INTERNATIONAL STANDARD ISO/IEC 9899:1999
TECHNICAL CORRIGENDUM 1

Published 2001-09-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Programming languages — C

TECHNICAL CORRIGENDUM 1

Langages de programmation — C

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to International Standard ISO/IEC 9899:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

ICS 35.060 **Ref. No. ISO/IEC 9899:1999/Cor.1:2001(E)**

Printed in Switzerland

Technical Corrigendum 1

1. *Page xiii, Foreword*
   In paragraph 5 item 45, change **`VA_COPY`** to **`va_copy`**.

2. *Page 33, 6.2.5*
   In paragraph 3 change sentence 2 to:

   > [#3] [...] If a member of the basic execution character set is stored in a **`char`** object, its value is guaranteed to be nonnegative.

3. *Page 127, 6.7.8*
   In paragraph 19, attach a footnote to "the same subobject":

   > [*] Any initializer for the subobject which is overridden and so not used to initialize that subobject might not be evaluated at all.

4. *Page 151, 6.10.3*
   In paragraph 5, change "arguments" to "parameters".

5. *Page 184, 7.4.1.12*
   In paragraph 2, change "(as defined in 6.4.4.2)" to "(as defined in 6.4.4.1)".

6. *Page 187, 7.6*
   In paragraph 5, attach a footnote to the wording:

   > if and only if the implementation supports the floating-point exception by means of the functions in 7.6.2.

   where the footnote is:

   > [*] The implementation supports an exception if there are circumstances where a call to at least one of the functions in 7.6.2, using the macro as the appropriate argument, will succeed. It is not necessary for all the functions to succeed all the time.

7. *Page 190, 7.6.2.1*
   In paragraph 1, change the result type from **`void`** to **`int`**.

8. *Page 190, 7.6.2.1*
   In paragraph 2, replace "clears" with "attempts to clear".

9. *Page 190, 7.6.2.1*
   Add a new heading and paragraph 3:

   > **Returns**
   >
   > [#3] The **`feclearexcept`** function returns zero if the **`excepts`** argument is zero or if all the specified exceptions were successfully cleared. Otherwise, it returns a nonzero value.

10. *Page 191, 7.6.2.2*
    In paragraph 1, change the result type from **`void`** to **`int`**.

11. *Page 191, 7.6.2.2*
    In paragraph 2, replace "stores" with "attempts to store".

12. *Page 191, 7.6.2.2*
Add a new heading and paragraph 3:

> **Returns**
>
> [#3] The **`fegetexceptflag`** function returns zero if the representation was successfully stored. Otherwise, it returns a nonzero value.

13. *Page 191, 7.6.2.3*
In paragraph 1, change the result type from **`void`** to **`int`**.

14. *Page 191, 7.6.2.3*
In paragraph 2, replace "raises" with "attempts to raise".

15. *Page 191, 7.6.2.3*
Add a new heading and paragraph 3:

> **Returns**
>
> [#3] The **`feraiseexcept`** function returns zero if the **`excepts`** argument is zero or if all the specified exceptions were successfully raised. Otherwise, it returns a nonzero value.

16. *Page 191, 7.6.2.4*
In paragraph 1, change the result type from **`void`** to **`int`**.

17. *Page 191, 7.6.2.4*
In paragraph 2, replace "sets" with "attempts to set".

18. *Page 191, 7.6.2.4*
Add a new heading and paragraph 3:

> **Returns**
>
> [#3] The **`fesetexceptflag`** function returns zero if the **`excepts`** argument is zero or if all the specified flags were successfully set to the appropriate state. Otherwise, it returns a nonzero value.

19. *Page 193, 7.6.3.2*
Replace paragraph 3 by:

> [#3] The **`fesetround`** function returns zero if and only if the requested rounding direction was established.

20. *Page 194, 7.6.4.1*
In paragraph 1, change the result type from **`void`** to **`int`**.

