**Task 1-1**

|  |
| --- |
| import Todo(todo) |
|  |  |
|  | |  | | --- | | Data Operacii=plus--+ = | |  | | Minus -- - | |  | | Multiply -- \* | |  | deriving (Show,Eq) |   data Term = IntConstant{ intValue :: Int } -- числовая константа |
|  | | Variable{ varName :: String } -- переменная |
|  | | BinaryTerm{ lhv :: Term, rhv :: Term } -- бинарная операция |
|  | deriving(Show,Eq) |
|  |  |
|  | -- Для бинарных операций необходима не только реализация, но и адекватные |
|  | -- ассоциативность и приоритет |
|  | (|+|) :: Term -> Term -> Term |
|  | (|+|) l r = BinaryTerm l r Plus |
|  |  |
|  | (|-|) :: Term -> Term -> Term |
|  | (|-|) l r = BinaryTerm l r Minus  (|\*|) :: Term -> Term -> Term  (|\*|)l r = BinaryTerm l r Multiply |
|  |  |
|  |  |
|  | -- Заменить переменную `varName` на `replacement` |
|  | -- во всём выражении `expression` |
|  | replaceVar :: String -> Term -> Term -> Term |
|  | replaceVar varName replacement expression = todo |
|  |  |
|  | -- Посчитать значение выражения `Term` |
|  | -- если оно состоит только из констант |
|  | evaluate :: Term -> Term |
|  | evaluate expression = todo |

|  |
| --- |
| module Task1\_2 where |
|  |  |
|  | import Todo(todo) |
|  |  |
|  | -- синус числа (формула Тейлора) |
|  | sin :: Double -> Double |
|  | sin x = todo |
|  |  |
|  | -- косинус числа (формула Тейлора) |
|  | cos :: Double -> Double |
|  | cos x = todo |
|  |  |
|  | -- наибольший общий делитель двух чисел  **gcd :: intergral a => a-> a-> a**  **gcd 0 0 = error “ HOД (0,0) НЕ ОПРЕДЕЛЁН”**  **gcd m n = gcd’ (abs m) (abs n) where**  **gcd’ m n = gcd’ n (rem m n )** |
|  | gcd :: Integer -> Integer -> Integer |
|  | gcd x y = todo |
|  |  |
|  | -- существует ли полный целочисленный квадрат в диапазоне [from, to)? |
|  | doesSquareBetweenExist :: Integer -> Integer -> Bool |
|  | doesSquareBetweenExist from to = todo |
|  |  |
|  | -- является ли дата корректной с учётом количества дней в месяце и |
|  | -- вискокосных годов?  isDateCorrect :: Integer -> Integer -> Integer -> Bool  isDateCorrect day month year = if day >= 1  then case month of  1 -> day <= 31  2 -> if isLeapYear year then day <= 29 else day <=28  3 -> day <= 31  4 -> day <= 30  5 -> day <= 31  6 -> day <= 30  7 -> day <= 31  8 -> day <= 31  9 -> day <= 30  10 -> day <= 31  11 -> day <= 30  12 -> day <= 31  isLeapYear :: Integer -> Bool  isLeapYear year = year `mod` 400 == 0 || (year `mod` 100 /= 0 && year `mod` 4 == 0) |
|  | isDateCorrect :: Integer -> Integer -> Integer -> Bool |
|  | isDateCorrect day month year = todo |
|  |  |
|  | -- возведение числа в степень, duh |
|  | -- готовые функции и плавающую арифметику использовать нельзя |
|  | pow :: Integer -> Integer -> Integer |
|  | pow x y = todo |
|  |  |
|  | -- является ли данное число простым? |
|  | isPrime :: Integer -> Bool |
|  | isPrime x = todo |
|  |  |
|  | type Point2D = (Double, Double) |
|  |  |
|  | -- рассчитайте площадь многоугольника по формуле Гаусса  **data** Point2D **=** Point2D {x **::** **Double**, y **::** **Double**}    getX **::** Point2D **->** **Double**  getX (Point2D x \_) **=** x    getY **::** Point2D **->** **Double**  getY (Point2D \_ y) **=** y    area **::** [**Double**] **->** [**Double**] **->** **Double**  area x y **=**  ((0.5**\***) . **abs** . **sum**) **$** **map** det **$** **zip** pxs pys  **where** xs **=** x  **++** [**head** x]             ys **=** y  **++** [**head** y]             pxs **=** **zip** xs (**drop** 1 xs)             pys **=** **zip** ys (**drop** 1 ys)             det ((x1,x2),(y1,y2)) **=** x1**\***y2 **-** y1**\***x2 |
|  | -- многоугольник задан списком координат |
|  | shapeArea :: [Point2D] -> Double |
|  | shapeArea points = todo |
|  |  |
|  | -- треугольник задан своими координатами. |
|  | -- функция должна вернуть |
|  | -- 0, если он тупоугольный |
|  | -- 1, если он остроугольный |
|  | -- 2, если он прямоугольный |
|  | -- -1, если это не треугольник |
|  | triangleKind :: Point2D -> Point2D -> Point2D -> Integer |
|  | triangleKind a b c = todo |