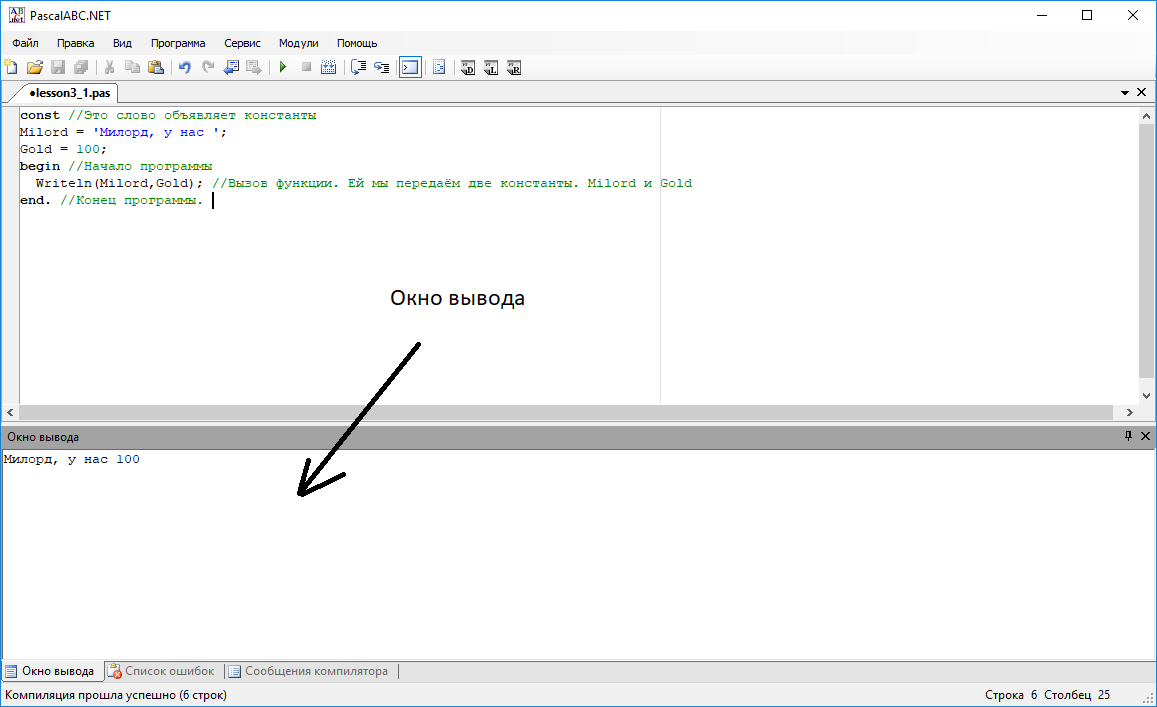
Урок 3. Построение программы. Функции.

В предыдущих уроках мы проводили изменение строковых и числовых констант в готовой программе, которая выводила нам сообщение о количестве золота и других ресурсов. В этом уроке мы попробует построить программу с нуля. В этом нам помогут **функции**, представляющие уже написанные кем-то части программы, выполняющие те или иные действия. В частности, существует функция **WriteLn**, которая позволяет написать любой текст в окно вывода. Она одинаково успешно работает как с числами, так и со строками. Чтобы вызвать функцию, необходимо ввести её имя, а в скобочках константу, который ей передаётся. Если констант несколько, то они разделяются запятыми. Например, вызов

Writeln(Gold, ' золота');

Выведет фразу «100 золота», если константа Gold равна 100.

Откройте файл «lesson3\_1.pas» и нажмите кнопку «Запустить». В результате, в окне вывода вы увидите сообщение «Милорд, у вас 100».



