Klocwork C/C++ Abstract Syntax Tree API

1.9.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

Deprecated List

Member ktc_error_isEnabled (p. **55) (const char ∗error_id)**

Use kwapi_cfgparam_errorIsEnabled

Member ktc_getPointedType (p. **36) (ktc_languageType_t ptr_type)**

Use **ktc_sema_getPointedType** (p. 25)

Member ktc_languageType_t (p. **12)**

Use **ktc_semanticInfo_t** (p. 12) type

Member ktc_languageTypeIsBuiltin (p. **36) (ktc_languageType_t type, int builtin_code)**

Use **ktc_sema_getBuiltinCode** (p. 23) instead

Member ktc_languageTypeIsPointer (p. **37) (ktc_languageType_t type)**

Use **ktc_sema_isPointer** (p. 31)

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 Obtaining configuration parameters for an error

Typedefs

- typedef struct ParameterNode * **kwapi_cfgparam_t**

Functions

- **kwapi_cfgparam_t kwapi_cfgparam_getRootParameterList** (const char *error)
- **kwapi_cfgparam_t kwapi_cfgparam_getListNodeByName** (**kwapi_cfgparam_t**, const char *name)
- const char * **kwapi_cfgparam_getName** (**kwapi_cfgparam_t**)
- const char * **kwapi_cfgparam_getType** (**kwapi_cfgparam_t**)
- unsigned int **kwapi_cfgparam_getListLength** (**kwapi_cfgparam_t**)
- **kwapi_cfgparam_t kwapi_cfgparam_getListNodeByIndex** (**kwapi_cfgparam_t**, unsigned int idx)
- int **kwapi_cfgparam_isParameter** (**kwapi_cfgparam_t**)
- const char * **kwapi_cfgparam_getParameterValue** (**kwapi_cfgparam_t**)
- const char * **kwapi_cfgparam_getParameterValueFromList** (**kwapi_cfgparam_t** parent, const char *paramName)
- const char * **kwapi_cfgparam_getConfigurationParameter** (const char *errorId, const char *paramName)
- const char *const * **kwapi_cfgparam_getCheckerErrors** (const char *checker_id)
- const char * **ktc_error_getConfigurationParameter** (const char *errorId, const char *paramName)
- int **kwapi_cfgparam_errorIsEnabled** (const char *error_id)

4.1.1 Detailed Description

Checker configuration XML file may store parameters that checker may access using functions from this section.

```
<error id="MYDEFECT" message="my message" severity="3" enabled="true">
  <parameter name="myparameter" value="yes"/>
</error>
```

4.1.2 Typedef Documentation

4.1.2.1 typedef struct ParameterNode* kwapi_cfgparam_t

4.1.3 Function Documentation

4.1.3.1 const char* ktc_error_getConfigurationParameter (const char * *errorId,* const char * *paramName*)

4.1.3.2 int kwapi_cfgparam_errorIsEnabled (const char * *error_id*)

Check that particular error was enabled in the configuration file

Parameters

error↩ _id	identifier of an error (the same id as in <error> tag in configuration file)

Returns

0 if error was disabled or was not present in the configuration file, 1 otherwise

4.1.3.3 const char* const* kwapi_cfgparam_getCheckerErrors (const char * *checker_id*)

Obtain a pointer to the internal array of error ids that are produced by a given checker

Parameters

checker↩ _id	Identifier of a checker

Returns

a pointer to a zero terminated array of strings containing error identifiers or 0 if checker was not configured

4.1.3.4 const char* kwapi_cfgparam_getConfigurationParameter (const char * *errorId,* const char * *paramName*)

Quick access to parameters for errors that do not require parameterlist'.

Remarks

Essentialy gets a parameter by name from a root list `kwapi_cfgparam_t el = kwapi_cfgparam↩ _getRootParameterList(errorId); if (el) { return kwapi_cfgparam_get↩ ParameterValueFromList(el, paramName); } else { return 0; }`

4.1.3.5 unsigned int kwapi_cfgparam_getListLength (kwapi_cfgparam_t)

Returns 0 for parameter's, list length for parameterlist's

4.1.3.6 kwapi_cfgparam_t kwapi_cfgparam_getListNodeByIndex (kwapi_cfgparam_t , unsigned int *idx*)

Returns 0 for parameters, parameter node at index idx for lists, or 0 if index is outside of bounds

4.1.3.7 kwapi_cfgparam_t kwapi_cfgparam_getListNodeByName (kwapi_cfgparam_t , const char * *name*)

Extract parameter node from a list by name,

Returns

0 if there is no parameter node with such name, or parameter node is not a parameterlist

4.1.3.8 const char* kwapi_cfgparam_getName (kwapi_cfgparam_t)

Get name of parameter node, or 0 if parameter has no name

4.1.3.9 const char* kwapi_cfgparam_getParameterValue (kwapi_cfgparam_t)

Return parameter string value for parameters, 0 for parameterlists

4.1.3.10 const char* kwapi_cfgparam_getParameterValueFromList (kwapi_cfgparam_t *parent,* const char * *paramName*)

Get parameter value from a parameter in the list by parameter name

Remarks

essentially a code: `kwapi_cfgparam_t el = kwapi_cfgparam_getListNodeByName(parent); if (el) { return kwapi_cfgparam_getParameterValue(paramName); } else { return 0; }`

4.1.3.11 kwapi_cfgparam_t kwapi_cfgparam_getRootParameterList (const char * *error*)

Get handler of root parameter list: it contains all parameter's and parameterlist's at the top level of <error> tag

4.1.3.12 const char* kwapi_cfgparam_getType (kwapi_cfgparam_t)

Get type of parameter node , or 0 if there is no type specified

4.1.3.13 int kwapi_cfgparam_isParameter (kwapi_cfgparam_t)

Returns 1 if node is parameter, 0 if it is parameterlist

4.2 Basic Abstract Syntax Tree traversal and checking routines

Macros

- #define **KTC_CUSTOM_TYPES**
- #define **ktc_require**(t, ttype) if (!**ktc_isTreeType**((t),(ttype))) { return 0; }

Typedefs

- typedef union CTree_Node * **ktc_tree_t**
- typedef union rs_element * **ktc_semanticInfo_t**
- typedef union rs_element * **ktc_languageType_t**
- typedef struct ktc_string_t_ * **ktc_string_t**
- typedef int64_t **ktc_long_long_t**
- typedef int **ktc_treeType_t**
- typedef int **ktc_childId_t**

Functions

- int **ktc_isTreeType** (**ktc_tree_t** t, **ktc_treeType_t** ttype)
- const char * **ktc_treeType_getName** (**ktc_tree_t** ttype)
- **ktc_tree_t ktc_proceed** (**ktc_tree_t** t, **ktc_childId_t** child_id)
- void **ktc_forAllSubtreeNodes** (**ktc_tree_t** t, int(*callback)(**ktc_tree_t**, void *), void *data)
- void **ktc_sema_forAllSubtreeNodes** (**ktc_semanticInfo_t** si, int(*callback)(**ktc_tree_t**, void *), void *data)
- int **ktc_isMacroExpansion** (**ktc_tree_t** t)
- int **ktc_isMacroExpansion2** (**ktc_tree_t** t)

4.2.1 Detailed Description

4.2.2 Macro Definition Documentation

4.2.2.1 #define KTC_CUSTOM_TYPES

4.2.2.2 #define ktc_require(*t, ttype*) if (!ktc_isTreeType((t),(ttype))) { return 0; }

Require that node has a particular kind and return from handler with 0 (not applicable) otherwise

Parameters

t	subtree root to check
ttype	required tree type

4.2.3 Typedef Documentation

4.2.3.1 typedef int ktc_childId_t

Type definition for child enumeration constants

4.2.3.2 typedef union rs_element* ktc_languageType_t

Opaque type representing C/C++ language type description Can safely casted to **ktc_semanticInfo_t** (p. 12) pointer type

Deprecated Use **ktc_semanticInfo_t** (p. 12) type

4.2.3.3 typedef int64_t ktc_long_long_t

Type definition for the largest available integer type

4.2.3.4 typedef union rs_element* ktc_semanticInfo_t

Opaque type for accessing semantic description of symbols. Can be safely casted to **ktc_languageType_t** (p. 12) pointer type described below

4.2.3.5 typedef struct ktc_string_t_* ktc_string_t

Opaque type for representing strings returned by KAST custom functions

4.2.3.6 typedef union CTree_Node* ktc_tree_t

Opaque type representing an Abstract Syntax subtree, starting with one node

4.2.3.7 typedef int ktc_treeType_t

Type definition for tree type

4.2.4 Function Documentation

4.2.4.1 void ktc_forAllSubtreeNodes (ktc_tree_t *t*, int(*)(ktc_tree_t, void *) *callback*, void * *data*)

From given subtree root, apply the given callback to all subtree nodes

Parameters

t	subtree root to start with
callback	callback to be applied The provided callback should return: 1 if no further traversal is required at all, 2 if no further children traversal is required, but Next links should be explored, and 0 otherwise. Node stack related routines should not be used from a callback

4.2.4.2 int ktc_isMacroExpansion (ktc_tree_t *t*)

Checks that subtree comes from a macro expansion.

Parameters

t	subtree root to test

Returns

1 if 't' comes from a macro expansion, 0 otherwise

4.2.4.3 int ktc_isMacroExpansion2 (ktc_tree_t *t*)

- Checks that one of the subtrees come from a macro expansion.

Parameters

t	subtree root to test

- **Returns**
- 1 if 't' comes from a macro expansion, 0 otherwise

4.2.4.4 int ktc_isTreeType (ktc_tree_t *t,* ktc_treeType_t *ttype*)

Checks that root of subtree has particular node type. Node type is a generalization of node kind. Node kind is an instance of one of node types. Node type, in turn, may be a subtype of another node type

Parameters

t	subtree root to test
ttype	tree type identifier

Returns

1 if 't' belongs to the type 'ttype', 0 otherwise

4.2.4.5 ktc_tree_t ktc_proceed (ktc_tree_t *t,* ktc_childId_t *child_id*)

From given subtree root, proceed to one of the edges designated by 'child_id'

Parameters

t	subtree root to start with
child_id	child edge identifier

Returns

subtree root that is a child of t via link designated by child_id

See also

gen-ktcAPI.h (p. 187)

4.2.4.6 void ktc_sema_forAllSubtreeNodes (ktc_semanticInfo_t *si,* int(∗)(ktc_tree_t, void ∗) *callback,* void ∗ *data*)

From subtree that corresponds to the given semantic element, apply the given callback to all subtree nodes

Parameters

si	semantic info
callback	callback to be applied The provided callback should return: 1 if no further traversal is required at all, 2 if no further children traversal is required, but Next links should be explored, and 0 otherwise. Node stack related routines should not be used from a callback

4.2.4.7 const char∗ ktc_treeType_getName (ktc_tree_t *ttype*)

Get name of tree type for particular tree vertex (without 'tid_' prefix). Mainly for debug purposes.

4.3 Accessing node stack

Functions

- int **ktc_nodeStackTop** (void)
- **ktc_tree_t ktc_nodeStackGet** (int n)

4.3.1 Detailed Description

4.3.2 Function Documentation

4.3.2.1 ktc_tree_t ktc_nodeStackGet (int *n*)

get node from n-th position at the stack

Parameters

n	an index in the stack in the range [0; top-1] where top is a value, returned by

See also

ktc_nodeStackTop (p. 15)

Returns

node that is located at the n-th index in the stack, 0 if n is outside of range

4.3.2.2 int ktc_nodeStackTop (void)

return top index of the node stack

4.4 Setting and clearing handlers for events during tree traversal

Typedefs

- typedef int(∗ **ktc_treeHook_t**) (**ktc_tree_t**)
- typedef void(∗ **ktc_eventHook_t**) (void)

Functions

- void **ktc_registerTreeHook** (int tree_event, **ktc_treeType_t** tt, **ktc_treeHook_t** p_hook)
- void **ktc_registerStartTraverseHook** (**ktc_eventHook_t** p_hook)
- void **ktc_registerSaveContextHook** (**ktc_eventHook_t** p_hook)
- void **ktc_registerRestoreContextHook** (**ktc_eventHook_t** p_hook)
- void **ktc_registerStopTraverseHook** (**ktc_eventHook_t** p_hook)
- void **ktc_unregisterTreeHook** (int tree_event, **ktc_treeType_t** tt, **ktc_treeHook_t** p_hook)

Variables

- int **KTC_TREE_EVENT_ON_ENTER**
- int **KTC_TREE_EVENT_ON_NEXT**
- int **KTC_TREE_EVENT_ON_LEAVE**

4.4.1 Detailed Description

4.4.2 Typedef Documentation

4.4.2.1 typedef void(∗ ktc_eventHook_t) (void)

4.4.2.2 typedef int(∗ ktc_treeHook_t) (ktc_tree_t)

4.4.3 Function Documentation

4.4.3.1 void ktc_registerRestoreContextHook (ktc_eventHook_t *p_hook*)

4.4.3.2 void ktc_registerSaveContextHook (ktc_eventHook_t *p_hook*)

4.4.3.3 void ktc_registerStartTraverseHook (ktc_eventHook_t *p_hook*)

4.4.3.4 void ktc_registerStopTraverseHook (ktc_eventHook_t *p_hook*)

4.4.3.5 void ktc_registerTreeHook (int *tree_event,* ktc_treeType_t *tt,* ktc_treeHook_t *p_hook*)

Register hook function for a certain kind of subtrees

Parameters

tree_event	tree event identifier
tt	tree type to hook on, may be tk_Any to hook on all tree kinds at once
p_hook	a pointer to a handler function

4.4.3.6 void ktc_unregisterTreeHook (int *tree_event,* ktc_treeType_t *tt,* ktc_treeHook_t *p_hook*)

Deregister hook from a hook table

Parameters

tree_event	tree event identifier
tt	tree type
p_hook	a hook address to deregister

4.4.4 Variable Documentation

4.4.4.1 int KTC_TREE_EVENT_ON_ENTER

'On Enter' traversal event identifier. Defines an event when some node is entered during depth-first traversal

4.4.4.2 int KTC_TREE_EVENT_ON_LEAVE

'On Leave' traversal event identifier. Defines an event when all links of a node (including Next) were traversed and traversal is about to enter a sibling node

4.4.4.3 int KTC_TREE_EVENT_ON_NEXT

'On Next' traversal event identifier. Defines an event when some node is left via 'Next' link to the next node in the list of nodes. Next link is always processed after all children's subtrees were fully traversed

4.5 Access to semantic information

Functions

- **ktc_semanticInfo_t ktc_getSemanticInfo** (**ktc_tree_t** t)
- **ktc_semanticInfo_t ktc_getCalledFunction** (**ktc_tree_t** t)
- **ktc_semanticInfo_t ktc_getAssociatedScope** (**ktc_tree_t** t)
- **ktc_semanticInfo_t ktc_sema_getOverridenMethod** (**ktc_tree_t** t)
- int **ktc_sema_isSameFunctions** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isSameVariables** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_haveSameSignature** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_haveSameFunctionType** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isSameClass** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isSameEnum** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isSameScope** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isNone** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isBitfield** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isScope** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isClass** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isUnion** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isTemplate** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isEnum** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isEnumConstant** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isType** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isVariable** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPointer** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isArray** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isReference** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFunctionType** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFunction** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isBuiltin** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isSpecialization** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isInstantiation** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isInstantiatedFunction** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isIntegerValue** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isObjectValue** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFunctionTemplateSet** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFunctionTemplate** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isNamespaceAlias** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isUsingDirective** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isUsingDeclaration** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isUsing** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isNamespace** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPOD** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFriend** (**ktc_semanticInfo_t** si)
- int **ktc_sema_hasMethods** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getClassTag** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isAnonymous** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isTypeParameter** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isVirtual** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPrivate** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isProtected** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPublic** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPureVirtual** (**ktc_semanticInfo_t** si)

- int **ktc_sema_isConstructor** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isDestructor** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isOperatorFunction** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isLocal** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isGlobal** (**ktc_semanticInfo_t** si)
- **ktc_tree_t ktc_sema_getVariableInitializer** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getVariableType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getVariableValue** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getNumberOfArguments** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFormalArgument** (**ktc_semanticInfo_t** si, int n)
- **ktc_semanticInfo_t ktc_sema_getReturnType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFunctionType** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getBuiltinCode** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getCVQualifiers** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isImmutable** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getPointedType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getArrayElementType** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getArraySize** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getReferencedType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getDefinedType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_skipTypedefs** (**ktc_semanticInfo_t** si, int ∗cvq)
- const char ∗ **ktc_sema_getIdentifier** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getIdentifierNo** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getNumber** (**ktc_semanticInfo_t** si)
- const char ∗ **ktc_sema_getTypedefedName** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_stripCVQ** (**ktc_semanticInfo_t** si)
- char ∗ **ktc_sema_getQualifiedName** (**ktc_semanticInfo_t** si)
- char ∗ **ktc_sema_getTypeName** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getScope** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getNumberOfBaseInfo** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getBaseInfo** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isBaseVirtual** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isBasePublic** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isBaseProtected** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isBasePrivate** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isConstMethod** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFirstByName** (**ktc_semanticInfo_t** scope, const char ∗name)
- **kw_array_t** ∗ **ktc_sema_getAllByName** (**ktc_semanticInfo_t** scope, const char ∗name)
- **ktc_semanticInfo_t ktc_sema_findFirstByName** (**ktc_semanticInfo_t** scope, const char ∗name)
- **ktc_semanticInfo_t ktc_sema_getGlobalScope** ()
- **kw_array_t** ∗ **ktc_sema_findAllByName** (**ktc_semanticInfo_t** scope, const char ∗name)
- void **ktc_sema_forAllClassDeclarations** (**ktc_semanticInfo_t** class_info, void(∗callback)(**ktc_semantic↩ Info_t**))
- void **ktc_sema_forAllScopeDeclarations** (**ktc_semanticInfo_t** scope_info, void(∗callback)(**ktc_↩ semanticInfo_t**))
- int **ktc_sema_functionOverloadsFunction** (**ktc_semanticInfo_t** func1, **ktc_semanticInfo_t** func2)
- int **ktc_sema_RelatedClasses** (**ktc_semanticInfo_t** class1, **ktc_semanticInfo_t** class2)

Variables

- const char ∗ **ktc_constructorName**
- const char ∗ **ktc_destructorName**

4.5.1 Detailed Description

4.5.2 Function Documentation

4.5.2.1 ktc_semanticInfo_t ktc_getAssociatedScope (ktc_tree_t *t*)

Get semantic description about a scope associated with a language construct described by node Applicable to nodes of the following kinds: NamespaceDecl, CompoundStmt, IfDeclStmt, SwitchDeclStmt, WhileDeclStmt, Do↩ DeclStmt, ForStmt

Returns

pointer to semantic information or 0 if no corresponding information available

4.5.2.2 ktc_semanticInfo_t ktc_getCalledFunction (ktc_tree_t *t*)

Get semantic description of a function called in a language construct described by node Applicable to nodes of the following kinds: CallExpr, TypeConvExpr, IndexExpr, CastExpr

Returns

pointer to semantic information or 0 if no corresponding information available

4.5.2.3 ktc_semanticInfo_t ktc_getSemanticInfo (ktc_tree_t *t*)

Get semantic description of language element described by node Semantic information can be retrieved only for nodes of the following kinds: NamespaceAlias, UsingDecl1, UsingDecl2, Enumr, Designator1, Designator2, Qualifier1, BasicIdentifier, QualifiedIdent, NewTypeId, NoTypeId, TypeId, ExprField, ExprIdent, ExprTypeid, DirDeclr1, Declr, ClassDeclr, TemplateArg2, TemplateArg3, TypeName, TypeOf2

Returns

pointer to semantic information or 0 if no semantic information available

4.5.2.4 kw_array_t* ktc_sema_findAllByName (ktc_semanticInfo_t *scope,* const char * *name*)

Get semantic descriptions for all entities with the given name in the given scope and all its parent scopes

Parameters

scope	scope info
name	entities name

Returns

array of entity semantic descriptions

4.5.2.5 ktc_semanticInfo_t ktc_sema_findFirstByName (ktc_semanticInfo_t *scope,* const char * *name*)

Get semantic information for first entity with the given name in the given scope and all its parent scopes

Parameters

scope	scope info
name	entity name

Returns

entity semantic info

4.5.2.6 void ktc_sema_forAllClassDeclarations (ktc_semanticInfo_t *class_info,* void(*)(ktc_semanticInfo_t) *callback*)

Perform the given action for all declarations in the given class

Parameters

class_info	class semantic info
callback	callback performing the given action

4.5.2.7 void ktc_sema_forAllScopeDeclarations (ktc_semanticInfo_t *scope_info,* void(*)(ktc_semanticInfo_t) *callback*)

Perform the given action for all declarations in the given scope

Parameters

scope_info	scope semantic info
callback	callback performing the given action

4.5.2.8 int ktc_sema_functionOverloadsFunction (ktc_semanticInfo_t *func1,* ktc_semanticInfo_t *func2*)

Check if function overloads another function in class hierarchy

Parameters

func1	first function info
func2	second function info

4.5.2.9 kw_array_t* ktc_sema_getAllByName (ktc_semanticInfo_t *scope,* const char * *name*)

Get semantic descriptions for all entities with the given name in the given scope

Parameters

scope	scope info
name	entities name

Returns

array of entity semantic descriptions

4.5.2.10 ktc_semanticInfo_t ktc_sema_getArrayElementType (ktc_semanticInfo_t *si*)

Get type of an array element

Parameters

si	semantic information on array type

Returns

0 if si is not an array type, semantic information on array element type otherwise

4.5.2.11 int ktc_sema_getArraySize (ktc_semanticInfo_t *si*)

Get size of an array

Parameters

si	semantic information on array type

Returns

-1 if si is not an array type, size of array otherwise

4.5.2.12 ktc_semanticInfo_t ktc_sema_getBaseInfo (ktc_semanticInfo_t *si,* int *i*)

Get base class for current class

Parameters

i	is between 0 and **ktc_sema_getNumberOfBaseInfo()** (p. 25)-1

Returns

pointer to semantic information or 0 if no corresponding information available

4.5.2.13 int ktc_sema_getBuiltinCode (ktc_semanticInfo_t *si*)

Get builtin type code by code item semantic information

Parameters

si	semantic description of the code item

Returns

operation code (see **Numerical codes of C/C++ built-in types** (p. 182)) or KTC_BUILTINTYPE_NONE if the item is not of builtin type

4.5.2.14 int ktc_sema_getClassTag (ktc_semanticInfo_t *si*)

Get class tag code for a type described by provided semantic information

4.5.2.15 int ktc_sema_getCVQualifiers (ktc_semanticInfo_t *si*)

Get const and volatile qualifiers of an entity described by 'si'

4.5.2.16 ktc_semanticInfo_t ktc_sema_getDefinedType (ktc_semanticInfo_t *si*)

Get type defined by typedef alias

Parameters

si	semantic information on typedef alias

Returns

0 if si is not a typedef alias, semantic information on typedef'd type otherwise

4.5.2.17 ktc_semanticInfo_t ktc_sema_getFirstByName (ktc_semanticInfo_t *scope,* const char * *name*)

Get semantic information for first entity with the given name in the given scope

Parameters

scope	scope info
name	entity name

Returns

entity semantic info

4.5.2.18 ktc_semanticInfo_t ktc_sema_getFormalArgument (ktc_semanticInfo_t *si,* int *n*)

Get semantic information of the n-th function type formal argument

Parameters

si	semantic description of the function type
n	formal parameter number. Must be in a range [0;N), where N - number returned by function 'ktc_sema_getNumberOfArguments'. Argument number 0 is non-zero only for class methods and desribes implicit this argument

Returns

semantic description of n-th formal argument, or 0 if 'si' is not a function type description.

4.5.2.19 ktc_semanticInfo_t ktc_sema_getFunctionType (ktc_semanticInfo_t *si*)

Get function type type from a function semantic description

Returns

function type if si is a semantic description of a function or 0 otherwise

4.5.2.20 ktc_semanticInfo_t ktc_sema_getGlobalScope ( )

Get semantic information for the global scope

Returns

global scope semantic info

4.5.2.21 const char* ktc_sema_getIdentifier (ktc_semanticInfo_t *si*)

Get semantic entity name

4.5.2.22 int ktc_sema_getIdentifierNo (ktc_semanticInfo_t *si*)

4.5.2.23 int ktc_sema_getNumber (ktc_semanticInfo_t *si*)

4.5.2.24 int ktc_sema_getNumberOfArguments (ktc_semanticInfo_t *si*)

Get number of arguments from function type

Returns

number of formal arguments for a given function type. 0 means that function type describes method with 'this' as only (and implicit) parameter. If 'si; is not a function type, -1 is returned.

4.5.2.25 int ktc_sema_getNumberOfBaseInfo (ktc_semanticInfo_t *si*)

Get number of base classes for current class

Returns

number of base classes

4.5.2.26 ktc_semanticInfo_t ktc_sema_getOverridenMethod (ktc_tree_t *t*)

Get semantic description of a base class method overriden with method that is associated with current node. Applicable to nodes of the following kinds: MemberDecl, MemberFunc, FuncDef

Returns

pointer to semantic information or 0 if no corresponding information available

4.5.2.27 ktc_semanticInfo_t ktc_sema_getPointedType (ktc_semanticInfo_t *si*)

Get type pointed by a pointer type

Parameters

si	semantic information on pointer type

Returns

0 if si is not a pointer type, semantic information on pointed type otherwise

4.5.2.28 char* ktc_sema_getQualifiedName (ktc_semanticInfo_t *si*)

Get fully qualified name

Returns

newly allocated string, that can be deallocated with **ktc_free()** (p. 58)

4.5.2.29 ktc_semanticInfo_t ktc_sema_getReferencedType (ktc_semanticInfo_t *si*)

Get type referenced by a reference type (C++ specific)

Parameters

si	semantic information on reference type

Returns

0 if si is not a reference type, semantic information on referenced type otherwise

4.5.2.30 ktc_semanticInfo_t ktc_sema_getReturnType (ktc_semanticInfo_t *si*)

Get function return type from function type semantic information

Returns

function return type if si is semantic information on function type or 0 if si is not a function type description

4.5.2.31 ktc_semanticInfo_t ktc_sema_getScope (ktc_semanticInfo_t *si*)

Get semantic description of class for current node.

