### 16.1.2 Predefined functions

TTCN‑3 contains a number of predefined (built-in) functions that need not be declared before use. These are summarized in table 14.

Table 14: List of TTCN‑3 predefined functions

| Category | Function | Keyword |
| --- | --- | --- |
| **Conversion functions**  | Convert **integer** value to **charstring** value | [**int2char**](#annex_PredefinedFunctions_int2char) |
| Convert **integer** value to **universal** **charstring** value | [**int2unichar**](#annex_PredefinedFunctions_int2unichar) |
| Convert **integer** value to **bitstring** value | [**int2bit**](#annex_PredefinedFunctions_int2bit) |
| Convert **integer** value to enumerated value | [**int2enum**](#annex_PredefinedFunctions_int2enum) |
| Convert **integer** value to **hexstring** value | [**int2hex**](#annex_PredefinedFunctions_int2hex) |
| Convert **integer** value to **octetstring** value | [**int2oct**](#annex_PredefinedFunctions_int2oct) |
| Convert **integer** value to **charstring** value  | [**int2str**](#annex_PredefinedFunctions_int2str) |
| Convert **integer** value to **float** value | [**int2float**](#annex_PredefinedFunctions_int2float) |
| Convert **float** value to **integer** value | [**float2int**](#annex_PredefinedFunctions_float2int) |
| Convert **charstring** value to **integer** value | [**char2int**](#annex_PredefinedFunctions_char2int) |
| Convert **charstring** value to **octetstring** value | [**char2oct**](#annex_PredefinedFunctions_char2oct) |
| Convert **universal charstring** value to **octetstring** value | [**unichar2oct**](#annex_PredefinedFunctions_unichar2oct) |
| Convert **universal** **charstring** value to **integer** value | [**unichar2int**](#annex_PredefinedFunctions_unichar2int) |
| Convert **bitstring** value to **integer** value | [**bit2int**](#annex_PredefinedFunctions_bit2int) |
| Convert **bitstring** value to **hexstring** value | [**bit2hex**](#annex_PredefinedFunctions_bit2hex) |
| Convert **bitstring** value to **octetstring** value | [**bit2oct**](#annex_PredefinedFunctions_bit2oct) |
| Convert **bitstring** value to **charstring** value | [**bit2str**](#annex_PredefinedFunctions_bit2str) |
| Convert **hexstring** value to **integer** value | [**hex2int**](#annex_PredefinedFunctions_bit2int) |
| Convert **hexstring** value to **bitstring** value | [**hex2bit**](#annex_PredefinedFunctions_hex2bit) |
| Convert **hexstring** value to **octetstring** value | [**hex2oct**](#annex_PredefinedFunctions_hex2oct) |
| Convert **hexstring** value to **charstring** value | [**hex2str**](#annex_PredefinedFunctions_hex2str) |
| Convert **octetstring** value to **integer** value | [**oct2int**](#annex_PredefinedFunctions_oct2int) |
| Convert **octetstring** value to **bitstring** value | [**oct2bit**](#annex_PredefinedFunctions_oct2bit) |
| Convert **octetstring** value to **hexstring** value | [**oct2hex**](#annex_PredefinedFunctions_oct2hex) |
| Convert **octetstring** value to **charstring** value | [**oct2str**](#annex_PredefinedFunctions_oct2str) |
| Convert **octetstring** value to **charstring** value, version II | [**oct2char**](#annex_PredefinedFunctions_oct2char) |
| Convert **octetstring** value to **universal charstring** value | [**oct2unichar**](#annex_PredefinedFunctions_oct2unichar) |
| Convert **charstring** value to **integer** value  | [**str2int**](#annex_PredefinedFunctions_str2int) |
| Convert **charstring** value to **hexstring** value | [**str2hex**](#annex_PredefinedFunctions_str2hex) |
| Convert **charstring** value to **octetstring** value | [**str2oct**](#annex_PredefinedFunctions_str2oct) |
| Convert **charstring** value to **float** value | [**str2float**](#annex_PredefinedFunctions_str2float) |
| Convert enumerated value to **integer** value | [**enum2int**](#annex_PredefinedFunctions_enum2int) |
| Convert value or template to **universal charstring** value | [**any2unistr**](#annex_PredefinedFunctions_any2unistr) |
| **Length/size functions** | Return the length of a value or template of any string type, **record of**, **set of** or **array** | [**lengthof**](#annex_PredefinedFunctions_lengthof) |
| Return the number of elements in a value or a template of a **record** or **set** | [**sizeof**](#annex_PredefinedFunctions_sizeof) |
| **Presence checking functions** | Determine if an optional field in a **record** or **set** value or template is present or is assigned a matching mechanism that cannot match an ommitted field (i.e. none of **omit**, *AnyValueOrNone* or **ifpresent**) | [**ispresent**](#annex_PredefinedFunctions_ispresent) |
| Determine which choice has been selected in a **union** value or template | [**ischosen**](#annex_PredefinedFunctions_ischosen) |
| Determine if a template evaluates to a concrete value | [**isvalue**](#annex_PredefinedFunctions_isvalue) |
| Determine if a template is uninitialized or not | [**isbound**](#annex_PredefinedFunctions_isbound) |
| Determine if a template contains certain matching mechanism | [**istemplatekind**](#annex_PredefinedFunctions_isTemplateKind) |
| **String/list handling functions** | Returns part of the input string matching the specified pattern group within a character pattern | [**regexp**](#annex_PredefinedFunctions_regexp) |
| Returns the specified portion of the input string/list value or template | [**substr**](#annex_PredefinedFunctions_substr) |
| Replaces a substring of a string with or inserts the input string into a string, and similarly for lists | [**replace**](#annex_PredefinedFunctions_replace) |
| **Codec functions** | Encode a value into a bitstring | [**encvalue**](#annex_PredefinedFunctions_encvalue) |
| Decode a bitstring into a value | [**decvalue**](#annex_PredefinedFunctions_decvalue) |
| Encode a value into a universal charstring | [**encvalue\_unichar**](#annex_PredefinedFunctions_encvalueUchar) |
| Decode a universal charstring into a value | [**decvalue\_unichar**](#annex_PredefinedFunctions_decvalueUchar) |
| Retrieve the type of string encoding | [**get\_stringencoding**](#annex_PredefinedFunctions_getStringenc) |
| Remove BOMs of UCS encoding schemes | [**remove\_bom**](#annex_PredefinedFunctions_removeBOM) |
| **Other functions** | Generate a random float number | [**rnd**](#annex_PredefinedFunctions_rnd) |
| Returns the name of the currently executing test case | [**testcasename**](#annex_PredefinedFunctions_testcasename) |
| Returns the host id of the test component or module | [**hostid**](#annex_PredefinedFunctions_hostid) |

