# Chapter 24
# Operator Overloading


*All the persons in this book are real and none is fictitious even in part.*

Flann O'Brien, The Hard Life

**Aims**
The aims of this chapter are to look at operator overloading in Fortran.

## 24.1 Introduction

In programming operator overloading can be regarded as a way of achieving polymorphism in that operators (e.g. +, −, * , / or =) can have different implementations depending on the types of their arguments.

In some programming languages overloading is defined by the language. In Fortran for example, the addition + operator invokes quite different code when used with integer, real or complex types.

Some languages allow the programmer to implement support for user defined types. Fortran introduced support for operator and assignment overloading in the 1990 standard.

## 24.2 Other Languages

Operator overloading is not new and several languages offer support for the feature including:



I. Chivers and J. Sleightholme, *Introduction to Programming with Fortran*,
https://doi.org/10.1007/978-3-319-75502-1_24

- Algol 68 - 1968
- Ada - Ada 83
- C++ - First standard, 1998
- Eiffel - 1986
- C# - 2001

Java, however does not.

## 24.3 Example 1: Overloading the Addition (+) Operator

The following example overloads the addition operator.

```
module t_position
  implicit none
  type position
    integer :: x
    integer :: y
    integer :: z
  end type position
  interface operator (+)
    module procedure new_position
  end interface operator (+)

contains
  function new_position(a, b)
    type (position), intent (in) :: a, b
    type (position) :: new_position

    new_position%x = a%x + b%x
    new_position%y = a%y + b%y
    new_position%z = a%z + b%z
  end function new_position
end module t_position

program ch2401
  use t_position
  implicit none
  type (position) :: a, b, c

  a%x = 10
  a%y = 10
  a%z = 10
  b%x = 20
```

```
    b%y = 20
    b%z = 20
    c = a + b
    print *, a
    print *, b
    print *, c
end program ch2401
```

We have extended the meaning of the addition operator so that we can write simple expressions in Fortran based on it and have our new position calculated using a user supplied function that actually implements the calculation of the new position.

## 24.4 Problem

**24.1** Compile and run this example. Overload the subtraction operator as well.