21. *Page 194, 7.6.4.1*
In paragraph 2, replace "stores" with "attempts to store".

22. *Page 194, 7.6.4.1*
Add a new heading and paragraph 3:

> **Returns**
>
> [#3] The **`fegetenv`** function returns zero if the environment was successfully stored. Otherwise, it returns a nonzero value.

23. *Page 194, 7.6.4.3*
In paragraph 1, change the result type from **`void`** to **`int`**.

24. *Page 194, 7.6.4.3*
In paragraph 2, replace "establishes" with "attempts to establish".

25. *Page 194, 7.6.4.3*
Add a new heading and paragraph 3:

> **Returns**
>
> [#3] The **`fesetenv`** function returns zero if the environment was successfully established. Otherwise, it returns a nonzero value.

26. *Page 195, 7.6.4.4*
In paragraph 1, change the result type from **`void`** to **`int`**.

27. *Page 195, 7.6.4.4*
In paragraph 2, replace "saves" with "attempts to save", replace "installs" by "install", and replace "raises" by "raise".

28. *Page 195, 7.6.4.4*
Add a new heading and paragraph 3:

> **Returns**
>
> [#3] The **`feupdateenv`** function returns zero if all the actions were successfully carried out. Otherwise, it returns a nonzero value.

29. *Page 195, 7.6.4.4*
Change to existing paragraph 3, also renumbering it as 4:

> [#4] EXAMPLE Hide spurious underflow floating-point exceptions:

```
#include <fenv.h>
double f(double x)
{
   #pragma STDC FENV_ACCESS ON
   double result;
   fenv_t save_env;
   if (feholdexcept(&save_env))
      return /* indication of an environmental problem */;
   // compute result
   if (/* test spurious underflow */)
      if (feclearexcept(FE_UNDERFLOW))
         return /* indication of an environmental problem */;
   if (feupdateenv(&save_env))
      return /* indication of an environmental problem */;
   return result;
}
```

30. *Page 205, 7.11.2.1*
In paragraph 3, change all occurrences of **`int_currency_symbol`** to **`int_curr_symbol`**.

31. *Page 208, 7.11.2.1*
Append to paragraph 5:

> For **`int_p_sep_by_space`** and **`int_n_sep_by_space`**, the fourth character of **`int_curr_symbol`** is used instead of a space.

32. *Page 253, 7.17*
Add a new heading and paragraph 4:

> **Recommended Practice**
>
> [#4] The types used for **`size_t`** and **`ptrdiff_t`** should not have an integer conversion rank greater than that of **`signed long`** unless the implementation supports objects large enough to make this necessary.

33. *Page 259, 7.18.4*
Add a new paragraph 3:

> [#3] Each invocation of one of these macros shall expand to an integer constant expression suitable for use in **`#if`** preprocessing directives. The type of the expression shall have the same type as would an expression of the corresponding type converted according to the integer promotions. The value of the expression shall be that of the argument.

34. *Page 259, 7.18.4.1*
Remove the first paragraph and footnote 221.

35. *Page 259, 7.18.4.1*
Change to existing paragraph 2, also renumbering it as 1:

> [#1] The macro **`INT`***N*`_C(`*value*`)` shall expand to an integer constant expression corresponding to the type **`int_least`***N*`_t`. The macro **`UINT`***N*`_C(`*value*`)` shall expand to an integer constant expression corresponding to the type **`uint_least`***N*`_t`. For example, if **`uint_least64_t`** is a name for the type **`unsigned long long int`**, then **`UINT64_C(0x123)`** might expand to the integer constant **`0x123ULL`**.

36. *Page 260, 7.18.4.2*
In paragraph 1, change both occurrences of "integer constant" to "integer constant expression".

37. *Page 274, 7.19.6.1*
In paragraph 4 item 2, change "decimal integer" to "nonnegative decimal integer".

