVruch;
const n=4; m=3; var a: array [1..n,1..m] of integer;
i,j:integer;
begin
```

```

for i:=1 to n do
begin
    for j:=1 to m do
        begin
            readln(a[i,j]);
        end;
    end;
    for i:=1 to n do
    begin
        for j:=1 to m do
            begin
                write(a[i,j]:3);
            end;
            writeln;
        end;
    end;
end.

```

9. Программа сортировки элементов массива пузырьковым методом:

```

Program SortPuzzyrok;
const n=20;
var a : array [1..n] of integer;
    i,j,k,prom : integer;
begin
    for i:=1 to n do
        begin
            a[i]:=random(101);
            write(a[i], ' ');
        end;
        writeln;
        for i:=2 to n do
            for j:=1 to n do
                if a[i]<a[j] then
                    begin
                        prom:=a[j];
                        a[j]:=a[i];
                        a[i]:=prom;
                    end;
            end;
        for i:=1 to n do
            begin
                write(a[i], ' ');
            end;
        end;
end.

```

10. Программа вычисления суммы элементов главной диагонали квадратной матрицы:

```

Program SummGlavDiag;

```

```

const n=3;
var a: array [1..n,1..n] of integer;
s,i,j:integer;
begin
s:=0;
  for i:=1 to n do
    begin
      for j:=1 to n do
        begin
          a[i,j]:=random(101);
          write(a[i,j]:5);
          if i=j then s:=s+a[i,j];
        end;
        writeln;
      end;
      writeln('Сумма эл. главной диаг.',s);
    end.

```

11. Программа вычисления суммы элементов вспомогательной диагонали:

```

Program SummVspomDiag;
const n=3;
var a: array [1..n,1..n] of integer; s,i,j:integer;
begin
s:=0;
  for i:=1 to n do
    begin
      for j:=1 to n do
        begin
          a[i,j]:=random(101);
          write(a[i,j]:5);
          if i=n-j+1 then
            s:=s+a[i,j]; end;
          writeln;
        end;
      end;
      writeln('Сумма элементов вспом. диаг.=
',s); end.

```