**Занятие № 18. Управляющие операторы.**

**Условные операторы. Операторы повторения и перехода.**

Программам, получающим от внешнего мира какие-то данные, например, поступающие с клавиатуры или из параметров командной строки, часто приходится анализировать полученную информацию и выполнять различные действия в зависимости от истинности или ложности некоторых условий. Конструкция, существующая в языке Java для реализации такого поведения, называется условным оператором.

Прежде, чем мы перейдем к изучению условного оператора, рассмотрим вопрос о том, что такое оператор. Оператор это единица действия в программе. Существует несколько разновидностей операторов. Одну из них мы уже знаем это оператор, построенный из выражения.

Любое выражение можно превратить в оператор, поставив после него точку с запятой. Надо заметить, что в Java, как и в С++, точка с запятой не разделяет операторы она превращает выражение в оператор. В частности, допустим пустой оператор, состоящий только из точки с запятой. Такой оператор ничего не делает и может встречаться везде, где структура языка требует наличия оператора.

Например, присваивание в Java это не оператор, это выражение; однако его можно превратить в оператор, поставив после соответствующего выражения точку с запятой.

Следующая разновидность операторов - так называемый составной оператор. Это последовательность операторов, заключенная в фигурные скобки. Выполнение составного оператора состоит в последовательном выполнении входящих в него операторов в том порядке, в котором они встречаются в составном операторе. Составной оператор нужен тогда, когда есть необходимость поставить несколько операторов в то место программы, где структура языка допускает только один оператор.

Синтаксис, т.е. форма записи, условного оператора в языке Java следующий: if (условие) оператор1 else оператор2.

В языке Java условием может быть только выражение, имеющее логический результат (например, применение операции сравнения или логической операции).

Смысл условного оператора состоит в том, что сначала вычисляется условие; если оно истинно, выполняется оператор 1 (он называется положительной альтернативой), иначе выполняется оператор2 (отрицательная альтернатива). Ключевое слово else и отрицательная альтернатива могут отсутствовать - в этом случае при ложном условии не выполняется никаких действий.

Синтаксис языка не позволяет записывать несколько операторов ни в ветви then, ни в ветви else. При необходимости составляется блок операторов в фигурных скобках. Соглашения "Code Conventions" рекомендуют всегда использовать фигурные скобки и размещать оператор на нескольких строках с отступами, как в следующем примере:

*if (а < х)*

*{*

*х = а + b;*

*}*

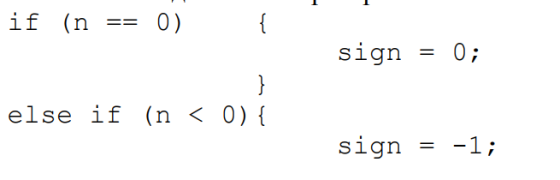
*else*

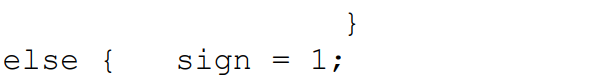
*{*

*х = а — b;*

*}*

Очень часто одним из операторов является снова условный оператор, например:





При этом может возникнуть такая ситуация ("dangling else"):

*int ind = 5, х = 100;*

*if (ind >= 10) if (ind <= 20) x = 0; else x = 1;*

Сохранит переменная x значение 0 или станет равной 1? Здесь необходимо волевое решение, и общее для большинства языков, в том числе и Java,. правило таково: ветвь else относится к ближайшему слева услдвию if, не имеющему своей ветви else. Поэтому в нашем примере переменная x останется равной 0.

