**Разработка урока по теме: «Рекурсия. Рекурсивные алгоритмы» Предметная область**: информатика;

**Участники**: учащиеся 10 класса;

**Цель урока**: создать условия для:

* формирования представления о рекурсивном объекте и рекурсивном алгоритме,
* освоения приёмов применения рекурсивных функции при составлении программ на языке программирования Pascal.

# Задачи урока:

***Развивающие:*** Развивать логическое мышление. Отрабатывать навыки работы на компьютере. Развивать у школьников умение выделять главное, существенное, обобщать имеющиеся факты, логически излагать мысли. Развивать умения и навыки сознательного и рационального использования современных компьютерных технологий в своей учебной деятельности.

***Образовательные:*** Сформировать понятия рекурсивного объекта и рекурсивного определения, познакомить учащихся с рекурсивными алгоритмами, научить ребят составлять программы с использованием рекурсивных функций; выражений на алгоритмический язык. Продолжить изучение подпрограмм. Показать сущность рекурсии на конкретных примерах. Способствовать приобретению опыта применения рекурсивных функции при составлении программ.

***Воспитательные:*** Создать условия для воспитания познавательных потребностей, интерес к предмету. Познавательных интересов, интеллектуальных и творческих способностей в информационной деятельности, в профильных областях и с применением специализированных профессиональных инструментов. Содействовать воспитанию информационной культуры учащихся.

**Оборудование:** Компьютерный класс с операционной системой Windows 10, мультимедийный проектор, видео и аудио фрагменты, презентация Microsoft Office Power Point по теме урока, видео и аудио фрагменты, раздаточный материал, компьютерный тест "Язык программирования Pascal. Рекурсия" в программе Тест Полякова К.Ю., приложение Pascal ABC.

**Тип урока**: урок изучения и первичного закрепления новых знаний.

# Структура урока:

1. Организация и мотивация учебной деятельности обучающихся.
2. Актуализация опорных знаний.
3. Целеполагание.
4. Изучение нового материала.
5. Закрепление нового материала. Самостоятельная работа учащихся в группах.
6. Работа в приложении Pascal ABC.
7. Промежуточная диагностика усвоения материала. Компьютерный тест.
8. Домашнее задание. Подведение итогов урока. Рефлексия.

# Ход урока

1. **Организационный момент (мотивационная беседа).**

*Учитель*: Здравствуйте, ребята! Начнем наш урок. Я надеюсь, что наше сотрудничество будет плодотворным.

# Актуализация знаний (создание ситуации успеха)

Учитель: что такое подпрограмма? (вспомогательный алгоритм)

Ученик: подпрограмма - это специальным образом оформленный алгоритм, который может многократно использоваться при решении более общей задачи.

Учитель: Какие виды подпрограмм существует в языке Паскаль?

Ученик: В языке Паскаль существует два вида подпрограмм: процедура (PROCEDURE) и функция ( FUNCTION ).

Учитель: В каком случае удобно использовать в программе процедуру?

Ученик: Процедуры используются в случаях, когда в подпрограмме необходимо получить несколько результатов.

Учитель: В каком случае удобно использовать в программе функцию? Ученик: Когда в подпрограмме необходимо получить единственный результат.

Учитель: Как происходит вызов подпрограммы?

Ученик: Вызов подпрограммы происходит при каждом употреблении ее имени в основной программе. При вызове подпрограммы устанавливается взаимно однозначное соответствие между фактическими и формальными параметрами, затем начинают выполняться команды, заданные в ней. После выполнения подпрограммы управление передается следующему оператору основной программы.

Учитель: Процедура или функция может содержать вызов других процедур или функций. А может ли процедура вызвать саму себя?

Ученик: *Возможные ответы учащихся:*

-нет, это замкнутый круг, это приведет к зацикливанию программы.

-да, при правильной организации вызова.

Учитель: Это возможно. Никакого парадокса здесь нет – компьютер лишь последовательно выполняет встретившиеся ему в программе команды и, если встречается вызов процедуры, просто начинает выполнять эту процедуру. Без разницы, какая процедура дала команду это делать. Собственно в этом и заключается сущность рекурсии.

# Целеполагание

Рекурсией называется ситуация, когда подпрограмма вызывает сама себя. Тема урока: "РЕКУРСИЯ"

Запишите тему урока. "РЕКУРСИЯ" (RECURCIО - возвращение)

Исходя из этого, что это первое знакомство с рекурсией, сформулируйте цель нашего урока.