Returns

pointer to semantic information or 0 if no corresponding information available

4.5.2.32 const char* ktc_sema_getTypedefedName (ktc_semanticInfo_t *si*)

Get name defined via typedef for anonymous structs and enums

4.5.2.33 char* ktc_sema_getTypeName (ktc_semanticInfo_t *si*)

Get a newly allocated string containing name of language type

Returns

newly allocated string, that can be deallocated with **ktc_free()** (p. 58)

4.5.2.34 ktc_tree_t ktc_sema_getVariableInitializer (ktc_semanticInfo_t *si*)

Get variable initializer from variable semantic information

Returns

tree node on variable initializer or 0 if si is not a variable description / does not have initializer

4.5.2.35 ktc_semanticInfo_t ktc_sema_getVariableType (ktc_semanticInfo_t *si*)

Get variable type from variable semantic information

Returns

semantic information on variable type or 0 if si is not a variable description

4.5.2.36 ktc_semanticInfo_t ktc_sema_getVariableValue (ktc_semanticInfo_t *si*)

Get variable value from variable semantic information

Returns

semantic information on variable value or 0 if si is not a variable description

4.5.2.37 int ktc_sema_hasMethods (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of a class with declared methods

4.5.2.38 int ktc_sema_haveSameFunctionType (ktc_semanticInfo_t *si1,* ktc_semanticInfo_t *si2*)

Check if function types are the close (regardless of the scope) Applicable to function types

Returns

1 if functions are the close, 0 otherwise

4.5.2.39 int ktc_sema_haveSameSignature (ktc_semanticInfo_t *si1,* ktc_semanticInfo_t *si2*)

Check if functions are the close (regardless of the scope) Applicable to descriptors of functions

Returns

1 if functions are the close, 0 otherwise

4.5.2.40 int ktc_sema_isAnonymous (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of an anonymous struct

4.5.2.41 int ktc_sema_isArray (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of array

4.5.2.42 int ktc_sema_isBasePrivate (ktc_semanticInfo_t *si,* int *i*)

Check if particular base of the given class is private

Parameters

si	derived class info
i	is between 0 and **ktc_sema_getNumberOfBaseInfo()** (p. 25)-1

Returns

1 if base is private, 0 otherwise

4.5.2.43 int ktc_sema_isBaseProtected (ktc_semanticInfo_t *si,* int *i*)

Check if particular base of the given class is protected

Parameters

si	derived class info
i	is between 0 and **ktc_sema_getNumberOfBaseInfo()** (p. 25)-1

Returns

1 if base is protected, 0 otherwise

4.5.2.44 int ktc_sema_isBasePublic (ktc_semanticInfo_t *si,* int *i*)

Check if particular base of the given class is public

Parameters

si	derived class info
i	is between 0 and **ktc_sema_getNumberOfBaseInfo()** (p. 25)-1

Returns

1 if base is public, 0 otherwise

4.5.2.45 int ktc_sema_isBaseVirtual (ktc_semanticInfo_t *si,* int *i*)

Check if particular base of the given class is virtual

Parameters

si	derived class info
i	is between 0 and **ktc_sema_getNumberOfBaseInfo()** (p. 25)-1

Returns

1 if base is virtual, 0 otherwise

4.5.2.46 int ktc_sema_isBitfield (ktc_semanticInfo_t *si*)

Check whether symbol described by semantic information is a bit field in a structure

Parameters

si	semantic information handle

Returns

0 if symbol is not a bit field, number of bits if symbol is a bit field

4.5.2.47 int ktc_sema_isBuiltin (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of built-in type

4.5.2.48 int ktc_sema_isClass (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of class

4.5.2.49 int ktc_sema_isConstMethod (ktc_semanticInfo_t *si*)

Check if particular method is constant

Parameters

si	method info

Returns

1 if method is constant, 0 otherwise

4.5.2.50 int ktc_sema_isConstructor (ktc_semanticInfo_t *si*)

Check whether given semantic info describes a class constructor

4.5.2.51 int ktc_sema_isDestructor (ktc_semanticInfo_t *si*)

Check whether given semantic info describes a class destructor

4.5.2.52 int ktc_sema_isEnum (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of enum

4.5.2.53 int ktc_sema_isEnumConstant (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of an enumeration constant

4.5.2.54 int ktc_sema_isFriend (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of a friend element

4.5.2.55 int ktc_sema_isFunction (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of function

4.5.2.56 int ktc_sema_isFunctionTemplate (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of function template

4.5.2.57 int ktc_sema_isFunctionTemplateSet (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of function template set

4.5.2.58 int ktc_sema_isFunctionType (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of function type

4.5.2.59 int ktc_sema_isGlobal (ktc_semanticInfo_t *si*)

Check whether given semantic info describes an entity declared in the global scope

4.5.2.60 int ktc_sema_isImmutable (ktc_semanticInfo_t *si*)

Check if an entity described by 'si' is immutable (has const qualifier or is an array)

Parameters

si	semantic description of the code item

Returns

1 for immutable items, 0 otherwise

4.5.2.61 int ktc_sema_isInstantiatedFunction (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of an instantiated function

4.5.2.62 int ktc_sema_isInstantiation (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of instantiation

4.5.2.63 int ktc_sema_isIntegerValue (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of integervalue

4.5.2.64 int ktc_sema_isLocal (ktc_semanticInfo_t *si*)

Check whether given semantic info describes an entity declared in a local scope

4.5.2.65 int ktc_sema_isNamespace (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of namespace

4.5.2.66 int ktc_sema_isNamespaceAlias (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of namespace alias

4.5.2.67 int ktc_sema_isNone (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of none

4.5.2.68 int ktc_sema_isObjectValue (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of objectvalue

4.5.2.69 int ktc_sema_isOperatorFunction (ktc_semanticInfo_t *si*)

Check whether given semantic info describes an operator function

4.5.2.70 int ktc_sema_isPOD (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of a POD type

4.5.2.71 int ktc_sema_isPointer (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of pointer

4.5.2.72 int ktc_sema_isPrivate (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of a private class member

4.5.2.73 int ktc_sema_isProtected (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of a protected class member

4.5.2.74 int ktc_sema_isPublic (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of a public class member

4.5.2.75 int ktc_sema_isPureVirtual (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of a pure virtual class method

4.5.2.76 int ktc_sema_isReference (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of reference

4.5.2.77 int ktc_sema_isSameClass (ktc_semanticInfo_t *si1,* ktc_semanticInfo_t *si2*)

Check if classes are the same Applicable to descriptors of classes

Returns

1 if classes are the same, 0 if classes are different

4.5.2.78 int ktc_sema_isSameEnum (ktc_semanticInfo_t *si1,* ktc_semanticInfo_t *si2*)

Check if enums are the same Applicable to descriptors of enums

Returns

1 if enums are the same, 0 if enums are different

4.5.2.79 int ktc_sema_isSameFunctions (ktc_semanticInfo_t *si1,* ktc_semanticInfo_t *si2*)

Check if functions are the same Applicable to descriptors of functions

Returns

1 if functions are the same, 0 if functions are different

4.5.2.80 int ktc_sema_isSameScope (ktc_semanticInfo_t *si1,* ktc_semanticInfo_t *si2*)

Check if scopes are the same Applicable to descriptors of scopes

Returns

1 if scopes are the same, 0 if scopes are different

4.5.2.81 int ktc_sema_isSameVariables (ktc_semanticInfo_t *si1,* ktc_semanticInfo_t *si2*)

Check if variables are the same Applicable to descriptors of variables

Returns

1 if variables are the same, 0 if variables are different

4.5.2.82 int ktc_sema_isScope (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of scope

4.5.2.83 int ktc_sema_isSpecialization (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of specialization

4.5.2.84 int ktc_sema_isTemplate (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of template

4.5.2.85 int ktc_sema_isType (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of type

4.5.2.86 int ktc_sema_isTypeParameter (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of a template type parameter

4.5.2.87 int ktc_sema_isUnion (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of union

4.5.2.88 int ktc_sema_isUsing (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of using

4.5.2.89 int ktc_sema_isUsingDeclaration (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of using declaration

4.5.2.90 int ktc_sema_isUsingDirective (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of using directive

4.5.2.91 int ktc_sema_isVariable (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of variable

4.5.2.92 int ktc_sema_isVirtual (ktc_semanticInfo_t *si*)

Check whether semantic information is a description of virtual method

4.5.2.93 int ktc_sema_RelatedClasses (ktc_semanticInfo_t *class1,* ktc_semanticInfo_t *class2*)

Check if two class are related through inheritance relationship

Parameters

class1	first class info
class2	second class info

4.5.2.94 ktc_semanticInfo_t ktc_sema_skipTypedefs (ktc_semanticInfo_t *si,* int * *cvq*)

Get type defined by all typedef alias

Parameters

si	semantic information on typedef alias
cvq	if not NULL saves const and volatile qualifiers

Returns

si if si is not a typedef alias, semantic information on all typedef'd types otherwise

4.5.2.95 ktc_semanticInfo_t ktc_sema_stripCVQ (ktc_semanticInfo_t *si*)

Get type identical to the given one but without 'const' and 'volatile' qualifiers

4.5.3 Variable Documentation

4.5.3.1 const char* ktc_constructorName

4.5.3.2 const char* ktc_destructorName

4.6 Functions for accessing type information

Modules

- **Attribute values for signedness of types**

Functions

- **ktc_languageType_t ktc_getLanguageType** (**ktc_tree_t** node)
- int **ktc_languageTypeSize** (**ktc_languageType_t** lang_type)
- int **ktc_languageTypeIsPointer** (**ktc_languageType_t** type)
- **ktc_languageType_t ktc_getPointedType** (**ktc_languageType_t** ptr_type)
- int **ktc_languageTypeIsBuiltin** (**ktc_languageType_t** type, int builtin_code)
- int **ktc_languageTypeSignedness** (**ktc_languageType_t** ctypeinfo)

4.6.1 Detailed Description

4.6.2 Function Documentation

4.6.2.1 ktc_languageType_t ktc_getLanguageType (ktc_tree_t *node*)

Get C/C++ type corresponding to subtree

Returns

NULL if type information is not available, pointer to type information otherwise

4.6.2.2 ktc_languageType_t ktc_getPointedType (ktc_languageType_t *ptr_type*)

Get type pointed by given type

Returns

pointed type if ptr_type is a pointer type, 0 otherwise.

Deprecated Use **ktc_sema_getPointedType** (p. 25)

4.6.2.3 int ktc_languageTypeIsBuiltin (ktc_languageType_t *type,* int *builtin_code*)

Deprecated Use **ktc_sema_getBuiltinCode** (p. 23) instead

4.6.2.4 int ktc_languageTypeIsPointer (ktc_languageType_t *type*)

Check that C/C++ type is actually a pointer type

Returns

1 if type is a pointer type, 0 otherwise.

Deprecated Use **ktc_sema_isPointer** (p. 31)

4.6.2.5 int ktc_languageTypeSignedness (ktc_languageType_t *ctypeinfo*)

Get 'signedness' of a type

Returns

'signedness' value (see **Attribute values for signedness of types** (p. 38))

4.6.2.6 int ktc_languageTypeSize (ktc_languageType_t *lang_type*)

Get size taken by value of the type (as in sizeof)

Returns

size taken by a value of a given type

Remarks

size calculation routines do not respect alignment, so size of structure is counted as a sum of its members' sizes. Structure containing one 'int' and one 'char' occupies 8 bits on 32-bit platforms with 4-byte alignment and 6 bytes on 32-bit platforms with 2-byte alignment. However, this routine will return 5.

4.7 Attribute values for signedness of types

Variables

- int **KTC_TYPESIGNEDNESS_NONE**
- int **KTC_TYPESIGNEDNESS_DEFAULT**
- int **KTC_TYPESIGNEDNESS_SIGNED**
- int **KTC_TYPESIGNEDNESS_UNSIGNED**

4.7.1 Detailed Description

4.7.2 Variable Documentation

4.7.2.1 int KTC_TYPESIGNEDNESS_DEFAULT

Default value for 'int', 'char', 'short', 'long' 'long long'

4.7.2.2 int KTC_TYPESIGNEDNESS_NONE

Type is neither signed nor unsigned. All built-in non-integer types and all non built-in types do not have signedness

4.7.2.3 int KTC_TYPESIGNEDNESS_SIGNED

Value for 'signed int|char|short|long|long long'

4.7.2.4 int KTC_TYPESIGNEDNESS_UNSIGNED

Value for 'unsigned int|char|short|long|long long'

4.8 Utility functions for tree access

Functions

- int **ktc_isCallTo** (**ktc_tree_t** node, const char ∗fn_name)
- **ktc_tree_t ktc_getCallArgument** (**ktc_tree_t** node, int n_arg)
- int **ktc_getNumberOfCallArguments** (**ktc_tree_t** node)
- int **ktc_isCharLiteral** (**ktc_tree_t** node)
- **ktc_long_long_t ktc_getIntegerValue** (**ktc_tree_t** t, int ∗error_flag)
- **ktc_long_long_t ktc_sema_getIntegerValue** (**ktc_semanticInfo_t** val, int ∗error_flag)
- const char ∗ **ktc_sema_getObjectName** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getIntValueType** (**ktc_semanticInfo_t** si)
- int **ktc_isNullPointerConstant** (**ktc_tree_t** t)
- **ktc_tree_t ktc_getSizeofArgument** (**ktc_tree_t** t)
- int **ktc_assembleStringConstant** (**ktc_tree_t** exprString, char ∗∗pbuf)
- int **ktc_isWideString** (**ktc_tree_t** exprString)
- int **ktc_isUTF16String** (**ktc_tree_t** exprString)
- int **ktc_isUTF32String** (**ktc_tree_t** exprString)
- const char ∗ **ktc_getNoIdent** (void)
- const char ∗ **ktc_getIdentifier** (**ktc_tree_t** t)
- int **ktc_getIdentifierNo** (**ktc_tree_t** t)
- char ∗ **ktc_getTokens** (**ktc_tree_t** t)
- int **ktc_is_NoToken** (**ktc_tree_t** t)
- int **ktc_is_Token** (**ktc_tree_t** t)
- char ∗ **ktc_getStringConstantValue** (**ktc_tree_t** t)
- **ktc_tree_t ktc_skipBrackets** (**ktc_tree_t** t)
- **ktc_tree_t ktc_getNameDeclarator** (**ktc_tree_t**)
- int **ktc_getOperation** (**ktc_tree_t** t)
- int **ktc_isOperationOverloaded** (**ktc_tree_t** t)
- int **ktc_getStorageClass** (**ktc_tree_t** t)
- int **ktc_getPointerOperator** (**ktc_tree_t** t)
- int **ktc_getTypeQualifiers** (**ktc_tree_t** t)
- int **ktc_getClassTag** (**ktc_tree_t** t)
- int **ktc_getBuiltinType** (**ktc_tree_t** t)
- int **ktc_getBuiltinTypeSize** (const char ∗type_name)
- int **ktc_getPointerSize** ()
- int **ktc_getCastSpecifier** (**ktc_tree_t** t)
- int **ktc_getFunctionSpecifier** (**ktc_tree_t** t)
- int **ktc_compareSubtrees** (**ktc_tree_t** t1, **ktc_tree_t** t2)

4.8.1 Detailed Description

4.8.2 Function Documentation

4.8.2.1 int ktc_assembleStringConstant (ktc_tree_t *exprString,* char ∗∗ *pbuf*)

Return buffer with a string constant assembled from a list of adjacent string literals. Buffer is allocated on heap, so calling procedure must take care to 'free' this buffer when it is no longer needed.

Remarks

Please not that string pointer returned via pointer in 'pbuf' is *not* a C zero-terminated string. String literal may contain zero bytes.

Parameters

exprString	tree root for the first literal in the list of adjacent string literals
pbuf	address of a pointer that will point to a newly allocated buffer with a string

Returns

length of the whole string buffer. If exprString is not of type tid_StringLiteralExpr, 0 is returned.

4.8.2.2 int ktc_compareSubtrees (ktc_tree_t *t1*, ktc_tree_t *t2*)

Compare two subtrees for structural equivalence. Subtrees are structurally equivalent if kinds of their root nodes are the same and all their children subtrees (excluding Next links) are structurally equivalent.

Parameters

t1	first subtree to compare
t2	second subtree to compare

Returns

1 if subtrees are structurally equivalent, 0 otherwise

4.8.2.3 int ktc_getBuiltinType (ktc_tree_t *t*)

Get builtin type code from the tid_BuiltinType subtree. This function is not applicable to other node types, and so only basic builtin types can be returned. Note that the returned type is not necessary the actual type of a declaration Use **ktc_sema_getBuiltinCode** (p. 23) function to get the complete information

Parameters

t	expression subtree, must be of tree type tid_BuiltinType

Returns

operation code (see **Numerical codes of C/C++ built-in types** (p. 182)) or KTC_BUILTINTYPE_NONE if preconditions are not met

4.8.2.4 int ktc_getBuiltinTypeSize (const char * *type_name*)

Get size (in bytes) of a builtin type by the type's name.

Parameters

type_name	name of builtin type

Returns

type size in bytes or -1 if the given type is unknown

4.8.2.5 ktc_tree_t ktc_getCallArgument (ktc_tree_t *node,* int *n_arg*)

Get n-th argument of a call, starting from 1

Parameters

node	call expression node
n_arg	number of an argument

Returns

a subtree for an n-th argument or 0 if 'n' is out of range or 'node' is not a call expression

4.8.2.6 int ktc_getCastSpecifier (ktc_tree_t *t*)

Get cast specifier code (for C++ -style cast expressions) from the tid_SpecialCastExpr subtree. This function is not applicable to other node types,

Parameters

t	expression subtree, must be of tree type tid_SpecialCastExpr

Returns

cast specifier code (see **Numerical codes for identifying different C++ -style cast expressions** (p. 185)) or -1 if t belongs to any other type.

4.8.2.7 int ktc_getClassTag (ktc_tree_t *t*)

Get class tag code from the tid_ClassType subtree.

Parameters

t	expression subtree, must be of tree type tid_ClassType

Returns

operation code (see **Numerical codes to differentiate struct/class/union declarations** (p. 181)) or KTC_↩ CLASSTAG_NONE if preconditions are not met

4.8.2.8 int ktc_getFunctionSpecifier (ktc_tree_t *t*)

Get function specifier code (for C++ member functions) from the tid_FunctionSpec subtree.

Parameters

t	expression subtree

Returns

function specifier code (see **Numerical codes for identifying inline/virtual/explicit/friend member function specifiers** (p. 184)), KTC_FUNCSPECIFIER_NONE is returned when no specifier is provided in the source or preconditions are not met.

4.8.2.9 const char* ktc_getIdentifier (ktc_tree_t *t*)

Get identifier string from tree nodes of subtypes: tid_IdExpr, tid_Name, tt_namespaceDef, tid_NamespaceAlias, tid_GotoStmt, tid_TypeName, tid_Enumerator, tid_NameDeclarator, tid_ParamName, tid_FieldDesignator, tid_↩ MemberDesignator, tid_Label, tid_ExprEnum, tid_NameSpec, tid_Dtor

4.8.2.10 int ktc_getIdentifierNo (ktc_tree_t *t*)

4.8.2.11 ktc_long_long_t ktc_getIntegerValue (ktc_tree_t *t,* int * *error_flag*)

Get integer value of constant of one of integer types or bool type (in C++)

Parameters

t	constant subtree
error_flag	set to non-zero value if it is impossible to convert constant into integer value

Returns

integer value of a constant, 0 is returned for 'false', 1 is returned for 'true'

4.8.2.12 ktc_tree_t ktc_getNameDeclarator (ktc_tree_t)

Get innermost declarator from a declarator sequence For example, in the declaration 'int *a[];', (after explicit breakdown it is 'int *((a)[]))))') the outermost declarator is '*a[]', then follows 'a[]', the innermost is 'a'.

4.8.2.13 const char* ktc_getNoIdent (void)

DEPRECATED!

4.8.2.14 int ktc_getNumberOfCallArguments (ktc_tree_t *node*)

Get number of call arguments

Parameters

node	call expression node

Returns

number of call arguments or (-1) if 'node' is not a call expression

4.8.2.15 int ktc_getOperation (ktc_tree_t *t*)

Get operation code from the Unary, Binary, Ternary or Filed expression subtree, from OperatorFunction subtree or from Operation subtree.

Parameters

t	expression subtree, must be of the following tree types: tid_UnaryExpr, tid_BinaryExpr, tid_MemberExpr, tid_OpFunc

Returns

operation code (see **Numerical codes of operations** (p. 175)) or KTC_OPCODE_NONE if preconditions are not met

4.8.2.16 int ktc_getPointerOperator (ktc_tree_t *t*)

Get pointer operator kind from tid_Ptr subtree.

Parameters

t	expression subtree, must be of tree type tid_Ptr

Returns

pointer operatior code code (see **Numerical codes for identifying declarator ptr types** (p. 186)) or KTC_↩ POINTEROPERATOR_NONE if preconditions are not met

4.8.2.17 int ktc_getPointerSize ( )

Get size (in bytes) of a pointer.

Returns

size of a pointer in bytes

4.8.2.18 ktc_tree_t ktc_getSizeofArgument (ktc_tree_t *t*)

Return argument of 'sizeof' regardless of the fact whether its argument is a type or an expression

4.8.2.19 int ktc_getStorageClass (ktc_tree_t *t*)

Get storage class code from the tid_StorageClass subtree.

Parameters

t	expression subtree, must be of tree type tid_StorageClass

Returns

operation code (see **Numerical codes of declaration storage class specifiers** (p. 179)) or KTC_STORA↩ GECLASS_NONE if preconditions are not met

4.8.2.20 char* ktc_getStringConstantValue (ktc_tree_t *t*)

Get string from StringLiteralExpr

Returns

newly allocated string, that can be deallocated with **ktc_free()** (p. 58)

4.8.2.21 char* ktc_getTokens (ktc_tree_t *t*)

Get identifier string from tree using Tokens

Returns

newly allocated string, that can be deallocated with **ktc_free()** (p. 58)

4.8.2.22 int ktc_getTypeQualifiers (ktc_tree_t *t*)

Get type qualifier flags from the tid_CVQualifier subtree.

Parameters

t	expression subtree, must be of tree type tid_CVQualifier

Returns

bit-or'ed superposition of type qualifier flags (see **Numerical codes of declaration type qualifiers (no qualifier, 'const', 'volatile' or 'restrict'). Actual values are bit-or'ed superpositions of these flags. Use '==' check for KTC_CVQUALIFIER_NONE and '&' for two other values** (p. 180)) or KTC_CVQUALIFIER_N↩ ONE if preconditions are not met

4.8.2.23 int ktc_is_NoToken (ktc_tree_t *t*)

Checks that root of subtree has 'NoToken' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoToken', 0 otherwise

4.8.2.24 int ktc_is_Token (ktc_tree_t *t*)

Checks that root of subtree has 'Token' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Token', 0 otherwise

4.8.2.25 int ktc_isCallTo (ktc_tree_t *node,* const char * *fn_name*)

Check that subtree is a representation of a call to a function with particular name.

Parameters

node	subtree root ot check
fn_name	function name

Returns

1 if subtree is a call of function with name 'fn_name', 0 otherwise

4.8.2.26 int ktc_isCharLiteral (ktc_tree_t *node*)

Checks if the node represents a char literal

Parameters

node	the node to test

Returns

1 if the node represents a char literal, 0 otherwise

4.8.2.27 int ktc_isNullPointerConstant (ktc_tree_t *t*)

Validate if the given tree represents a "null pointer constant". According to the C standard section 6.3.2.3, a null pointer constant is "an integer constant expression with the value 0, or such an expression cast to type void ∗."

Here are some examples of accepted null pointer constants: 0 (void ∗) 0 (((0))) (((1 - 1))) (1 ^ 1)

Parameters

t	the tree

Returns

1 if the tree represents a null pointer constant, 0 otherwise.

4.8.2.28 int ktc_isOperationOverloaded (ktc_tree_t *t*)

Check if operation is overloaded

Parameters

t	expression subtree, must be of the following tree types: tid_UnaryExpr, tid_BinaryExpr, tid_MemberExpr

4.8.2.29 int ktc_isUTF16String (ktc_tree_t *exprString*)

Return 1 if the given string literal is a utf-16 character string, 0 otherwise

Parameters

exprString	tree root for the first literal in the list of adjacent string literals

Returns

1 if the given string literal is a utf-16 character string, 0 otherwise

4.8.2.30 int ktc_isUTF32String (ktc_tree_t *exprString*)

Return 1 if the given string literal is a utf-32 character string, 0 otherwise

Parameters

exprString	tree root for the first literal in the list of adjacent string literals

Returns

1 if the given string literal is a utf-32 character string, 0 otherwise

4.8.2.31 int ktc_isWideString (ktc_tree_t *exprString*)

Return 1 if the given string literal is a wide character string, 0 otherwise

Parameters

exprString	tree root for the first literal in the list of adjacent string literals

Returns

1 if the given string literal is a wide character string, 0 otherwise

4.8.2.32 ktc_long_long_t ktc_sema_getIntegerValue (ktc_semanticInfo_t *val,* int * *error_flag*)

Get integer value of constant of one of integer types or bool type (in C++)

Parameters

val	semantic info
error_flag	set to non-zero value if it is impossible to convert constant into integer value

Returns

integer value of a constant, 0 is returned for 'false', 1 is returned for 'true'

4.8.2.33 int ktc_sema_getIntValueType (ktc_semanticInfo_t *si*)

Get the original type of an int value

Parameters

value	semantic info

Returns

the KTC_BUILTIN type code

4.8.2.34 const char* ktc_sema_getObjectName (ktc_semanticInfo_t *si*)

Get object name string of an object value

Parameters

object	semantic info

Returns

the name string or NULL on error. The string pointer should not be freed.

