**Конспект урока по информатике для 10 класса по теме «Циклы на языке Паскаль. Цикл с параметром»**

**Цель урока:**Изучить понятие цикл в Паскаль, виды циклов, подробнее изучить цикл с параметром.

**Задачи урока:**

**образовательная:** познакомить учащихся с применением циклов в языке Паскаль.

**развивающая:**развитие логического мышления, понятийного мышления, познавательного интереса.

**воспитательная:** воспитание аккуратности, дисциплинированности, самостоятельности, интереса к предмету.

**Тип урока**: урок объяснения нового материала

**Используемые на уроке методы работы**: фронтальная, индивидуально-групповая.

**Оборудование урока:**

1.компьютеры;

2.демонстрация презентации PowerPoint «Циклы в Паскаль».

**План урока:**

1.Организационная часть (2 мин)

2.Актуализация знаний (3 мин)

3.Изучение нового материала (30 мин)

4.Закрепление нового материала (5 мин)

5.Подведение итогов урока (3 мин)

6.Домашнее задание (2 мин)

**Ход урока**

1. **Организационная часть.**

Сообщение темы и цели урока, плана работы на уроке.

1. **Актуализация знаний.**

Учитель задает вопросы по пройденному материалу, а также проверяет задачи заданные на дом, для самостоятельной работы:

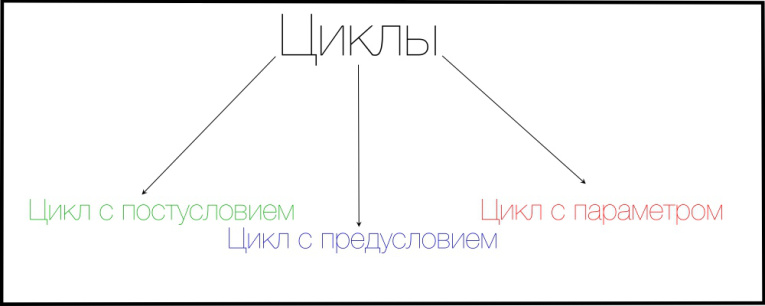
* + Циклические алгоритмы используются для…
  + Арифметический цикл отличается от логического тем, что…
  + Вспомогательный алгоритм – это …
  + Главный алгоритм - это… за каждый правильный ответ-оценка.

1. **Изучение нового материала**.

Учитель для представления нового материала использует презентацию по теме «Циклы в Паскаль».

**ЦИКЛЫ В ПАСКАЛЬ**

Существует три вида алгоритмов: линейный, разветвляющийся и циклический. Мы с вами уже знаем, как реализовывать [линейные](https://infourok.ru/go.html?href=http%3A%2F%2Flearnpascal.ru%2Fvvedenie-v-paskal%2Fstruktura-programmy.html) и [разветвляющиеся](https://infourok.ru/go.html?href=http%3A%2F%2Flearnpascal.ru%2Foperator-if%2Fblok-shema-if.html) алгоритмы на Паскале. Приступим к изучению последнего типа алгоритмов.  
В языке Pascal, как и в большинстве языков программирования, существует три типа циклических конструкций.



Любой цикл состоит из тела и заголовка. Тело цикла — это набор повторяющихся операторов, а условие — это логическое выражение, в зависимости от результата которого и происходит повторное выполнение цикла.

Возьмем одну задачу, которую будем решать, используя различные виды циклов.

**While, или цикл с предусловием (пока истинно условие цикла, повторяется выполнение тела цикла)**

Как вы, наверное, уже поняли из названия, **while** — это цикл, в котором условие стоит перед телом. Причем тело цикла выполняется тогда и только тогда, когда условие **истинно**; как только условие становится **неправдой**, выполнение цикла прекращается.

While имеет формат:

**while** < условие> **do**<тело цикла>;  {Пока .. делай ..}

Данный цикл подходит только для одного оператора, если же вы хотите использовать несколько операторов в своем коде, вам следует заключить их в операторные скобки — **begin**и **end;** . Решение задачи.

**Пример 1. Вывести все цифры от 1 до числа, введенного с клавиатуры.**

**program с1;**

**var**

**i, N: integer; { объявляем переменные }**

**begin**

**i := 1; { Присваиваем i значение 1 }**

**readln(N); { Считываем последнее число }**

**while i <= N do {Как только i станет больше N, цикл прекратится }**

**begin**

**write(i); {Выводим i}**

**Inc(i); {увеличиваем i на один.}**

**end; { закрываем скобки }**

**end.**

**Repeat, или цикл с постусловием**

**Repeat**  — полная противоположность **while**. **Repeat** — это цикл, в котором условие стоит после тела Причем оно выполняется тогда и только тогда, когда результат условия **неправда**;как только логическое выражение становится , выполнение цикла прекращается.

Repeat имеет формат:

**repeat**{ повторяй … }  
< тело цикла>;  
**until** {до…} <условие>

**Begin** и **end** не требуются.

**Пример 2. Вывести все цифры от 1 до числа, введенного с клавиатуры.**

Решение задачи.

