PHYSICS 210 – Fall 2009

INTRODUCTION TO MAPLE PROGRAMMING

Thursday, October 1

Introduction to Maple Programming

- **NOTE:** Due to time constraints, cannot provide full details; refer to Programming Guides (Introductory / Advanced) for more information (available online in PDF form)

- Places where I'm not "telling the whole truth" will be marked with (*), but I'm trying to ensure that no procedures that you write will be injured by my omission of certain details.

Overview

- **How does one program in Maple?**
  - **Principal unit of Maple usage: Maple statement**
    - Assignment statements
      - a := 2;
      - l := [1, 2, 3, 4];
      - f := x - > x^3;
    - Other statements
      - expand((x + y)^12);
      - plot(sin(x) , x = -2*Pi .. 2*Pi);
  - **Principal unit of Maple programming: Maple procedure**
    - All of the maple commands that we have seen are procedures

Overview

- **How does one program in Maple?**
  - **IMPORTANT:** Can also “program” in Maple using the equivalent of a (basic) bash script
  - **Maple script** → sequence of Maple statements, prepared in
    1. Worksheet
    2. Text file: use **read** command to input
  - Script will often constitute the equivalent of a “main program” in other languages: top level program unit invokes sub-programs (procedures) as necessary

Overview (cont.)

- **What programming paradigms does Maple support?** (Sep 22)
  - Maple provides support for both
    - **FUNCTIONAL** programming
    - **PROCEDURAL (IMPERATIVE)** programming
  - **FUNCTIONAL** programming
    - Every Maple procedure normally returns a value = some Maple expression
    - Other elements of functional programming that we have already see
      - Defining (mathematical) functions using the arrow notation
      - **map** command

Overview (cont.)

- **PROCEDURAL** programming
  - **Recall: Data and Algorithms**
  - **Data**
    - Maple supports **many** different data types
      - » Various types of numbers (integer, rational, floating point …)
      - » General algebraic expressions
      - » Lists
      - » Sets
      - » Arrays
      - » Tables

        .

        .

        .

Overview (cont.)

- **PROCEDURAL** programming
  - **Algorithms**
    - Usual set of “control structures”
      - » **Sequence:** (script / procedure can have an arbitrary number of statements
      - » **Selection: if** statement
      - » **Iteration (looping): for-while-do** statement

Overview (cont.)

- **Syntax and Semantics**
  - **Syntax and Semantics → Language**
    - **Syntax = Grammar:** Rules to combine basic elements into statements
      - Defines valid forms of input
      - Examples:
        - » How do we represent multiplication?
          > a * b;
        - » How do we define a string?
          > “Foo”;
        - » What does the definition of a procedure look like?
          (later)

Overview (cont.)

- **Syntax = Grammar:** Rules to combine basic elements into statements

  – Syntactically invalid input → syntax error

  > - - 2;

  > 1.e-3 ; (needs to be written as **1.0e-3**, **1.00e-3**, etc)

Overview (cont.)

- **Syntax and Semantics**
  - **Syntax and Semantics → Language**
    - **Semantics = Additional information / meaning that syntax cannot captures**
      - Defines what Maple does when a given statement is executed
      - Examples:
        - » Is **a / b / c** equal to **a / (b / c)** or **(a / b) / c** ?
        - » What is the value of **i** after the execution of this loop?

          > for i from 1 to 5 do print(i^2) end do;