4.8.2.35 ktc_tree_t ktc_skipBrackets (ktc_tree_t *t*)

Drop redundant parentheses around expression in expression tree

Returns

a node that is a root of subtree of first non-parenthesized expression, for example, ((((a+(b)))))) will be traversed down to a+(b)

4.9 Working with tree positions

Typedefs

- typedef void ∗ **ktc_position_t**
- typedef **ktc_position_t ktc_position**

Functions

- **ktc_position_t ktc_position_new** (int line, int col, const char ∗fname)
- **ktc_position_t ktc_position_copy** (**ktc_position_t** pos)
- void **ktc_position_delete** (**ktc_position_t** pos)
- **ktc_position_t ktc_getStartPosition** (**ktc_tree_t** t)
- **ktc_position_t ktc_getEndPosition** (**ktc_tree_t** t)
- int **ktc_position_getLine** (**ktc_position_t** pos)
- void **ktc_position_setLine** (**ktc_position_t** pos, int line)
- int **ktc_position_getColumn** (**ktc_position_t** pos)
- char ∗ **ktc_position_getFileName** (**ktc_position_t** pos)

4.9.1 Detailed Description

4.9.2 Typedef Documentation

4.9.2.1 typedef ktc_position_t ktc_position

Alias for ktc_position_t - for compatibility with 1.0 - 1.1 API versions

4.9.2.2 typedef void∗ ktc_position_t

Opaque data type that describes position (file, line, and column)

4.9.3 Function Documentation

4.9.3.1 ktc_position_t ktc_getEndPosition (ktc_tree_t *t*)

Get end position of the code fragment that is translated to a particular subtree

Parameters

t	subtree to get end position of

4.9.3.2 ktc_position_t ktc_getStartPosition (ktc_tree_t *t*)

Get start position of the code fragment that is translated to a particular subtree

Parameters

t	subtree to get start position of

4.9.3.3 ktc_position_t ktc_position_copy (ktc_position_t *pos*)

Copy previously allocated position

Parameters

position	to copy

Returns

copied position, that can be deallocated with **ktc_free()** (p. 58)

4.9.3.4 void ktc_position_delete (ktc_position_t *pos*)

Delete newly allocated position

4.9.3.5 int ktc_position_getColumn (ktc_position_t *pos*)

Get column from position

4.9.3.6 char* ktc_position_getFileName (ktc_position_t *pos*)

Get file name from position

4.9.3.7 int ktc_position_getLine (ktc_position_t *pos*)

Get line from position

4.9.3.8 ktc_position_t ktc_position_new (int *line,* int *col,* const char * *fname*)

Create newly allocated position with line, col and file name

Parameters

line,column	and filename to set

Returns

newly allocated position, that can be deallocated with **ktc_free()** (p. 58)

4.9.3.9 void ktc_position_setLine (ktc_position_t *pos*, int *line*)

- Set line to position

4.10 for defect

Typedefs

- typedef void * **ktc_autofix_t**

Functions

- **ktc_autofix_t ktc_autofix_new** ()
- void **ktc_autofix_delete** (**ktc_autofix_t** a)
- void **ktc_autofix_addSegment** (**ktc_autofix_t** a, const char *repl, const char *file, int line1, int col1, int line2, int col2)

4.10.1 Detailed Description

4.10.2 Typedef Documentation

4.10.2.1 typedef void* ktc_autofix_t

Opaque data type that describes autofix (file, line, and column)

4.10.3 Function Documentation

4.10.3.1 void ktc_autofix_addSegment (ktc_autofix_t *a,* const char * *repl,* const char * *file,* int *line1,* int *col1,* int *line2,* int *col2*)

4.10.3.2 void ktc_autofix_delete (ktc_autofix_t *a*)

4.10.3.3 ktc_autofix_t ktc_autofix_new ( )

Create newly allocated position with line, col and file name

Parameters

t	subtree to get start position of

Returns

newly allocated position, that can be deallocated with **ktc_autofix_delete()** (p. 54)

4.11 Accessing compiler configuration

Functions

- int **ktc_error_isEnabled** (const char ∗error_id)
- const char ∗ **ktc_error_getConfigurationParameter** (const char ∗error_id, const char ∗param_name)
- int **ktc_getFrontendLanguage** (void)
- int **ktc_getFrontendDialect** (void)
- int **ktc_hasBuiltinWideChar** (void)

Variables

- const int **KTC_LANGUAGE_C**
- const int **KTC_LANGUAGE_CXX**
- const int **KTC_DIALECT_STD**
- const int **KTC_DIALECT_GNU**
- const int **KTC_DIALECT_MS**
- const int **KTC_DIALECT_ARM**
- const int **KTC_DIALECT_GHS**
- const int **KTC_DIALECT_SUN**
- const int **KTC_DIALECT_TI**

4.11.1 Detailed Description

4.11.2 Function Documentation

4.11.2.1 const char∗ ktc_error_getConfigurationParameter (const char ∗ *error_id,* const char ∗ *param_name*)

Get configuration parameter for an error

Parameters

param_name	name of the configuration parameter
error_id	identifier of an error (the same id as in <error> tag in configuration file)

Returns

parameter value or "" if error was not configured or there is no such parameter

4.11.2.2 int ktc_error_isEnabled (const char ∗ *error_id*)

Deprecated Use kwapi_cfgparam_errorIsEnabled

4.11.2.3 int ktc_getFrontendDialect (void)

Returns

dialect of the current compilation (KTC_DIALECT_STD / KTC_DIALECT_GNU / KTC_DIALECT_MS / KTC↩ _DIALECT_ARM / KTC_DIALECT_GHS / KTC_DIALECT_SUN / KTC_DIALECT_TI

4.11.2.4 int ktc_getFrontendLanguage (void)

Returns

programming language of the current compilation (KTC_LANGUAGE_CXX or KTC_LANGUAGE_C).

4.11.2.5 int ktc_hasBuiltinWideChar (void)

Check if wchar_t is a built-in type

4.11.3 Variable Documentation

4.11.3.1 const int KTC_DIALECT_ARM

Current translation unit is compiled in armcc compiler emulation mode

4.11.3.2 const int KTC_DIALECT_GHS

Current translation unit is compiled in Green Hills compiler emulation mode

4.11.3.3 const int KTC_DIALECT_GNU

Current translation unit is compiled in gnu emulation mode

4.11.3.4 const int KTC_DIALECT_MS

Current translation unit is compiled in MS cl emulation mode

4.11.3.5 const int KTC_DIALECT_STD

Current translation unit is compiled in standard mode

4.11.3.6 const int KTC_DIALECT_SUN

Current translation unit is compiled in Sun compiler emulation mode

4.11.3.7 const int KTC_DIALECT_TI

Current translation unit is compiled in Texas Instruments compiler emulation mode

4.11.3.8 const int KTC_LANGUAGE_C

Current translation unit is compiled by C frontend

4.11.3.9 const int KTC_LANGUAGE_CXX

Current translation unit is compiled by C++ frontend

4.12 Working with warning and error messages

Typedefs

- typedef void * **ktc_message_t**
- typedef void * **ktc_message**

Functions

- **ktc_message_t ktc_message_new** (const char *error_id)
- void **ktc_message_setPosition** (**ktc_message_t** msg, **ktc_position_t** pos)
- void **ktc_message_addAttribute** (**ktc_message_t** msg, const char *attr_string)
- void **ktc_message_addAnchorAttribute** (**ktc_message_t** msg, const char *attr_string)
- void **ktc_message_addTraceBySemanticsInfo** (**ktc_message_t** msg, **ktc_semanticInfo_t** sema)
- void **ktc_message_addEvent** (**ktc_message_t** msg, **ktc_position_t** pos, const char *str)
- void * **ktc_event_new** (**ktc_position_t** pos, const char *str)
- void **ktc_event_setParameter** (void *event, const char *name, const char *value)
- void **ktc_message_addEventEx** (void *msg, void *event)
- void **ktc_message_render** (**ktc_message_t** msg)
- void **ktc_message_render_wi_autofix** (**ktc_message_t** msg, **ktc_autofix_t** a)
- void **ktc_message_delete** (**ktc_message_t** msg)
- void **ktc_message_setFunction** (void *function)
- void **ktc_message_unsetFunction** ()
- int **ktc_isStatic** (**ktc_tree_t** node)
- int **ktc_sema_isStatic** (**ktc_semanticInfo_t** info)
- int **ktc_isNameIncluded** (const char *fname)
- int **ktc_isIncluded** (**ktc_tree_t** node)
- int **ktc_sema_isIncluded** (**ktc_semanticInfo_t** si)
- void **ktc_free** (void *ptr)
- int **ktc_isConstructor** (**ktc_tree_t** node)
- int **ktc_sema_getFieldNumber** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFieldType** (**ktc_semanticInfo_t** si, int fieldpos)
- **ktc_position_t ktc_sema_getPosition** (**ktc_semanticInfo_t** si)
- **ktc_tree_t ktc_sema_getAST** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getPrimaryTemplate** (**ktc_semanticInfo_t** si)
- **kw_array_t** * **ktc_sema_getInstantiationParameters** (**ktc_semanticInfo_t** si)
- **kw_array_t** * **ktc_sema_getEnumerators** (**ktc_semanticInfo_t** si)
- int **ktc_isDeclaration** (**ktc_tree_t** info)
- int **ktc_isDefinition** (**ktc_tree_t** info)
- int **ktc_sema_isFuncDef** (**ktc_semanticInfo_t** info)
- **ktc_semanticInfo_t ktc_sema_getInstantiatedClass** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFunctionFromTemplate** (**ktc_semanticInfo_t** info)
- **kw_array_t** * **ktc_sema_getFunctionTemplateSpecializations** (**ktc_semanticInfo_t** info)
- **kw_array_t** * **ktc_sema_getFunctionTemplateInstantiations** (**ktc_semanticInfo_t** info)
- **ktc_semanticInfo_t ktc_sema_getInstantiationOrigin** (**ktc_semanticInfo_t** info)
- short **ktc_sema_checkBitField** (**ktc_semanticInfo_t** info)
- int **ktc_sema_isFunctionPointer** (**ktc_semanticInfo_t** info)
- **ktc_semanticInfo_t ktc_sema_getFunctionPointerType** (**ktc_semanticInfo_t** info)
- int **ktc_sema_isInline** (**ktc_semanticInfo_t** info)

4.12.1 Detailed Description

4.12.2 Typedef Documentation

4.12.2.1 typedef void* ktc_message

Alias for ktc_message_t - for compatibility with 1.0 - 1.1 API versions

4.12.2.2 typedef void* ktc_message_t

Opaque type for warning/error message data structure

4.12.3 Function Documentation

4.12.3.1 void* ktc_event_new (ktc_position_t *pos,* const char * *str*)

4.12.3.2 void ktc_event_setParameter (void * *event,* const char * *name,* const char * *value*)

4.12.3.3 void ktc_free (void * *ptr*)

Deallocate memory

Parameters

ptr	is pointer to memoty allocated from ktc_ functions

4.12.3.4 int ktc_isConstructor (ktc_tree_t *node*)

Check that given node corresponds to a class constructor

Returns

1 if true; 0 if false

4.12.3.5 int ktc_isDeclaration (ktc_tree_t *info*)

Check that given node corresponds to a class or enum declaration

Returns

1 if true; 0 if false

4.12.3.6 int ktc_isDefinition (ktc_tree_t *info*)

Check that given node corresponds to a class or enum definition

Returns

1 if true; 0 if false

4.12.3.7 int ktc_isIncluded (ktc_tree_t *node*)

Check that given node is included via header file

Returns

1 if is; 0 otherwise

4.12.3.8 int ktc_isNameIncluded (const char * *fname*)

Check that given file name is a name of a header file

Returns

1 if is; 0 otherwise

4.12.3.9 int ktc_isStatic (ktc_tree_t *node*)

Check that given node corresponds to a static declaration

Returns

1 if true, 0 if false

4.12.3.10 void ktc_message_addAnchorAttribute (ktc_message_t *msg,* const char * *attr_string*)

4.12.3.11 void ktc_message_addAttribute (ktc_message_t *msg,* const char * *attr_string*)

Add arbitrary attribute to the message, they may be referred by {n} in the format string. 'n' is an number of an attribute (starting from 0)

Parameters

msg	message to add attribute to
attr_string	attribute value

4.12.3.12 void ktc_message_addEvent (ktc_message_t *msg*, ktc_position_t *pos*, const char * *str*)

4.12.3.13 void ktc_message_addEventEx (void * *msg*, void * *event*)

4.12.3.14 void ktc_message_addTraceBySemanticsInfo (ktc_message_t *msg*, ktc_semanticInfo_t *sema*)

4.12.3.15 void ktc_message_delete (ktc_message_t *msg*)

Free memory occupied by internal message data structures

4.12.3.16 ktc_message_t ktc_message_new (const char * *error_id*)

Allocate memory for an error report

Parameters

error_id	identifier of an error (the same id as in <error> tag in configuration file)

Returns

Allocated structure for keeping message details

4.12.3.17 void ktc_message_render (ktc_message_t *msg*)

Put message into the object file

4.12.3.18 void ktc_message_render_wi_autofix (ktc_message_t *msg*, ktc_autofix_t *a*)

Put message into the object file

4.12.3.19 void ktc_message_setFunction (void * *function*)

Set current function

4.12.3.20 void ktc_message_setPosition (ktc_message_t *msg*, ktc_position_t *pos*)

Set position for the message

Parameters

msg	message to set position for
pos	position

4.12.3.21 void ktc_message_unsetFunction ( )

Set current file

4.12.3.22 short ktc_sema_checkBitField (ktc_semanticInfo_t *info*)

Get number of bits for a bitfield

4.12.3.23 ktc_tree_t ktc_sema_getAST (ktc_semanticInfo_t *si*)

Get subtree that corresponds to the given semantic element

Parameters

si	semantic info

Returns

the corresponding subtree

4.12.3.24 kw_array_t* ktc_sema_getEnumerators (ktc_semanticInfo_t *si*)

Get actual enumerators for a given enum

Parameters

si	semantic description of enum

Returns

array of enumerators' semantic descriptions

4.12.3.25 int ktc_sema_getFieldNumber (ktc_semanticInfo_t *si*)

Get number of struct fields

Parameters

struct	semantic info

Returns

0 if the semantic info does not correspond to the struct, number of fields otherwise

4.12.3.26 ktc_semanticInfo_t ktc_sema_getFieldType (ktc_semanticInfo_t *si,* int *fieldpos*)

Get type of struct or class field

Parameters

struct	or class semantic info, number of field

Returns

0 if the semantic info does not correspond to the struct, the semantic type of the field otherwise

4.12.3.27 ktc_semanticInfo_t ktc_sema_getFunctionFromTemplate (ktc_semanticInfo_t *info*)

Get function from semantic information about function template

Returns

semantic information or 0 if si is not a function template description

4.12.3.28 ktc_semanticInfo_t ktc_sema_getFunctionPointerType (ktc_semanticInfo_t *info*)

Returns

the semantic information decribing a function pointer type, 0 if not found.

4.12.3.29 kw_array_t* ktc_sema_getFunctionTemplateInstantiations (ktc_semanticInfo_t *info*)

Get all instantiations for a function template

4.12.3.30 kw_array_t* ktc_sema_getFunctionTemplateSpecializations (ktc_semanticInfo_t *info*)

Get all specializations for a function template

4.12.3.31 ktc_semanticInfo_t ktc_sema_getInstantiatedClass (ktc_semanticInfo_t *si*)

Get class semantic info for an instantiation

4.12.3.32 ktc_semanticInfo_t ktc_sema_getInstantiationOrigin (ktc_semanticInfo_t *info*)

Get a specialization the given instantiation was created from

4.12.3.33 kw_array_t* ktc_sema_getInstantiationParameters (ktc_semanticInfo_t *si*)

Get actual template parameters for a given instantiation

Parameters

si	semantic description of an instantiation

Returns

array of parameters' semantic descriptions

4.12.3.34 ktc_position_t ktc_sema_getPosition (ktc_semanticInfo_t *si*)

Get position of the declaration that corresponds to the given semantic element

Parameters

si	semantic info

Returns

the corresponding position

4.12.3.35 ktc_semanticInfo_t ktc_sema_getPrimaryTemplate (ktc_semanticInfo_t *si*)

Get primary template for the given instantiation or specialization

Parameters

si	semantic description of an instantiation or a specialization

Returns

semantic description of the primary template

4.12.3.36 int ktc_sema_isFuncDef (ktc_semanticInfo_t *info*)

Check that given node corresponds to a function definition

Returns

1 if true; 0 if false

4.12.3.37 int ktc_sema_isFunctionPointer (ktc_semanticInfo_t *info*)

Check if the given semantic information decribes a function pointer

4.12.3.38 int ktc_sema_isIncluded (ktc_semanticInfo_t *si*)

Check that given sema info corresponds to an included entity

Returns

1 if is; 0 otherwise

4.12.3.39 int ktc_sema_isInline (ktc_semanticInfo_t *info*)

Check if the given function is inline

4.12.3.40 int ktc_sema_isStatic (ktc_semanticInfo_t *info*)

Check that given entity corresponds to a static declaration

Returns

1 if true, 0 if false

4.13 Tree type identifiers

Variables

- const **ktc_treeType_t tid_Any**
- const **ktc_treeType_t tid_Node**
- const **ktc_treeType_t tid_TranslationUnit**
- const **ktc_treeType_t tid_DeclOrStmts**
- const **ktc_treeType_t tid_AnyFuncBody**
- const **ktc_treeType_t tid_Handlers**
- const **ktc_treeType_t tid_ExceptHandler**
- const **ktc_treeType_t tid_FinallyHandler**
- const **ktc_treeType_t tid_MaybeException**
- const **ktc_treeType_t tid_TemplateParams**
- const **ktc_treeType_t tid_DeclSpecs**
- const **ktc_treeType_t tid_AttributeSpecs**
- const **ktc_treeType_t tid_Attributes**
- const **ktc_treeType_t tid_PropertyFuncs**
- const **ktc_treeType_t tid_MemberDecls**
- const **ktc_treeType_t tid_Enumerators**
- const **ktc_treeType_t tid_MaybeDeclarator**
- const **ktc_treeType_t tid_ParamNames**
- const **ktc_treeType_t tid_Initializers**
- const **ktc_treeType_t tid_Designators**
- const **ktc_treeType_t tid_AnyLabel**
- const **ktc_treeType_t tid_Exprs**
- const **ktc_treeType_t tid_AnyNameQualifier**
- const **ktc_treeType_t tid_AnyNames**
- const **ktc_treeType_t tid_TemplateArgs**
- const **ktc_treeType_t tid_BaseSpecs**
- const **ktc_treeType_t tid_MaybeLambdaDeclarator**
- const **ktc_treeType_t tid_LambdaIntroducer**
- const **ktc_treeType_t tid_AnyCapture**
- const **ktc_treeType_t tid_MaybeTypeId**
- const **ktc_treeType_t tid_MaybeNewInitializer**
- const **ktc_treeType_t tid_MemberInitializers**
- const **ktc_treeType_t tid_MaybeCtorInitializer**
- const **ktc_treeType_t tid_MaybeExceptionSpec**
- const **ktc_treeType_t tid_NoDeclOrStmt**
- const **ktc_treeType_t tid_DeclEllipsis**
- const **ktc_treeType_t tid_DeclOrStmt**
- const **ktc_treeType_t tid_FuncBody**
- const **ktc_treeType_t tid_FuncTryBlock**
- const **ktc_treeType_t tid_PromisedFuncBody**
- const **ktc_treeType_t tid_NoHandler**
- const **ktc_treeType_t tid_Handler**
- const **ktc_treeType_t tid_NoException**
- const **ktc_treeType_t tid_Exception**
- const **ktc_treeType_t tid_DefaultException**
- const **ktc_treeType_t tid_UnparsedException**
- const **ktc_treeType_t tid_NoTemplateParam**
- const **ktc_treeType_t tid_TemplateParam**
- const **ktc_treeType_t tid_NoDeclSpec**
- const **ktc_treeType_t tid_DeclSpec**

- const **ktc_treeType_t tid_NoAttributeSpec**
- const **ktc_treeType_t tid_AttributeSpec**
- const **ktc_treeType_t tid_NoAttribute**
- const **ktc_treeType_t tid_AnyAttribute**
- const **ktc_treeType_t tid_NoPropertyFunc**
- const **ktc_treeType_t tid_AnyPropertyFunc**
- const **ktc_treeType_t tid_NoMemberDecl**
- const **ktc_treeType_t tid_AnyMemberDecl**
- const **ktc_treeType_t tid_NoEnumerator**
- const **ktc_treeType_t tid_AnyEnumerator**
- const **ktc_treeType_t tid_NoDeclarator**
- const **ktc_treeType_t tid_AnyDeclarator**
- const **ktc_treeType_t tid_NoParamName**
- const **ktc_treeType_t tid_AnyParamName**
- const **ktc_treeType_t tid_NoInitializer**
- const **ktc_treeType_t tid_AnyInitializer**
- const **ktc_treeType_t tid_NoDesignator**
- const **ktc_treeType_t tid_AnyDesignator**
- const **ktc_treeType_t tid_Label**
- const **ktc_treeType_t tid_CaseLabel**
- const **ktc_treeType_t tid_DefaultLabel**
- const **ktc_treeType_t tid_CaseRangeLabel**
- const **ktc_treeType_t tid_UnparsedLabel**
- const **ktc_treeType_t tid_NoExpr**
- const **ktc_treeType_t tid_AnyExpr**
- const **ktc_treeType_t tid_NoNameQualifier**
- const **ktc_treeType_t tid_GlobalScope**
- const **ktc_treeType_t tid_SuperScope**
- const **ktc_treeType_t tid_AnyNameSpec**
- const **ktc_treeType_t tid_NoName**
- const **ktc_treeType_t tid_AnyName**
- const **ktc_treeType_t tid_NoTemplateArg**
- const **ktc_treeType_t tid_AnyTemplateArg**
- const **ktc_treeType_t tid_NoBaseSpec**
- const **ktc_treeType_t tid_BaseSpec**
- const **ktc_treeType_t tid_NoLambdaDeclarator**
- const **ktc_treeType_t tid_LambdaDeclarator**
- const **ktc_treeType_t tid_CaptureDefault**
- const **ktc_treeType_t tid_Capture**
- const **ktc_treeType_t tid_NoCapture**
- const **ktc_treeType_t tid_NoTypeId**
- const **ktc_treeType_t tid_TypeId**
- const **ktc_treeType_t tid_NoNewInitializer**
- const **ktc_treeType_t tid_NewInitializer**
- const **ktc_treeType_t tid_NoMemberInitializer**
- const **ktc_treeType_t tid_MemberInitializer**
- const **ktc_treeType_t tid_NoCtorInitializer**
- const **ktc_treeType_t tid_CtorInitializer**
- const **ktc_treeType_t tid_NoExceptionSpec**
- const **ktc_treeType_t tid_ExceptionSpec**
- const **ktc_treeType_t tid_DenyThrowSpec**
- const **ktc_treeType_t tid_AnyDecl**
- const **ktc_treeType_t tid_AnyStmt**
- const **ktc_treeType_t tid_AnyTypeParam**
- const **ktc_treeType_t tid_Param**

- const **ktc_treeType_t tid_AutoType**
- const **ktc_treeType_t tid_ConstExpr**
- const **ktc_treeType_t tid_StorageClass**
- const **ktc_treeType_t tid_AlignAsExpr**
- const **ktc_treeType_t tid_AlignAsType**
- const **ktc_treeType_t tid_AnyTypeName**
- const **ktc_treeType_t tid_FuncSpec**
- const **ktc_treeType_t tid_AttributeDeclSpec**
- const **ktc_treeType_t tid_UnparsedDeclSpec**
- const **ktc_treeType_t tid_Attribute**
- const **ktc_treeType_t tid_AttributeWithArgs**
- const **ktc_treeType_t tid_PropertyAttribute**
- const **ktc_treeType_t tid_PropertyPutFunc**
- const **ktc_treeType_t tid_PropertyGetFunc**
- const **ktc_treeType_t tid_UnparsedPropertyFunc**
- const **ktc_treeType_t tid_MemberDecl**
- const **ktc_treeType_t tid_MemberFunc**
- const **ktc_treeType_t tid_AccessSpecification**
- const **ktc_treeType_t tid_MemberTemplate**
- const **ktc_treeType_t tid_MemberUsingDecl**
- const **ktc_treeType_t tid_UnparsedMemberDecl**
- const **ktc_treeType_t tid_PromisedMemberDecl**
- const **ktc_treeType_t tid_Enumerator**
- const **ktc_treeType_t tid_UnparsedEnumerator**
- const **ktc_treeType_t tid_AnyNonPtrDeclarator**
- const **ktc_treeType_t tid_PtrDeclarator**
- const **ktc_treeType_t tid_BitFieldDeclarator**
- const **ktc_treeType_t tid_AttributedDeclarator**
- const **ktc_treeType_t tid_InitializedDeclarator**
- const **ktc_treeType_t tid_UnparsedDeclarator**
- const **ktc_treeType_t tid_ParamName**
- const **ktc_treeType_t tid_UnparsedParamName**
- const **ktc_treeType_t tid_CopyInitializer**
- const **ktc_treeType_t tid_InitClause**
- const **ktc_treeType_t tid_TruncatedInitClause**
- const **ktc_treeType_t tid_DirectInitializer**
- const **ktc_treeType_t tid_UnparsedInitializer**
- const **ktc_treeType_t tid_FieldDesignator**
- const **ktc_treeType_t tid_MemberDesignator**
- const **ktc_treeType_t tid_IndexDesignator**
- const **ktc_treeType_t tid_RangeDesignator**
- const **ktc_treeType_t tid_IdExpr**
- const **ktc_treeType_t tid_BoolLiteralExpr**
- const **ktc_treeType_t tid_LiteralExpr**
- const **ktc_treeType_t tid_UserLiteralExpr**
- const **ktc_treeType_t tid_StringLiteralExpr**
- const **ktc_treeType_t tid_UserStringLiteralExpr**
- const **ktc_treeType_t tid_NullptrLiteralExpr**
- const **ktc_treeType_t tid_MemberExpr**
- const **ktc_treeType_t tid_CallExpr**
- const **ktc_treeType_t tid_TypeConvExpr**
- const **ktc_treeType_t tid_IndexExpr**
- const **ktc_treeType_t tid_UnaryExpr**
- const **ktc_treeType_t tid_ThrowExpr**
- const **ktc_treeType_t tid_BinaryExpr**

- const **ktc_treeType_t tid_ConditionalExpr**
- const **ktc_treeType_t tid_SizeOfExpr**
- const **ktc_treeType_t tid_AlignOfExpr**
- const **ktc_treeType_t tid_CastExpr**
- const **ktc_treeType_t tid_SpecialCastExpr**
- const **ktc_treeType_t tid_ParensExpr**
- const **ktc_treeType_t tid_ThisExpr**
- const **ktc_treeType_t tid_NewExpr**
- const **ktc_treeType_t tid_DeleteExpr**
- const **ktc_treeType_t tid_TypeTypeIdExpr**
- const **ktc_treeType_t tid_ExprTypeIdExpr**
- const **ktc_treeType_t tid_StmtExpr**
- const **ktc_treeType_t tid_InitializerExpr**
- const **ktc_treeType_t tid_LambdaExpr**
- const **ktc_treeType_t tid_UnparsedExpr**
- const **ktc_treeType_t tid_NameSpec**
- const **ktc_treeType_t tid_TemplateSpec**
- const **ktc_treeType_t tid_TypeOfSpec**
- const **ktc_treeType_t tid_UnparsedNameQualifier**
- const **ktc_treeType_t tid_QualifiedName**
- const **ktc_treeType_t tid_UnqualifiedName**
- const **ktc_treeType_t tid_ExprArg**
- const **ktc_treeType_t tid_TypeArg**
- const **ktc_treeType_t tid_TemplateTypeArg**
- const **ktc_treeType_t tid_FuncDef**
- const **ktc_treeType_t tid_Decl**
- const **ktc_treeType_t tid_TemplateDecl**
- const **ktc_treeType_t tid_LinkageSpec**
- const **ktc_treeType_t tid_ExplicitInstantiation**
- const **ktc_treeType_t tid_NamespaceDecl**
- const **ktc_treeType_t tid_NamespaceAlias**
- const **ktc_treeType_t tid_AnyUsing**
- const **ktc_treeType_t tid_AliasDecl**
- const **ktc_treeType_t tid_AsmDef**
- const **ktc_treeType_t tid_StaticAssertDecl**
- const **ktc_treeType_t tid_UnparsedDecl**
- const **ktc_treeType_t tid_LabeledStmt**
- const **ktc_treeType_t tid_ExprStmt**
- const **ktc_treeType_t tid_CompoundStmt**
- const **ktc_treeType_t tid_IfStmt**
- const **ktc_treeType_t tid_IfDeclStmt**
- const **ktc_treeType_t tid_SwitchStmt**
- const **ktc_treeType_t tid_SwitchDeclStmt**
- const **ktc_treeType_t tid_WhileStmt**
- const **ktc_treeType_t tid_WhileDeclStmt**
- const **ktc_treeType_t tid_DoStmt**
- const **ktc_treeType_t tid_DoDeclStmt**
- const **ktc_treeType_t tid_ForStmt**
- const **ktc_treeType_t tid_ForEachStmt**
- const **ktc_treeType_t tid_ForRangeStmt**
- const **ktc_treeType_t tid_GotoStmt**
- const **ktc_treeType_t tid_ContinueStmt**
- const **ktc_treeType_t tid_BreakStmt**
- const **ktc_treeType_t tid_ReturnStmt**
- const **ktc_treeType_t tid_TryStmt**

- const **ktc_treeType_t tid_TryExceptStmt**
- const **ktc_treeType_t tid_TryFinallyStmt**
- const **ktc_treeType_t tid_LeaveStmt**
- const **ktc_treeType_t tid_AsmStmt**
- const **ktc_treeType_t tid_UnparsedStmt**
- const **ktc_treeType_t tid_TypeParam**
- const **ktc_treeType_t tid_TemplateTypeParam**
- const **ktc_treeType_t tid_CVQualifier**
- const **ktc_treeType_t tid_ReservedTypeSpec**
- const **ktc_treeType_t tid_TypeName**
- const **ktc_treeType_t tid_EnumType**
- const **ktc_treeType_t tid_ClassType**
- const **ktc_treeType_t tid_AnyTypeOf**
- const **ktc_treeType_t tid_NameDeclarator**
- const **ktc_treeType_t tid_ParensDeclarator**
- const **ktc_treeType_t tid_ArrayDeclarator**
- const **ktc_treeType_t tid_FuncDeclarator**
- const **ktc_treeType_t tid_KRFuncDeclarator**
- const **ktc_treeType_t tid_Name**
- const **ktc_treeType_t tid_Dtor**
- const **ktc_treeType_t tid_SuffixFunc**
- const **ktc_treeType_t tid_OpFunc**
- const **ktc_treeType_t tid_ConvFunc**
- const **ktc_treeType_t tid_TemplateName**
- const **ktc_treeType_t tid_AnyPseudoDtor**
- const **ktc_treeType_t tid_UnparsedName**
- const **ktc_treeType_t tid_UsingDecl**
- const **ktc_treeType_t tid_UsingDirective**
- const **ktc_treeType_t tid_TypeAdjective**
- const **ktc_treeType_t tid_BuiltinType**
- const **ktc_treeType_t tid_TypeOfExpr**
- const **ktc_treeType_t tid_TypeOfType**
- const **ktc_treeType_t tid_PseudoDtor**
- const **ktc_treeType_t tid_QualifiedPseudoDtor**

4.13.1 Detailed Description

This header file is generated by a plug-in. Do not do any changes here as they will be lost.