*Возможные ответы учащихся:*

Продолжить изучение подпрограмм. Узнать, что такое рекурсия, какими свойствами она обладает, как выполняется рекурсивный алгоритм.

Цели на слайде:

* + Познакомится с понятием рекурсивного объекта и рекурсивного определения.
  + Познакомиться с рекурсивными алгоритмами.
  + Рассмотреть программы с использованием рекурсивных подпрограмм.

# Изучение нового материала.

Что такое рекурсивный объект и каковы его свойства?

В жизни на не раз приходилось сталкиваться с рекурсией.

## Прием рекурсии в живописи.

Гравюра голландского художника Мориса Эшера «Рисующие руки» - одна из лучших иллюстраций понятия рекурсии.

Существует специальный раздел графики – фрактальная графика. В основе которой рекурсивное изображение одного и того же узора в самом себе.

## Прием рекурсии в архитектуре.

Сооружения с элементами фрактала "Треугольник Серпинского" - это Эйфелева Башня в Париже;

Исторический музей, Москва;

Фрактальность современных архитектурных форм.

## Рекурсия в математике:

Идеи рекурсии известны людям издавна. Рекурсивные определения как мощный аналитический аппарат используются во многих областях науки, особенно в математике.

1. Арифметическая прогрессия: а)а1=а0;

б) аn=аn-1+d.

1. Геометрическая прогрессия: а) а1=а0;

б) аn=а n-1\*q.

3) Факториал an=n! n!=1\*2\*3\*4\*5\*б\*...\*n. а)а1=1;

б) аn=n\*аn-1.

1. Числа Фибоначчи. x1=x2=1

xn=xn-1+xn-2 при n > 2

Каждый элемент ряда Фибоначчи является суммой двух предшествующих элементов, т.е. 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,…

Мы рассмотрели примеры рекурсивных объектов. **Рекурсивным** называется любой объект, который частично определяется через себя. Т.е. объект определяется в терминах более простого случая этого же объекта.

## Свойства рекурсивных объектов.

* + Простота построения;
  + Несхожесть конечного результата с начальными данными;
  + Внутреннее самоподобие.

## Рекурсия в программировании

**Рекурсия** — это такой способ организации вспомогательного алгоритма (подпрограммы), при котором эта подпрограмма вызывает *сама себя*.

В языке программирования Pascal рекурсивностью могут обладать как функции, так и процедуры.

Рекурсивная функция (или процедура) обязательно должна содержать в себе **условие окончания рекурсивности или граничное условие**, чтобы не вызвать зацикливания программы. Потому что бесконечный вызов рекурсии приводит к переполнению стека и возникновению ошибки времени исполнения.

Большая часть всех шуток о рекурсии касается бесконечной рекурсии, в которой нет условия выхода. *«Чтобы понять рекурсию, нужно сначала понять рекурсию». (цитата на слайде).*

## Примеры рекурсивных алгоритмов.

Общая форма записи:

**Procedure Rec(<список формальных параметров>); Begin**

**…**

**If <проверка условия> Then Rec(n-1);**

**…**

**End;**

|  |  |
| --- | --- |
| Пример рекурсивной процедуры: | Пример рекурсивной функции: |
| procedure Rec(a: integer); begin  if a>0 then Rec(a-1); writeln(a); end; | function Rec(n: integer): integer; begin  if n>0 then Rec(n-1);  writeln(n); end; |

Типичная конструкция рекурсивной процедуры имеет вид:

**Procedure Rec (t: Integer) ;**

**Begin**

**действия на входе в рекурсию; If проверка условия**

**Then Rec (t+1) ;**

**действия на выходе из рекурсии; End;**

**Трассировочная таблица**

|  |  |  |
| --- | --- | --- |
| **i** | **i>1** | **Rec(i-1)** |
| 5 | Да | **Rec** (4) |
| 4 | Да | **Rec** (3) |
| 3 | Да | **Rec** (2) |
| 2 | да | **Rec** (1) |
| 1 | нет | - |

**Задание:**

Пример рекурсивной процедуры:

**Program n1;**

**procedure Rec(i: integer); begin**

**writeln(i);**

**if i>1 then Rec(i-1);**

**end; begin**

**Rec(5);**

**end.**

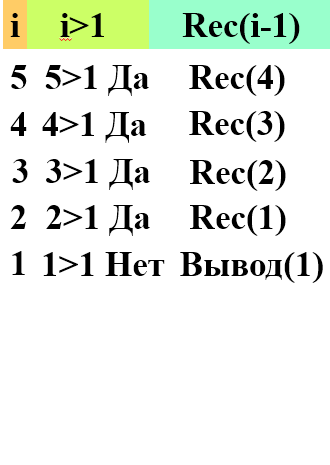
процедура с выполнением действий **на рекурсивном спуске *(выводит 5,4,3,2,1).***

Количество одновременно выполняемых процедур называют ***глубиной рекурсии*.**

А сейчас поработаем в группах. Исследуйте, что получится при вызове описанных на карточках процедур.

