Тема 6 б)

Контрольные задания

1. Написать функциональный оператор *GM* и процедуру *gm*, которые возвращают геометрическое среднее двух вещественных чисел для чисел одинаковых знаков и модуль произведения для чисел разных знаков. Для проверки рассмотреть целые числа и десятичные дроби, числа одинаковых и разных знаков.

Указание. Геометрическим средним чисел *a* и *b* называется $\sqrt{ab}$.

2. Написать процедуру *gm_pos*, которая возвращает геометрическое среднее двух чисел и принимает только положительные целые числа (добавить декларацию типа **`posint`** для аргументов). Дополнительно задать значения по умолчанию: a=1, b=1. Выполнить проверку работы процедуры и объяснить выдаваемые результаты (числа для проверки: 13 и 17, 13.5 и 17.5, −13 и 17, 13 и −17).

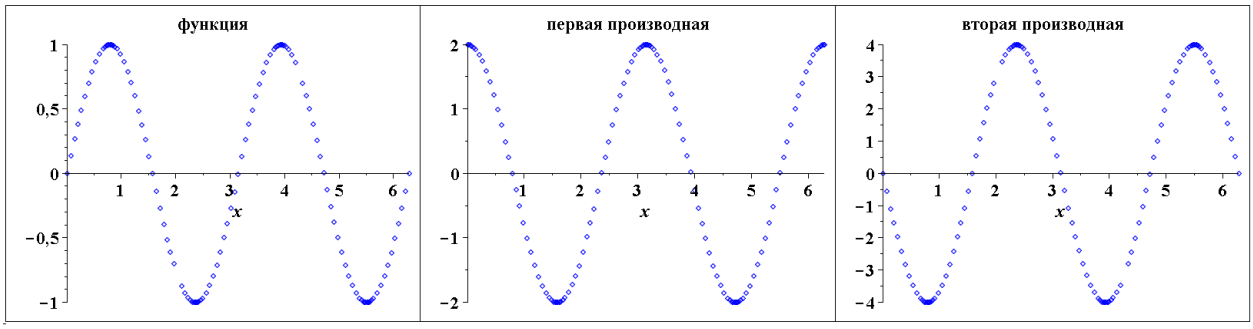
3. Написать процедуру *testSC*, которая проверяет, является ли число одновременно полным квадратом и полным кубом. В случае положительного ответа процедура должна выдавать true, в случае отрицательного ответа – false. Для проверки на полный квадрат использовать функцию **`issqr(n)`**, для проверки на полный куб нужно написать вспомогательную пользовательскую процедуру *IsCub*. Проверить работу процедуры на различных числах.

Указание. Можно посмотреть, как реализована процедура **`issqr`** в Maple (см. в лекциях вывод кода процедуры на экран), и написать похожую процедуру *IsCub*, которая бы определяла, является ли число полным кубом.

Для проверки. Первые несколько чисел, которые являются одновременно полными квадратами и кубами находятся во множестве пересечения полных квадратов и полных кубов.

4. Модифицировать процедуру из предыдущего примера таким образом, чтобы она могла принимать как один, так и несколько аргументов (т.е. тестировать сразу несколько чисел).
   Указание. Использовать зарезервированные имена **`_passed`** и **`_npassed`**.

5. Написать процедуру *PlotDerivatives*, которая выводит график функции, ее первой производной и ее второй производной на трех рисунках, расположенных рядом. Процедура должна содержать два обязательных аргумента: функцию (тип **`algebraic`**) и интервал вывода графика по оси абсцисс (тип **`range(realcons)`**). Остальными аргументами могут быть опции графического вывода, которые должны передаваться непосредственно в графические команды **`plot`**. Вывести подходящий заголовок для каждого графика, например:



Протестировать работу процедуры на командах:

```
> plotDerivatives(arcsin(x),-1..1);
>      plotDerivatives(sin(2*x),0..2*Pi,color=blue,style=point,
   font=[TIMES,BOLD,16]);
```

Указание. Использовать зарезервированное имя **_rest** и подключить пакет **plots** в теле процедуры. Для вывода трех графиков, расположенных рядом, задать массив 1x3 из трех команд **plot** и использовать команду **display** из пакета **plots** для вывода этого массива на экран. Задание массива:

```
A:=Array(1..3,[plot(…),plot(…),plot(…)]);display(A)
```

6. Написать рекурсивную процедуру *myfactorial*, которая вычисляет n! для любого неотрицательного целого n (тип **nonnegint**) без использования знака факториала. При несоответствии типа аргумента процедура должна выводить сообщение о том, что число не является натуральным и возвращать само число.