4.13.2 Variable Documentation

4.13.2.1 const ktc_treeType_t tid_AccessSpecification

Constant to identify 'AccessSpecification' AST node type

4.13.2.2 const ktc_treeType_t tid_AliasDecl

Constant to identify 'AliasDecl' AST node type

4.13.2.3 const ktc_treeType_t tid_AlignAsExpr

Constant to identify 'AlignAsExpr' AST node type

4.13.2.4 const ktc_treeType_t tid_AlignAsType

Constant to identify 'AlignAsType' AST node type

4.13.2.5 const ktc_treeType_t tid_AlignOfExpr

Constant to identify 'AlignOfExpr' AST node type

4.13.2.6 const ktc_treeType_t tid_Any

Constant to identify any AST node type

4.13.2.7 const ktc_treeType_t tid_AnyAttribute

Constant to identify 'AnyAttribute' AST node type

4.13.2.8 const ktc_treeType_t tid_AnyCapture

Constant to identify 'AnyCapture' AST node type

4.13.2.9 const ktc_treeType_t tid_AnyDecl

Constant to identify 'AnyDecl' AST node type

4.13.2.10 const ktc_treeType_t tid_AnyDeclarator

Constant to identify 'AnyDeclarator' AST node type

4.13.2.11 const ktc_treeType_t tid_AnyDesignator

Constant to identify 'AnyDesignator' AST node type

4.13.2.12 const ktc_treeType_t tid_AnyEnumerator

Constant to identify 'AnyEnumerator' AST node type

4.13.2.13 const ktc_treeType_t tid_AnyExpr

Constant to identify 'AnyExpr' AST node type

4.13.2.14 const ktc_treeType_t tid_AnyFuncBody

Constant to identify 'AnyFuncBody' AST node type

4.13.2.15 const ktc_treeType_t tid_AnyInitializer

Constant to identify 'AnyInitializer' AST node type

4.13.2.16 const ktc_treeType_t tid_AnyLabel

Constant to identify 'AnyLabel' AST node type

4.13.2.17 const ktc_treeType_t tid_AnyMemberDecl

Constant to identify 'AnyMemberDecl' AST node type

4.13.2.18 const ktc_treeType_t tid_AnyName

Constant to identify 'AnyName' AST node type

4.13.2.19 const ktc_treeType_t tid_AnyNameQualifier

Constant to identify 'AnyNameQualifier' AST node type

4.13.2.20 const ktc_treeType_t tid_AnyNames

Constant to identify 'AnyNames' AST node type

4.13.2.21 const ktc_treeType_t tid_AnyNameSpec

Constant to identify 'AnyNameSpec' AST node type

4.13.2.22 const ktc_treeType_t tid_AnyNonPtrDeclarator

Constant to identify 'AnyNonPtrDeclarator' AST node type

4.13.2.23 const ktc_treeType_t tid_AnyParamName

Constant to identify 'AnyParamName' AST node type

4.13.2.24 const ktc_treeType_t tid_AnyPropertyFunc

Constant to identify 'AnyPropertyFunc' AST node type

4.13.2.25 const ktc_treeType_t tid_AnyPseudoDtor

Constant to identify 'AnyPseudoDtor' AST node type

4.13.2.26 const ktc_treeType_t tid_AnyStmt

Constant to identify 'AnyStmt' AST node type

4.13.2.27 const ktc_treeType_t tid_AnyTemplateArg

Constant to identify 'AnyTemplateArg' AST node type

4.13.2.28 const ktc_treeType_t tid_AnyTypeName

Constant to identify 'AnyTypeName' AST node type

4.13.2.29 const ktc_treeType_t tid_AnyTypeOf

Constant to identify 'AnyTypeOf' AST node type

4.13.2.30 const ktc_treeType_t tid_AnyTypeParam

Constant to identify 'AnyTypeParam' AST node type

4.13.2.31 const ktc_treeType_t tid_AnyUsing

Constant to identify 'AnyUsing' AST node type

4.13.2.32 const ktc_treeType_t tid_ArrayDeclarator

Constant to identify 'ArrayDeclarator' AST node type

4.13.2.33 const ktc_treeType_t tid_AsmDef

Constant to identify 'AsmDef' AST node type

4.13.2.34 const ktc_treeType_t tid_AsmStmt

Constant to identify 'AsmStmt' AST node type

4.13.2.35 const ktc_treeType_t tid_Attribute

Constant to identify 'Attribute' AST node type

4.13.2.36 const ktc_treeType_t tid_AttributedDeclarator

Constant to identify 'AttributedDeclarator' AST node type

4.13.2.37 const ktc_treeType_t tid_AttributeDeclSpec

Constant to identify 'AttributeDeclSpec' AST node type

4.13.2.38 const ktc_treeType_t tid_Attributes

Constant to identify 'Attributes' AST node type

4.13.2.39 const ktc_treeType_t tid_AttributeSpec

Constant to identify 'AttributeSpec' AST node type

4.13.2.40 const ktc_treeType_t tid_AttributeSpecs

Constant to identify 'AttributeSpecs' AST node type

4.13.2.41 const ktc_treeType_t tid_AttributeWithArgs

Constant to identify 'AttributeWithArgs' AST node type

4.13.2.42 const ktc_treeType_t tid_AutoType

Constant to identify 'AutoType' AST node type

4.13.2.43 const ktc_treeType_t tid_BaseSpec

Constant to identify 'BaseSpec' AST node type

4.13.2.44 const ktc_treeType_t tid_BaseSpecs

Constant to identify 'BaseSpecs' AST node type

4.13.2.45 const ktc_treeType_t tid_BinaryExpr

Constant to identify 'BinaryExpr' AST node type

4.13.2.46 const ktc_treeType_t tid_BitFieldDeclarator

Constant to identify 'BitFieldDeclarator' AST node type

4.13.2.47 const ktc_treeType_t tid_BoolLiteralExpr

Constant to identify 'BoolLiteralExpr' AST node type

4.13.2.48 const ktc_treeType_t tid_BreakStmt

Constant to identify 'BreakStmt' AST node type

4.13.2.49 const ktc_treeType_t tid_BuiltinType

Constant to identify 'BuiltinType' AST node type

4.13.2.50 const ktc_treeType_t tid_CallExpr

Constant to identify 'CallExpr' AST node type

4.13.2.51 const ktc_treeType_t tid_Capture

Constant to identify 'Capture' AST node type

4.13.2.52 const ktc_treeType_t tid_CaptureDefault

Constant to identify 'CaptureDefault' AST node type

4.13.2.53 const ktc_treeType_t tid_CaseLabel

Constant to identify 'CaseLabel' AST node type

4.13.2.54 const ktc_treeType_t tid_CaseRangeLabel

Constant to identify 'CaseRangeLabel' AST node type

4.13.2.55 const ktc_treeType_t tid_CastExpr

Constant to identify 'CastExpr' AST node type

4.13.2.56 const ktc_treeType_t tid_ClassType

Constant to identify 'ClassType' AST node type

4.13.2.57 const ktc_treeType_t tid_CompoundStmt

Constant to identify 'CompoundStmt' AST node type

4.13.2.58 const ktc_treeType_t tid_ConditionalExpr

Constant to identify 'ConditionalExpr' AST node type

4.13.2.59 const ktc_treeType_t tid_ConstExpr

Constant to identify 'ConstExpr' AST node type

4.13.2.60 const ktc_treeType_t tid_ContinueStmt

Constant to identify 'ContinueStmt' AST node type

4.13.2.61 const ktc_treeType_t tid_ConvFunc

Constant to identify 'ConvFunc' AST node type

4.13.2.62 const ktc_treeType_t tid_CopyInitializer

Constant to identify 'CopyInitializer' AST node type

4.13.2.63 const ktc_treeType_t tid_CtorInitializer

Constant to identify 'CtorInitializer' AST node type

4.13.2.64 const ktc_treeType_t tid_CVQualifier

Constant to identify 'CVQualifier' AST node type

4.13.2.65 const ktc_treeType_t tid_Decl

Constant to identify 'Decl' AST node type

4.13.2.66 const ktc_treeType_t tid_DeclEllipsis

Constant to identify 'DeclEllipsis' AST node type

4.13.2.67 const ktc_treeType_t tid_DeclOrStmt

Constant to identify 'DeclOrStmt' AST node type

4.13.2.68 const ktc_treeType_t tid_DeclOrStmts

Constant to identify 'DeclOrStmts' AST node type

4.13.2.69 const ktc_treeType_t tid_DeclSpec

Constant to identify 'DeclSpec' AST node type

4.13.2.70 const ktc_treeType_t tid_DeclSpecs

Constant to identify 'DeclSpecs' AST node type

4.13.2.71 const ktc_treeType_t tid_DefaultException

Constant to identify 'DefaultException' AST node type

4.13.2.72 const ktc_treeType_t tid_DefaultLabel

Constant to identify 'DefaultLabel' AST node type

4.13.2.73 const ktc_treeType_t tid_DeleteExpr

Constant to identify 'DeleteExpr' AST node type

4.13.2.74 const ktc_treeType_t tid_DenyThrowSpec

Constant to identify 'DenyThrowSpec' AST node type

4.13.2.75 const ktc_treeType_t tid_Designators

Constant to identify 'Designators' AST node type

4.13.2.76 const ktc_treeType_t tid_DirectInitializer

Constant to identify 'DirectInitializer' AST node type

4.13.2.77 const ktc_treeType_t tid_DoDeclStmt

Constant to identify 'DoDeclStmt' AST node type

4.13.2.78 const ktc_treeType_t tid_DoStmt

Constant to identify 'DoStmt' AST node type

4.13.2.79 const ktc_treeType_t tid_Dtor

Constant to identify 'Dtor' AST node type

4.13.2.80 const ktc_treeType_t tid_Enumerator

Constant to identify 'Enumerator' AST node type

4.13.2.81 const ktc_treeType_t tid_Enumerators

Constant to identify 'Enumerators' AST node type

4.13.2.82 const ktc_treeType_t tid_EnumType

Constant to identify 'EnumType' AST node type

4.13.2.83 const ktc_treeType_t tid_ExceptHandler

Constant to identify 'ExceptHandler' AST node type

4.13.2.84 const ktc_treeType_t tid_Exception

Constant to identify 'Exception' AST node type

4.13.2.85 const ktc_treeType_t tid_ExceptionSpec

Constant to identify 'ExceptionSpec' AST node type

4.13.2.86 const ktc_treeType_t tid_ExplicitInstantiation

Constant to identify 'ExplicitInstantiation' AST node type

4.13.2.87 const ktc_treeType_t tid_ExprArg

Constant to identify 'ExprArg' AST node type

4.13.2.88 const ktc_treeType_t tid_Exprs

Constant to identify 'Exprs' AST node type

4.13.2.89 const ktc_treeType_t tid_ExprStmt

Constant to identify 'ExprStmt' AST node type

4.13.2.90 const ktc_treeType_t tid_ExprTypeIdExpr

Constant to identify 'ExprTypeIdExpr' AST node type

4.13.2.91 const ktc_treeType_t tid_FieldDesignator

Constant to identify 'FieldDesignator' AST node type

4.13.2.92 const ktc_treeType_t tid_FinallyHandler

Constant to identify 'FinallyHandler' AST node type

4.13.2.93 const ktc_treeType_t tid_ForEachStmt

Constant to identify 'ForEachStmt' AST node type

4.13.2.94 const ktc_treeType_t tid_ForRangeStmt

Constant to identify 'ForRangeStmt' AST node type

4.13.2.95 const ktc_treeType_t tid_ForStmt

Constant to identify 'ForStmt' AST node type

4.13.2.96 const ktc_treeType_t tid_FuncBody

Constant to identify 'FuncBody' AST node type

4.13.2.97 const ktc_treeType_t tid_FuncDeclarator

Constant to identify 'FuncDeclarator' AST node type

4.13.2.98 const ktc_treeType_t tid_FuncDef

Constant to identify 'FuncDef' AST node type

4.13.2.99 const ktc_treeType_t tid_FuncSpec

Constant to identify 'FuncSpec' AST node type

4.13.2.100 const ktc_treeType_t tid_FuncTryBlock

Constant to identify 'FuncTryBlock' AST node type

4.13.2.101 const ktc_treeType_t tid_GlobalScope

Constant to identify 'GlobalScope' AST node type

4.13.2.102 const ktc_treeType_t tid_GotoStmt

Constant to identify 'GotoStmt' AST node type

4.13.2.103 const ktc_treeType_t tid_Handler

Constant to identify 'Handler' AST node type

4.13.2.104 const ktc_treeType_t tid_Handlers

Constant to identify 'Handlers' AST node type

4.13.2.105 const ktc_treeType_t tid_IdExpr

Constant to identify 'IdExpr' AST node type

4.13.2.106 const ktc_treeType_t tid_IfDeclStmt

Constant to identify 'IfDeclStmt' AST node type

4.13.2.107 const ktc_treeType_t tid_IfStmt

Constant to identify 'IfStmt' AST node type

4.13.2.108 const ktc_treeType_t tid_IndexDesignator

Constant to identify 'IndexDesignator' AST node type

4.13.2.109 const ktc_treeType_t tid_IndexExpr

Constant to identify 'IndexExpr' AST node type

4.13.2.110 const ktc_treeType_t tid_InitClause

Constant to identify 'InitClause' AST node type

4.13.2.111 const ktc_treeType_t tid_InitializedDeclarator

Constant to identify 'InitializedDeclarator' AST node type

4.13.2.112 const ktc_treeType_t tid_InitializerExpr

Constant to identify 'InitializerExpr' AST node type

4.13.2.113 const ktc_treeType_t tid_Initializers

Constant to identify 'Initializers' AST node type

4.13.2.114 const ktc_treeType_t tid_KRFuncDeclarator

Constant to identify 'KRFuncDeclarator' AST node type

4.13.2.115 const ktc_treeType_t tid_Label

Constant to identify 'Label' AST node type

4.13.2.116 const ktc_treeType_t tid_LabeledStmt

Constant to identify 'LabeledStmt' AST node type

4.13.2.117 const ktc_treeType_t tid_LambdaDeclarator

Constant to identify 'LambdaDeclarator' AST node type

4.13.2.118 const ktc_treeType_t tid_LambdaExpr

Constant to identify 'LambdaExpr' AST node type

4.13.2.119 const ktc_treeType_t tid_LambdaIntroducer

Constant to identify 'LambdaIntroducer' AST node type

4.13.2.120 const ktc_treeType_t tid_LeaveStmt

Constant to identify 'LeaveStmt' AST node type

4.13.2.121 const ktc_treeType_t tid_LinkageSpec

Constant to identify 'LinkageSpec' AST node type

4.13.2.122 const ktc_treeType_t tid_LiteralExpr

Constant to identify 'LiteralExpr' AST node type

4.13.2.123 const ktc_treeType_t tid_MaybeCtorInitializer

Constant to identify 'MaybeCtorInitializer' AST node type

4.13.2.124 const ktc_treeType_t tid_MaybeDeclarator

Constant to identify 'MaybeDeclarator' AST node type

4.13.2.125 const ktc_treeType_t tid_MaybeException

Constant to identify 'MaybeException' AST node type

4.13.2.126 const ktc_treeType_t tid_MaybeExceptionSpec

Constant to identify 'MaybeExceptionSpec' AST node type

4.13.2.127 const ktc_treeType_t tid_MaybeLambdaDeclarator

Constant to identify 'MaybeLambdaDeclarator' AST node type

4.13.2.128 const ktc_treeType_t tid_MaybeNewInitializer

Constant to identify 'MaybeNewInitializer' AST node type

4.13.2.129 const ktc_treeType_t tid_MaybeTypeId

Constant to identify 'MaybeTypeId' AST node type

4.13.2.130 const ktc_treeType_t tid_MemberDecl

Constant to identify 'MemberDecl' AST node type

4.13.2.131 const ktc_treeType_t tid_MemberDecls

Constant to identify 'MemberDecls' AST node type

4.13.2.132 const ktc_treeType_t tid_MemberDesignator

Constant to identify 'MemberDesignator' AST node type

4.13.2.133 const ktc_treeType_t tid_MemberExpr

Constant to identify 'MemberExpr' AST node type

4.13.2.134 const ktc_treeType_t tid_MemberFunc

Constant to identify 'MemberFunc' AST node type

4.13.2.135 const ktc_treeType_t tid_MemberInitializer

Constant to identify 'MemberInitializer' AST node type

4.13.2.136 const ktc_treeType_t tid_MemberInitializers

Constant to identify 'MemberInitializers' AST node type

4.13.2.137 const ktc_treeType_t tid_MemberTemplate

Constant to identify 'MemberTemplate' AST node type

4.13.2.138 const ktc_treeType_t tid_MemberUsingDecl

Constant to identify 'MemberUsingDecl' AST node type

4.13.2.139 const ktc_treeType_t tid_Name

Constant to identify 'Name' AST node type

4.13.2.140 const ktc_treeType_t tid_NameDeclarator

Constant to identify 'NameDeclarator' AST node type

4.13.2.141 const ktc_treeType_t tid_NamespaceAlias

Constant to identify 'NamespaceAlias' AST node type

4.13.2.142 const ktc_treeType_t tid_NamespaceDecl

Constant to identify 'NamespaceDecl' AST node type

4.13.2.143 const ktc_treeType_t tid_NameSpec

Constant to identify 'NameSpec' AST node type

4.13.2.144 const ktc_treeType_t tid_NewExpr

Constant to identify 'NewExpr' AST node type

4.13.2.145 const ktc_treeType_t tid_NewInitializer

Constant to identify 'NewInitializer' AST node type

4.13.2.146 const ktc_treeType_t tid_NoAttribute

Constant to identify 'NoAttribute' AST node type

4.13.2.147 const ktc_treeType_t tid_NoAttributeSpec

Constant to identify 'NoAttributeSpec' AST node type

4.13.2.148 const ktc_treeType_t tid_NoBaseSpec

Constant to identify 'NoBaseSpec' AST node type

4.13.2.149 const ktc_treeType_t tid_NoCapture

Constant to identify 'NoCapture' AST node type

4.13.2.150 const ktc_treeType_t tid_NoCtorInitializer

Constant to identify 'NoCtorInitializer' AST node type

4.13.2.151 const ktc_treeType_t tid_Node

Constant to identify 'Node' AST node type

4.13.2.152 const ktc_treeType_t tid_NoDeclarator

Constant to identify 'NoDeclarator' AST node type

4.13.2.153 const ktc_treeType_t tid_NoDeclOrStmt

Constant to identify 'NoDeclOrStmt' AST node type

4.13.2.154 const ktc_treeType_t tid_NoDeclSpec

Constant to identify 'NoDeclSpec' AST node type

4.13.2.155 const ktc_treeType_t tid_NoDesignator

Constant to identify 'NoDesignator' AST node type

4.13.2.156 const ktc_treeType_t tid_NoEnumerator

Constant to identify 'NoEnumerator' AST node type

4.13.2.157 const ktc_treeType_t tid_NoException

Constant to identify 'NoException' AST node type

4.13.2.158 const ktc_treeType_t tid_NoExceptionSpec

Constant to identify 'NoExceptionSpec' AST node type

4.13.2.159 const ktc_treeType_t tid_NoExpr

Constant to identify 'NoExpr' AST node type

4.13.2.160 const ktc_treeType_t tid_NoHandler

Constant to identify 'NoHandler' AST node type

4.13.2.161 const ktc_treeType_t tid_NoInitializer

Constant to identify 'NoInitializer' AST node type

4.13.2.162 const ktc_treeType_t tid_NoLambdaDeclarator

Constant to identify 'NoLambdaDeclarator' AST node type

4.13.2.163 const ktc_treeType_t tid_NoMemberDecl

Constant to identify 'NoMemberDecl' AST node type

4.13.2.164 const ktc_treeType_t tid_NoMemberInitializer

Constant to identify 'NoMemberInitializer' AST node type

4.13.2.165 const ktc_treeType_t tid_NoName

Constant to identify 'NoName' AST node type

4.13.2.166 const ktc_treeType_t tid_NoNameQualifier

Constant to identify 'NoNameQualifier' AST node type

4.13.2.167 const ktc_treeType_t tid_NoNewInitializer

Constant to identify 'NoNewInitializer' AST node type

4.13.2.168 const ktc_treeType_t tid_NoParamName

Constant to identify 'NoParamName' AST node type

4.13.2.169 const ktc_treeType_t tid_NoPropertyFunc

Constant to identify 'NoPropertyFunc' AST node type

4.13.2.170 const ktc_treeType_t tid_NoTemplateArg

Constant to identify 'NoTemplateArg' AST node type

4.13.2.171 const ktc_treeType_t tid_NoTemplateParam

Constant to identify 'NoTemplateParam' AST node type

4.13.2.172 const ktc_treeType_t tid_NoTypeId

Constant to identify 'NoTypeId' AST node type

4.13.2.173 const ktc_treeType_t tid_NullptrLiteralExpr

Constant to identify 'NullptrLiteralExpr' AST node type

4.13.2.174 const ktc_treeType_t tid_OpFunc

Constant to identify 'OpFunc' AST node type

4.13.2.175 const ktc_treeType_t tid_Param

Constant to identify 'Param' AST node type

4.13.2.176 const ktc_treeType_t tid_ParamName

Constant to identify 'ParamName' AST node type

4.13.2.177 const ktc_treeType_t tid_ParamNames

Constant to identify 'ParamNames' AST node type

4.13.2.178 const ktc_treeType_t tid_ParensDeclarator

Constant to identify 'ParensDeclarator' AST node type

4.13.2.179 const ktc_treeType_t tid_ParensExpr

Constant to identify 'ParensExpr' AST node type

4.13.2.180 const ktc_treeType_t tid_PromisedFuncBody

Constant to identify 'PromisedFuncBody' AST node type

4.13.2.181 const ktc_treeType_t tid_PromisedMemberDecl

Constant to identify 'PromisedMemberDecl' AST node type

4.13.2.182 const ktc_treeType_t tid_PropertyAttribute

Constant to identify 'PropertyAttribute' AST node type

4.13.2.183 const ktc_treeType_t tid_PropertyFuncs

Constant to identify 'PropertyFuncs' AST node type

4.13.2.184 const ktc_treeType_t tid_PropertyGetFunc

Constant to identify 'PropertyGetFunc' AST node type

4.13.2.185 const ktc_treeType_t tid_PropertyPutFunc

Constant to identify 'PropertyPutFunc' AST node type

4.13.2.186 const ktc_treeType_t tid_PseudoDtor

Constant to identify 'PseudoDtor' AST node type

4.13.2.187 const ktc_treeType_t tid_PtrDeclarator

Constant to identify 'PtrDeclarator' AST node type

4.13.2.188 const ktc_treeType_t tid_QualifiedName

Constant to identify 'QualifiedName' AST node type

4.13.2.189 const ktc_treeType_t tid_QualifiedPseudoDtor

Constant to identify 'QualifiedPseudoDtor' AST node type

4.13.2.190 const ktc_treeType_t tid_RangeDesignator

Constant to identify 'RangeDesignator' AST node type

4.13.2.191 const ktc_treeType_t tid_ReservedTypeSpec

Constant to identify 'ReservedTypeSpec' AST node type

4.13.2.192 const ktc_treeType_t tid_ReturnStmt

Constant to identify 'ReturnStmt' AST node type

4.13.2.193 const ktc_treeType_t tid_SizeOfExpr

Constant to identify 'SizeOfExpr' AST node type

4.13.2.194 const ktc_treeType_t tid_SpecialCastExpr

Constant to identify 'SpecialCastExpr' AST node type

4.13.2.195 const ktc_treeType_t tid_StaticAssertDecl

Constant to identify 'StaticAssertDecl' AST node type

4.13.2.196 const ktc_treeType_t tid_StmtExpr

Constant to identify 'StmtExpr' AST node type

4.13.2.197 const ktc_treeType_t tid_StorageClass

Constant to identify 'StorageClass' AST node type

4.13.2.198 const ktc_treeType_t tid_StringLiteralExpr

Constant to identify 'StringLiteralExpr' AST node type

4.13.2.199 const ktc_treeType_t tid_SuffixFunc

Constant to identify 'SuffixFunc' AST node type

4.13.2.200 const ktc_treeType_t tid_SuperScope

Constant to identify 'SuperScope' AST node type

4.13.2.201 const ktc_treeType_t tid_SwitchDeclStmt

Constant to identify 'SwitchDeclStmt' AST node type

4.13.2.202 const ktc_treeType_t tid_SwitchStmt

Constant to identify 'SwitchStmt' AST node type

4.13.2.203 const ktc_treeType_t tid_TemplateArgs

Constant to identify 'TemplateArgs' AST node type

4.13.2.204 const ktc_treeType_t tid_TemplateDecl

Constant to identify 'TemplateDecl' AST node type

4.13.2.205 const ktc_treeType_t tid_TemplateName

Constant to identify 'TemplateName' AST node type