Карточка №2: Исследуйте, что получится при вызове описанной ниже процедуры Rec(5).

**Program n2;**

**procedure Rec(i: integer); begin**

**if i>1 then Rec(i-1);**

**writeln(i);**

**end;**

**begin**

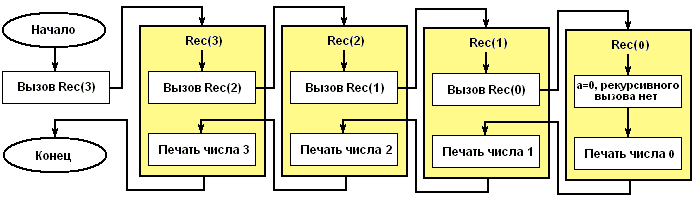
**Rec(5);**

**end.**

**процедура с выполнением действий на**

**рекурсивном подъеме *(выводит 1,2,3,4,5)***

***Учитель: (на слайде) пояснение.***

******

# Самостоятельная работа учащихся в группах.

## Рекурсивные алгоритмы на ЕГЭ

**Задание 11** - Умение исполнить рекурсивный алгоритм

Способ решения: последовательное выполнение операций от начального определения до определения с введенным в алгоритм значением

**Задание:** Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:

F(1) = 1

F(n) = F(n–1) + n, при n >1

Чему равно значение функции F(5)? В ответе запишите только натуральное число. Пояснение. Последовательно находим:

F(2) = F(1) + 2 = 3,

F(3) = F(2) + 3 = 6,

F(4) = F(3) + 4 = 10,

F(5) = F(4) +5 = 15.

Ответ: 15

## Работа в приложении Pascal ABC.

Классический пример, без которого не обходятся ни в одном рассказе о рекурсии, — определение факториала. С одной стороны, факториал определяется так: *n*!=1\*2\*3\*...\**n*, где n∈ Ν. Факториа́л числа n это— произведение всех [натуральных](http://ru.math.wikia.com/wiki/%D0%9D%D0%B0%D1%82%D1%83%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) чисел до n включительно. По определению

𝐧! = 𝐧 ∗ (𝐧 − 𝟏)!, при 𝐧 > 𝟏

𝟏, при 𝐧 <= 𝟏

*Граничным условием в данном случае является n<=1.*

Реализуем приведённые выше рекурсивное определение факториала в виде функции на языке **Pascal.**

Работаем в группах. Дифференцированные задания: На карточках предложены программы.

Для первой группы: в программе имеются пропуски. Вам необходимо дописать пропущенные операторы ***(граничное условие и вызов рекурсивной функции).***

Для второй группы: в программе допущены ошибки и имеются пропуски. Вам необходимо найти ошибки и исправить.

## Карточка 1 (уровень сложности 1)

**Program Factorial;**

**var n:integer;**

**function fac ( n : integer): integer; begin**

**if … then fac := 1**

**else fac := …**

**end; begin**

**write('n = '); readln(n);**

**writeln(n,'! = ', fac(n)); end.**

## Программа без ошибок:

**Program Factorial;**

**var n:integer;**

**function fac ( n : integer): integer; begin**

**if n <= 1 then fac := 1**

**else fac := n \* fac ( n -1 );**

**end; begin**

**write('n = '); readln(n);**

**writeln(n,'! = ', fac(n)); end.**

1. **Работа в приложении Pascal ABC.**

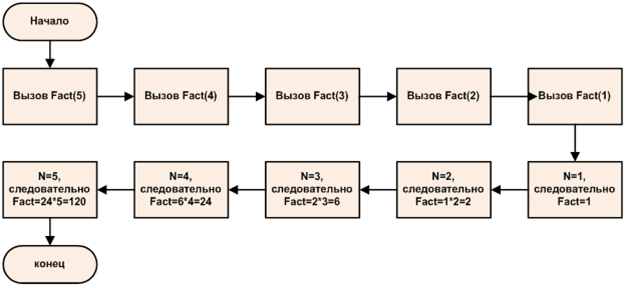
Откройте приложение **Pascal ABC.** Откройте программу **Factorial,** в соответствии с номером карточки, отредактируйте программу и исполните для n=1, 5, 10.

