CH2120 Introduction to Programming in Fortran

Classes 4-6

# Welcome to Numerical Methods

## Code

**program** mainProgram

**implicit** **none**

**write**(\*,\*) "Welcome to Numerical Methods!"

**end** **program** mainProgram

## Output

Welcome to Numerical Methods!

## Ideas

* Feasibility of the set up
* Keywords/statements: **program**, **end** **program**, **write**
* Fortran is not case sensitive. **program**, **PROGRAM**, **PrOgRaM** are interpreted the same way by the Fortran compiler. Recommended: lower case for keywords, camelCase (e.g. mainProgram) for naming.

# Basic Arithmetic Operations

## Code

**program** mainProgram

**implicit** **none**

**write**(\*,\*) "Welcome to Numerical Methods!"

**write**(\*,\*) 2.1 + 4.5

**write**(\*,\*) 2.1 - 4.5

**write**(\*,\*) 2.1 \* 4.5

**write**(\*,\*) 2.1 / 4.5

**end** **program** mainProgram

## Output

Welcome to Numerical Methods!

6.59999990

-2.40000010

9.44999981

0.466666639

## Ideas

* Basic arithmetic operations
* Results are not exact.

# Variables

## Code

**program** mainProgram

**implicit** **none**

**integer** x, y, a, b, c, d

**write**(\*,\*) "Welcome to Numerical Methods!"

x = 2

y = 3

a = x + y

b = x - y

c = x \* y

d = x / y

**write**(\*,\*) a

**write**(\*,\*) b

**write**(\*,\*) c

**write**(\*,\*) d

**end** **program** mainProgram

## Output

Welcome to Numerical Methods!

5

-1

6

0

## Ideas

* Fortran variables: type, name, value
* Variable type: **integer**
* Can declare multiple variables of the same type on the same line, separated by commas
* Variables are declared – all in one place – immediately after the **implicit** **none** statement
* Naming—
  + Variable names can be up to 63 characters long.
  + The first character must be a letter.
  + Rest of the characters can be letters, numbers, and underscores.
  + Variable names are not case sensitive. Recommended: camelCase.
  + Fortran doesn’t have reserve words. Fortran keywords can be used as variable names. This is terrible practice and is to be avoided.
* The **write** statement is flexible: string of characters, numbers, variables.

# Writing Clean Code

## Code

**program** mainProgram

**implicit** **none**

**integer** integer1, integer2

**integer** sumOfIntegers, diffOfIntegers, prodOfIntegers, quotOfIntegers

**write**(\*,\*) "Welcome to Numerical Methods!"

integer1 = 2

integer2 = 3

sumOfIntegers = integer1 + integer2

diffOfIntegers = integer1 - integer2

prodOfIntegers = integer1 \* integer2

quotOfIntegers = integer1 / integer2

**write**(\*,\*) sumOfIntegers

**write**(\*,\*) diffOfIntegers

**write**(\*,\*) prodOfIntegers

**write**(\*,\*) quotOfIntegers

**end** **program** mainProgram

## Output

Welcome to Numerical Methods!

5

-1

6

0

## Ideas

1. Indent for better readability. Use the tab key to indent individual lines. You only need to indent the first line of the block. Eclipse automatically indents the next line after you press the Enter key.
2. Add a space before and after mathematical operations (+, -, \*, /, =, etc.)
3. Add a space after commas.
4. Separate blocks of related code with whitespace (press the Enter key), like paragraphs in a book. In this code, for example, we’ve separated blocks of code for: (1) declaration of variables, (2) assigning values to the variables, (3) calculations, and (4) writing the variables to the console.
5. Use descriptive variable names.