4.13.2.206 const ktc_treeType_t tid_TemplateParam

Constant to identify 'TemplateParam' AST node type

4.13.2.207 const ktc_treeType_t tid_TemplateParams

Constant to identify 'TemplateParams' AST node type

4.13.2.208 const ktc_treeType_t tid_TemplateSpec

Constant to identify 'TemplateSpec' AST node type

4.13.2.209 const ktc_treeType_t tid_TemplateTypeArg

Constant to identify 'TemplateTypeArg' AST node type

4.13.2.210 const ktc_treeType_t tid_TemplateTypeParam

Constant to identify 'TemplateTypeParam' AST node type

4.13.2.211 const ktc_treeType_t tid_ThisExpr

Constant to identify 'ThisExpr' AST node type

4.13.2.212 const ktc_treeType_t tid_ThrowExpr

Constant to identify 'ThrowExpr' AST node type

4.13.2.213 const ktc_treeType_t tid_TranslationUnit

Constant to identify 'TranslationUnit' AST node type

4.13.2.214 const ktc_treeType_t tid_TruncatedInitClause

Constant to identify 'TruncatedInitClause' AST node type

4.13.2.215 const ktc_treeType_t tid_TryExceptStmt

Constant to identify 'TryExceptStmt' AST node type

4.13.2.216 const ktc_treeType_t tid_TryFinallyStmt

Constant to identify 'TryFinallyStmt' AST node type

4.13.2.217 const ktc_treeType_t tid_TryStmt

Constant to identify 'TryStmt' AST node type

4.13.2.218 const ktc_treeType_t tid_TypeAdjective

Constant to identify 'TypeAdjective' AST node type

4.13.2.219 const ktc_treeType_t tid_TypeArg

Constant to identify 'TypeArg' AST node type

4.13.2.220 const ktc_treeType_t tid_TypeConvExpr

Constant to identify 'TypeConvExpr' AST node type

4.13.2.221 const ktc_treeType_t tid_TypeId

Constant to identify 'TypeId' AST node type

4.13.2.222 const ktc_treeType_t tid_TypeName

Constant to identify 'TypeName' AST node type

4.13.2.223 const ktc_treeType_t tid_TypeOfExpr

Constant to identify 'TypeOfExpr' AST node type

4.13.2.224 const ktc_treeType_t tid_TypeOfSpec

Constant to identify 'TypeOfSpec' AST node type

4.13.2.225 const ktc_treeType_t tid_TypeOfType

Constant to identify 'TypeOfType' AST node type

4.13.2.226 const ktc_treeType_t tid_TypeParam

Constant to identify 'TypeParam' AST node type

4.13.2.227 const ktc_treeType_t tid_TypeTypeIdExpr

Constant to identify 'TypeTypeIdExpr' AST node type

4.13.2.228 const ktc_treeType_t tid_UnaryExpr

Constant to identify 'UnaryExpr' AST node type

4.13.2.229 const ktc_treeType_t tid_UnparsedDecl

Constant to identify 'UnparsedDecl' AST node type

4.13.2.230 const ktc_treeType_t tid_UnparsedDeclarator

Constant to identify 'UnparsedDeclarator' AST node type

4.13.2.231 const ktc_treeType_t tid_UnparsedDeclSpec

Constant to identify 'UnparsedDeclSpec' AST node type

4.13.2.232 const ktc_treeType_t tid_UnparsedEnumerator

Constant to identify 'UnparsedEnumerator' AST node type

4.13.2.233 const ktc_treeType_t tid_UnparsedException

Constant to identify 'UnparsedException' AST node type

4.13.2.234 const ktc_treeType_t tid_UnparsedExpr

Constant to identify 'UnparsedExpr' AST node type

4.13.2.235 const ktc_treeType_t tid_UnparsedInitializer

Constant to identify 'UnparsedInitializer' AST node type

4.13.2.236 const ktc_treeType_t tid_UnparsedLabel

Constant to identify 'UnparsedLabel' AST node type

4.13.2.237 const ktc_treeType_t tid_UnparsedMemberDecl

Constant to identify 'UnparsedMemberDecl' AST node type

4.13.2.238 const ktc_treeType_t tid_UnparsedName

Constant to identify 'UnparsedName' AST node type

4.13.2.239 const ktc_treeType_t tid_UnparsedNameQualifier

Constant to identify 'UnparsedNameQualifier' AST node type

4.13.2.240 const ktc_treeType_t tid_UnparsedParamName

Constant to identify 'UnparsedParamName' AST node type

4.13.2.241 const ktc_treeType_t tid_UnparsedPropertyFunc

Constant to identify 'UnparsedPropertyFunc' AST node type

4.13.2.242 const ktc_treeType_t tid_UnparsedStmt

Constant to identify 'UnparsedStmt' AST node type

4.13.2.243 const ktc_treeType_t tid_UnqualifiedName

Constant to identify 'UnqualifiedName' AST node type

4.13.2.244 const ktc_treeType_t tid_UserLiteralExpr

Constant to identify 'UserLiteralExpr' AST node type

4.13.2.245 const ktc_treeType_t tid_UserStringLiteralExpr

Constant to identify 'UserStringLiteralExpr' AST node type

4.13.2.246 const ktc_treeType_t tid_UsingDecl

Constant to identify 'UsingDecl' AST node type

4.13.2.247 const ktc_treeType_t tid_UsingDirective

Constant to identify 'UsingDirective' AST node type

4.13.2.248 const ktc_treeType_t tid_WhileDeclStmt

Constant to identify 'WhileDeclStmt' AST node type

4.13.2.249 const ktc_treeType_t tid_WhileStmt

Constant to identify 'WhileStmt' AST node type

4.14 Tree type checking predicates

Functions

- int **ktc_is_Node** (**ktc_tree_t** t)
- int **ktc_is_TranslationUnit** (**ktc_tree_t** t)
- int **ktc_is_DeclOrStmts** (**ktc_tree_t** t)
- int **ktc_is_AnyFuncBody** (**ktc_tree_t** t)
- int **ktc_is_Handlers** (**ktc_tree_t** t)
- int **ktc_is_ExceptHandler** (**ktc_tree_t** t)
- int **ktc_is_FinallyHandler** (**ktc_tree_t** t)
- int **ktc_is_MaybeException** (**ktc_tree_t** t)
- int **ktc_is_TemplateParams** (**ktc_tree_t** t)
- int **ktc_is_DeclSpecs** (**ktc_tree_t** t)
- int **ktc_is_AttributeSpecs** (**ktc_tree_t** t)
- int **ktc_is_Attributes** (**ktc_tree_t** t)
- int **ktc_is_PropertyFuncs** (**ktc_tree_t** t)
- int **ktc_is_MemberDecls** (**ktc_tree_t** t)
- int **ktc_is_Enumerators** (**ktc_tree_t** t)
- int **ktc_is_MaybeDeclarator** (**ktc_tree_t** t)
- int **ktc_is_ParamNames** (**ktc_tree_t** t)
- int **ktc_is_Initializers** (**ktc_tree_t** t)
- int **ktc_is_Designators** (**ktc_tree_t** t)
- int **ktc_is_AnyLabel** (**ktc_tree_t** t)
- int **ktc_is_Exprs** (**ktc_tree_t** t)
- int **ktc_is_AnyNameQualifier** (**ktc_tree_t** t)
- int **ktc_is_AnyNames** (**ktc_tree_t** t)
- int **ktc_is_TemplateArgs** (**ktc_tree_t** t)
- int **ktc_is_BaseSpecs** (**ktc_tree_t** t)
- int **ktc_is_MaybeLambdaDeclarator** (**ktc_tree_t** t)
- int **ktc_is_LambdaIntroducer** (**ktc_tree_t** t)
- int **ktc_is_AnyCapture** (**ktc_tree_t** t)
- int **ktc_is_MaybeTypeId** (**ktc_tree_t** t)
- int **ktc_is_MaybeNewInitializer** (**ktc_tree_t** t)
- int **ktc_is_MemberInitializers** (**ktc_tree_t** t)
- int **ktc_is_MaybeCtorInitializer** (**ktc_tree_t** t)
- int **ktc_is_MaybeExceptionSpec** (**ktc_tree_t** t)
- int **ktc_is_NoDeclOrStmt** (**ktc_tree_t** t)
- int **ktc_is_DeclEllipsis** (**ktc_tree_t** t)
- int **ktc_is_DeclOrStmt** (**ktc_tree_t** t)
- int **ktc_is_FuncBody** (**ktc_tree_t** t)
- int **ktc_is_FuncTryBlock** (**ktc_tree_t** t)
- int **ktc_is_PromisedFuncBody** (**ktc_tree_t** t)
- int **ktc_is_NoHandler** (**ktc_tree_t** t)
- int **ktc_is_Handler** (**ktc_tree_t** t)
- int **ktc_is_NoException** (**ktc_tree_t** t)
- int **ktc_is_Exception** (**ktc_tree_t** t)
- int **ktc_is_DefaultException** (**ktc_tree_t** t)
- int **ktc_is_UnparsedException** (**ktc_tree_t** t)
- int **ktc_is_NoTemplateParam** (**ktc_tree_t** t)
- int **ktc_is_TemplateParam** (**ktc_tree_t** t)
- int **ktc_is_NoDeclSpec** (**ktc_tree_t** t)
- int **ktc_is_DeclSpec** (**ktc_tree_t** t)
- int **ktc_is_NoAttributeSpec** (**ktc_tree_t** t)

- int **ktc_is_AttributeSpec** (**ktc_tree_t** t)
- int **ktc_is_NoAttribute** (**ktc_tree_t** t)
- int **ktc_is_AnyAttribute** (**ktc_tree_t** t)
- int **ktc_is_NoPropertyFunc** (**ktc_tree_t** t)
- int **ktc_is_AnyPropertyFunc** (**ktc_tree_t** t)
- int **ktc_is_NoMemberDecl** (**ktc_tree_t** t)
- int **ktc_is_AnyMemberDecl** (**ktc_tree_t** t)
- int **ktc_is_NoEnumerator** (**ktc_tree_t** t)
- int **ktc_is_AnyEnumerator** (**ktc_tree_t** t)
- int **ktc_is_NoDeclarator** (**ktc_tree_t** t)
- int **ktc_is_AnyDeclarator** (**ktc_tree_t** t)
- int **ktc_is_NoParamName** (**ktc_tree_t** t)
- int **ktc_is_AnyParamName** (**ktc_tree_t** t)
- int **ktc_is_NoInitializer** (**ktc_tree_t** t)
- int **ktc_is_AnyInitializer** (**ktc_tree_t** t)
- int **ktc_is_NoDesignator** (**ktc_tree_t** t)
- int **ktc_is_AnyDesignator** (**ktc_tree_t** t)
- int **ktc_is_Label** (**ktc_tree_t** t)
- int **ktc_is_CaseLabel** (**ktc_tree_t** t)
- int **ktc_is_DefaultLabel** (**ktc_tree_t** t)
- int **ktc_is_CaseRangeLabel** (**ktc_tree_t** t)
- int **ktc_is_UnparsedLabel** (**ktc_tree_t** t)
- int **ktc_is_NoExpr** (**ktc_tree_t** t)
- int **ktc_is_AnyExpr** (**ktc_tree_t** t)
- int **ktc_is_NoNameQualifier** (**ktc_tree_t** t)
- int **ktc_is_GlobalScope** (**ktc_tree_t** t)
- int **ktc_is_SuperScope** (**ktc_tree_t** t)
- int **ktc_is_AnyNameSpec** (**ktc_tree_t** t)
- int **ktc_is_NoName** (**ktc_tree_t** t)
- int **ktc_is_AnyName** (**ktc_tree_t** t)
- int **ktc_is_NoTemplateArg** (**ktc_tree_t** t)
- int **ktc_is_AnyTemplateArg** (**ktc_tree_t** t)
- int **ktc_is_NoBaseSpec** (**ktc_tree_t** t)
- int **ktc_is_BaseSpec** (**ktc_tree_t** t)
- int **ktc_is_NoLambdaDeclarator** (**ktc_tree_t** t)
- int **ktc_is_LambdaDeclarator** (**ktc_tree_t** t)
- int **ktc_is_CaptureDefault** (**ktc_tree_t** t)
- int **ktc_is_Capture** (**ktc_tree_t** t)
- int **ktc_is_NoCapture** (**ktc_tree_t** t)
- int **ktc_is_NoTypeId** (**ktc_tree_t** t)
- int **ktc_is_TypeId** (**ktc_tree_t** t)
- int **ktc_is_NoNewInitializer** (**ktc_tree_t** t)
- int **ktc_is_NewInitializer** (**ktc_tree_t** t)
- int **ktc_is_NoMemberInitializer** (**ktc_tree_t** t)
- int **ktc_is_MemberInitializer** (**ktc_tree_t** t)
- int **ktc_is_NoCtorInitializer** (**ktc_tree_t** t)
- int **ktc_is_CtorInitializer** (**ktc_tree_t** t)
- int **ktc_is_NoExceptionSpec** (**ktc_tree_t** t)
- int **ktc_is_ExceptionSpec** (**ktc_tree_t** t)
- int **ktc_is_DenyThrowSpec** (**ktc_tree_t** t)
- int **ktc_is_AnyDecl** (**ktc_tree_t** t)
- int **ktc_is_AnyStmt** (**ktc_tree_t** t)
- int **ktc_is_AnyTypeParam** (**ktc_tree_t** t)
- int **ktc_is_Param** (**ktc_tree_t** t)
- int **ktc_is_AutoType** (**ktc_tree_t** t)

- int **ktc_is_ConstExpr** (**ktc_tree_t** t)
- int **ktc_is_StorageClass** (**ktc_tree_t** t)
- int **ktc_is_AlignAsExpr** (**ktc_tree_t** t)
- int **ktc_is_AlignAsType** (**ktc_tree_t** t)
- int **ktc_is_AnyTypeName** (**ktc_tree_t** t)
- int **ktc_is_FuncSpec** (**ktc_tree_t** t)
- int **ktc_is_AttributeDeclSpec** (**ktc_tree_t** t)
- int **ktc_is_UnparsedDeclSpec** (**ktc_tree_t** t)
- int **ktc_is_Attribute** (**ktc_tree_t** t)
- int **ktc_is_AttributeWithArgs** (**ktc_tree_t** t)
- int **ktc_is_PropertyAttribute** (**ktc_tree_t** t)
- int **ktc_is_PropertyPutFunc** (**ktc_tree_t** t)
- int **ktc_is_PropertyGetFunc** (**ktc_tree_t** t)
- int **ktc_is_UnparsedPropertyFunc** (**ktc_tree_t** t)
- int **ktc_is_MemberDecl** (**ktc_tree_t** t)
- int **ktc_is_MemberFunc** (**ktc_tree_t** t)
- int **ktc_is_AccessSpecification** (**ktc_tree_t** t)
- int **ktc_is_MemberTemplate** (**ktc_tree_t** t)
- int **ktc_is_MemberUsingDecl** (**ktc_tree_t** t)
- int **ktc_is_UnparsedMemberDecl** (**ktc_tree_t** t)
- int **ktc_is_PromisedMemberDecl** (**ktc_tree_t** t)
- int **ktc_is_Enumerator** (**ktc_tree_t** t)
- int **ktc_is_UnparsedEnumerator** (**ktc_tree_t** t)
- int **ktc_is_AnyNonPtrDeclarator** (**ktc_tree_t** t)
- int **ktc_is_PtrDeclarator** (**ktc_tree_t** t)
- int **ktc_is_BitFieldDeclarator** (**ktc_tree_t** t)
- int **ktc_is_AttributedDeclarator** (**ktc_tree_t** t)
- int **ktc_is_InitializedDeclarator** (**ktc_tree_t** t)
- int **ktc_is_UnparsedDeclarator** (**ktc_tree_t** t)
- int **ktc_is_ParamName** (**ktc_tree_t** t)
- int **ktc_is_UnparsedParamName** (**ktc_tree_t** t)
- int **ktc_is_CopyInitializer** (**ktc_tree_t** t)
- int **ktc_is_InitClause** (**ktc_tree_t** t)
- int **ktc_is_TruncatedInitClause** (**ktc_tree_t** t)
- int **ktc_is_DirectInitializer** (**ktc_tree_t** t)
- int **ktc_is_UnparsedInitializer** (**ktc_tree_t** t)
- int **ktc_is_FieldDesignator** (**ktc_tree_t** t)
- int **ktc_is_MemberDesignator** (**ktc_tree_t** t)
- int **ktc_is_IndexDesignator** (**ktc_tree_t** t)
- int **ktc_is_RangeDesignator** (**ktc_tree_t** t)
- int **ktc_is_IdExpr** (**ktc_tree_t** t)
- int **ktc_is_BoolLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_LiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_UserLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_StringLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_UserStringLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_NullptrLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_MemberExpr** (**ktc_tree_t** t)
- int **ktc_is_CallExpr** (**ktc_tree_t** t)
- int **ktc_is_TypeConvExpr** (**ktc_tree_t** t)
- int **ktc_is_IndexExpr** (**ktc_tree_t** t)
- int **ktc_is_UnaryExpr** (**ktc_tree_t** t)
- int **ktc_is_ThrowExpr** (**ktc_tree_t** t)
- int **ktc_is_BinaryExpr** (**ktc_tree_t** t)
- int **ktc_is_ConditionalExpr** (**ktc_tree_t** t)

- int **ktc_is_SizeOfExpr** (**ktc_tree_t** t)
- int **ktc_is_AlignOfExpr** (**ktc_tree_t** t)
- int **ktc_is_CastExpr** (**ktc_tree_t** t)
- int **ktc_is_SpecialCastExpr** (**ktc_tree_t** t)
- int **ktc_is_ParensExpr** (**ktc_tree_t** t)
- int **ktc_is_ThisExpr** (**ktc_tree_t** t)
- int **ktc_is_NewExpr** (**ktc_tree_t** t)
- int **ktc_is_DeleteExpr** (**ktc_tree_t** t)
- int **ktc_is_TypeTypeIdExpr** (**ktc_tree_t** t)
- int **ktc_is_ExprTypeIdExpr** (**ktc_tree_t** t)
- int **ktc_is_StmtExpr** (**ktc_tree_t** t)
- int **ktc_is_InitializerExpr** (**ktc_tree_t** t)
- int **ktc_is_LambdaExpr** (**ktc_tree_t** t)
- int **ktc_is_UnparsedExpr** (**ktc_tree_t** t)
- int **ktc_is_NameSpec** (**ktc_tree_t** t)
- int **ktc_is_TemplateSpec** (**ktc_tree_t** t)
- int **ktc_is_TypeOfSpec** (**ktc_tree_t** t)
- int **ktc_is_UnparsedNameQualifier** (**ktc_tree_t** t)
- int **ktc_is_QualifiedName** (**ktc_tree_t** t)
- int **ktc_is_UnqualifiedName** (**ktc_tree_t** t)
- int **ktc_is_ExprArg** (**ktc_tree_t** t)
- int **ktc_is_TypeArg** (**ktc_tree_t** t)
- int **ktc_is_TemplateTypeArg** (**ktc_tree_t** t)
- int **ktc_is_FuncDef** (**ktc_tree_t** t)
- int **ktc_is_Decl** (**ktc_tree_t** t)
- int **ktc_is_TemplateDecl** (**ktc_tree_t** t)
- int **ktc_is_LinkageSpec** (**ktc_tree_t** t)
- int **ktc_is_ExplicitInstantiation** (**ktc_tree_t** t)
- int **ktc_is_NamespaceDecl** (**ktc_tree_t** t)
- int **ktc_is_NamespaceAlias** (**ktc_tree_t** t)
- int **ktc_is_AnyUsing** (**ktc_tree_t** t)
- int **ktc_is_AliasDecl** (**ktc_tree_t** t)
- int **ktc_is_AsmDef** (**ktc_tree_t** t)
- int **ktc_is_StaticAssertDecl** (**ktc_tree_t** t)
- int **ktc_is_UnparsedDecl** (**ktc_tree_t** t)
- int **ktc_is_LabeledStmt** (**ktc_tree_t** t)
- int **ktc_is_ExprStmt** (**ktc_tree_t** t)
- int **ktc_is_CompoundStmt** (**ktc_tree_t** t)
- int **ktc_is_IfStmt** (**ktc_tree_t** t)
- int **ktc_is_IfDeclStmt** (**ktc_tree_t** t)
- int **ktc_is_SwitchStmt** (**ktc_tree_t** t)
- int **ktc_is_SwitchDeclStmt** (**ktc_tree_t** t)
- int **ktc_is_WhileStmt** (**ktc_tree_t** t)
- int **ktc_is_WhileDeclStmt** (**ktc_tree_t** t)
- int **ktc_is_DoStmt** (**ktc_tree_t** t)
- int **ktc_is_DoDeclStmt** (**ktc_tree_t** t)
- int **ktc_is_ForStmt** (**ktc_tree_t** t)
- int **ktc_is_ForEachStmt** (**ktc_tree_t** t)
- int **ktc_is_ForRangeStmt** (**ktc_tree_t** t)
- int **ktc_is_GotoStmt** (**ktc_tree_t** t)
- int **ktc_is_ContinueStmt** (**ktc_tree_t** t)
- int **ktc_is_BreakStmt** (**ktc_tree_t** t)
- int **ktc_is_ReturnStmt** (**ktc_tree_t** t)
- int **ktc_is_TryStmt** (**ktc_tree_t** t)
- int **ktc_is_TryExceptStmt** (**ktc_tree_t** t)

- int **ktc_is_TryFinallyStmt** (**ktc_tree_t** t)
- int **ktc_is_LeaveStmt** (**ktc_tree_t** t)
- int **ktc_is_AsmStmt** (**ktc_tree_t** t)
- int **ktc_is_UnparsedStmt** (**ktc_tree_t** t)
- int **ktc_is_TypeParam** (**ktc_tree_t** t)
- int **ktc_is_TemplateTypeParam** (**ktc_tree_t** t)
- int **ktc_is_CVQualifier** (**ktc_tree_t** t)
- int **ktc_is_ReservedTypeSpec** (**ktc_tree_t** t)
- int **ktc_is_TypeName** (**ktc_tree_t** t)
- int **ktc_is_EnumType** (**ktc_tree_t** t)
- int **ktc_is_ClassType** (**ktc_tree_t** t)
- int **ktc_is_AnyTypeOf** (**ktc_tree_t** t)
- int **ktc_is_NameDeclarator** (**ktc_tree_t** t)
- int **ktc_is_ParensDeclarator** (**ktc_tree_t** t)
- int **ktc_is_ArrayDeclarator** (**ktc_tree_t** t)
- int **ktc_is_FuncDeclarator** (**ktc_tree_t** t)
- int **ktc_is_KRFuncDeclarator** (**ktc_tree_t** t)
- int **ktc_is_Name** (**ktc_tree_t** t)
- int **ktc_is_Dtor** (**ktc_tree_t** t)
- int **ktc_is_SuffixFunc** (**ktc_tree_t** t)
- int **ktc_is_OpFunc** (**ktc_tree_t** t)
- int **ktc_is_ConvFunc** (**ktc_tree_t** t)
- int **ktc_is_TemplateName** (**ktc_tree_t** t)
- int **ktc_is_AnyPseudoDtor** (**ktc_tree_t** t)
- int **ktc_is_UnparsedName** (**ktc_tree_t** t)
- int **ktc_is_UsingDecl** (**ktc_tree_t** t)
- int **ktc_is_UsingDirective** (**ktc_tree_t** t)
- int **ktc_is_TypeAdjective** (**ktc_tree_t** t)
- int **ktc_is_BuiltinType** (**ktc_tree_t** t)
- int **ktc_is_TypeOfExpr** (**ktc_tree_t** t)
- int **ktc_is_TypeOfType** (**ktc_tree_t** t)
- int **ktc_is_PseudoDtor** (**ktc_tree_t** t)
- int **ktc_is_QualifiedPseudoDtor** (**ktc_tree_t** t)

4.14.1 Detailed Description

4.14.2 Function Documentation

4.14.2.1 int ktc_is_AccessSpecification (ktc_tree_t *t*)

Checks that root of subtree has 'AccessSpecification' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AccessSpecification', 0 otherwise

4.14.2.2 int ktc_is_AliasDecl (ktc_tree_t *t*)

Checks that root of subtree has 'AliasDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AliasDecl', 0 otherwise

4.14.2.3 int ktc_is_AlignAsExpr (ktc_tree_t *t*)

Checks that root of subtree has 'AlignAsExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AlignAsExpr', 0 otherwise

4.14.2.4 int ktc_is_AlignAsType (ktc_tree_t *t*)

Checks that root of subtree has 'AlignAsType' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AlignAsType', 0 otherwise

4.14.2.5 int ktc_is_AlignOfExpr (ktc_tree_t *t*)

Checks that root of subtree has 'AlignOfExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AlignOfExpr', 0 otherwise

4.14.2.6 int ktc_is_AnyAttribute (ktc_tree_t *t*)

Checks that root of subtree has 'AnyAttribute' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyAttribute', 0 otherwise

4.14.2.7 int ktc_is_AnyCapture (ktc_tree_t *t*)

Checks that root of subtree has 'AnyCapture' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyCapture', 0 otherwise

4.14.2.8 int ktc_is_AnyDecl (ktc_tree_t *t*)

Checks that root of subtree has 'AnyDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyDecl', 0 otherwise

4.14.2.9 int ktc_is_AnyDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'AnyDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyDeclarator', 0 otherwise

4.14.2.10 int ktc_is_AnyDesignator (ktc_tree_t *t*)

Checks that root of subtree has 'AnyDesignator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyDesignator', 0 otherwise

4.14.2.11 int ktc_is_AnyEnumerator (ktc_tree_t *t*)

Checks that root of subtree has 'AnyEnumerator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyEnumerator', 0 otherwise

4.14.2.12 int ktc_is_AnyExpr (ktc_tree_t *t*)

Checks that root of subtree has 'AnyExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyExpr', 0 otherwise

4.14.2.13 int ktc_is_AnyFuncBody (ktc_tree_t *t*)

Checks that root of subtree has 'AnyFuncBody' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyFuncBody', 0 otherwise

4.14.2.14 int ktc_is_AnyInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'AnyInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyInitializer', 0 otherwise

4.14.2.15 int ktc_is_AnyLabel (ktc_tree_t *t*)

Checks that root of subtree has 'AnyLabel' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyLabel', 0 otherwise

4.14.2.16 int ktc_is_AnyMemberDecl (ktc_tree_t *t*)

Checks that root of subtree has 'AnyMemberDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyMemberDecl', 0 otherwise

4.14.2.17 int ktc_is_AnyName (ktc_tree_t *t*)

Checks that root of subtree has 'AnyName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyName', 0 otherwise

4.14.2.18 int ktc_is_AnyNameQualifier (ktc_tree_t *t*)

Checks that root of subtree has 'AnyNameQualifier' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyNameQualifier', 0 otherwise

4.14.2.19 int ktc_is_AnyNames (ktc_tree_t *t*)

Checks that root of subtree has 'AnyNames' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyNames', 0 otherwise

4.14.2.20 int ktc_is_AnyNameSpec (ktc_tree_t *t*)

Checks that root of subtree has 'AnyNameSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyNameSpec', 0 otherwise

4.14.2.21 int ktc_is_AnyNonPtrDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'AnyNonPtrDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyNonPtrDeclarator', 0 otherwise

4.14.2.22 int ktc_is_AnyParamName (ktc_tree_t *t*)