Примеры: n=1 1!=1

n=5 5!=120

n=10 10!= 3628800

Рассмотрим выполнение рекурсивной функции при n=5, т.е. вычислим 5! Блок-схема:



Рассмотрим последовательность вызовов этой функции для n=5. (Слайд 25).

1 вызов (n=5) у:=F(5), так как n<>1, то управление передается на ветку иначе и функция F:=5\*F(4).

2 вызов (n=4) n<>1, то F:=4\*F(3). З вызов (n=З) n<>1, то F:=3\*F(2).

4 вызов (n=2) n<>1, то F:=2\*F(1)=2\*1=2..

Возвращаемся назад поднимаясь вверх по цепочке рекурсивных вызовов. 3 вызов (n=3), , то F:=3\*F(2)=З\*2=6.

2 вызов (n=4) , то F:=4\*F(3)=4\*6=24.

1 вызов (n=5), то F:=5\*F(4)=5\*24=120.

Значение 120 присваивается переменной и выводится на экран. Определите глубину рекурсии.

Ответ: 5

**Вывод:** Программы, в которых используются рекурсивные подпрограммы, отличаются простотой, наглядностью и компактностью текста.

Однако за эту простоту приходится расплачиваться неэкономным пользованием оперативной памяти, кроме того рекурсивный вызов подпрограммы иногда существенно замедляет работу программы.

При исполнении на компьютере рекурсии, содержащей такой бесконечный вызов приводит к переполнению стека и возникновению ошибки времени исполнения.

1. **Промежуточная диагностика усвоения материала. Компьютерный тест.** Компьютерный тест "Язык программирования Pascal. Рекурсия" в программе Тест Полякова К.Ю.

# IX. Домашнее задание. Подведение итогов урока. Рефлексия.

# Подведение итогов

Проверка и обобщение полученных знаний проводится в форме беседы, сопровождающейся краткими записями основных моментов в тетрадях. Вопросы для проведения беседы:

1. Что такое рекурсия?
2. Что такое рекурсивный объект?
3. Свойства рекурсивного объекта?
4. Приведите примеры рекурсивного определения в математике.
5. Что называется *глубиной рекурсии?*
6. Как выполняется рекурсивный алгоритм?
7. Что такое зацикливание и как его избежать? (условие окончания рекурсивности)

# Рефлексия

Оцените, насколько данная тема показалась вам сложной? Оцените, насколько данная тема показалась вам интересной? Оцените, насколько данный урок был для вас информативным? Оцените, насколько данный урок был для вас полезным?

Оцените, как вы поняли данную тему?

Я рада, что после первой встречи с рекурсией вы настроены весьма оптимистично, в отличие от героев мультфильма.

*Фрагмент видео.*

# Список использованных источников:

1. Угринович Н.Д. Информатика и ИКТ: учебник для 11 класса. М.:БИНОМ. Лаборатория знаний, 2012
2. Семакин И.Г., Хеннер Е.К. Задачник-практикум в 2 т.: Том 1. М.: БИНОМ. Лаборатория знаний, 2012
3. Л. Залогова, М. Плаксин и др. Задачник-практикум, М.:БИНОМ. Лаборатория знаний, 2013
4. В.Б. Попов. Turbo Pascal для школьников, М. Финансы и сиатистика, 2002
5. Д.М. Златопольский. Я иду на урок информатики, М. Первое сентября, 2012
6. Н. Культин. Turbo Pascal в задачах и примерах, СПб, БХВ, 2008
7. А.И. Гусева. Учимся информатике, М. Диалог-МИФИ, 2010
8. Е.В. Андреева. Программирование – это так просто, М. МЦНМО, 2009
9. В.Ю. Гуревич. Сборник задач по основам информатики, Минск, 2011 10.Л.М. Поддубная, В.Ф. Шаньгин. Мне нравится Паскаль. М. Радио, 2012 11.С.М. Окулов. Основы программирования, М. Юнимедиастайл, 2002 12.Ю.А. Аляев. Практикум по программированию. М, Финансы и статистика,

2004

13.Д. Гуденко, Д. Петроченко. Сборник задач по программированию, СПб, ПИТЕР, 2003

14.ФЦИОР [http://www.fcior.edu.ru](http://www.fcior.edu.ru/) 15.ЕК ЦОР [http://school-collection.edu.ru](http://school-collection.edu.ru/)