# Descriptive Output

## Code

**program** mainProgram

**implicit** **none**

**integer** integer1, integer2

**integer** sumOfIntegers, diffOfIntegers, prodOfIntegers, quotOfIntegers

**write**(\*,\*) "Arithmetic operations for integers"

**write**(\*,\*)

integer1 = 3

integer2 = 2

sumOfIntegers = integer1 + integer2

diffOfIntegers = integer1 - integer2

prodOfIntegers = integer1 \* integer2

quotOfIntegers = integer1 / integer2

**write**(\*,\*) "The input--"

**write**(\*,\*) "Integer 1 = ", integer1

**write**(\*,\*) "Integer 2 = ", integer2

**write**(\*,\*)

**write**(\*,\*) "The output--"

**write**(\*,\*) "The sum of ", integer1, " and ", integer2, " is ", sumOfIntegers

**write**(\*,\*) "The difference between ", integer1, " and ", integer2, " is ", diffOfIntegers

**write**(\*,\*) "The product of ", integer1, " and ", integer2, " is ", prodOfIntegers

**write**(\*,\*) "The quotient of ", integer1, " and ", integer2, " is ", quotOfIntegers

**end** **program** mainProgram

## Output

Arithmetic operations for integers

The input--

Integer 1 = 3

Integer 2 = 2

The output--

The sum of 3 and 2 is 5

The difference between 3 and 2 is 1

The product of 3 and 2 is 6

The quotient of 3 and 2 is 1

## Ideas

* Fortran is not a visual language. It’s meant for scientific, numerical computing. Formatting options are, therefore, limited.
* A single **write** statement can be used to print multiple items (separated by commas) to the console, including strings of characters, integers, real numbers, etc.

# The format Statement

## Code

**program** mainProgram

**implicit** **none**

**integer** integer1, integer2

**integer** sumOfIntegers, diffOfIntegers, prodOfIntegers, quotOfIntegers

**write**(\*,\*) "Arithmetic operations for integers"

**write**(\*,\*)

integer1 = 3

integer2 = 2

sumOfIntegers = integer1 + integer2

diffOfIntegers = integer1 - integer2

prodOfIntegers = integer1 \* integer2

quotOfIntegers = integer1 / integer2

**write**(\*,\*) "The input--"

**write**(\*,10) "Integer 1 = ", integer1

**write**(\*,10) "Integer 2 = ", integer2

10 **format**(a12, i1)

**write**(\*,\*)

**write**(\*,\*) "The output--"

**write**(\*,20) "The sum of ", integer1, " and ", integer2, " is ", sumOfIntegers

**write**(\*,20) "The difference between ", integer1, " and ", integer2, " is ", diffOfIntegers

**write**(\*,20) "The product of ", integer1, " and ", integer2, " is ", prodOfIntegers

**write**(\*,20) "The quotient of ", integer1, " and ", integer2, " is ", quotOfIntegers

20 **format**(a23, i1, a5, i1, a4, i1)

**end** **program** mainProgram

## Output

Arithmetic operations for integers

The input--

Integer 1 = 3

Integer 2 = 2

The output--

The sum of 3 and 2 is 5

The difference between 3 and 2 is 1

The product of 3 and 2 is 6

The quotient of 3 and 2 is 1

## Ideas

* **write**(\*,\*). The first \* instructs the compiler to write the output to the console (as opposed to a file). The second \* tells the compiler the format in which we wish to view the output.
* The **format** statement.
  + Must have a label (an integer to identify the line on which it appears).
  + Replace the second \* in **write**(\*,\*) by the label to use the format specified.
  + A single format statement can be used by multiple write statements (that use the same format).
  + a23: allocate space on the console for a string 23 characters wide. When printing a pre-defined string using write (“text written within quotes”), manually count the number of characters in the string (including spaces) to arrive at the a-number.
  + i1: allocate space on the console for an integer 1 character wide.
  + Items are right-aligned on the console.
  + Play around with the format statement till you get a satisfactory output.
  + Can appear anywhere in the code (not necessarily after or before the write statement that uses it.)