Checks that root of subtree has 'AnyParamName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyParamName', 0 otherwise

4.14.2.23 int ktc_is_AnyPropertyFunc (ktc_tree_t *t*)

Checks that root of subtree has 'AnyPropertyFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyPropertyFunc', 0 otherwise

4.14.2.24 int ktc_is_AnyPseudoDtor (ktc_tree_t *t*)

Checks that root of subtree has 'AnyPseudoDtor' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyPseudoDtor', 0 otherwise

4.14.2.25 int ktc_is_AnyStmt (ktc_tree_t *t*)

Checks that root of subtree has 'AnyStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyStmt', 0 otherwise

4.14.2.26 int ktc_is_AnyTemplateArg (ktc_tree_t *t*)

Checks that root of subtree has 'AnyTemplateArg' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyTemplateArg', 0 otherwise

4.14.2.27 int ktc_is_AnyTypeName (ktc_tree_t *t*)

Checks that root of subtree has 'AnyTypeName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyTypeName', 0 otherwise

4.14.2.28 int ktc_is_AnyTypeOf (ktc_tree_t *t*)

Checks that root of subtree has 'AnyTypeOf' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyTypeOf', 0 otherwise

4.14.2.29 int ktc_is_AnyTypeParam (ktc_tree_t *t*)

Checks that root of subtree has 'AnyTypeParam' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyTypeParam', 0 otherwise

4.14.2.30 int ktc_is_AnyUsing (ktc_tree_t *t*)

Checks that root of subtree has 'AnyUsing' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AnyUsing', 0 otherwise

4.14.2.31 int ktc_is_ArrayDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'ArrayDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ArrayDeclarator', 0 otherwise

4.14.2.32 int ktc_is_AsmDef (ktc_tree_t *t*)

Checks that root of subtree has 'AsmDef' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AsmDef', 0 otherwise

4.14.2.33 int ktc_is_AsmStmt (ktc_tree_t *t*)

Checks that root of subtree has 'AsmStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AsmStmt', 0 otherwise

4.14.2.34 int ktc_is_Attribute (ktc_tree_t *t*)

Checks that root of subtree has 'Attribute' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Attribute', 0 otherwise

4.14.2.35 int ktc_is_AttributedDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'AttributedDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AttributedDeclarator', 0 otherwise

4.14.2.36 int ktc_is_AttributeDeclSpec (ktc_tree_t *t*)

Checks that root of subtree has 'AttributeDeclSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AttributeDeclSpec', 0 otherwise

4.14.2.37 int ktc_is_Attributes (ktc_tree_t *t*)

Checks that root of subtree has 'Attributes' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Attributes', 0 otherwise

4.14.2.38 int ktc_is_AttributeSpec (ktc_tree_t *t*)

Checks that root of subtree has 'AttributeSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AttributeSpec', 0 otherwise

4.14.2.39 int ktc_is_AttributeSpecs (ktc_tree_t *t*)

Checks that root of subtree has 'AttributeSpecs' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AttributeSpecs', 0 otherwise

4.14.2.40 int ktc_is_AttributeWithArgs (ktc_tree_t *t*)

Checks that root of subtree has 'AttributeWithArgs' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AttributeWithArgs', 0 otherwise

4.14.2.41 int ktc_is_AutoType (ktc_tree_t *t*)

Checks that root of subtree has 'AutoType' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'AutoType', 0 otherwise

4.14.2.42 int ktc_is_BaseSpec (ktc_tree_t *t*)

Checks that root of subtree has 'BaseSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'BaseSpec', 0 otherwise

4.14.2.43 int ktc_is_BaseSpecs (ktc_tree_t *t*)

Checks that root of subtree has 'BaseSpecs' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'BaseSpecs', 0 otherwise

4.14.2.44 int ktc_is_BinaryExpr (ktc_tree_t *t*)

Checks that root of subtree has 'BinaryExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'BinaryExpr', 0 otherwise

4.14.2.45 int ktc_is_BitFieldDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'BitFieldDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'BitFieldDeclarator', 0 otherwise

4.14.2.46 int ktc_is_BoolLiteralExpr (ktc_tree_t *t*)

Checks that root of subtree has 'BoolLiteralExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'BoolLiteralExpr', 0 otherwise

4.14.2.47 int ktc_is_BreakStmt (ktc_tree_t *t*)

Checks that root of subtree has 'BreakStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'BreakStmt', 0 otherwise

4.14.2.48 int ktc_is_BuiltinType (ktc_tree_t *t*)

Checks that root of subtree has 'BuiltinType' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'BuiltinType', 0 otherwise

4.14.2.49 int ktc_is_CallExpr (ktc_tree_t *t*)

Checks that root of subtree has 'CallExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CallExpr', 0 otherwise

4.14.2.50 int ktc_is_Capture (ktc_tree_t *t*)

Checks that root of subtree has 'Capture' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Capture', 0 otherwise

4.14.2.51 int ktc_is_CaptureDefault (ktc_tree_t *t*)

Checks that root of subtree has 'CaptureDefault' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CaptureDefault', 0 otherwise

4.14.2.52 int ktc_is_CaseLabel (ktc_tree_t *t*)

Checks that root of subtree has 'CaseLabel' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CaseLabel', 0 otherwise

4.14.2.53 int ktc_is_CaseRangeLabel (ktc_tree_t *t*)

Checks that root of subtree has 'CaseRangeLabel' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CaseRangeLabel', 0 otherwise

4.14.2.54 int ktc_is_CastExpr (ktc_tree_t *t*)

Checks that root of subtree has 'CastExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CastExpr', 0 otherwise

4.14.2.55 int ktc_is_ClassType (ktc_tree_t *t*)

Checks that root of subtree has 'ClassType' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ClassType', 0 otherwise

4.14.2.56 int ktc_is_CompoundStmt (ktc_tree_t *t*)

Checks that root of subtree has 'CompoundStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CompoundStmt', 0 otherwise

4.14.2.57 int ktc_is_ConditionalExpr (ktc_tree_t *t*)

Checks that root of subtree has 'ConditionalExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ConditionalExpr', 0 otherwise

4.14.2.58 int ktc_is_ConstExpr (ktc_tree_t *t*)

Checks that root of subtree has 'ConstExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ConstExpr', 0 otherwise

4.14.2.59 int ktc_is_ContinueStmt (ktc_tree_t *t*)

Checks that root of subtree has 'ContinueStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ContinueStmt', 0 otherwise

4.14.2.60 int ktc_is_ConvFunc (ktc_tree_t *t*)

Checks that root of subtree has 'ConvFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ConvFunc', 0 otherwise

4.14.2.61 int ktc_is_CopyInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'CopyInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CopyInitializer', 0 otherwise

4.14.2.62 int ktc_is_CtorInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'CtorInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CtorInitializer', 0 otherwise

4.14.2.63 int ktc_is_CVQualifier (ktc_tree_t *t*)

Checks that root of subtree has 'CVQualifier' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'CVQualifier', 0 otherwise

4.14.2.64 int ktc_is_Decl (ktc_tree_t *t*)

Checks that root of subtree has 'Decl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Decl', 0 otherwise

4.14.2.65 int ktc_is_DeclEllipsis (ktc_tree_t *t*)

Checks that root of subtree has 'DeclEllipsis' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DeclEllipsis', 0 otherwise

4.14.2.66 int ktc_is_DeclOrStmt (ktc_tree_t *t*)

Checks that root of subtree has 'DeclOrStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DeclOrStmt', 0 otherwise

4.14.2.67 int ktc_is_DeclOrStmts (ktc_tree_t *t*)

Checks that root of subtree has 'DeclOrStmts' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DeclOrStmts', 0 otherwise

4.14.2.68 int ktc_is_DeclSpec (ktc_tree_t *t*)

Checks that root of subtree has 'DeclSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DeclSpec', 0 otherwise

4.14.2.69 int ktc_is_DeclSpecs (ktc_tree_t *t*)

Checks that root of subtree has 'DeclSpecs' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DeclSpecs', 0 otherwise

4.14.2.70 int ktc_is_DefaultException (ktc_tree_t *t*)

Checks that root of subtree has 'DefaultException' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DefaultException', 0 otherwise

4.14.2.71 int ktc_is_DefaultLabel (ktc_tree_t *t*)

Checks that root of subtree has 'DefaultLabel' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DefaultLabel', 0 otherwise

4.14.2.72 int ktc_is_DeleteExpr (ktc_tree_t *t*)

Checks that root of subtree has 'DeleteExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DeleteExpr', 0 otherwise

4.14.2.73 int ktc_is_DenyThrowSpec (ktc_tree_t *t*)

Checks that root of subtree has 'DenyThrowSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DenyThrowSpec', 0 otherwise

4.14.2.74 int ktc_is_Designators (ktc_tree_t *t*)

Checks that root of subtree has 'Designators' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Designators', 0 otherwise

4.14.2.75 int ktc_is_DirectInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'DirectInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DirectInitializer', 0 otherwise

4.14.2.76 int ktc_is_DoDeclStmt (ktc_tree_t *t*)

Checks that root of subtree has 'DoDeclStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DoDeclStmt', 0 otherwise

4.14.2.77 int ktc_is_DoStmt (ktc_tree_t *t*)

Checks that root of subtree has 'DoStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'DoStmt', 0 otherwise

4.14.2.78 int ktc_is_Dtor (ktc_tree_t *t*)

Checks that root of subtree has 'Dtor' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Dtor', 0 otherwise

4.14.2.79 int ktc_is_Enumerator (ktc_tree_t *t*)

Checks that root of subtree has 'Enumerator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Enumerator', 0 otherwise

4.14.2.80 int ktc_is_Enumerators (ktc_tree_t *t*)

Checks that root of subtree has 'Enumerators' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Enumerators', 0 otherwise

4.14.2.81 int ktc_is_EnumType (ktc_tree_t *t*)

Checks that root of subtree has 'EnumType' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'EnumType', 0 otherwise

4.14.2.82 int ktc_is_ExceptHandler (ktc_tree_t *t*)

Checks that root of subtree has 'ExceptHandler' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ExceptHandler', 0 otherwise

4.14.2.83 int ktc_is_Exception (ktc_tree_t *t*)

Checks that root of subtree has 'Exception' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Exception', 0 otherwise

4.14.2.84 int ktc_is_ExceptionSpec (ktc_tree_t *t*)

Checks that root of subtree has 'ExceptionSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ExceptionSpec', 0 otherwise

4.14.2.85 int ktc_is_ExplicitInstantiation (ktc_tree_t *t*)

Checks that root of subtree has 'ExplicitInstantiation' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ExplicitInstantiation', 0 otherwise

4.14.2.86 int ktc_is_ExprArg (ktc_tree_t *t*)

Checks that root of subtree has 'ExprArg' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ExprArg', 0 otherwise

4.14.2.87 int ktc_is_Exprs (ktc_tree_t *t*)

Checks that root of subtree has 'Exprs' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Exprs', 0 otherwise

4.14.2.88 int ktc_is_ExprStmt (ktc_tree_t *t*)

Checks that root of subtree has 'ExprStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ExprStmt', 0 otherwise

4.14.2.89 int ktc_is_ExprTypeIdExpr (ktc_tree_t *t*)

Checks that root of subtree has 'ExprTypeIdExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ExprTypeIdExpr', 0 otherwise

4.14.2.90 int ktc_is_FieldDesignator (ktc_tree_t *t*)

Checks that root of subtree has 'FieldDesignator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'FieldDesignator', 0 otherwise

4.14.2.91 int ktc_is_FinallyHandler (ktc_tree_t *t*)

Checks that root of subtree has 'FinallyHandler' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'FinallyHandler', 0 otherwise

4.14.2.92 int ktc_is_ForEachStmt (ktc_tree_t *t*)

Checks that root of subtree has 'ForEachStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ForEachStmt', 0 otherwise

4.14.2.93 int ktc_is_ForRangeStmt (ktc_tree_t *t*)

Checks that root of subtree has 'ForRangeStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ForRangeStmt', 0 otherwise

4.14.2.94 int ktc_is_ForStmt (ktc_tree_t *t*)

Checks that root of subtree has 'ForStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ForStmt', 0 otherwise

4.14.2.95 int ktc_is_FuncBody (ktc_tree_t *t*)

Checks that root of subtree has 'FuncBody' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'FuncBody', 0 otherwise

4.14.2.96 int ktc_is_FuncDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'FuncDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'FuncDeclarator', 0 otherwise

4.14.2.97 int ktc_is_FuncDef (ktc_tree_t *t*)

Checks that root of subtree has 'FuncDef' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'FuncDef', 0 otherwise

4.14.2.98 int ktc_is_FuncSpec (ktc_tree_t *t*)

Checks that root of subtree has 'FuncSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'FuncSpec', 0 otherwise

4.14.2.99 int ktc_is_FuncTryBlock (ktc_tree_t *t*)

Checks that root of subtree has 'FuncTryBlock' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'FuncTryBlock', 0 otherwise

4.14.2.100 int ktc_is_GlobalScope (ktc_tree_t *t*)

Checks that root of subtree has 'GlobalScope' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'GlobalScope', 0 otherwise

4.14.2.101 int ktc_is_GotoStmt (ktc_tree_t *t*)

Checks that root of subtree has 'GotoStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'GotoStmt', 0 otherwise

4.14.2.102 int ktc_is_Handler (ktc_tree_t *t*)

Checks that root of subtree has 'Handler' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Handler', 0 otherwise

4.14.2.103 int ktc_is_Handlers (ktc_tree_t *t*)

Checks that root of subtree has 'Handlers' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Handlers', 0 otherwise

4.14.2.104 int ktc_is_IdExpr (ktc_tree_t *t*)

Checks that root of subtree has 'IdExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'IdExpr', 0 otherwise

4.14.2.105 int ktc_is_IfDeclStmt (ktc_tree_t *t*)

Checks that root of subtree has 'IfDeclStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'IfDeclStmt', 0 otherwise

4.14.2.106 int ktc_is_IfStmt (ktc_tree_t *t*)

Checks that root of subtree has 'IfStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'IfStmt', 0 otherwise

4.14.2.107 int ktc_is_IndexDesignator (ktc_tree_t *t*)

Checks that root of subtree has 'IndexDesignator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'IndexDesignator', 0 otherwise

4.14.2.108 int ktc_is_IndexExpr (ktc_tree_t *t*)

Checks that root of subtree has 'IndexExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'IndexExpr', 0 otherwise

4.14.2.109 int ktc_is_InitClause (ktc_tree_t *t*)

Checks that root of subtree has 'InitClause' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'InitClause', 0 otherwise

4.14.2.110 int ktc_is_InitializedDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'InitializedDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'InitializedDeclarator', 0 otherwise

4.14.2.111 int ktc_is_InitializerExpr (ktc_tree_t *t*)

Checks that root of subtree has 'InitializerExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'InitializerExpr', 0 otherwise

4.14.2.112 int ktc_is_Initializers (ktc_tree_t *t*)

Checks that root of subtree has 'Initializers' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Initializers', 0 otherwise

4.14.2.113 int ktc_is_KRFuncDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'KRFuncDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'KRFuncDeclarator', 0 otherwise

4.14.2.114 int ktc_is_Label (ktc_tree_t *t*)

Checks that root of subtree has 'Label' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Label', 0 otherwise

4.14.2.115 int ktc_is_LabeledStmt (ktc_tree_t *t*)

Checks that root of subtree has 'LabeledStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'LabeledStmt', 0 otherwise

4.14.2.116 int ktc_is_LambdaDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'LambdaDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'LambdaDeclarator', 0 otherwise

4.14.2.117 int ktc_is_LambdaExpr (ktc_tree_t *t*)

Checks that root of subtree has 'LambdaExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'LambdaExpr', 0 otherwise

4.14.2.118 int ktc_is_LambdaIntroducer (ktc_tree_t *t*)

Checks that root of subtree has 'LambdaIntroducer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'LambdaIntroducer', 0 otherwise

4.14.2.119 int ktc_is_LeaveStmt (ktc_tree_t *t*)

Checks that root of subtree has 'LeaveStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'LeaveStmt', 0 otherwise

4.14.2.120 int ktc_is_LinkageSpec (ktc_tree_t *t*)

Checks that root of subtree has 'LinkageSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'LinkageSpec', 0 otherwise

4.14.2.121 int ktc_is_LiteralExpr (ktc_tree_t *t*)

Checks that root of subtree has 'LiteralExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'LiteralExpr', 0 otherwise

4.14.2.122 int ktc_is_MaybeCtorInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'MaybeCtorInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MaybeCtorInitializer', 0 otherwise

4.14.2.123 int ktc_is_MaybeDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'MaybeDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MaybeDeclarator', 0 otherwise

4.14.2.124 int ktc_is_MaybeException (ktc_tree_t *t*)

Checks that root of subtree has 'MaybeException' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MaybeException', 0 otherwise

4.14.2.125 int ktc_is_MaybeExceptionSpec (ktc_tree_t *t*)

Checks that root of subtree has 'MaybeExceptionSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MaybeExceptionSpec', 0 otherwise

4.14.2.126 int ktc_is_MaybeLambdaDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'MaybeLambdaDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MaybeLambdaDeclarator', 0 otherwise

4.14.2.127 int ktc_is_MaybeNewInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'MaybeNewInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MaybeNewInitializer', 0 otherwise

4.14.2.128 int ktc_is_MaybeTypeId (ktc_tree_t *t*)

Checks that root of subtree has 'MaybeTypeId' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MaybeTypeId', 0 otherwise

4.14.2.129 int ktc_is_MemberDecl (ktc_tree_t *t*)

Checks that root of subtree has 'MemberDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberDecl', 0 otherwise

4.14.2.130 int ktc_is_MemberDecls (ktc_tree_t *t*)

Checks that root of subtree has 'MemberDecls' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberDecls', 0 otherwise

4.14.2.131 int ktc_is_MemberDesignator (ktc_tree_t *t*)

Checks that root of subtree has 'MemberDesignator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberDesignator', 0 otherwise

4.14.2.132 int ktc_is_MemberExpr (ktc_tree_t *t*)

Checks that root of subtree has 'MemberExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberExpr', 0 otherwise

4.14.2.133 int ktc_is_MemberFunc (ktc_tree_t *t*)

Checks that root of subtree has 'MemberFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberFunc', 0 otherwise

4.14.2.134 int ktc_is_MemberInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'MemberInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberInitializer', 0 otherwise

4.14.2.135 int ktc_is_MemberInitializers (ktc_tree_t *t*)

Checks that root of subtree has 'MemberInitializers' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberInitializers', 0 otherwise

4.14.2.136 int ktc_is_MemberTemplate (ktc_tree_t *t*)

Checks that root of subtree has 'MemberTemplate' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberTemplate', 0 otherwise

4.14.2.137 int ktc_is_MemberUsingDecl (ktc_tree_t *t*)

Checks that root of subtree has 'MemberUsingDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'MemberUsingDecl', 0 otherwise

4.14.2.138 int ktc_is_Name (ktc_tree_t *t*)

Checks that root of subtree has 'Name' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Name', 0 otherwise

4.14.2.139 int ktc_is_NameDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'NameDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NameDeclarator', 0 otherwise

4.14.2.140 int ktc_is_NamespaceAlias (ktc_tree_t *t*)

Checks that root of subtree has 'NamespaceAlias' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NamespaceAlias', 0 otherwise

4.14.2.141 int ktc_is_NamespaceDecl (ktc_tree_t *t*)

Checks that root of subtree has 'NamespaceDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NamespaceDecl', 0 otherwise

4.14.2.142 int ktc_is_NameSpec (ktc_tree_t *t*)

Checks that root of subtree has 'NameSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NameSpec', 0 otherwise

4.14.2.143 int ktc_is_NewExpr (ktc_tree_t *t*)

Checks that root of subtree has 'NewExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NewExpr', 0 otherwise

4.14.2.144 int ktc_is_NewInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'NewInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NewInitializer', 0 otherwise

4.14.2.145 int ktc_is_NoAttribute (ktc_tree_t *t*)

Checks that root of subtree has 'NoAttribute' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoAttribute', 0 otherwise

4.14.2.146 int ktc_is_NoAttributeSpec (ktc_tree_t *t*)

Checks that root of subtree has 'NoAttributeSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoAttributeSpec', 0 otherwise

4.14.2.147 int ktc_is_NoBaseSpec (ktc_tree_t *t*)

Checks that root of subtree has 'NoBaseSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoBaseSpec', 0 otherwise

4.14.2.148 int ktc_is_NoCapture (ktc_tree_t *t*)

Checks that root of subtree has 'NoCapture' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoCapture', 0 otherwise

4.14.2.149 int ktc_is_NoCtorInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'NoCtorInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoCtorInitializer', 0 otherwise

4.14.2.150 int ktc_is_Node (ktc_tree_t *t*)

Checks that root of subtree has 'Node' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Node', 0 otherwise

4.14.2.151 int ktc_is_NoDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'NoDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoDeclarator', 0 otherwise

4.14.2.152 int ktc_is_NoDeclOrStmt (ktc_tree_t *t*)

Checks that root of subtree has 'NoDeclOrStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoDeclOrStmt', 0 otherwise

4.14.2.153 int ktc_is_NoDeclSpec (ktc_tree_t *t*)

Checks that root of subtree has 'NoDeclSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoDeclSpec', 0 otherwise

4.14.2.154 int ktc_is_NoDesignator (ktc_tree_t *t*)

Checks that root of subtree has 'NoDesignator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoDesignator', 0 otherwise

4.14.2.155 int ktc_is_NoEnumerator (ktc_tree_t *t*)

Checks that root of subtree has 'NoEnumerator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoEnumerator', 0 otherwise

4.14.2.156 int ktc_is_NoException (ktc_tree_t *t*)

Checks that root of subtree has 'NoException' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoException', 0 otherwise

4.14.2.157 int ktc_is_NoExceptionSpec (ktc_tree_t *t*)

Checks that root of subtree has 'NoExceptionSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoExceptionSpec', 0 otherwise

4.14.2.158 int ktc_is_NoExpr (ktc_tree_t *t*)

Checks that root of subtree has 'NoExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoExpr', 0 otherwise

4.14.2.159 int ktc_is_NoHandler (ktc_tree_t *t*)

Checks that root of subtree has 'NoHandler' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoHandler', 0 otherwise

4.14.2.160 int ktc_is_NoInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'NoInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoInitializer', 0 otherwise

4.14.2.161 int ktc_is_NoLambdaDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'NoLambdaDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoLambdaDeclarator', 0 otherwise

4.14.2.162 int ktc_is_NoMemberDecl (ktc_tree_t *t*)

Checks that root of subtree has 'NoMemberDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoMemberDecl', 0 otherwise

4.14.2.163 int ktc_is_NoMemberInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'NoMemberInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoMemberInitializer', 0 otherwise

4.14.2.164 int ktc_is_NoName (ktc_tree_t *t*)

Checks that root of subtree has 'NoName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoName', 0 otherwise

4.14.2.165 int ktc_is_NoNameQualifier (ktc_tree_t *t*)

Checks that root of subtree has 'NoNameQualifier' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoNameQualifier', 0 otherwise

4.14.2.166 int ktc_is_NoNewInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'NoNewInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoNewInitializer', 0 otherwise

4.14.2.167 int ktc_is_NoParamName (ktc_tree_t *t*)

Checks that root of subtree has 'NoParamName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoParamName', 0 otherwise

4.14.2.168 int ktc_is_NoPropertyFunc (ktc_tree_t *t*)

Checks that root of subtree has 'NoPropertyFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoPropertyFunc', 0 otherwise

4.14.2.169 int ktc_is_NoTemplateArg (ktc_tree_t *t*)

Checks that root of subtree has 'NoTemplateArg' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoTemplateArg', 0 otherwise

4.14.2.170 int ktc_is_NoTemplateParam (ktc_tree_t *t*)

Checks that root of subtree has 'NoTemplateParam' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoTemplateParam', 0 otherwise

4.14.2.171 int ktc_is_NoTypeId (ktc_tree_t *t*)

Checks that root of subtree has 'NoTypeId' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NoTypeId', 0 otherwise

4.14.2.172 int ktc_is_NullptrLiteralExpr (ktc_tree_t *t*)

Checks that root of subtree has 'NullptrLiteralExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'NullptrLiteralExpr', 0 otherwise

4.14.2.173 int ktc_is_OpFunc (ktc_tree_t *t*)

Checks that root of subtree has 'OpFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'OpFunc', 0 otherwise

4.14.2.174 int ktc_is_Param (ktc_tree_t *t*)

Checks that root of subtree has 'Param' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'Param', 0 otherwise

4.14.2.175 int ktc_is_ParamName (ktc_tree_t *t*)

Checks that root of subtree has 'ParamName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ParamName', 0 otherwise

4.14.2.176 int ktc_is_ParamNames (ktc_tree_t *t*)

Checks that root of subtree has 'ParamNames' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ParamNames', 0 otherwise

4.14.2.177 int ktc_is_ParensDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'ParensDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ParensDeclarator', 0 otherwise

4.14.2.178 int ktc_is_ParensExpr (ktc_tree_t *t*)

Checks that root of subtree has 'ParensExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ParensExpr', 0 otherwise

4.14.2.179 int ktc_is_PromisedFuncBody (ktc_tree_t *t*)

Checks that root of subtree has 'PromisedFuncBody' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'PromisedFuncBody', 0 otherwise

4.14.2.180 int ktc_is_PromisedMemberDecl (ktc_tree_t *t*)

Checks that root of subtree has 'PromisedMemberDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'PromisedMemberDecl', 0 otherwise

4.14.2.181 int ktc_is_PropertyAttribute (ktc_tree_t *t*)

Checks that root of subtree has 'PropertyAttribute' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'PropertyAttribute', 0 otherwise

4.14.2.182 int ktc_is_PropertyFuncs (ktc_tree_t *t*)

Checks that root of subtree has 'PropertyFuncs' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'PropertyFuncs', 0 otherwise

4.14.2.183 int ktc_is_PropertyGetFunc (ktc_tree_t *t*)

Checks that root of subtree has 'PropertyGetFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'PropertyGetFunc', 0 otherwise

4.14.2.184 int ktc_is_PropertyPutFunc (ktc_tree_t *t*)

Checks that root of subtree has 'PropertyPutFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'PropertyPutFunc', 0 otherwise

4.14.2.185 int ktc_is_PseudoDtor (ktc_tree_t *t*)

Checks that root of subtree has 'PseudoDtor' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'PseudoDtor', 0 otherwise

4.14.2.186 int ktc_is_PtrDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'PtrDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'PtrDeclarator', 0 otherwise

4.14.2.187 int ktc_is_QualifiedName (ktc_tree_t *t*)

Checks that root of subtree has 'QualifiedName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'QualifiedName', 0 otherwise

4.14.2.188 int ktc_is_QualifiedPseudoDtor (ktc_tree_t *t*)

Checks that root of subtree has 'QualifiedPseudoDtor' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'QualifiedPseudoDtor', 0 otherwise

4.14.2.189 int ktc_is_RangeDesignator (ktc_tree_t *t*)

Checks that root of subtree has 'RangeDesignator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'RangeDesignator', 0 otherwise

