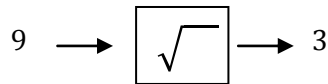


Fortran 77: 5. Intrinsic Functions

A function is a general term which maps one entity – the input or argument – on to another number – the output or result. There are a number of functions that are important in mathematics science and engineering. Such a set of functions can be found on any scientific calculator¹. For example on a calculator we can apply the square root function to the number '9' to get the result '3':

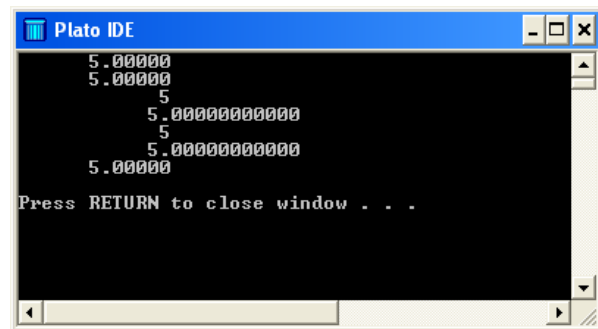


Similarly, programming languages like Fortran also have a number of built-in functions which act on the standard data types and can be called by the programmer. In this document the pre-defined or intrinsic functions in Fortran 77 are listed.

Type Conversions

The functions REAL, INT and DBLE can turn the type of the argument into the REAL, INT and DOUBLE types respectively. FLOAT transforms a value of integer type to the same value in the REAL type.

```
C *****
C TYPECONV
C *****
  PROGRAM TYPECONV
C INT -> REAL
  WRITE(*,*) FLOAT(5)
  WRITE(*,*) REAL(5)
C REAL -> INT
  WRITE(*,*) INT(5.0E0)
C INT -> DOUBLE
  WRITE(*,*) DBLE(5)
C DOUBLE -> INT
  WRITE(*,*) INT(5.0D0)
C FLOAT -> DOUBLE
  WRITE(*,*) DBLE(5.0E0)
C DOUBLE -> FLOAT
  WRITE(*,*) REAL(5.0D0)
  STOP
  END
```



Often converting a real number to an integer involves approximation and this is covered in the next section.

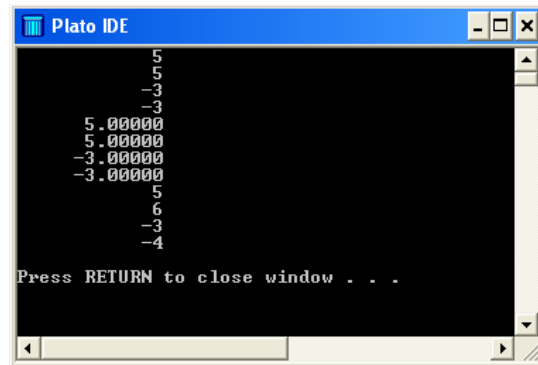
Truncating or Rounding REAL to INT: INT, AINT and NINT

The functions INT, AINT, and NINT convert real numbers to a whole number, which usually involves approximation. INT truncates the number, removing the decimal part and returns an integer. Note that INT returns the greatest whole number that is also less than the argument when the argument is positive. When the argument is negative, INT returns the least whole number that is greater than the argument. AINT acts the same as INT but returns the result as a

¹ [Standard Mathematical Functions: Windows Scientific Calculator](#)

real number. NINT returns the nearest integer to the given argument. The following program shows typical results from using INT, AINT, and NINT.

```
C *****
C WHOLE
C *****
PROGRAM WHOLE
C INT - Truncate to integer
WRITE(*,*) INT(5.3)
WRITE(*,*) INT(5.8)
WRITE(*,*) INT(-3.2)
WRITE(*,*) INT(-3.9)
C AINT - Truncate to real
WRITE(*,*) AINT(5.3)
WRITE(*,*) AINT(5.8)
WRITE(*,*) AINT(-3.2)
WRITE(*,*) AINT(-3.9)
C NINT - Round to integer
WRITE(*,*) NINT(5.3)
WRITE(*,*) NINT(5.8)
WRITE(*,*) NINT(-3.2)
WRITE(*,*) NINT(-3.9)
STOP
END
```