Изменить этот порядок можно с помощью блока:

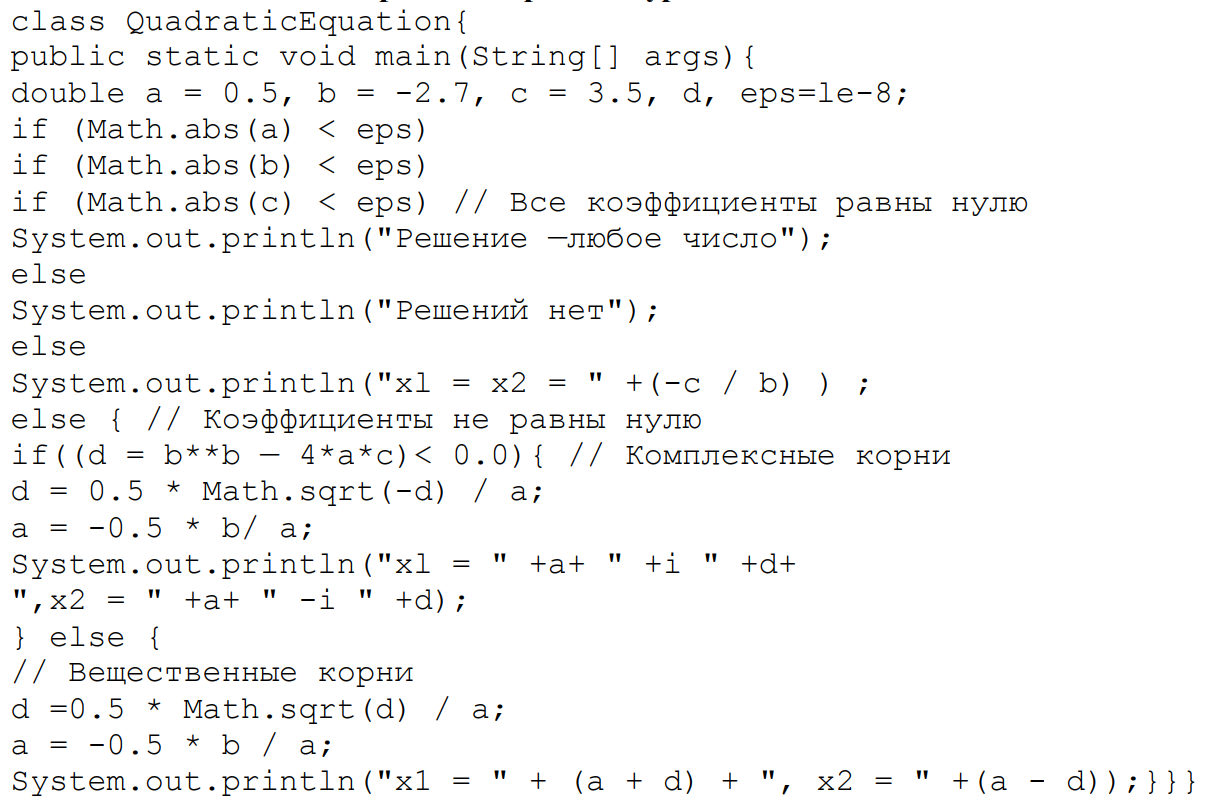
if (ind > 10) {if (ind < 20) x = 0; else x = 1;}

Вообще не стоит увлекаться сложными вложенными условными операторами. Проверки условий занимают много времени. По возможности лучше использовать логические операции, например, в нашем примере можно написать

if (ind >= 10 && ind <= 20) х = 0; else х = 1;

В приведенном ниже листинге вычисляются корни квадратного уравнения ах 2 + bx + с = 0 для любых коэффициентов, в том числе и нулевых.

Листинг Вычисление корней квадратного уравнения



В этой программе использованы методы вычисления модуля abs() и квадратного корня sqrt о вещественного числа из встроенного в Java API класса Math. Поскольку все вычисления с вещественными числами производятся приближенно, мы считаем, что коэффициент уравнения равен нулю, если его модуль меньше 0,00000001. Обратите внимание на то, как в методе println о используется сцепление строк, и на то, как операция присваивания при вычислении дискриминанта вложена в логическое выражение.