4.14.2.190 int ktc_is_ReservedTypeSpec (ktc_tree_t *t*)

Checks that root of subtree has 'ReservedTypeSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ReservedTypeSpec', 0 otherwise

4.14.2.191 int ktc_is_ReturnStmt (ktc_tree_t *t*)

Checks that root of subtree has 'ReturnStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ReturnStmt', 0 otherwise

4.14.2.192 int ktc_is_SizeOfExpr (ktc_tree_t *t*)

Checks that root of subtree has 'SizeOfExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'SizeOfExpr', 0 otherwise

4.14.2.193 int ktc_is_SpecialCastExpr (ktc_tree_t *t*)

Checks that root of subtree has 'SpecialCastExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'SpecialCastExpr', 0 otherwise

4.14.2.194 int ktc_is_StaticAssertDecl (ktc_tree_t *t*)

Checks that root of subtree has 'StaticAssertDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'StaticAssertDecl', 0 otherwise

4.14.2.195 int ktc_is_StmtExpr (ktc_tree_t *t*)

Checks that root of subtree has 'StmtExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'StmtExpr', 0 otherwise

4.14.2.196 int ktc_is_StorageClass (ktc_tree_t *t*)

Checks that root of subtree has 'StorageClass' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'StorageClass', 0 otherwise

4.14.2.197 int ktc_is_StringLiteralExpr (ktc_tree_t *t*)

Checks that root of subtree has 'StringLiteralExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'StringLiteralExpr', 0 otherwise

4.14.2.198 int ktc_is_SuffixFunc (ktc_tree_t *t*)

Checks that root of subtree has 'SuffixFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'SuffixFunc', 0 otherwise

4.14.2.199 int ktc_is_SuperScope (ktc_tree_t *t*)

Checks that root of subtree has 'SuperScope' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'SuperScope', 0 otherwise

4.14.2.200 int ktc_is_SwitchDeclStmt (ktc_tree_t *t*)

Checks that root of subtree has 'SwitchDeclStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'SwitchDeclStmt', 0 otherwise

4.14.2.201 int ktc_is_SwitchStmt (ktc_tree_t *t*)

Checks that root of subtree has 'SwitchStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'SwitchStmt', 0 otherwise

4.14.2.202 int ktc_is_TemplateArgs (ktc_tree_t *t*)

Checks that root of subtree has 'TemplateArgs' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TemplateArgs', 0 otherwise

4.14.2.203 int ktc_is_TemplateDecl (ktc_tree_t *t*)

Checks that root of subtree has 'TemplateDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TemplateDecl', 0 otherwise

4.14.2.204 int ktc_is_TemplateName (ktc_tree_t *t*)

Checks that root of subtree has 'TemplateName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TemplateName', 0 otherwise

4.14.2.205 int ktc_is_TemplateParam (ktc_tree_t *t*)

Checks that root of subtree has 'TemplateParam' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TemplateParam', 0 otherwise

4.14.2.206 int ktc_is_TemplateParams (ktc_tree_t *t*)

Checks that root of subtree has 'TemplateParams' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TemplateParams', 0 otherwise

4.14.2.207 int ktc_is_TemplateSpec (ktc_tree_t *t*)

Checks that root of subtree has 'TemplateSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TemplateSpec', 0 otherwise

4.14.2.208 int ktc_is_TemplateTypeArg (ktc_tree_t *t*)

Checks that root of subtree has 'TemplateTypeArg' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TemplateTypeArg', 0 otherwise

4.14.2.209 int ktc_is_TemplateTypeParam (ktc_tree_t *t*)

Checks that root of subtree has 'TemplateTypeParam' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TemplateTypeParam', 0 otherwise

4.14.2.210 int ktc_is_ThisExpr (ktc_tree_t *t*)

Checks that root of subtree has 'ThisExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ThisExpr', 0 otherwise

4.14.2.211 int ktc_is_ThrowExpr (ktc_tree_t *t*)

Checks that root of subtree has 'ThrowExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'ThrowExpr', 0 otherwise

4.14.2.212 int ktc_is_TranslationUnit (ktc_tree_t *t*)

Checks that root of subtree has 'TranslationUnit' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TranslationUnit', 0 otherwise

4.14.2.213 int ktc_is_TruncatedInitClause (ktc_tree_t *t*)

Checks that root of subtree has 'TruncatedInitClause' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TruncatedInitClause', 0 otherwise

4.14.2.214 int ktc_is_TryExceptStmt (ktc_tree_t *t*)

Checks that root of subtree has 'TryExceptStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TryExceptStmt', 0 otherwise

4.14.2.215 int ktc_is_TryFinallyStmt (ktc_tree_t *t*)

Checks that root of subtree has 'TryFinallyStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TryFinallyStmt', 0 otherwise

4.14.2.216 int ktc_is_TryStmt (ktc_tree_t *t*)

Checks that root of subtree has 'TryStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TryStmt', 0 otherwise

4.14.2.217 int ktc_is_TypeAdjective (ktc_tree_t *t*)

Checks that root of subtree has 'TypeAdjective' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeAdjective', 0 otherwise

4.14.2.218 int ktc_is_TypeArg (ktc_tree_t *t*)

Checks that root of subtree has 'TypeArg' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeArg', 0 otherwise

4.14.2.219 int ktc_is_TypeConvExpr (ktc_tree_t *t*)

Checks that root of subtree has 'TypeConvExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeConvExpr', 0 otherwise

4.14.2.220 int ktc_is_TypeId (ktc_tree_t *t*)

Checks that root of subtree has 'TypeId' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeId', 0 otherwise

4.14.2.221 int ktc_is_TypeName (ktc_tree_t *t*)

Checks that root of subtree has 'TypeName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeName', 0 otherwise

4.14.2.222 int ktc_is_TypeOfExpr (ktc_tree_t *t*)

Checks that root of subtree has 'TypeOfExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeOfExpr', 0 otherwise

4.14.2.223 int ktc_is_TypeOfSpec (ktc_tree_t *t*)

Checks that root of subtree has 'TypeOfSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeOfSpec', 0 otherwise

4.14.2.224 int ktc_is_TypeOfType (ktc_tree_t *t*)

Checks that root of subtree has 'TypeOfType' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeOfType', 0 otherwise

4.14.2.225 int ktc_is_TypeParam (ktc_tree_t *t*)

Checks that root of subtree has 'TypeParam' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeParam', 0 otherwise

4.14.2.226 int ktc_is_TypeTypeIdExpr (ktc_tree_t *t*)

Checks that root of subtree has 'TypeTypeIdExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'TypeTypeIdExpr', 0 otherwise

4.14.2.227 int ktc_is_UnaryExpr (ktc_tree_t *t*)

Checks that root of subtree has 'UnaryExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnaryExpr', 0 otherwise

4.14.2.228 int ktc_is_UnparsedDecl (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedDecl', 0 otherwise

4.14.2.229 int ktc_is_UnparsedDeclarator (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedDeclarator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedDeclarator', 0 otherwise

4.14.2.230 int ktc_is_UnparsedDeclSpec (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedDeclSpec' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedDeclSpec', 0 otherwise

4.14.2.231 int ktc_is_UnparsedEnumerator (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedEnumerator' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedEnumerator', 0 otherwise

4.14.2.232 int ktc_is_UnparsedException (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedException' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedException', 0 otherwise

4.14.2.233 int ktc_is_UnparsedExpr (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedExpr', 0 otherwise

4.14.2.234 int ktc_is_UnparsedInitializer (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedInitializer' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedInitializer', 0 otherwise

4.14.2.235 int ktc_is_UnparsedLabel (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedLabel' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedLabel', 0 otherwise

4.14.2.236 int ktc_is_UnparsedMemberDecl (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedMemberDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedMemberDecl', 0 otherwise

4.14.2.237 int ktc_is_UnparsedName (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedName', 0 otherwise

4.14.2.238 int ktc_is_UnparsedNameQualifier (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedNameQualifier' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedNameQualifier', 0 otherwise

4.14.2.239 int ktc_is_UnparsedParamName (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedParamName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedParamName', 0 otherwise

4.14.2.240 int ktc_is_UnparsedPropertyFunc (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedPropertyFunc' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedPropertyFunc', 0 otherwise

4.14.2.241 int ktc_is_UnparsedStmt (ktc_tree_t *t*)

Checks that root of subtree has 'UnparsedStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnparsedStmt', 0 otherwise

4.14.2.242 int ktc_is_UnqualifiedName (ktc_tree_t *t*)

Checks that root of subtree has 'UnqualifiedName' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UnqualifiedName', 0 otherwise

4.14.2.243 int ktc_is_UserLiteralExpr (ktc_tree_t *t*)

Checks that root of subtree has 'UserLiteralExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UserLiteralExpr', 0 otherwise

4.14.2.244 int ktc_is_UserStringLiteralExpr (ktc_tree_t *t*)

Checks that root of subtree has 'UserStringLiteralExpr' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UserStringLiteralExpr', 0 otherwise

4.14.2.245 int ktc_is_UsingDecl (ktc_tree_t *t*)

Checks that root of subtree has 'UsingDecl' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UsingDecl', 0 otherwise

4.14.2.246 int ktc_is_UsingDirective (ktc_tree_t *t*)

Checks that root of subtree has 'UsingDirective' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'UsingDirective', 0 otherwise

4.14.2.247 int ktc_is_WhileDeclStmt (ktc_tree_t *t*)

Checks that root of subtree has 'WhileDeclStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'WhileDeclStmt', 0 otherwise

4.14.2.248 int ktc_is_WhileStmt (ktc_tree_t *t*)

Checks that root of subtree has 'WhileStmt' AST node type.

Parameters

t	subtree root to test

Returns

1 if 't' belongs to the type 'WhileStmt', 0 otherwise

4.15 Child link identifiers

Variables

- const **ktc_childId_t cid_Next**
- const **ktc_childId_t cid_Decls**
- const **ktc_childId_t cid_Expr**
- const **ktc_childId_t cid_Stmt**
- const **ktc_childId_t cid_LambdaCapture**
- const **ktc_childId_t cid_CtorInit**
- const **ktc_childId_t cid_Handlers**
- const **ktc_childId_t cid_Exception**
- const **ktc_childId_t cid_DeclSpecs**
- const **ktc_childId_t cid_Declarator**
- const **ktc_childId_t cid_Attributes**
- const **ktc_childId_t cid_Designators**
- const **ktc_childId_t cid_Lower**
- const **ktc_childId_t cid_Upper**
- const **ktc_childId_t cid_Qualifier**
- const **ktc_childId_t cid_Name**
- const **ktc_childId_t cid_Params**
- const **ktc_childId_t cid_Throw**
- const **ktc_childId_t cid_TrailingReturnType**
- const **ktc_childId_t cid_Args**
- const **ktc_childId_t cid_MemberInitializers**
- const **ktc_childId_t cid_TypeIds**
- const **ktc_childId_t cid_Decl**
- const **ktc_childId_t cid_Type**
- const **ktc_childId_t cid_AttributeSpec**
- const **ktc_childId_t cid_PropertyFuncs**
- const **ktc_childId_t cid_Declarators**
- const **ktc_childId_t cid_FuncBody**
- const **ktc_childId_t cid_TemplateParams**
- const **ktc_childId_t cid_MemberDecl**
- const **ktc_childId_t cid_NameSpec**
- const **ktc_childId_t cid_CVQualifiers**
- const **ktc_childId_t cid_Bits**
- const **ktc_childId_t cid_Init**
- const **ktc_childId_t cid_Inits**
- const **ktc_childId_t cid_Index**
- const **ktc_childId_t cid_Adjacent**
- const **ktc_childId_t cid_Literal**
- const **ktc_childId_t cid_Func**
- const **ktc_childId_t cid_Left**
- const **ktc_childId_t cid_Right**
- const **ktc_childId_t cid_Cond**
- const **ktc_childId_t cid_Then**
- const **ktc_childId_t cid_Else**
- const **ktc_childId_t cid_Placement**
- const **ktc_childId_t cid_Initializer**
- const **ktc_childId_t cid_Introducer**
- const **ktc_childId_t cid_TemplateName**
- const **ktc_childId_t cid_KRParams**
- const **ktc_childId_t cid_Label**

- const **ktc_childId_t cid_Stmts**
- const **ktc_childId_t cid_Handler**
- const **ktc_childId_t cid_Default**
- const **ktc_childId_t cid_Base**
- const **ktc_childId_t cid_Enumerators**
- const **ktc_childId_t cid_AttributeSpecs**
- const **ktc_childId_t cid_BaseSpecs**
- const **ktc_childId_t cid_MemberDecls**
- const **ktc_childId_t cid_Size**
- const **ktc_childId_t cid_ConversionType**

4.15.1 Detailed Description

4.15.2 Variable Documentation

4.15.2.1 const ktc_childId_t cid_Adjacent

Constant to identify 'Adjacent' child of AST nodes

4.15.2.2 const ktc_childId_t cid_Args

Constant to identify 'Args' child of AST nodes

4.15.2.3 const ktc_childId_t cid_Attributes

Constant to identify 'Attributes' child of AST nodes

4.15.2.4 const ktc_childId_t cid_AttributeSpec

Constant to identify 'AttributeSpec' child of AST nodes

4.15.2.5 const ktc_childId_t cid_AttributeSpecs

Constant to identify 'AttributeSpecs' child of AST nodes

4.15.2.6 const ktc_childId_t cid_Base

Constant to identify 'Base' child of AST nodes

4.15.2.7 const ktc_childId_t cid_BaseSpecs

Constant to identify 'BaseSpecs' child of AST nodes

4.15.2.8 const ktc_childId_t cid_Bits

Constant to identify 'Bits' child of AST nodes

4.15.2.9 const ktc_childId_t cid_Cond

Constant to identify 'Cond' child of AST nodes

4.15.2.10 const ktc_childId_t cid_ConversionType

Constant to identify 'ConversionType' child of AST nodes

4.15.2.11 const ktc_childId_t cid_CtorInit

Constant to identify 'CtorInit' child of AST nodes

4.15.2.12 const ktc_childId_t cid_CVQualifiers

Constant to identify 'CVQualifiers' child of AST nodes

4.15.2.13 const ktc_childId_t cid_Decl

Constant to identify 'Decl' child of AST nodes

4.15.2.14 const ktc_childId_t cid_Declarator

Constant to identify 'Declarator' child of AST nodes

4.15.2.15 const ktc_childId_t cid_Declarators

Constant to identify 'Declarators' child of AST nodes

4.15.2.16 const ktc_childId_t cid_Decls

Constant to identify 'Decls' child of AST nodes

4.15.2.17 const ktc_childId_t cid_DeclSpecs

Constant to identify 'DeclSpecs' child of AST nodes

4.15.2.18 const ktc_childId_t cid_Default

Constant to identify 'Default' child of AST nodes

4.15.2.19 const ktc_childId_t cid_Designators

Constant to identify 'Designators' child of AST nodes

4.15.2.20 const ktc_childId_t cid_Else

Constant to identify 'Else' child of AST nodes

4.15.2.21 const ktc_childId_t cid_Enumerators

Constant to identify 'Enumerators' child of AST nodes

4.15.2.22 const ktc_childId_t cid_Exception

Constant to identify 'Exception' child of AST nodes

4.15.2.23 const ktc_childId_t cid_Expr

Constant to identify 'Expr' child of AST nodes

4.15.2.24 const ktc_childId_t cid_Func

Constant to identify 'Func' child of AST nodes

4.15.2.25 const ktc_childId_t cid_FuncBody

Constant to identify 'FuncBody' child of AST nodes

4.15.2.26 const ktc_childId_t cid_Handler

Constant to identify 'Handler' child of AST nodes

4.15.2.27 const ktc_childId_t cid_Handlers

Constant to identify 'Handlers' child of AST nodes

4.15.2.28 const ktc_childId_t cid_Index

Constant to identify 'Index' child of AST nodes

4.15.2.29 const ktc_childId_t cid_Init

Constant to identify 'Init' child of AST nodes

4.15.2.30 const ktc_childId_t cid_Initializer

Constant to identify 'Initializer' child of AST nodes

4.15.2.31 const ktc_childId_t cid_Inits

Constant to identify 'Inits' child of AST nodes

4.15.2.32 const ktc_childId_t cid_Introducer

Constant to identify 'Introducer' child of AST nodes

4.15.2.33 const ktc_childId_t cid_KRParams

Constant to identify 'KRParams' child of AST nodes

4.15.2.34 const ktc_childId_t cid_Label

Constant to identify 'Label' child of AST nodes

4.15.2.35 const ktc_childId_t cid_LambdaCapture

Constant to identify 'LambdaCapture' child of AST nodes

4.15.2.36 const ktc_childId_t cid_Left

Constant to identify 'Left' child of AST nodes

4.15.2.37 const ktc_childId_t cid_Literal

Constant to identify 'Literal' child of AST nodes

4.15.2.38 const ktc_childId_t cid_Lower

Constant to identify 'Lower' child of AST nodes

4.15.2.39 const ktc_childId_t cid_MemberDecl

Constant to identify 'MemberDecl' child of AST nodes

4.15.2.40 const ktc_childId_t cid_MemberDecls

Constant to identify 'MemberDecls' child of AST nodes

4.15.2.41 const ktc_childId_t cid_MemberInitializers

Constant to identify 'MemberInitializers' child of AST nodes

4.15.2.42 const ktc_childId_t cid_Name

Constant to identify 'Name' child of AST nodes

4.15.2.43 const ktc_childId_t cid_NameSpec

Constant to identify 'NameSpec' child of AST nodes

4.15.2.44 const ktc_childId_t cid_Next

Constant to identify 'Next' child of AST nodes

4.15.2.45 const ktc_childId_t cid_Params

Constant to identify 'Params' child of AST nodes

4.15.2.46 const ktc_childId_t cid_Placement

Constant to identify 'Placement' child of AST nodes

4.15.2.47 const ktc_childId_t cid_PropertyFuncs

Constant to identify 'PropertyFuncs' child of AST nodes

4.15.2.48 const ktc_childId_t cid_Qualifier

Constant to identify 'Qualifier' child of AST nodes

4.15.2.49 const ktc_childId_t cid_Right

Constant to identify 'Right' child of AST nodes

4.15.2.50 const ktc_childId_t cid_Size

Constant to identify 'Size' child of AST nodes

4.15.2.51 const ktc_childId_t cid_Stmt

Constant to identify 'Stmt' child of AST nodes

4.15.2.52 const ktc_childId_t cid_Stmts

Constant to identify 'Stmts' child of AST nodes

4.15.2.53 const ktc_childId_t cid_TemplateName

Constant to identify 'TemplateName' child of AST nodes

4.15.2.54 const ktc_childId_t cid_TemplateParams

Constant to identify 'TemplateParams' child of AST nodes

4.15.2.55 const ktc_childId_t cid_Then

Constant to identify 'Then' child of AST nodes

4.15.2.56 const ktc_childId_t cid_Throw

Constant to identify 'Throw' child of AST nodes

4.15.2.57 const ktc_childId_t cid_TrailingReturnType

Constant to identify 'TrailingReturnType' child of AST nodes

4.15.2.58 const ktc_childId_t cid_Type

Constant to identify 'Type' child of AST nodes

4.15.2.59 const ktc_childId_t cid_TypeIds

Constant to identify 'TypeIds' child of AST nodes

4.15.2.60 const ktc_childId_t cid_Upper

Constant to identify 'Upper' child of AST nodes

4.16 Numerical codes of operations

Variables

- const int **KTC_OPCODE_NONE**
- const int **KTC_OPCODE_ASSIGN**
- const int **KTC_OPCODE_MULASSIGN**
- const int **KTC_OPCODE_DIVASSIGN**
- const int **KTC_OPCODE_MODASSIGN**
- const int **KTC_OPCODE_ADDASSIGN**
- const int **KTC_OPCODE_SUBASSIGN**
- const int **KTC_OPCODE_ASLASSIGN**
- const int **KTC_OPCODE_ASRASSIGN**
- const int **KTC_OPCODE_ANDASSIGN**
- const int **KTC_OPCODE_XORASSIGN**
- const int **KTC_OPCODE_ORASSIGN**
- const int **KTC_OPCODE_COMMA**
- const int **KTC_OPCODE_COND**
- const int **KTC_OPCODE_LOGOR**
- const int **KTC_OPCODE_LOGAND**
- const int **KTC_OPCODE_BITOR**
- const int **KTC_OPCODE_BITXOR**
- const int **KTC_OPCODE_BITAND**
- const int **KTC_OPCODE_EQ**
- const int **KTC_OPCODE_NE**
- const int **KTC_OPCODE_LT**
- const int **KTC_OPCODE_GT**
- const int **KTC_OPCODE_LE**
- const int **KTC_OPCODE_GE**
- const int **KTC_OPCODE_ASR**
- const int **KTC_OPCODE_ASL**
- const int **KTC_OPCODE_ADD**
- const int **KTC_OPCODE_SUB**
- const int **KTC_OPCODE_MUL**
- const int **KTC_OPCODE_DIV**
- const int **KTC_OPCODE_MOD**
- const int **KTC_OPCODE_PREINC**
- const int **KTC_OPCODE_PREDEC**
- const int **KTC_OPCODE_SIZEOF**
- const int **KTC_OPCODE_DEREF**
- const int **KTC_OPCODE_ADDRESS**
- const int **KTC_OPCODE_PLUS**
- const int **KTC_OPCODE_MINUS**
- const int **KTC_OPCODE_BITNOT**
- const int **KTC_OPCODE_LOGNOT**
- const int **KTC_OPCODE_POSTINC**
- const int **KTC_OPCODE_POSTDEC**
- const int **KTC_OPCODE_FIELD**
- const int **KTC_OPCODE_FIELDREF**
- const int **KTC_OPCODE_DOTAST**
- const int **KTC_OPCODE_DEREFAST**
- const int **KTC_OPCODE_ROUND_BRACKETS**
- const int **KTC_OPCODE_SQUARE_BRACKETS**
- const int **KTC_OPCODE_THROW**
- const int **KTC_OPCODE_MIN**
- const int **KTC_OPCODE_MAX**

4.16.1 Detailed Description

4.16.2 Variable Documentation

4.16.2.1 const int KTC_OPCODE_ADD

4.16.2.2 const int KTC_OPCODE_ADDASSIGN

4.16.2.3 const int KTC_OPCODE_ADDRESS

4.16.2.4 const int KTC_OPCODE_ANDASSIGN

4.16.2.5 const int KTC_OPCODE_ASL

4.16.2.6 const int KTC_OPCODE_ASLASSIGN

4.16.2.7 const int KTC_OPCODE_ASR

4.16.2.8 const int KTC_OPCODE_ASRASSIGN

4.16.2.9 const int KTC_OPCODE_ASSIGN

4.16.2.10 const int KTC_OPCODE_BITAND

4.16.2.11 const int KTC_OPCODE_BITNOT

4.16.2.12 const int KTC_OPCODE_BITOR

4.16.2.13 const int KTC_OPCODE_BITXOR

4.16.2.14 const int KTC_OPCODE_COMMA

4.16.2.15 const int KTC_OPCODE_COND

4.16.2.16 const int KTC_OPCODE_DEREF

4.16.2.17 const int KTC_OPCODE_DEREFAST

4.16.2.18 const int KTC_OPCODE_DIV

4.16.2.19 const int KTC_OPCODE_DIVASSIGN

4.16.2.20 const int KTC_OPCODE_DOTAST

4.16.2.21 const int KTC_OPCODE_EQ

4.16.2.22 const int KTC_OPCODE_FIELD

4.16.2.23 const int KTC_OPCODE_FIELDREF

4.16.2.24 const int KTC_OPCODE_GE

4.16.2.25 const int KTC_OPCODE_GT

4.16.2.26 const int KTC_OPCODE_LE

4.16.2.27 const int KTC_OPCODE_LOGAND

4.16.2.28 const int KTC_OPCODE_LOGNOT

4.16.2.29 const int KTC_OPCODE_LOGOR

4.16.2.30 const int KTC_OPCODE_LT

4.16.2.31 const int KTC_OPCODE_MAX

4.16.2.32 const int KTC_OPCODE_MIN

4.16.2.33 const int KTC_OPCODE_MINUS

4.16.2.34 const int KTC_OPCODE_MOD

4.16.2.35 const int KTC_OPCODE_MODASSIGN

4.16.2.36 const int KTC_OPCODE_MUL

4.16.2.37 const int KTC_OPCODE_MULASSIGN

4.16.2.38 const int KTC_OPCODE_NE

4.16.2.39 const int KTC_OPCODE_NONE

4.16.2.40 const int KTC_OPCODE_ORASSIGN

4.16.2.41 const int KTC_OPCODE_PLUS

4.16.2.42 const int KTC_OPCODE_POSTDEC

4.16.2.43 const int KTC_OPCODE_POSTINC

4.16.2.44 const int KTC_OPCODE_PREDEC

4.16.2.45 const int KTC_OPCODE_PREINC

4.16.2.46 const int KTC_OPCODE_ROUND_BRACKETS

4.16.2.47 const int KTC_OPCODE_SIZEOF

4.16.2.48 const int KTC_OPCODE_SQUARE_BRACKETS

4.16.2.49 const int KTC_OPCODE_SUB

4.16.2.50 const int KTC_OPCODE_SUBASSIGN

4.16.2.51 const int KTC_OPCODE_THROW

4.16.2.52 const int KTC_OPCODE_XORASSIGN

4.17 Numerical codes of declaration storage class specifiers

Variables

- const int **KTC_STORAGECLASS_NONE**
- const int **KTC_STORAGECLASS_TYPEDEF**
- const int **KTC_STORAGECLASS_EXTERN**
- const int **KTC_STORAGECLASS_STATIC**
- const int **KTC_STORAGECLASS_AUTO**
- const int **KTC_STORAGECLASS_REGISTER**
- const int **KTC_STORAGECLASS_MUTABLE**
- const int **KTC_STORAGECLASS_THREADLOCAL**

4.17.1 Detailed Description

4.17.2 Variable Documentation

4.17.2.1 const int KTC_STORAGECLASS_AUTO

4.17.2.2 const int KTC_STORAGECLASS_EXTERN

4.17.2.3 const int KTC_STORAGECLASS_MUTABLE

4.17.2.4 const int KTC_STORAGECLASS_NONE

4.17.2.5 const int KTC_STORAGECLASS_REGISTER

4.17.2.6 const int KTC_STORAGECLASS_STATIC

4.17.2.7 const int KTC_STORAGECLASS_THREADLOCAL

4.17.2.8 const int KTC_STORAGECLASS_TYPEDEF

4.18 Numerical codes of declaration type qualifiers (no qualifier, 'const', 'volatile' or 'restrict'). Actual values are bit-or'ed superpositions of these flags. Use '==' check for KTC_CVQUALIFIER_NONE and '&' for two other values

Variables

- const int **KTC_CVQUALIFIER_NONE**
- const int **KTC_CVQUALIFIER_CONST**
- const int **KTC_CVQUALIFIER_VOLATILE**
- const int **KTC_CVQUALIFIER_RESTRICT**