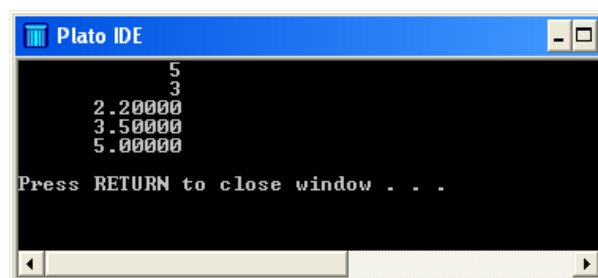


Other effective type conversions will be covered in this document.

Absolute value: ABS

The function ABS returns the magnitude or size of a number. For integer or real numbers this is simply a matter of ignoring the sign. For complex numbers ABS returns the absolute value of a complex number².

```
C *****
C ABS1
C *****
PROGRAM ABS1
C ABS for integers
WRITE(*,*) ABS(5)
WRITE(*,*) ABS(-3)
C ABS for real numbers
WRITE(*,*) ABS(2.2)
WRITE(*,*) ABS(-3.5)
C ABS for a complex number
WRITE(*,*) ABS((3,4))
STOP
END
```

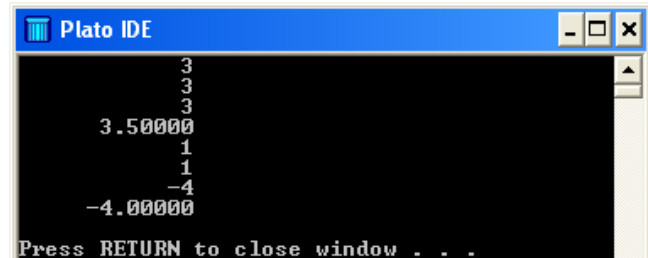


² [Complex Numbers](#)

Maximum and Minimum of a set: MAX and MIN

The functions MAX and MIN return the maximum and minimum values of a set of numbers. An example program and results are given below.

```
C *****
C MAXMIN
C *****
PROGRAM MAXMIN
C MAX
WRITE(*,*) MAX(1,3)
WRITE(*,*) MAX(1,3,2)
WRITE(*,*) MAX(1,3,2,-4)
WRITE(*,*) MAX(1,3,2,-4,3.5)
C MIN
WRITE(*,*) MIN(1,3)
WRITE(*,*) MIN(1,3,2)
WRITE(*,*) MIN(1,3,2,-4)
WRITE(*,*) MIN(1,3,2,-4,3.5)
STOP
END
```

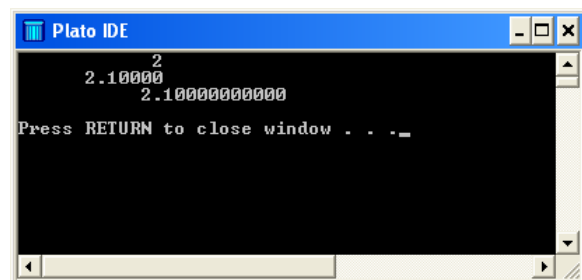


Note that if the arguments are all of integer type then an integer type is returned. If any of the arguments are of real type then a real type is returned.

MOD: the remainder after integer division

The MOD function had two arguments and it returns the first argument *modulo* the second argument. In other words, in modular arithmetic³ the modulo (or MOD) function returns the remainder after the first argument is divided by the second argument. In Fortran 77, MOD may have real (and double precision) arguments, as well as integer. The following program and results demonstrate the MOD function.

```
C *****
C MODULAR
C *****
PROGRAM MODULAR
C INT -> REAL
WRITE(*,*) MOD(22,5)
WRITE(*,*) MOD(22.5,5.1)
WRITE(*,*) MOD(22.5D0,5.1D0)
STOP
END
```



³ [Integer Division and Modular Arithmetic](#)

