Chapter 3

Programming in MuPAD - Declarations

Many mathematicians consider programming as a skill that is completely irrelevant to mathematics. However, when looking at mathematics, we may notice that working in mathematics requires many activities that are not available in visual CAS. For instance, while solving mathematical problems, we have to explore algorithms, use recursive functions, or create complex constructions in 2D or 3D geometry. All these activities consist of many steps and their nature resembles writing a program. These kinds of activities can be practiced if we use a computer program where a programming language is available. This is the point where you will appreciate MuPAD.

Programming in MuPAD is structured very similarly to writing programs in Pascal. However, in MuPAD you can represent mathematical concepts in a more efficient and flexible way. Thus, you can follow the natural way of mathematical thinking with greater ease.

In this chapter you will start learning the basics of MuPAD programming. In the later chapters, we will expand this knowledge.

3.1 Writing Formulae in MuPAD

In the previous chapter, we had tried a few very simple formulae with MuPAD. The first thing that you had perhaps observed is that we cannot omit the multiplication symbol and brackets for functions arguments. For example, in real life we usually write $3x \sin x$ or $\sin x \cos y$, while in MuPAD we have to type in $3*x*sin(x)$ or $\sin(x)*\cos(y)$. It is important to realize that, for example, $3x$ and $3*x$ mean completely different things in MuPAD. In fact, $3*x$ means the multiplication 3 times x, but $3x$ is just a nonsense expression that will produce an error message. This example suggests that we should consider for a moment how we can build formulae in MuPAD so they will mean what we want them to mean.
Up to now, you have seen only a very limited set of MuPAD arithmetical operations: +, −, *, /, and taking to the power ^, for example, x^3 meaning \( x^3 \). MuPAD knows a number of other operations and functions used in mathematics. You can use not only basic arithmetic operations but also functions like \( \tan x \), \( \sin x \), \( \ln x \) and many others. Creating formulae with these operations and functions can sometimes be confusing. For example, how do we write the formula \( x^2y \) in MuPAD's notation, and how will \( x \times y^2 \) be evaluated? What will be calculated first, \( x \times y \) or \( y^2 \)?

Let's learn some general rules.

1. **Brackets are the strongest.** Operations enclosed in brackets are calculated first. This means that in the expression \( 3*(3+9) \), MuPAD will first calculate \( 3+9 \) and then the multiplication. The same is true for \( 5*\sin(x*0.01) \). First, \( x*0.01 \) will be calculated, then \( \sin(x*0.01) \) and finally multiplication by 5.

2. **Next goes “^”**. For example, in the formula \( 2^3*5 \) MuPAD will first calculate \( 2^3 \) (i.e. \( 2^3 \)) and then it will multiply it by 5.

3. **Operations * and / are stronger than + and −.** Thus, in the formula \( 5+3/2 \) or \( 3/2+5 \), the operation \( 3/2 \) will be calculated first and then 5 will be added.

4. **Binary operations not using brackets (like mod, div, etc.) are the weakest.** For example, \( 2+3 \mod 5 \) really means \( (2+3) \mod 5 \) and thus, \( 2+3 \mod 5 = 0 \). Note that “mod” denotes the binary operation modulo.

5. If you are not sure how your formula will work, use brackets.

Sometimes, long and complicated formulae may look very crowded, and it is very hard to read and understand them. For instance try to read a formula like this one: \( 3*x^5-8*x^4+9*x^3-2*x^2+x-5 \). It is difficult to understand, isn’t it?

You should always write code that is easy to read and understand. Let’s make an agreement that will help us to develop clear MuPAD code.

**Rule 1** There are always spaces after keywords and surrounding binary operations.