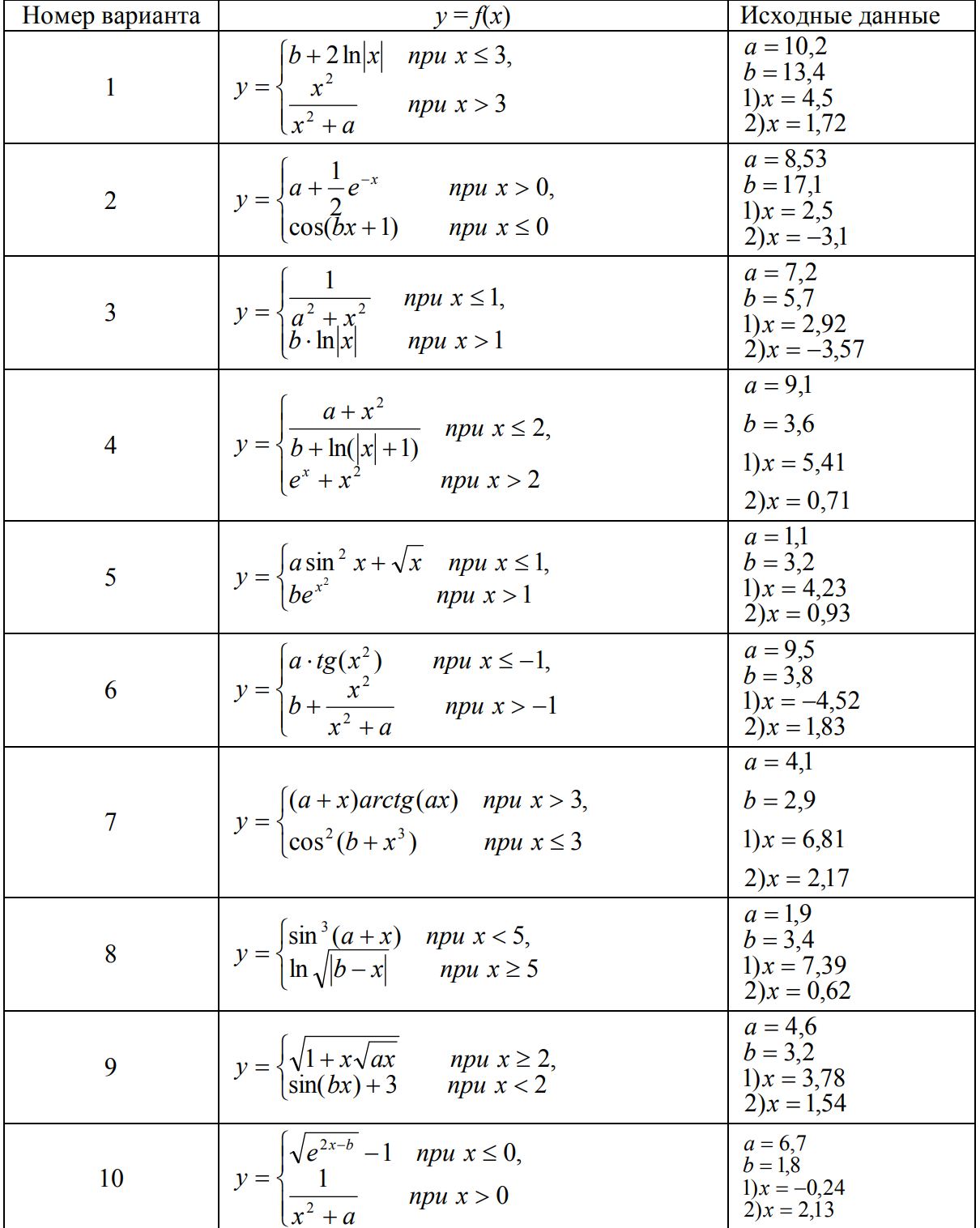
Задание для самостоятельной работы.