**program с2;**

**var i, N: integer; { объявляем переменные }**

**begin**

**i := 1; { Присваиваем i значение 1 }**

**readln(N); { Считываем последнее число }**

**repeat {после repeat не требуется begin и end }**

**write(i); {Выводим i}**

**Inc(i); {увеличиваем i на один.}**

**until i = N + 1;**

**end.**

**For, или цикл с параметром**

**For** — это  цикл, в котором тело выполняется заданное количество раз. При выполнении оператора for сначала вычисляется значение выражения начальное\_значение, затем вычисляется значение выражения конечное\_значение, далее управляющая переменная цикла последовательно пробегает все значения от начального до конечного. В том случае, когда начальное значение оказывается больше конечного значения, тело цикла не будет выполняться вовсе. Начальное и конечное значения остаются неизменными в ходе выполнения всего цикла fог. Параметр цикла i, если он целого типа, пробегает все значения с приращением 1, и его текущее значение не должно изменяться операторами внутри цикла. Такое изменение не запрещено правилами языка, но его последствия будут непредсказуемы. После завершения цикла параметр i считается неопределенным.

**Существует две формы записи этого цикла:**

**Первая форма**

**for** <счетчик1> := <значение1> **to** <конечное\_значение> **do** <тело цикла>;

После каждой итерации значение <счетчик1> будет увеличиваться на 1.

<значение1> — это начальное значение счетчика. Это может быть переменная или число.  
< конечное\_значение> : как только значение <счетчик1> станет больше <конечное\_значение>, выполнение цикла прекратится.

Если требуется написать несколько операторов в теле цикла, используем**begin** и**end**.

И <счетчик1>, и <конечное\_значение>, и <значение1> —  переменные **целого** типа.

Чаще всего в качестве счетчика используется переменная i.

**Вторая форма**

1. for <счетчик2> := <значение2> downto <конечное\_значение> do <оператор1>;

После каждой итерации значение <счетчик2> будет уменьшатся на 1.

<значение2> — это начальное значение счетчика.  
< конечное\_значение> : как только значение <счетчик2> станет меньше <конечное\_значение>, выполнение цикла прекратится.

**Два важных примечания:**

1. Цикл повторяется, пока значение счетчика лежит в отрезке [значение ; конечное\_значение].
2. Изменять значение счетчика внутри тела**нельзя**!

**Пример 3:**Даны целые числа K и N  (N > 0). Вывести N раз число K.

Организовываем простой цикл от 1 до требуемого числа.

**program z1;**

**var**

**K, N, i: integer;**

**begin**

**read(K, N);**{считываем 2 числа}

**for i := 1 to N do write(K);**{выводим N раз число К на экран}

**end.**

**Пример 4:** Даны два целых числа A и B (A < B). Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.

Так как A < B, то цикл должен будет выводить все числа от А до B. Чтобы сосчитать количество чисел, используем формулу: <конечное\_значение> — <начальное\_значение> + 1.

1. program for2;
2. var
3. A, B, i, count: integer;
4. begin
5. read(A, B);
6. for i := A to B do write(i, ' '); {выписываем числа от меньшего к большему}
7. count := B - A + 1; {считаем количество чисел}
8. writeln;
9. write( 'Количество чисел - ', count);
10. end.

**Пример 5:**Даны два целых числа A и B (A < B). Найти сумму квадратов всех целых чисел от A до B включительно.

В цикле одновременно суммируем квадраты всех чисел. Чтобы высчитать квадрат, используем функцию [Sqr](https://infourok.ru/go.html?href=http%3A%2F%2Flearnpascal.ru%2Fvvedenie-v-paskal%2Fmatematica-v-pasca.html).

**program summa2;**

**var**

**A, B, i, S: integer;**

**begin**

**S:=0;**

**readln(A, B);**

**for i := A to B do S := S + Sqr(i); {складываем все квадраты}**

**writeln ('Сумма квадратов - ', S);**

**end.**

**4**. Закрепление материала

– Программа вычисления суммы с использованием цикла со счетчиком дана далее в двух вариантах (ва риант с to и вариант с downto).

program summa3;

const

n = 100;

vaг

i : Integeг;

sum: Real;

begin

sum := 0;

for i := 1 to n do {При первом выполнении цикла i равняется 1, к sum добавляется 1, затем i = 2, к sum добавляется 1/2}

sum := sum + 1 / i; {Затем i = 3, к sum добавляется 1/3, и так продолжается до i = n}

WгiteLn('cуммa ‘, n,' элементов = ', sum:10:5);

end.

Второй вариант:

program summa4;

n = 100;

vaг

i : Integeг;

sum: Real;

begin

sum := 0;

foг i := n downto 1 do

sum := sum + 1 / i;

WriteLn (‘Сумма ’, n, ‘элементов = ’, sum:10:5);

end.

– Самостоятельно разберите работу циклов for в обоих случаях.

– Какую разновидность цикла лучше выбрать в каждом конкретном случае?

Использовать цикл for в том случае, когда точно знаете, сколько раз должно быть выполнено тело цикла. В противном случае обратитесь к другим циклам: используйте цикл repeat, если необходимо, чтобы тело цикла выполнялось по крайней мере один раз, используйте цикл while, если необходимо, чтобы проверка была произведена прежде, чем будет выполняться тело цикла.

**5**. – Итак, мы познакомились с циклами, важнейшей составной частью большинства программ. Мы выяснили, что циклическое действие, которое выполняется, будет конечным, если существует условие, при выполнении (или невыполнении) которого цикл должен завершиться. Ограничение может быть и по числу повторений цикла. Цикл в программе при каждом новом повторении выполняет действие, которое хотя бы немного отличается о предыдущего. Завершим мы тему “Циклы” решением задач на следующем уроке

**6. Домашнее Задание:** В каждой задаче вначале нужно придумать алгоритм, а затем написать программу.

1)Даны целые числа a,b,k, вывести на экран k раз произведение чисел a и b

2)Выведите на экран сумму кубов чисел от 5 до 11