Complex number manipulation: CMPLX, DCMPLX, REAL, DIMAG, AIMAG, DIMAG and CONJG

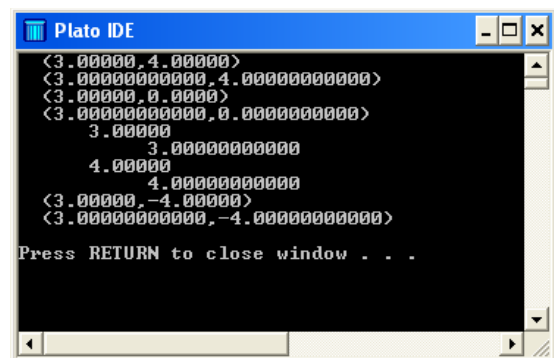
Complex numbers⁴ are each composed of a real and imaginary part. CMPLX creates a single precision complex number with real part equal to the first argument and imaginary part equal to the second argument. DCMPLX is the double precision equivalent of CMPLX. If CMPLX or DCMPLX have only one argument then they return the real number in a complex type.

REAL returns the real part of its complex argument. DBLE is the double precision equivalent of REAL. AIMAG returns the imaginary part of its complex argument. DIMAG is the double precision equivalent of AIMAG.

CONJG returns the complex conjugate of a complex number.

Typical results from using these functions are given in the following.

```
C *****
C COMPLEX
C *****
PROGRAM COMPLEX
C CMPLX and DCMPLX: creates a complex number
C from its real and imaginary parts
C CMPLX is for single precision
C DCMPLX is for double precision
WRITE(*,*) CMPLX(3,4)
WRITE(*,*) DCMPLX(3,4)
C CMPLX and DCMPLX can convert a real number in real type
C to a real number in single or double precision complex type
WRITE(*,*) CMPLX(3)
WRITE(*,*) DCMPLX(3)
C REAL, DBLE: real part of a complex number
C REAL-single precision, DBLE-double precision
WRITE(*,*) REAL(CMPLX(3,4))
WRITE(*,*) DBLE(DCMPLX(3,4))
C IMAG, DIMAG: imaginary part of a complex number
C IMAG-single precision, DIMAG-double precision
WRITE(*,*) AIMAG(CMPLX(3,4))
WRITE(*,*) DIMAG(DCMPLX(3,4))
C CONJG : the complex conjugate
WRITE(*,*) CONJG(CMPLX(3,4))
WRITE(*,*) CONJG(DCMPLX(3,4))
STOP
END
```



```
Plato IDE
<3.000000, 4.000000>
<3.000000000000, 4.000000000000>
<3.000000, 0.000000>
<3.000000000000, 0.000000000000>
3.000000
3.000000000000
4.000000
4.000000000000
<3.000000, -4.000000>
<3.000000000000, -4.000000000000>
Press RETURN to close window . . .
```

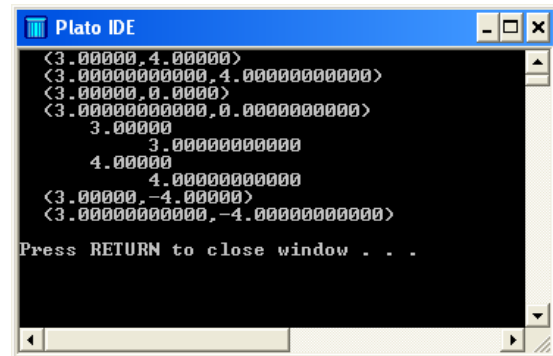
Square root function: SQRT

The square root⁵ of a number is equivalent to raising it to the power of 0.5. The square root function can be applied to real and complex numbers⁴ in both single and double precision. The following program and results demonstrate the SQRT function.