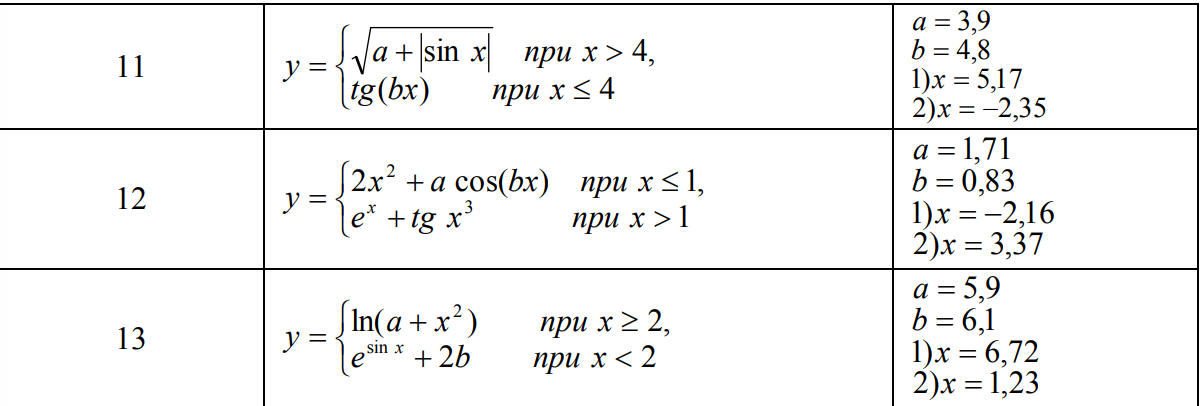
1. Написать программу, вводящую с клавиатуры число и выводящую сообщение «less», если его значение меньше 100, «not less» - иначе.
2. Написать программу, вводящую с клавиатуры - оценку (целое число от 2 до 5) и выводящую эту оценку словом (5 - отлично, 4 - хорошо, 3 - удовлетворительно, 2 - неудовлетворительно).
3. Написать программу, которая по названию дня недели печатает его номер (воскресенье - 0, понедельник - 1, ..., суббота - 6)
4. Успеваемость студентов оценивается по 100-балльной шкале. Оценкам от 91 до 100 соответствует разряд А, от 81 до 90 — разряд В, и т. д., оценкам не более 50 — разряд F. Написать программу, вводящую с клавиатуры оценку по 100-балльной шкале и печатающую соответствующий ей разряд.
5. Индивидуальное задание для бригад.

