Differences between Modula-2 R10 and classic Modula-2

This document is a brief summary of differences between Modula-2 R10 and classic Modula-2 as defined in the fourth edition of “Programming in Modula-2” by N. Wirth (1984).

Modula-2 R10 is a modernised revision of classic Modula-2 defined in the Modula-2 R10 language specification by B. Kowarsch and R. Sutcliffe (2012). Backwards compatibility of source code was not a consideration of the revision. Instead, migration of source code from classic Modula-2 is expected to be carried out via source-to-source translation tools.

The language specification is available at *`http://modula2.net/resources/M2R10.pdf`*

Removed Features

The following features were removed in Modula-2 R10:

- Local modules
- Variant records
- `EXPORT` statement
- `WITH DO` statement
- Octal number literals
- Synonyms `~`, `&` and `<>`
- Type conversion functions
- `CONST` declaration of type aliases
- 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
- Pervasive functions `SUCC` and `PRED` replace `INC` and `DEC`
- Pseudo-function `CAST` in module `SYSTEM` replaces type-transfer syntax [ISO]
- Auto-casting formal open array parameters are prefixed with `CAST`
- Delimiters `<*` and `*>` replace `(*$` and `*)` as pragma delimiters [ISO]

Mandatory Features That Were Previously Optional

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

- Variables are always exported immutable
- Low-level intrinsics in pseudo-module `SYSTEM`

Revised Syntax

The following syntax was changed in Modula-2 R10:

- Nesting of comments is limited to a maximum of ten levels
- Opaque types are declared using new reserved word `OPAQUE`
- A subrange type declaration must always specify the base type
- Variables are declared at fixed addresses using pragma `ADDR`
- The control variable of a `FOR` loop is declared in the loop's header
- The index of an array declaration must be an unsigned whole number
- Named elements of sets and enumerations must be qualified with their type

---

[ISO] Features directly adopted from or similar to ISO Modula-2.

Revised Semantics

The following semantics were changed in Modula-2 R10:

- Array indices are always zero-based
- Strict name equivalence instead of lose name equivalence
- Character literals are assignment compatible with `ARRAY OF CHAR`
- The control variable of a `FOR` loop is immutable within its scope
- The scope of the control variable of a `FOR` loop is the loop's body

Added Features

The following features were added in Modula-2 R10:

- Single-line comments
- Conditional compilation
- Various language defined pragmas
- Structured literals and structured value constructors ISO
- Import qualifiers for import-all and re-export
- Extensible enumeration and record types
- Built-in `UNICHAR` type for unicode characters
- Pervasive types `OCTET`, `LONGBITSET` and `LONGCARD`
- Immutable `CONST` parameters and pointer target types
- Concatenation of string literals using the + operator ISO
- `ASSCOCIATIVE ARRAY` type constructor for unordered collection types
- `FOR IN` loop for iteration of ordinals, enumerations, arrays, sets and collections
- Use of built-in operators and pervasive procedures with library defined data types
- Type-safe variadic procedures
- Type-safe indeterminate record types
- Foreign function interface to C using pragma `FFI`
- A new pseudo-module `ATOMIC` providing atomic intrinsics
- A new pseudo-module `COMPILER` providing constants and intrinsics for introspection
- A new pseudo-module `RUNTIME` providing a standard interface to the runtime system

Convenience Features

- C style literals using `0b`, `0u` and `0x` prefixing
- C style escape sequences `\0`, `\n`, `\r`, `\t`, `\\`, `\'` and `\"` in string literals
- C style postfix increment and decrement notation in statements but not expressions

Optional Features

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

- Pseudo-module `ASSEMBLER` for inline assembly code
- Various language defined optional pragmas

Outstanding Features (Phase II)

The following features will be revised, respectively added in phase II of the revision:

- Pseudo-module `COROUTINE` for coroutine based concurrency
- Pseudo-module `ACTOR` for actor based concurrency

Standard Library

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

ISO Features directly adopted from or similar to ISO Modula-2.