⁴ [Complex Numbers](#)

⁵ [Powers and Roots](#)

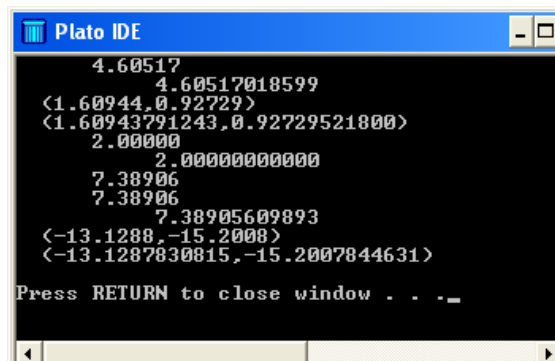
```
C *****
C SQUAREROOT
C *****
PROGRAM SQUAREROOT
C SQRT returns the squareroot of its argument
C The squareroot of a number is equivalent to it
C raised to a power of 0.5
C Single precision SQRT of real number
WRITE(*,*) SQRT(16.0)
WRITE(*,*) 16.0**0.5
C Double precision SQRT of real number
WRITE(*,*) SQRT(16.0D0)
WRITE(*,*) 16.0D0**0.5D0
C Single precision SQRT of complex number
WRITE(*,*) SQRT((1.0,1.0))
WRITE(*,*) (1.0,1.0)**0.5
C Double precision SQRT of complex number
WRITE(*,*) SQRT((1.0D0,1.0D0))
WRITE(*,*) (1.0D0,1.0D0)**0.5D0
STOP
END
```



Logarithm and Exponential Functions⁶: LOG, LOG10 and EXP

The LOG function returns the natural or Napierian logarithm (base e) and it can be applied to real or complex numbers in single or double precision. The LOG10 function returns the logarithm (base 10) and it can be applied to real numbers in single or double precision. The function EXP returns the e to the power of the real or complex argument⁷ in double or single precision.

```
C *****
C LOGEXP
C *****
PROGRAM LOGEXP
C LOG returns the logarithm (base e) of its argument
C it may be applied to double precision and complex arguments
WRITE(*,*) LOG(100.0)
WRITE(*,*) LOG(100.0D0)
WRITE(*,*) LOG((3.0,4.0))
WRITE(*,*) LOG((3.0D0,4.0D0))
C LOG10 returns the logarithm (base 10) of its argument
C it may be applied to double precision but not complex arguments
WRITE(*,*) LOG10(100.0)
WRITE(*,*) LOG10(100.0D0)
C EXP returns e to the power of its argument
C it may be applied to double precision and complex arguments
WRITE(*,*) EXP(2.0)
WRITE(*,*) EXP(2.0)
WRITE(*,*) EXP(2.0D0)
WRITE(*,*) EXP((3.0,4.0))
WRITE(*,*) EXP((3.0D0,4.0D0))
STOP
END
```



⁶ [Logarithm and Exponential Functions](#)

⁷ [Standard functions applied to complex numbers](#)

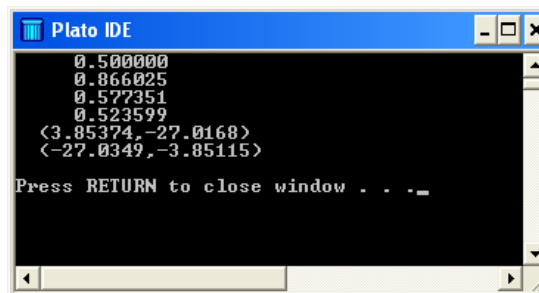
Trigonometric Functions: SIN, COS, TAN, ASIN, ACOS, ATAN and ATAN2

The trigonometric functions are originally defined for triangles⁸. However, they can be extended to be defined for the real numbers in general, and then also for complex numbers. The argument for trig functions is assumed to be in radians and for the inverse functions the result is presumed to be in radians. In Fortran 77 all functions can be applied to real numbers (single or double precision), but only sine and cosines of complex numbers are available. The following table lists the available functions.