# User Input – The read Statement

## Code

**program** mainProgram

**implicit** **none**

**integer** integer1, integer2

**integer** sumOfIntegers, diffOfIntegers, prodOfIntegers, quotOfIntegers

**write**(\*,\*) "Arithmetic operations for integers"

**write**(\*,\*)

**write**(\*,\*) "Enter the first integer:"

**read**(\*,\*) integer1

**write**(\*,\*) "Enter the second integer:"

**read**(\*,\*) integer2

sumOfIntegers = integer1 + integer2

diffOfIntegers = integer1 - integer2

prodOfIntegers = integer1 \* integer2

quotOfIntegers = integer1 / integer2

**write**(\*,\*) "The input--"

**write**(\*,10) "Integer 1 = ", integer1

**write**(\*,10) "Integer 2 = ", integer2

10 **format**(a12, i1)

**write**(\*,\*)

**write**(\*,\*) "The output--"

**write**(\*,20) "The sum of ", integer1, " and ", integer2, " is ", sumOfIntegers

**write**(\*,20) "The difference between ", integer1, " and ", integer2, " is ", diffOfIntegers

**write**(\*,20) "The product of ", integer1, " and ", integer2, " is ", prodOfIntegers

**write**(\*,20) "The quotient of ", integer1, " and ", integer2, " is ", quotOfIntegers

20 **format**(a23, i1, a5, i1, a4, i1)

**end** **program** mainProgram

## Output

Arithmetic operations for real numbers

Enter the first integer:

3

Enter the second integer:

2

The input--

Integer 1 = 3

Integer 2 = 2

The output--

The sum of 3 and 2 is 5

The difference between 3 and 2 is 1

The product of 3 and 2 is 6

The quotient of 3 and 2 is 1

## Ideas

* **read**(\*,\*) to get user input from the Console.
* What happens if the user enters a real number, instead of an integer?

# Variable Type: Real Numbers

## Code

**program** mainProgram

**implicit** **none**

**real** number1, number2

**real** sumOfNumbers, diffOfNumbers, prodOfNumbers, quotOfNumbers

**write**(\*,\*) "Arithmetic operations for real numbers"

**write**(\*,\*)

**write**(\*,\*) "Enter the first number:"

**read**(\*,\*) number1

**write**(\*,\*) "Enter the second number:"

**read**(\*,\*) number2

sumOfNumbers = number1 + number2

diffOfNumbers = number1 - number2

prodOfNumbers = number1 \* number2

quotOfNumbers = number1 / number2

**write**(\*,\*) "The input--"

**write**(\*,10) "Number 1 = ", number1

**write**(\*,10) "Number 2 = ", number2

10 **format**(a12, f8.2)

**write**(\*,\*)

**write**(\*,\*) "The output--"

**write**(\*,20) "The sum of ", number1, " and ", number2, " is ", sumOfNumbers

**write**(\*,20) "The difference between ", number1, " and ", number2, " is ", diffOfNumbers

**write**(\*,20) "The product of ", number1, " and ", number2, " is ", prodOfNumbers

**write**(\*,20) "The quotient of ", number1, " and ", number2, " is ", quotOfNumbers

20 **format**(a23, f8.2, a5, f8.2, a4, f8.2)

**end** **program** mainProgram

## Output

Arithmetic operations for real numbers

Enter the first number:

4.5

Enter the second number:

2.1

The input--

Number 1 = 4.50

Number 2 = 2.10

The output--

The sum of 4.50 and 2.10 is 6.60

The difference between 4.50 and 2.10 is 2.40

The product of 4.50 and 2.10 is 9.45

The quotient of 4.50 and 2.10 is 2.14

## Ideas

* Formatting of real numbers using f.
* f8.2: allocate space on the console for a real number 8 characters wide. Of these 8 spaces, use 2 spaces for the fractional part of the number. 1 space is used for the decimal point, leaving 8 – 2 – 1 = 5 spaces for the integer part of the number.