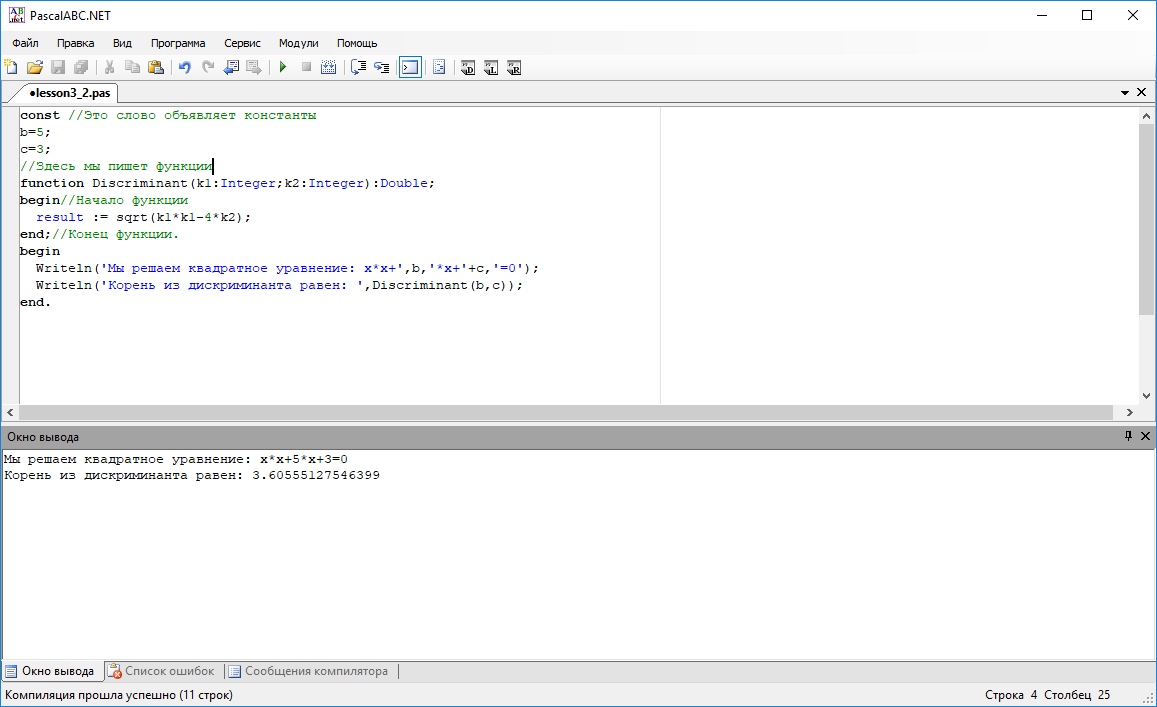
Обратите внимание, что у нас появилось два новых слова.

**Begin -**означает начало выполнения программы.

**end**. **-**означает конец выполнения программы.

Между ними располагается сама программа , которая состоит из одного вызова *функции*. Для успешного выполнения урока вызовите *функцию WriteLn* несколько раз, так, чтобы программа, как и в предыдущих уроках сообщала нам количество золота, дерева, железа, твёрдых ресурсов, а также их стоимость. Также добавьте в первый вызов функции строчку, чтобы в итоге выводилось полное сообщение «Милорд, у нас 100 золота».

Также мы можем писать *функции* сами. Запустите вторую часть урока 3, которая располагается в файле «lesson3\_2.pas».



Сначала с помощью слова **function** мы объявляем, что это функция. Далее идёт имя этой функции. Далее в скобочках идут параметры, которые мы передаём функции. В прошлом уроке мы познакомились с целочисленным типом константы. Чтобы объявить ,что k1 – это целое число , необходимо записать выражение:

k1:Integer;

Если бы параметр k1 был бы строкой, то мы бы записали

k1:String;

После скобочек пишется тип результата. Это новый для нас тип, означающий десятичную дробь. Этот тип обозначается как **Double.** Его применение связанно с тем, что наша функция возвращает корень из дискриминанта. Сама функция записывается как:

result := sqrt(k1\*k1-4\*k2);

Слово result обозначает результат функции. := Означает присваивание, то что слева, то есть результат становиться равным тому, что справа. Обратите внимание, что наша функция использует результат вычисления другой функции, sqrt, которая вычисляет квадратный корень из заданного нами выражения.

В основной программе мы выводим результат работы этой функции. Если дискриминант меньше нуля, то результатом будет «не число».

Для завершения урока напишите функции Solve1 и Solve2 , которые бы полностью вычисляли первый и второй корень квадратного уравнения. Если корень один, то обе эти функции должны возвращать одинаковый результат.

С помощью функции *WriteLn*  выведите оба корня уравнения.