Function	Mathematical notation	Fortran 77 function	Applied to
sine	sin	SIN	real or complex
cosine	cos	COS	real or complex
tangent	tan	TAN	real
arcsine	arcsin, \sin^{-1}	ASIN	real
arccosine	arccos, \cos^{-1}	ACOS	real
arctangent	arctan, \tan^{-1}	ATAN	real

The following program demonstrates the use of the trigonometric functions in Fortran 77. Only single precision arguments are applied here, but the functions can also be applied in double precision.

```
C *****  
C TRIG FUNCTIONS  
C *****  
PROGRAM TRIG  
C sin of 30deg, pi/6  
WRITE(*,*) SIN(0.523599)  
C cos of 30deg, pi/6  
WRITE(*,*) COS(0.523599)  
C tan of 30deg, pi/6  
WRITE(*,*) TAN(0.523599)  
C arcsin of 0.5  
WRITE(*,*) ASIN(0.5)  
C Sine of a complex number  
WRITE(*,*) SIN((3,4))  
C Cosine of a complex number  
WRITE(*,*) COS((3,4))  
STOP  
END
```



ATAN2

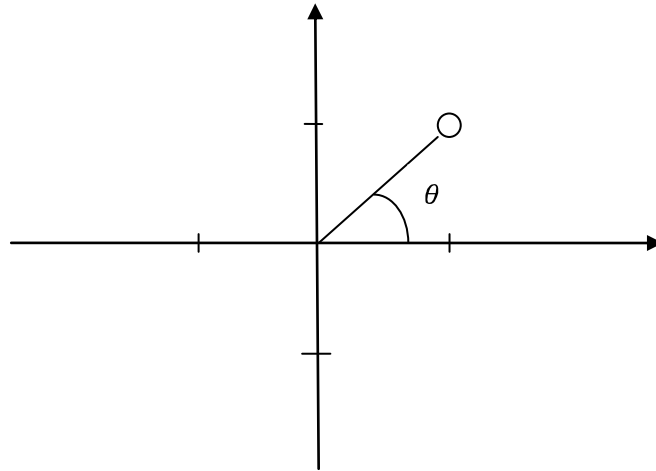
The result from calling arctan (ATAN) is a number in the range $(-\frac{\pi}{2}, \frac{\pi}{2})$ or $(-90^\circ, 90^\circ)$. However, often we require the angle to be in the range $(-\pi, \pi)$ or $(-180^\circ, 180^\circ)$.

For example a complex number⁹ can have an argument in the range $(-180^\circ, 180^\circ)$. If we use arctan (ATAN) to find the argument of the complex number $1+i$ then – thinking of its position on

⁸ [Trigonometry](#)

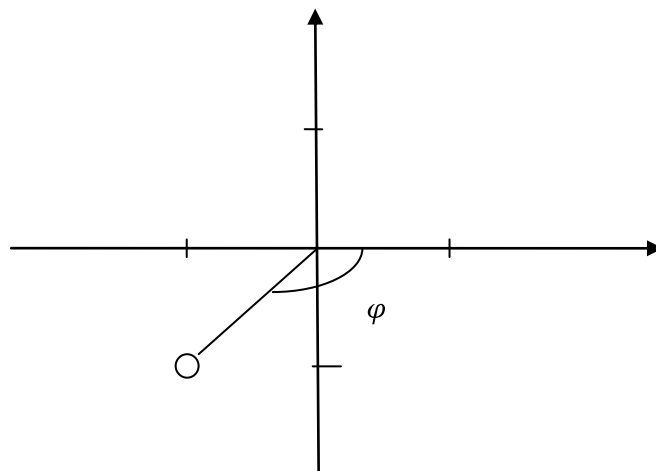
⁹ [Complex Numbers](#)

the Argand diagram – the tangent of the argument is equal to the imaginary part divided by the real part. The situation is illustrated in the following diagram.



$$\tan \theta = \frac{1}{1} = 1 \text{ and if we find the arctangent of 1 we get } \frac{\pi}{4}.$$