38. *Page 279, 7.19.6.1*
Change paragraph 12 to:

> [#12] For **`a`** and **`A`** conversions, if **`FLT_RADIX`** is not a power of 2 and the result is not exactly representable in the given precision, the result should be one of the two adjacent numbers in hexadecimal floating style with the given precision, with the extra stipulation that the error should have a correct sign for the current rounding direction.

39. *Page 281, 7.19.6.2*
In paragraph 3 item 2, change "nonzero decimal integer" to "decimal integer greater than zero".

40. *Page 293, 7.19.6.12*
Change **`vsprintf`** to **`vsnprintf`** in the synopsis.

41. *Page 294, 7.19.6.14*
In paragraph 3, change the reference from **`vscanf`** to **`vsscanf`**.

42. *Page 308, 7.20.1.3*
Change paragraph 8 to:

> [#8] If the subject sequence has the hexadecimal form, **FLT_RADIX** is not a power of 2, and the result is not exactly representable, the result should be one of the two numbers in the appropriate internal format that are adjacent to the hexadecimal floating source value, with the extra stipulation that the error should have a correct sign for the current rounding direction.

43. *Page 349, 7.24.2.1*
In paragraph 4 item 2, change "decimal integer" to "nonnegative decimal integer".

44. *Page 354, 7.24.2.1*
Change paragraph 12 to:

> [#12] For **a** and **A** conversions, if **FLT_RADIX** is not a power of 2 and the result is not exactly representable in the given precision, the result should be one of the two adjacent numbers in hexadecimal floating style with the given precision, with the extra stipulation that the error should have a correct sign for the current rounding direction.

45. *Page 355, 7.24.2.2*
In paragraph 3 item 2, change "nonzero decimal integer" to "decimal integer greater than zero".

46. *Page 372, 7.24.4.1.1*
Change paragraph 8 to:

> [#8] If the subject sequence has the hexadecimal form, **FLT_RADIX** is not a power of 2, and the result is not exactly representable, the result should be one of the two numbers in the appropriate internal format that are adjacent to the hexadecimal floating source value, with the extra stipulation that the error should have a correct sign for the current rounding direction.

47. *Page 388, 7.24.6.3.2*
In paragraph 4, change the label of the case "*positive*" to "*between* 1 *and* **n** *inclusive*".

48. *Page 419, B.5*
Change:

```
void feclearexcept(int excepts);
```

to:

```
int feclearexcept(int excepts);
```

49. *Page 419, B.5*
Change:

```
void fegetexceptflag(fexcept_t *flagp, int excepts);
```

to:

```
int fegetexceptflag(fexcept_t *flagp, int excepts);
```

50. *Page 419, B.5*
Change:

```
void feraiseexcept(int excepts);
```

to:

```
int feraiseexcept(int excepts);
```

51. *Page 419, B.5*
Change:

```
void fesetexceptflag(const fexcept_t *flagp,
int excepts);
```

to:

```
int fesetexceptflag(const fexcept_t *flagp,
int excepts);
```

52. *Page 419, B.5*
Change:

```
void fegetenv(fenv_t *envp);
```

to:

```
int fegetenv(fenv_t *envp);
```

53. *Page 419, B.5*
Change:

```
void fesetenv(const fenv_t *envp);
```

to:

```
int fesetenv(const fenv_t *envp);
```

54. *Page 419, B.5*
Change:

```
void feupdateenv(const fenv_t *envp);
```

to:

```
int feupdateenv(const fenv_t *envp);
```

55. *Page 433, B.23*
Change:

```
int wmemcmp(wchar_t * restrict s1,
const wchar_t * restrict s2, size_t n);
```

to:

```
int wmemcmp(const wchar_t *s1, const wchar_t *s2,
size_t n);
```

56. *Page 433, B.23*
wmemcmp should immediately follow **wcsncpy**, **wmemcpy** and **wmemmove** should immediately follow **wcsxfrm**, and **wcslen** should immediately follow **wmemchr**.

57. *Page 485, Annex I*
In paragraph 2 item 11, change "enumeration type" to "enumerated type".