***Syntactical Structure***

**int2char** "(" *SingleExpression* ")" |

**int2unichar** "(" *SingleExpression* ")" |

**int2bit** "(" *SingleExpression* "," *SingleExpression* ")" |

**int2enum** "(" *SingleExpression* "," *SingleExpression* ")" |

**int2hex** "(" *SingleExpression* "," *SingleExpression* ")" |

**int2oct** "(" *SingleExpression* "," *SingleExpression* ")" |

**int2str** "(" *SingleExpression* ")" |

**int2float** "(" *SingleExpression* ")" |

**float2int** "(" *SingleExpression* ")" |

**char2int** "(" *SingleExpression* ")" |

**char2oct** "(" *SingleExpression* ")" |

**unichar2int** "(" *SingleExpression* ")" |

**unichar2oct** "(" *SingleExpression* ["," *SingleExpression*] ")" |

**bit2int** "(" *SingleExpression* ")" |

**bit2hex** "(" *SingleExpression* ")" |

**bit2oct** "(" *SingleExpression* ")" |

**bit2str** "(" *SingleExpression* ")" |

**hex2int** "(" *SingleExpression* ")" |

**hex2bit** "(" *SingleExpression* ")" |

**hex2oct** "(" *SingleExpression* ")" |

**hex2str** "(" *SingleExpression* ")" |

**oct2int** "(" *SingleExpression* ")" |

**oct2bit** "(" *SingleExpression* ")" |

**oct2hex** "(" *SingleExpression* ")" |

**oct2str** "(" *SingleExpression* ")" |

**oct2char** "(" *SingleExpression* ")" |

**oct2unichar** "(" *SingleExpression* ["," *SingleExpression*] ")" |

**str2int** "(" *SingleExpression* ")" |

**str2hex** "(" *SingleExpression* ")" |

**str2oct** "(" *SingleExpression* ")" |

**str2float** "(" *SingleExpression* ")" |

**enum2int** "(" *SingleExpression* ")" |

**any2unistr** "(" *SingleExpression* ")" |

**lengthof** "(" *TemplateInstance* ")" |

**sizeof** "(" *TemplateInstance* ")" |

**ispresent** "(" *TemplateInstance* ")" |

**ischosen** "(" *TemplateInstance* ")" |

**isvalue** "(" *TemplateInstance* ")" |

**isbound** "(" *TemplateInstance* ")" |

**istemplatekind** "(" *TemplateInstance* "," *TemplateInstance* ")" |

**regexp** ["**@nocase**"] "(" *TemplateInstance* "," *TemplateInstance* "," *SingleExpression* ")" |

**substr** "(" *TemplateInstance* "," *SingleExpression* "," *SingleExpression* ")" |

**replace** "(" *SingleExpression* "," *SingleExpression* "," *SingleExpression* "," *SingleExpression* ")" |

**encvalue** "("*TemplateInstance* **[**"**,**" *SingleExpression***]** ")" |

**decvalue** "("*SingleExpression*","*SingleExpression* **[**"," *SingleExpression***]** ")" |

**encvalue\_unichar** "("*TemplateInstance* ["," *SingleExpression*] ["," *SingleExpression*]")" |

**decvalue\_unichar** "("*SingleExpression*","*SingleExpression*[","*SingleExpression***]** ["**,**" *SingleExpression*]")" |

**get\_stringencoding** "("*SingleExpression*")"

**remove\_bom** ("*SingleExpression*")"

**rnd** "(" [ *SingleExpression* ] ")" |

**testcasename** "()"

**hostid** "(" [ *SingleExpression* ]")"

***Semantic Description***

The description of predefined functions is given in annex C.

***Restrictions***

In addition to the general static rules of TTCN‑3 given in clause 5, the following restrictions apply:

a) When a predefined function is invoked:

1) the number of the actual parameters shall be the same as the number of the formal parameters; and

2) each actual parameter shall evaluate to an element of its corresponding formal parameter's type; and

3) all actual in and inout parameters shall be initialized with the following exceptions:

* the actual in parameter passed to the predefined functions isvalue, ischosen, ispresent and isbound may be uninitialized or even contain non-evaluable reference expressions;
* any\_string\_or\_sequence\_type parameters of the functions lengthof, substr and replace may be partially initialized;
* the invalue parameter of the any2unistr function may be uninitialized or partially initialized;
* the encoded\_value parameter of the decvalue and decvalue\_unichar function may be uninitialized.

b) Restrictions on invoking functions from specific places are described in clause 16.1.4.

***Examples***

 **var hexstring** v\_h:= **bit2hex** ('111010111'B);

 **var octetstring** v\_o:= **substr** ('01AB23CD'O, 1, 2);