### 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** |
| Convert **integer** value to **universal** **charstring** value | **int2unichar** |
| Convert **integer** value to **bitstring** value | **int2bit** |
| Convert **integer** value to enumerated value | **int2enum** |
| Convert **integer** value to **hexstring** value | **int2hex** |
| Convert **integer** value to **octetstring** value | **int2oct** |
| Convert **integer** value to **charstring** value  | **int2str** |
| Convert **integer** value to **float** value | **int2float** |
| Convert **float** value to **integer** value | **float2int** |
| Convert **charstring** value to **integer** value | **char2int** |
| Convert **charstring** value to **octetstring** value | **char2oct** |
| Convert **universal charstring** value to **octetstring** value | **unichar2oct** |
| Convert **universal** **charstring** value to **integer** value | **unichar2int** |
| Convert **bitstring** value to **integer** value | **bit2int** |
| Convert **bitstring** value to **hexstring** value | **bit2hex** |
| Convert **bitstring** value to **octetstring** value | **bit2oct** |
| Convert **bitstring** value to **charstring** value | **bit2str** |
| Convert **hexstring** value to **integer** value | **hex2int** |
| Convert **hexstring** value to **bitstring** value | **hex2bit** |
| Convert **hexstring** value to **octetstring** value | **hex2oct** |
| Convert **hexstring** value to **charstring** value | **hex2str** |
| Convert **octetstring** value to **integer** value | **oct2int** |
| Convert **octetstring** value to **bitstring** value | **oct2bit** |
| Convert **octetstring** value to **hexstring** value | **oct2hex** |
| Convert **octetstring** value to **charstring** value | **oct2str** |
| Convert **octetstring** value to **charstring** value, version II | **oct2char** |
| Convert **octetstring** value to **universal charstring** value | **oct2unichar** |
| Convert **charstring** value to **integer** value  | **str2int** |
| Convert **charstring** value to **hexstring** value | **str2hex** |
| Convert **charstring** value to **octetstring** value | **str2oct** |
| Convert **charstring** value to **float** value | **str2float** |
| Convert enumerated value to **integer** value | **enum2int** |
| Convert value or template to **charstring** value | **Ttcn2char** |
| **Length/size functions** | Return the length of a value or template of any string type, **record of**, **set of** or **array** | **lengthof** |
| Return the number of elements in a value or a template of a **record** or **set** | **sizeof** |
| **Presence checking functions** | Determine if an optional field in a **record** or **set** value or template is present | **ispresent** |
| Determine which choice has been selected in a **union** value or template | **ischosen** |
| Determine if a template evaluates to a concrete value | **isvalue** |
| Determine if a template is uninitialized or not | **isbound** |
| **String/list handling functions** | Returns part of the input string matching the specified pattern group within a character pattern | **regexp** |
| Returns the specified portion of the input string/list value or template | **substr** |
| Replaces a substring of a string with or inserts the input string into a string, and similarly for lists | **replace** |
| **Codec functions** | Encode a value into a bitstring | **encvalue** |
| Decode a bitstring into a value | **decvalue** |
| Encode a value into a universal charstring | **encvalue\_unichar** |
| Decode a universal charstring into a value | **decvalue\_unichar** |
| **Other functions** | Generate a random float number | **rnd** |
| Returns the name of the currently executing test case | **testcasename** |
| Returns the host id of the test component or module | **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* ")" |

**ttcn2char** "(" *SingleExpression* ")" |

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

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

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

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

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

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

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

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

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

**encvalue** "("*TemplateInstance*")" |

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

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

**decvalue\_unichar** "("*SingleExpression*","*SingleExpression* **[,** *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 and inout parameter passed to the predefined functions isvalue, ischosen, ispresent and isbound may be uninitialized or even contain non-evaluable reference expressions
* the any\_string\_or\_sequence\_type parameters of the functions lengthof, substr and replace may be partially initialized.
* the invalue parameter of the ttcn2char function may be uninitialized or partially initialized

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

***Examples***

 **var hexstring** h:= **bit2hex** ('111010111'B);

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

## C.1.33 Value or template to charstring

 **ttcn2char**(**in template** any\_type invalue) **return charstring**

This function converts the content of a value or template to a single **charstring**. The resulting **charstring** is the same as the string produced by the log operation containing the same operand as the one passed to the **ttcn2char** function. The value or template passed as a parameter to the **ttcn2char** function may be completely or partially initialized.

The general error causes in clause 16.1.2 apply.

EXAMPLE:

 **var integer** v\_int1 :=5, v\_int2;

 **var template integer vmw\_int** :=?;

 **var charstring** v\_chr1, v\_chr2, v\_chr3;

 v\_chr1 := **ttcn2char(v\_int1)**; // after the assignment v\_chr1 will be "5"

 v\_chr2 := **ttcn2char(v\_int2)**; // after the assignment v\_chr2 will be "UNINITIALIZED"

 v\_chr3 := **ttcn2char(vmw\_int)**; // after the assignment v\_chr3 will be "?"