Указание. Использовать команду **return** в теле процедуры. Рекуррентная формула для вычисления n! имеет вид:

$$n! = \begin{cases} 1, & n = 0; \\ (n-1)! \cdot n, & n \geqslant 1. \end{cases}$$

Для проверки. Использовать команду **n!**

7. Написать процедуру *DigitCount2*, которая вычисляет сумму цифр заданного двузначного натурального числа. Использовать декларацию типа для аргумента (тип **posint**). В случае, если заданное число не является двузначным, организовать аварийный выход из процедуры с сообщением об ошибке (использовать команду **error** с подстановкой значения введенного числа). Проверить работу процедуры.

Указание. Для получения частного и остатка от деления на 10 можно использовать команды **iquo(n,10)** и **irem(n,10)**.

8. Задать функцию $e^x \cdot \sin(x)$. Найти производные этой функции с первой по шестую. Сохранить результат (пять производных) в два разных файла: файл внутреннего формата Maple (расширение **.m**) и обычный файл Maple (расширение **.mw**). Считать полученный m-файл и вывести значение пятой производной. Открыть mw-файл и посмотреть его содержимое.

9. Вычислить приближенные значения производных функции из предыдущего задания при $x = \frac{\pi}{3}$. Используя команду **fprintf**, записать эти значения в текстовый файл в две строки в формате float, значения разделять пробелами (символ перехода на новую строку: **\n)**. Открыть и посмотреть полученный текстовый файл. Считать и вывести на экран строки полученного файла, используя команды **readdata** и **readline**. Сравнить результаты.

Задания на дополнительные баллы

Maplet, сложная процедуоа (задания на бонусные баллы по материалам приложения к лекции 6, а также лекций 6 и 5)

1. С помощью ассистента Maplet Builder или команд Maple создать графическое приложение Maplet, которое предлагает пользователю ввести функцию одной переменной (“Enter a function f(x)”) и по нажатию кнопок выводит график самой функции, а также значения и графики ее первой и второй производной. *Указание*. Для вывода значения производной можно использовать поле TextField (для которого установить свойство editable=false, т.е. нередактируемое текстовое поле) или поле MathML Viewer (для вывода выражения в виде 2D Math). При использовании поля MathML Viewer событие по нажатию кнопки для отображения выражения, введенного в текстовом поле, должно экспортировать это поле в MathML: MathML[Export](TextField1).

2. Написать процедуру *backsub,* которая решает систему линейных алгебраических уравнений Ax=b с верхнетреугольной матрицей A методом обратной подстановки. Метод состоит в следующем. Сначала вычисляется последняя неизвестная $x_n = b_n / a_{nn}$, затем оставшиеся неизвестные вычисляются по правилу:

$$x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{kj} x_j}{a_{kk}} \text{ для } k = n-1,\ n-2,\ ...,1..$$

Процедура должна включать:

- декларацию типов аргументов (Matrix, Vector)
- подключение необходимых пакетов
- проверку типов аргументов и вывод соответствующих пользовательских сообщений об ошибках
- вывод сообщений об ошибках в случаях, если размеры матрицы не соответствуют размерам вектора, матрица не является верхнетреугольной, матрица является вырожденной.

Протестировать работу процедуры на заданном примере и произвольной матрице. Сравнить результат с работой команд **BackwardSubstitute** и **LinearSolve** пакета **LinearAlgebra**.

$$
\begin{aligned}
3x_1 - 2x_2 + x_3 - x_4 &= 8 \\
4x_2 - x_3 + 2x_4 &= -3 \\
2x_3 + 3x_4 &= 11 \\
5x_4 &= 15
\end{aligned}
$$

Затем попробовать ввести не весь набор аргументов, аргументы неверного типа, избыточное количество аргументов, и посмотреть, какие выдаются сообщения об ошибках. В каких случаях выдаются пользовательские сообщения об ошибках?

Указание. Матрицу из примера можно задать командой

```
M:=Matrix(4,[[3],[-2,4],[1,-1,2],[-
   1,2,3,5]],shape=triangular,scan= columns);
```

Произвольная матрица:

```
M:=RandomMatrix(4,outputoptions=[shape=triangular]);
M:=RandomMatrix(4,outputoptions=[shape=triangular,datatype=float
   ]);
```

Для определения, является ли матрица верхнетреугольной, можно привести ее к верхнетреугольной форме и сравнить полученную матрицу с исходной. Команда сравнения матриц на равенство элементов: **`Equal`** из пакета **`LinearAlgebra`**.