# Functions: Arithmetic Operations

## Code

### mainProgram.f08

**program** mainProgram

**implicit** **none**

**real** sumOf, diffBetween, prodOf, quotOf

**real** number1, number2

**real** sumOfNumbers, diffBetweenNumbers, prodOfNumbers, quotOfNumbers

**write**(\*,\*) "Arithmetic Operations"

**write**(\*,\*) "Enter the first number: "

**read**(\*,\*) number1

**write**(\*,\*) "Enter the second number: "

**read**(\*,\*) number2

sumOfNumbers = sumOf(number1, number2)

diffBetweenNumbers = diffBetween(number1, number2)

prodOfNumbers = prodOf(number1, number2)

quotOfNumbers = quotOf(number1, number2)

**write**(\*,\*) "The input--"

**write**(\*,10) "Number 1 = ", number1

**write**(\*,10) "Number 2 = ", number2

10 **format**(a12, f8.2)

**write**(\*,\*)

**write**(\*,\*) "The output--"

**write**(\*,20) "The sum of ", number1, " and ", number2, " is ", sumOfNumbers

**write**(\*,20) "The difference of ", number1, " and ", number2, " is ", diffBetweenNumbers

**write**(\*,20) "The product of ", number1, " and ", number2, " is ", prodOfNumbers

**write**(\*,20) "The quotient of ", number1, " and ", number2, " is ", quotOfNumbers

20 **format**(a18, f8.2, a5, f8.2, a4, f8.2)

**end** **program** mainProgram

### sumOf.f08

**real** **function** sumOf(inputNumber1, inputNumber2)

**implicit** **none**

**real** inputNumber1, inputNumber2

sumOf = inputNumber1 + inputNumber2

**end** **function** sumOf

### diffBetween.f08

**real** **function** diffBetween(inputNumber1, inputNumber2)

**implicit** **none**

**real** inputNumber1, inputNumber2

diffbetween = inputNumber1 - inputNumber2

**end** **function** diffBetween

### prodOf.f08

**real** **function** prodOf(inputNumber1, inputNumber2)

**implicit** **none**

**real** inputNumber1, inputNumber2

prodOf = inputNumber1 \* inputNumber2

**end** **function** prodOf

### quotOf.f08

**real** **function** quotOf(inputNumber1, inputNumber2)

**implicit** **none**

**real** inputNumber1, inputNumber2

quotOf = inputNumber1 / inputNumber2

**end** **function** quotOf

## Output

Arithmetic Operations

Enter the first number:

10

Enter the second number:

20

The input--

Number 1 = 10.00

Number 2 = 20.00

The output--

The sum of 10.00 and 20.00 is 30.00

The difference of 10.00 and 20.00 is -10.00

The product of 10.00 and 20.00 is 200.00

The quotient of 10.00 and 20.00 is 0.50

## Ideas

* Fortran function: a type of subprogram. Can be invoked from the main program and other subprograms. Reusable code. Modular programming makes it easier to extend functionality and isolate bugs.
* Like a mathematical function: one or more inputs (independent variables), exactly one output (dependent variable).
* Each function should perform one task, and perform it well.
* Input variables: arguments or parameters.
* Creating a new function--
  + **File** – **New** – **Fortran Source File** – Enter the name of the file along with the extension **.f08** (e.g., sumOf.f08, prodOf.f08, etc.). Alternatively, functions can also be written in the same source file as the main program (outside the program-end program block), or in a source file containing other (similar) subprograms.
  + In the newly created source file, replace the keyword **program** by **function** in the default code generated by Eclipse.
* Defining a function (in the newly created source file)--
  + Specify the type for the function output (**real**, in this case).
  + Specify the input variables/arguments (inputNumber1, inputNumber2).
  + Declare the types of all input arguments within the body of the function, immediately after **implicit** **none**.
  + Write the code in the body of the function (**function** – **end function** block).
  + Assign a return value to the name of the function (consistent with the function type).
* Invoking a function (in the main program)--
  + Declare the newly defined functions in the main program (**real**), before the first executable statement.
  + Invoke the functions from the main program, wherever appropriate.
* Recommended: Source files of related functions should be moved to a separate Fortran Source Folder. Eclipse locates all the referenced source files irrespective of the subfolder in which they’re placed, as long as they’re placed within the main project folder.