

# Differences between Modula-2 R10 and classic Modula-2

June 30, 2010

Modula-2 R10 is a modern revision of classic Modula-2 using N.Wirth's fourth edition of "Programming in Modula-2" as a starting point. The primary design goals were type safety, a strong emphasis on readability of source code, consistency and suitability as a core language for domain specific supersets. Target areas of application are systems implementation, engineering and mathematics. The moniker R10 is a shorthand for "Revision 2010". The language description is at <http://modula2.net/resources/M2R10.pdf>

## Omitted Features

The following features were omitted in Modula-2 R10:

- No local modules
- No variant records
- No `EXPORT` statement
- No `WITH DO` statement
- No synonyms (`~`, `&`, `<>`)
- No octal number literals
- No type conversion functions
- No anonymous types except for one-dimensional arrays

## Replaced Features

The following features were replaced in Modula-2 R10:

- Radix 2 replaces radix 8 in binary literals
- Radix 16 replaces radix 8 in character code literals
- Suffix `u` replaces suffix `c` in character code literals
- Extensible record types replace variant record types
- `ALIAS OF` type constructor replaces `CONST` declaration of type aliases
- Type conversion operator `::` replaces conversion functions
- increment `++` and decrement `--` statements replace `INC` and `DEC`
- Pseudo-function `CAST` replaces type-transfer syntax\*
- Auto-casting formal open array parameters are prefixed with `CAST`
- Pragma delimiters `<*` and `*>` replace `(* $` and `*)` delimiters\*

## Features That Were Previously Optional

The following features of Modula-2 R10 were optional-only in classic Modula-2:

- Variables may be declared at fixed addresses
- Variables are always exported immutable
- Low-level intrinsics in pseudo-module `SYSTEM`

## Revised Syntax

The following syntax was changed in Modula-2 R10:

- Constant expressions may not be type or range designators
- The index of an array declaration may not be a type or range
- A subrange type declaration must always specify the base type

## Revised Semantics

The following semantics were changed in Modula-2 R10:

- Array indices are always zero-based
- Module initialisation order is language defined
- Strict name equivalence instead of loose name equivalence
- Character literals are assignment compatible with `ARRAY OF CHAR`
- Named elements of sets and enumerations must be referenced qualified
- The control variable of a `FOR` loop is declared in the loop's header
- The scope of the control variable of a `FOR` loop is the loop's body
- Nesting of comments is limited to ten levels (one outer and nine inner comments)

## Newly Added Features

The following features were added in Modula-2 R10:

- Conditional compilation
- Language defined pragmas
- Single line comments, using prefix `//`
- Structured literals and structured value constructors\*
- Import qualifiers for import-all and re-export
- Extensible enumeration and record types
- Built-in `UNICHAR` type for unicode characters
- Types `OCTET`, `LONGBITSET` and `LONGCARD`
- Immutable `CONST` parameters and pointer target types
- Concatenation of string literals using the `+` operator\*
- Escape sequences `\0`, `\n`, `\r`, `\t`, `\\`, `\'` and `\"` in string literals
- `ASSOCIATIVE ARRAY` type constructor for ordered collections
- `FOR IN` loop for iteration of ordinals, enumerations, arrays, sets and collections
- Abstract data types may bind procedures to operators and built-in functions
- Records may have a single array field whose size is determined at runtime
- Type-safe variadic parameters and foreign function interface to C
- Atomic intrinsics via pseudo-module `ATOMIC`

## Newly Added Optional Features

The following features were added as optional features in Modula-2 R10:

- Pseudo-module `ASSEMBLER` for inline assembly code

## Standard Library

The standard library of Modula-2 R10 has been completely redesigned.

## Backwards Compatibility

Migration of source code written in classic Modula-2 to Modula-2 R10 will be assisted by a standard source-to-source translator utility.

---

\* pseudo function `CAST`, pragma delimiters, structured value constructors and string literal concatenation were adopted from ISO Modula-2.