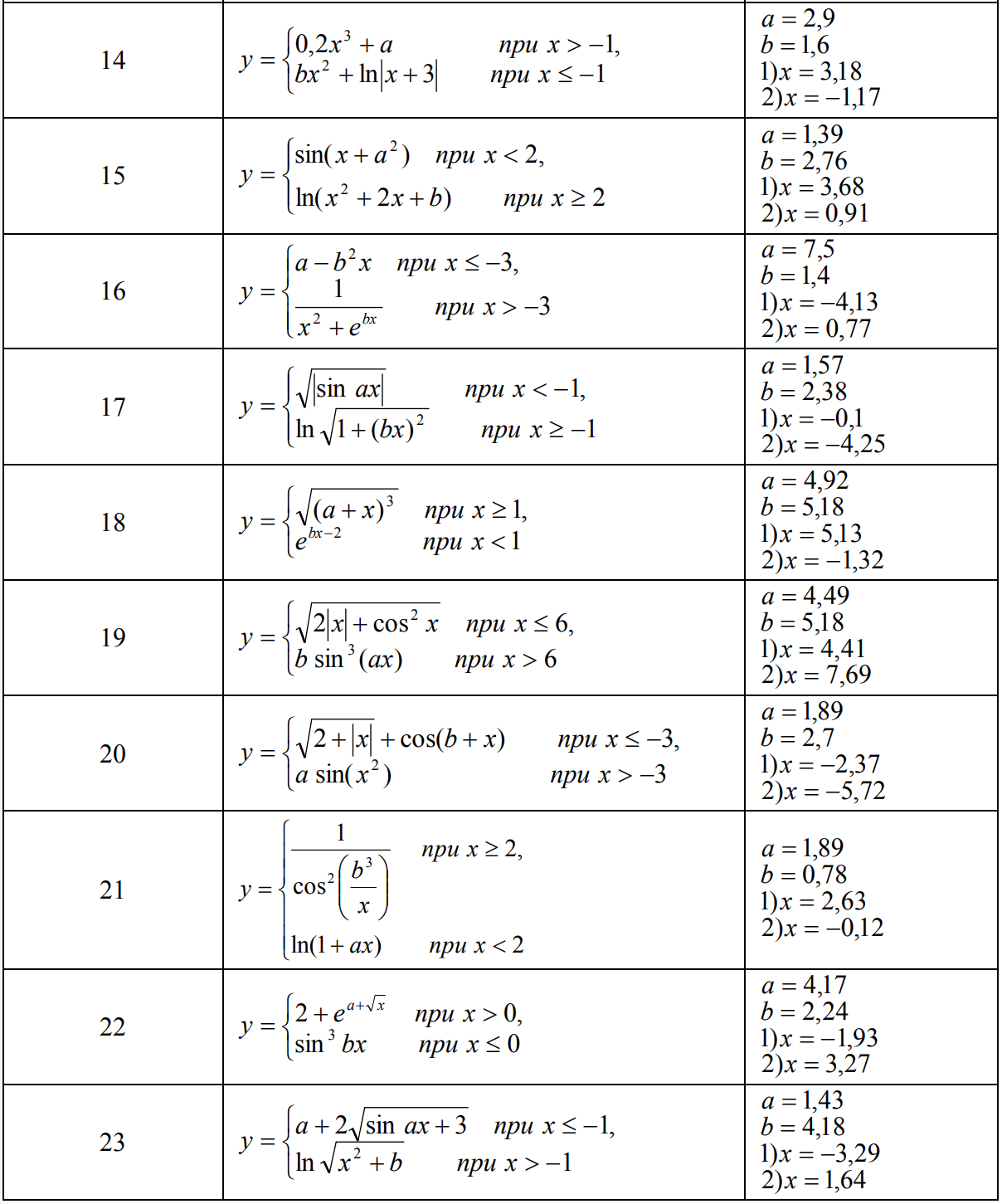
Варианты заданий (номер варианта задания 1 - 30 определяется как увеличенный на 1 остаток от деления по модулю 30 двух последних цифр зачетной книжки (студенческого билета) студента или суммы двух последних цифр зачетных книжек (студенческих билетов) бригады студентов.

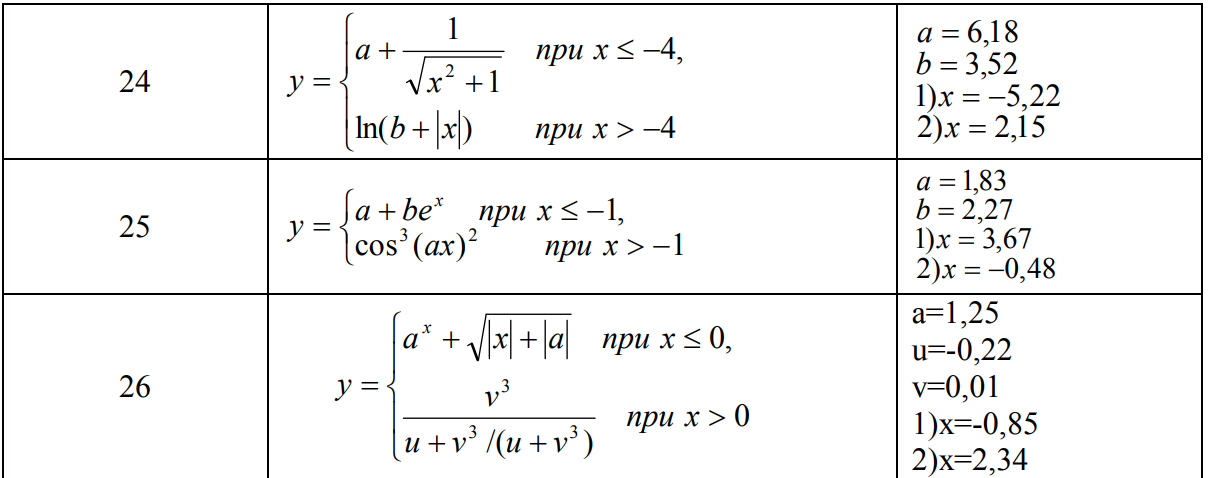
Задание 5.1. Ветвление с двумя альтернативами