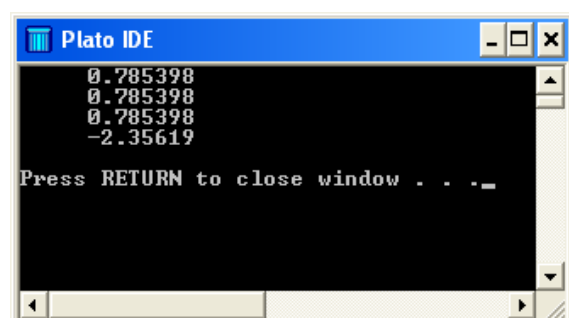
However, if we carry out the same method to find the argument of the complex number (-1,-1) we obtain the same answer.



$$\tan \varphi = \frac{-1}{-1} = 1 \text{ and if we find the arctangent of 1 we get } \frac{\pi}{4} ?$$

The answer is actually $-\frac{3\pi}{4}$ and this can be found using the ATAN2 function. The ATAN2 function is demonstrated in the following program.

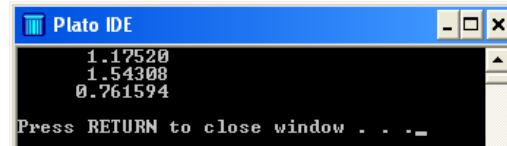
```
C *****
C ATAN
C *****
PROGRAM ATAN1
C Finding the angle to (1,1)
WRITE(*,*) ATAN(1.0/1.0)
WRITE(*,*) ATAN2(1.0,1.0)
C Finding the angle to (-1.0,-1.0)
WRITE(*,*) ATAN((-1.0)/(-1.0))
WRITE(*,*) ATAN2(-1.0,-1.0)
STOP
END
```



The Hyperbolic Functions¹⁰: SINH, COSH, TANH

Fortran 77 includes the hyperbolic functions sinh (SINH), cosh (COSH) and tanh (TANH). The following program demonstrates the functions in single precision.

```
C *****
C HYPERBOLIC
C *****
PROGRAM HYP
WRITE(*,*) SINH(1.0)
WRITE(*,*) COSH(1.0)
WRITE(*,*) TANH(1.0)
STOP
END
```



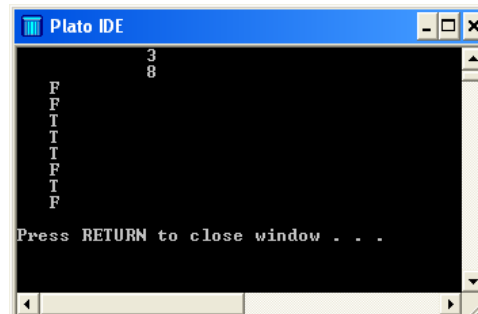
The screenshot shows a window titled "Plato IDE" with a black background and white text. The output of the program is displayed as follows:

```
1.17520
1.54308
0.761594
Press RETURN to close window . . .
```

String Functions

A string is a sequence of characters, eg a word or a sentence. Fortran 77 has a number of string functions. LEN returns the length of the string. Strings can be compared for lexicographic (similar to alphabetical but includes other characters and is carried out with respect to the ASCII codes) ordering using the functions LGE (larger than or equal to), LGT (larger than), LLE (less than or equal to), and LLT (less than). The following program demonstrates the string functions.

```
C *****
C STRING FUNCTIONS
C *****
PROGRAM STRINGFN
WRITE(*,*) LEN("DOG")
WRITE(*,*) LEN("ELEPHANT")
WRITE(*,*) LGE("DOG","ELEPHANT")
WRITE(*,*) LGT("DOG","ELEPHANT")
WRITE(*,*) LLE("DOG","ELEPHANT")
WRITE(*,*) LLT("DOG","ELEPHANT")
WRITE(*,*) LGE("DOG","DOG")
WRITE(*,*) LGT("DOG","DOG")
WRITE(*,*) LLE("DOG","DOG")
WRITE(*,*) LLT("DOG","DOG")
STOP
END
```



The screenshot shows a window titled "Plato IDE" with a black background and white text. The output of the program is displayed as follows:

```
3
8
F
F
T
T
F
T
F
Press RETURN to close window . . .
```

¹⁰ [Hyperbolic Functions](#)