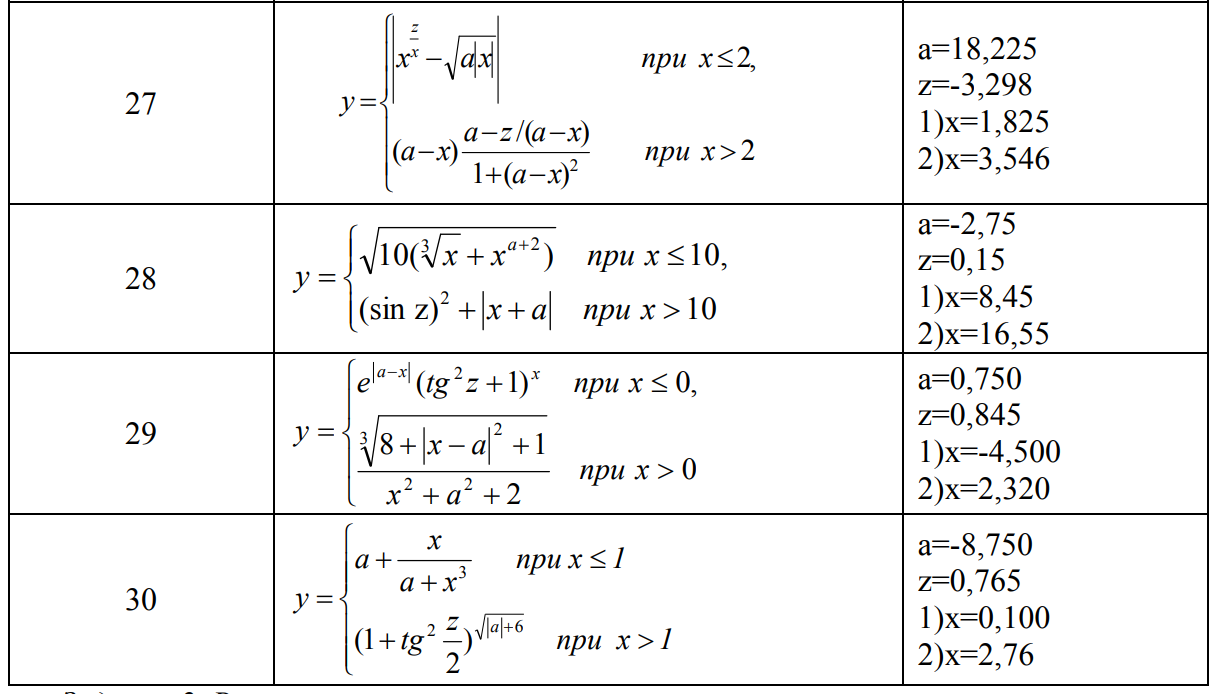
Составить программу для вычисления значений функции y = f(x) при произвольных значениях х. Получить результат работы программы для двух заданных значений х.











*Задание 5.2. Ветвление ветвления*

Составить программу для вычисления значений функции y=f(x) при произвольных значениях x.