4.18.1 Detailed Description

4.18.2 Variable Documentation

4.18.2.1 const int KTC_CVQUALIFIER_CONST

4.18.2.2 const int KTC_CVQUALIFIER_NONE

4.18.2.3 const int KTC_CVQUALIFIER_RESTRICT

4.18.2.4 const int KTC_CVQUALIFIER_VOLATILE

4.19 Numerical codes to differentiate struct/class/union declarations

Variables

- const int **KTC_CLASSTAG_NONE**
- const int **KTC_CLASSTAG_STRUCT**
- const int **KTC_CLASSTAG_UNION**
- const int **KTC_CLASSTAG_CLASS**

4.19.1 Detailed Description

4.19.2 Variable Documentation

4.19.2.1 const int KTC_CLASSTAG_CLASS

4.19.2.2 const int KTC_CLASSTAG_NONE

4.19.2.3 const int KTC_CLASSTAG_STRUCT

4.19.2.4 const int KTC_CLASSTAG_UNION

4.20 Numerical codes of C/C++ built-in types

Variables

- const int **KTC_BUILTINTYPE_NONE**
- const int **KTC_BUILTINTYPE_VOID**
- const int **KTC_BUILTINTYPE_BOOL**
- const int **KTC_BUILTINTYPE_WCHAR_T**
- const int **KTC_BUILTINTYPE_CHAR**
- const int **KTC_BUILTINTYPE_SIGNEDCHAR**
- const int **KTC_BUILTINTYPE_UNSIGNEDCHAR**
- const int **KTC_BUILTINTYPE_SHORTINT**
- const int **KTC_BUILTINTYPE_SIGNEDSHORTINT**
- const int **KTC_BUILTINTYPE_UNSIGNEDSHORTINT**
- const int **KTC_BUILTINTYPE_INT**
- const int **KTC_BUILTINTYPE_SIGNEDINT**
- const int **KTC_BUILTINTYPE_UNSIGNEDINT**
- const int **KTC_BUILTINTYPE_LONGINT**
- const int **KTC_BUILTINTYPE_SIGNEDLONGINT**
- const int **KTC_BUILTINTYPE_UNSIGNEDLONGINT**
- const int **KTC_BUILTINTYPE_LONGLONGINT**
- const int **KTC_BUILTINTYPE_SIGNEDLONGLONGINT**
- const int **KTC_BUILTINTYPE_UNSIGNEDLONGLONGINT**
- const int **KTC_BUILTINTYPE_FLOAT**
- const int **KTC_BUILTINTYPE_DOUBLE**
- const int **KTC_BUILTINTYPE_LONGDOUBLE**

4.20.1 Detailed Description

4.20.2 Variable Documentation

4.20.2.1 const int KTC_BUILTINTYPE_BOOL

4.20.2.2 const int KTC_BUILTINTYPE_CHAR

4.20.2.3 const int KTC_BUILTINTYPE_DOUBLE

4.20.2.4 const int KTC_BUILTINTYPE_FLOAT

4.20.2.5 const int KTC_BUILTINTYPE_INT

4.20.2.6 const int KTC_BUILTINTYPE_LONGDOUBLE

4.20.2.7 const int KTC_BUILTINTYPE_LONGINT

4.20.2.8 const int KTC_BUILTINTYPE_LONGLONGINT

4.20.2.9 const int KTC_BUILTINTYPE_NONE

4.20.2.10 const int KTC_BUILTINTYPE_SHORTINT

4.20.2.11 const int KTC_BUILTINTYPE_SIGNEDCHAR

4.20.2.12 const int KTC_BUILTINTYPE_SIGNEDINT

4.20.2.13 const int KTC_BUILTINTYPE_SIGNEDLONGINT

4.20.2.14 const int KTC_BUILTINTYPE_SIGNEDLONGLONGINT

4.20.2.15 const int KTC_BUILTINTYPE_SIGNEDSHORTINT

4.20.2.16 const int KTC_BUILTINTYPE_UNSIGNEDCHAR

4.20.2.17 const int KTC_BUILTINTYPE_UNSIGNEDINT

4.20.2.18 const int KTC_BUILTINTYPE_UNSIGNEDLONGINT

4.20.2.19 const int KTC_BUILTINTYPE_UNSIGNEDLONGLONGINT

4.20.2.20 const int KTC_BUILTINTYPE_UNSIGNEDSHORTINT

4.20.2.21 const int KTC_BUILTINTYPE_VOID

4.20.2.22 const int KTC_BUILTINTYPE_WCHAR_T

4.21 Numerical codes for identifying inline/virtual/explicit/friend member function specifiers

Variables

- const int **KTC_FUNCSPECIFIER_NONE**
- const int **KTC_FUNCSPECIFIER_INLINE**
- const int **KTC_FUNCSPECIFIER_VIRTUAL**
- const int **KTC_FUNCSPECIFIER_EXPLICIT**
- const int **KTC_FUNCSPECIFIER_FRIEND**

4.21.1 Detailed Description

4.21.2 Variable Documentation

4.21.2.1 const int KTC_FUNCSPECIFIER_EXPLICIT

4.21.2.2 const int KTC_FUNCSPECIFIER_FRIEND

4.21.2.3 const int KTC_FUNCSPECIFIER_INLINE

4.21.2.4 const int KTC_FUNCSPECIFIER_NONE

4.21.2.5 const int KTC_FUNCSPECIFIER_VIRTUAL

4.22 Numerical codes for identifying different C++ -style cast expressions

Variables

- const int **KTC_CASTSPECIFIER_DYNAMIC**
- const int **KTC_CASTSPECIFIER_STATIC**
- const int **KTC_CASTSPECIFIER_REINTERPRET**
- const int **KTC_CASTSPECIFIER_CONST**

4.22.1 Detailed Description

4.22.2 Variable Documentation

4.22.2.1 const int KTC_CASTSPECIFIER_CONST

4.22.2.2 const int KTC_CASTSPECIFIER_DYNAMIC

4.22.2.3 const int KTC_CASTSPECIFIER_REINTERPRET

4.22.2.4 const int KTC_CASTSPECIFIER_STATIC

4.23 Numerical codes for identifying declarator ptr types

Variables

- const int **KTC_POINTEROPERATOR_NONE**
- const int **KTC_POINTEROPERATOR_POINTER**
- const int **KTC_POINTEROPERATOR_REFERENCE**
- const int **KTC_POINTEROPERATOR_RVALUE**

4.23.1 Detailed Description

4.23.2 Variable Documentation

4.23.2.1 const int KTC_POINTEROPERATOR_NONE

4.23.2.2 const int KTC_POINTEROPERATOR_POINTER

4.23.2.3 const int KTC_POINTEROPERATOR_REFERENCE

4.23.2.4 const int KTC_POINTEROPERATOR_RVALUE

Chapter 5

File Documentation

5.1 gen-ktcAPI.h File Reference

```
#include "kwapi.h"
#include "ktcMainAPI.h"
```

Functions

- int **ktc_is_Node** (**ktc_tree_t** t)
- int **ktc_is_TranslationUnit** (**ktc_tree_t** t)
- int **ktc_is_DeclOrStmts** (**ktc_tree_t** t)
- int **ktc_is_AnyFuncBody** (**ktc_tree_t** t)
- int **ktc_is_Handlers** (**ktc_tree_t** t)
- int **ktc_is_ExceptHandler** (**ktc_tree_t** t)
- int **ktc_is_FinallyHandler** (**ktc_tree_t** t)
- int **ktc_is_MaybeException** (**ktc_tree_t** t)
- int **ktc_is_TemplateParams** (**ktc_tree_t** t)
- int **ktc_is_DeclSpecs** (**ktc_tree_t** t)
- int **ktc_is_AttributeSpecs** (**ktc_tree_t** t)
- int **ktc_is_Attributes** (**ktc_tree_t** t)
- int **ktc_is_PropertyFuncs** (**ktc_tree_t** t)
- int **ktc_is_MemberDecls** (**ktc_tree_t** t)
- int **ktc_is_Enumerators** (**ktc_tree_t** t)
- int **ktc_is_MaybeDeclarator** (**ktc_tree_t** t)
- int **ktc_is_ParamNames** (**ktc_tree_t** t)
- int **ktc_is_Initializers** (**ktc_tree_t** t)
- int **ktc_is_Designators** (**ktc_tree_t** t)
- int **ktc_is_AnyLabel** (**ktc_tree_t** t)
- int **ktc_is_Exprs** (**ktc_tree_t** t)
- int **ktc_is_AnyNameQualifier** (**ktc_tree_t** t)
- int **ktc_is_AnyNames** (**ktc_tree_t** t)
- int **ktc_is_TemplateArgs** (**ktc_tree_t** t)
- int **ktc_is_BaseSpecs** (**ktc_tree_t** t)
- int **ktc_is_MaybeLambdaDeclarator** (**ktc_tree_t** t)
- int **ktc_is_LambdaIntroducer** (**ktc_tree_t** t)
- int **ktc_is_AnyCapture** (**ktc_tree_t** t)

- int **ktc_is_MaybeTypeId** (**ktc_tree_t** t)
- int **ktc_is_MaybeNewInitializer** (**ktc_tree_t** t)
- int **ktc_is_MemberInitializers** (**ktc_tree_t** t)
- int **ktc_is_MaybeCtorInitializer** (**ktc_tree_t** t)
- int **ktc_is_MaybeExceptionSpec** (**ktc_tree_t** t)
- int **ktc_is_NoDeclOrStmt** (**ktc_tree_t** t)
- int **ktc_is_DeclEllipsis** (**ktc_tree_t** t)
- int **ktc_is_DeclOrStmt** (**ktc_tree_t** t)
- int **ktc_is_FuncBody** (**ktc_tree_t** t)
- int **ktc_is_FuncTryBlock** (**ktc_tree_t** t)
- int **ktc_is_PromisedFuncBody** (**ktc_tree_t** t)
- int **ktc_is_NoHandler** (**ktc_tree_t** t)
- int **ktc_is_Handler** (**ktc_tree_t** t)
- int **ktc_is_NoException** (**ktc_tree_t** t)
- int **ktc_is_Exception** (**ktc_tree_t** t)
- int **ktc_is_DefaultException** (**ktc_tree_t** t)
- int **ktc_is_UnparsedException** (**ktc_tree_t** t)
- int **ktc_is_NoTemplateParam** (**ktc_tree_t** t)
- int **ktc_is_TemplateParam** (**ktc_tree_t** t)
- int **ktc_is_NoDeclSpec** (**ktc_tree_t** t)
- int **ktc_is_DeclSpec** (**ktc_tree_t** t)
- int **ktc_is_NoAttributeSpec** (**ktc_tree_t** t)
- int **ktc_is_AttributeSpec** (**ktc_tree_t** t)
- int **ktc_is_NoAttribute** (**ktc_tree_t** t)
- int **ktc_is_AnyAttribute** (**ktc_tree_t** t)
- int **ktc_is_NoPropertyFunc** (**ktc_tree_t** t)
- int **ktc_is_AnyPropertyFunc** (**ktc_tree_t** t)
- int **ktc_is_NoMemberDecl** (**ktc_tree_t** t)
- int **ktc_is_AnyMemberDecl** (**ktc_tree_t** t)
- int **ktc_is_NoEnumerator** (**ktc_tree_t** t)
- int **ktc_is_AnyEnumerator** (**ktc_tree_t** t)
- int **ktc_is_NoDeclarator** (**ktc_tree_t** t)
- int **ktc_is_AnyDeclarator** (**ktc_tree_t** t)
- int **ktc_is_NoParamName** (**ktc_tree_t** t)
- int **ktc_is_AnyParamName** (**ktc_tree_t** t)
- int **ktc_is_NoInitializer** (**ktc_tree_t** t)
- int **ktc_is_AnyInitializer** (**ktc_tree_t** t)
- int **ktc_is_NoDesignator** (**ktc_tree_t** t)
- int **ktc_is_AnyDesignator** (**ktc_tree_t** t)
- int **ktc_is_Label** (**ktc_tree_t** t)
- int **ktc_is_CaseLabel** (**ktc_tree_t** t)
- int **ktc_is_DefaultLabel** (**ktc_tree_t** t)
- int **ktc_is_CaseRangeLabel** (**ktc_tree_t** t)
- int **ktc_is_UnparsedLabel** (**ktc_tree_t** t)
- int **ktc_is_NoExpr** (**ktc_tree_t** t)
- int **ktc_is_AnyExpr** (**ktc_tree_t** t)
- int **ktc_is_NoNameQualifier** (**ktc_tree_t** t)
- int **ktc_is_GlobalScope** (**ktc_tree_t** t)
- int **ktc_is_SuperScope** (**ktc_tree_t** t)
- int **ktc_is_AnyNameSpec** (**ktc_tree_t** t)
- int **ktc_is_NoName** (**ktc_tree_t** t)
- int **ktc_is_AnyName** (**ktc_tree_t** t)
- int **ktc_is_NoTemplateArg** (**ktc_tree_t** t)
- int **ktc_is_AnyTemplateArg** (**ktc_tree_t** t)
- int **ktc_is_NoBaseSpec** (**ktc_tree_t** t)

- int **ktc_is_BaseSpec** (**ktc_tree_t** t)
- int **ktc_is_NoLambdaDeclarator** (**ktc_tree_t** t)
- int **ktc_is_LambdaDeclarator** (**ktc_tree_t** t)
- int **ktc_is_CaptureDefault** (**ktc_tree_t** t)
- int **ktc_is_Capture** (**ktc_tree_t** t)
- int **ktc_is_NoCapture** (**ktc_tree_t** t)
- int **ktc_is_NoTypeId** (**ktc_tree_t** t)
- int **ktc_is_TypeId** (**ktc_tree_t** t)
- int **ktc_is_NoNewInitializer** (**ktc_tree_t** t)
- int **ktc_is_NewInitializer** (**ktc_tree_t** t)
- int **ktc_is_NoMemberInitializer** (**ktc_tree_t** t)
- int **ktc_is_MemberInitializer** (**ktc_tree_t** t)
- int **ktc_is_NoCtorInitializer** (**ktc_tree_t** t)
- int **ktc_is_CtorInitializer** (**ktc_tree_t** t)
- int **ktc_is_NoExceptionSpec** (**ktc_tree_t** t)
- int **ktc_is_ExceptionSpec** (**ktc_tree_t** t)
- int **ktc_is_DenyThrowSpec** (**ktc_tree_t** t)
- int **ktc_is_AnyDecl** (**ktc_tree_t** t)
- int **ktc_is_AnyStmt** (**ktc_tree_t** t)
- int **ktc_is_AnyTypeParam** (**ktc_tree_t** t)
- int **ktc_is_Param** (**ktc_tree_t** t)
- int **ktc_is_AutoType** (**ktc_tree_t** t)
- int **ktc_is_ConstExpr** (**ktc_tree_t** t)
- int **ktc_is_StorageClass** (**ktc_tree_t** t)
- int **ktc_is_AlignAsExpr** (**ktc_tree_t** t)
- int **ktc_is_AlignAsType** (**ktc_tree_t** t)
- int **ktc_is_AnyTypeName** (**ktc_tree_t** t)
- int **ktc_is_FuncSpec** (**ktc_tree_t** t)
- int **ktc_is_AttributeDeclSpec** (**ktc_tree_t** t)
- int **ktc_is_UnparsedDeclSpec** (**ktc_tree_t** t)
- int **ktc_is_Attribute** (**ktc_tree_t** t)
- int **ktc_is_AttributeWithArgs** (**ktc_tree_t** t)
- int **ktc_is_PropertyAttribute** (**ktc_tree_t** t)
- int **ktc_is_PropertyPutFunc** (**ktc_tree_t** t)
- int **ktc_is_PropertyGetFunc** (**ktc_tree_t** t)
- int **ktc_is_UnparsedPropertyFunc** (**ktc_tree_t** t)
- int **ktc_is_MemberDecl** (**ktc_tree_t** t)
- int **ktc_is_MemberFunc** (**ktc_tree_t** t)
- int **ktc_is_AccessSpecification** (**ktc_tree_t** t)
- int **ktc_is_MemberTemplate** (**ktc_tree_t** t)
- int **ktc_is_MemberUsingDecl** (**ktc_tree_t** t)
- int **ktc_is_UnparsedMemberDecl** (**ktc_tree_t** t)
- int **ktc_is_PromisedMemberDecl** (**ktc_tree_t** t)
- int **ktc_is_Enumerator** (**ktc_tree_t** t)
- int **ktc_is_UnparsedEnumerator** (**ktc_tree_t** t)
- int **ktc_is_AnyNonPtrDeclarator** (**ktc_tree_t** t)
- int **ktc_is_PtrDeclarator** (**ktc_tree_t** t)
- int **ktc_is_BitFieldDeclarator** (**ktc_tree_t** t)
- int **ktc_is_AttributedDeclarator** (**ktc_tree_t** t)
- int **ktc_is_InitializedDeclarator** (**ktc_tree_t** t)
- int **ktc_is_UnparsedDeclarator** (**ktc_tree_t** t)
- int **ktc_is_ParamName** (**ktc_tree_t** t)
- int **ktc_is_UnparsedParamName** (**ktc_tree_t** t)
- int **ktc_is_CopyInitializer** (**ktc_tree_t** t)
- int **ktc_is_InitClause** (**ktc_tree_t** t)

- int **ktc_is_TruncatedInitClause** (**ktc_tree_t** t)
- int **ktc_is_DirectInitializer** (**ktc_tree_t** t)
- int **ktc_is_UnparsedInitializer** (**ktc_tree_t** t)
- int **ktc_is_FieldDesignator** (**ktc_tree_t** t)
- int **ktc_is_MemberDesignator** (**ktc_tree_t** t)
- int **ktc_is_IndexDesignator** (**ktc_tree_t** t)
- int **ktc_is_RangeDesignator** (**ktc_tree_t** t)
- int **ktc_is_IdExpr** (**ktc_tree_t** t)
- int **ktc_is_BoolLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_LiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_UserLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_StringLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_UserStringLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_NullptrLiteralExpr** (**ktc_tree_t** t)
- int **ktc_is_MemberExpr** (**ktc_tree_t** t)
- int **ktc_is_CallExpr** (**ktc_tree_t** t)
- int **ktc_is_TypeConvExpr** (**ktc_tree_t** t)
- int **ktc_is_IndexExpr** (**ktc_tree_t** t)
- int **ktc_is_UnaryExpr** (**ktc_tree_t** t)
- int **ktc_is_ThrowExpr** (**ktc_tree_t** t)
- int **ktc_is_BinaryExpr** (**ktc_tree_t** t)
- int **ktc_is_ConditionalExpr** (**ktc_tree_t** t)
- int **ktc_is_SizeOfExpr** (**ktc_tree_t** t)
- int **ktc_is_AlignOfExpr** (**ktc_tree_t** t)
- int **ktc_is_CastExpr** (**ktc_tree_t** t)
- int **ktc_is_SpecialCastExpr** (**ktc_tree_t** t)
- int **ktc_is_ParensExpr** (**ktc_tree_t** t)
- int **ktc_is_ThisExpr** (**ktc_tree_t** t)
- int **ktc_is_NewExpr** (**ktc_tree_t** t)
- int **ktc_is_DeleteExpr** (**ktc_tree_t** t)
- int **ktc_is_TypeTypeIdExpr** (**ktc_tree_t** t)
- int **ktc_is_ExprTypeIdExpr** (**ktc_tree_t** t)
- int **ktc_is_StmtExpr** (**ktc_tree_t** t)
- int **ktc_is_InitializerExpr** (**ktc_tree_t** t)
- int **ktc_is_LambdaExpr** (**ktc_tree_t** t)
- int **ktc_is_UnparsedExpr** (**ktc_tree_t** t)
- int **ktc_is_NameSpec** (**ktc_tree_t** t)
- int **ktc_is_TemplateSpec** (**ktc_tree_t** t)
- int **ktc_is_TypeOfSpec** (**ktc_tree_t** t)
- int **ktc_is_UnparsedNameQualifier** (**ktc_tree_t** t)
- int **ktc_is_QualifiedName** (**ktc_tree_t** t)
- int **ktc_is_UnqualifiedName** (**ktc_tree_t** t)
- int **ktc_is_ExprArg** (**ktc_tree_t** t)
- int **ktc_is_TypeArg** (**ktc_tree_t** t)
- int **ktc_is_TemplateTypeArg** (**ktc_tree_t** t)
- int **ktc_is_FuncDef** (**ktc_tree_t** t)
- int **ktc_is_Decl** (**ktc_tree_t** t)
- int **ktc_is_TemplateDecl** (**ktc_tree_t** t)
- int **ktc_is_LinkageSpec** (**ktc_tree_t** t)
- int **ktc_is_ExplicitInstantiation** (**ktc_tree_t** t)
- int **ktc_is_NamespaceDecl** (**ktc_tree_t** t)
- int **ktc_is_NamespaceAlias** (**ktc_tree_t** t)
- int **ktc_is_AnyUsing** (**ktc_tree_t** t)
- int **ktc_is_AliasDecl** (**ktc_tree_t** t)
- int **ktc_is_AsmDef** (**ktc_tree_t** t)

- int **ktc_is_StaticAssertDecl** (**ktc_tree_t** t)
- int **ktc_is_UnparsedDecl** (**ktc_tree_t** t)
- int **ktc_is_LabeledStmt** (**ktc_tree_t** t)
- int **ktc_is_ExprStmt** (**ktc_tree_t** t)
- int **ktc_is_CompoundStmt** (**ktc_tree_t** t)
- int **ktc_is_IfStmt** (**ktc_tree_t** t)
- int **ktc_is_IfDeclStmt** (**ktc_tree_t** t)
- int **ktc_is_SwitchStmt** (**ktc_tree_t** t)
- int **ktc_is_SwitchDeclStmt** (**ktc_tree_t** t)
- int **ktc_is_WhileStmt** (**ktc_tree_t** t)
- int **ktc_is_WhileDeclStmt** (**ktc_tree_t** t)
- int **ktc_is_DoStmt** (**ktc_tree_t** t)
- int **ktc_is_DoDeclStmt** (**ktc_tree_t** t)
- int **ktc_is_ForStmt** (**ktc_tree_t** t)
- int **ktc_is_ForEachStmt** (**ktc_tree_t** t)
- int **ktc_is_ForRangeStmt** (**ktc_tree_t** t)
- int **ktc_is_GotoStmt** (**ktc_tree_t** t)
- int **ktc_is_ContinueStmt** (**ktc_tree_t** t)
- int **ktc_is_BreakStmt** (**ktc_tree_t** t)
- int **ktc_is_ReturnStmt** (**ktc_tree_t** t)
- int **ktc_is_TryStmt** (**ktc_tree_t** t)
- int **ktc_is_TryExceptStmt** (**ktc_tree_t** t)
- int **ktc_is_TryFinallyStmt** (**ktc_tree_t** t)
- int **ktc_is_LeaveStmt** (**ktc_tree_t** t)
- int **ktc_is_AsmStmt** (**ktc_tree_t** t)
- int **ktc_is_UnparsedStmt** (**ktc_tree_t** t)
- int **ktc_is_TypeParam** (**ktc_tree_t** t)
- int **ktc_is_TemplateTypeParam** (**ktc_tree_t** t)
- int **ktc_is_CVQualifier** (**ktc_tree_t** t)
- int **ktc_is_ReservedTypeSpec** (**ktc_tree_t** t)
- int **ktc_is_TypeName** (**ktc_tree_t** t)
- int **ktc_is_EnumType** (**ktc_tree_t** t)
- int **ktc_is_ClassType** (**ktc_tree_t** t)
- int **ktc_is_AnyTypeOf** (**ktc_tree_t** t)
- int **ktc_is_NameDeclarator** (**ktc_tree_t** t)
- int **ktc_is_ParensDeclarator** (**ktc_tree_t** t)
- int **ktc_is_ArrayDeclarator** (**ktc_tree_t** t)
- int **ktc_is_FuncDeclarator** (**ktc_tree_t** t)
- int **ktc_is_KRFuncDeclarator** (**ktc_tree_t** t)
- int **ktc_is_Name** (**ktc_tree_t** t)
- int **ktc_is_Dtor** (**ktc_tree_t** t)
- int **ktc_is_SuffixFunc** (**ktc_tree_t** t)
- int **ktc_is_OpFunc** (**ktc_tree_t** t)
- int **ktc_is_ConvFunc** (**ktc_tree_t** t)
- int **ktc_is_TemplateName** (**ktc_tree_t** t)
- int **ktc_is_AnyPseudoDtor** (**ktc_tree_t** t)
- int **ktc_is_UnparsedName** (**ktc_tree_t** t)
- int **ktc_is_UsingDecl** (**ktc_tree_t** t)
- int **ktc_is_UsingDirective** (**ktc_tree_t** t)
- int **ktc_is_TypeAdjective** (**ktc_tree_t** t)
- int **ktc_is_BuiltinType** (**ktc_tree_t** t)
- int **ktc_is_TypeOfExpr** (**ktc_tree_t** t)
- int **ktc_is_TypeOfType** (**ktc_tree_t** t)
- int **ktc_is_PseudoDtor** (**ktc_tree_t** t)
- int **ktc_is_QualifiedPseudoDtor** (**ktc_tree_t** t)
- int **ktc_get_child_index** (**ktc_treeType_t** tt, **ktc_childId_t** child_id)

Variables

- const **ktc_treeType_t tid_Any**
- const **ktc_treeType_t tid_Node**
- const **ktc_treeType_t tid_TranslationUnit**
- const **ktc_treeType_t tid_DeclOrStmts**
- const **ktc_treeType_t tid_AnyFuncBody**
- const **ktc_treeType_t tid_Handlers**
- const **ktc_treeType_t tid_ExceptHandler**
- const **ktc_treeType_t tid_FinallyHandler**
- const **ktc_treeType_t tid_MaybeException**
- const **ktc_treeType_t tid_TemplateParams**
- const **ktc_treeType_t tid_DeclSpecs**
- const **ktc_treeType_t tid_AttributeSpecs**
- const **ktc_treeType_t tid_Attributes**
- const **ktc_treeType_t tid_PropertyFuncs**
- const **ktc_treeType_t tid_MemberDecls**
- const **ktc_treeType_t tid_Enumerators**
- const **ktc_treeType_t tid_MaybeDeclarator**
- const **ktc_treeType_t tid_ParamNames**
- const **ktc_treeType_t tid_Initializers**
- const **ktc_treeType_t tid_Designators**
- const **ktc_treeType_t tid_AnyLabel**
- const **ktc_treeType_t tid_Exprs**
- const **ktc_treeType_t tid_AnyNameQualifier**
- const **ktc_treeType_t tid_AnyNames**
- const **ktc_treeType_t tid_TemplateArgs**
- const **ktc_treeType_t tid_BaseSpecs**
- const **ktc_treeType_t tid_MaybeLambdaDeclarator**
- const **ktc_treeType_t tid_LambdaIntroducer**
- const **ktc_treeType_t tid_AnyCapture**
- const **ktc_treeType_t tid_MaybeTypeId**
- const **ktc_treeType_t tid_MaybeNewInitializer**
- const **ktc_treeType_t tid_MemberInitializers**
- const **ktc_treeType_t tid_MaybeCtorInitializer**
- const **ktc_treeType_t tid_MaybeExceptionSpec**
- const **ktc_treeType_t tid_NoDeclOrStmt**
- const **ktc_treeType_t tid_DeclEllipsis**
- const **ktc_treeType_t tid_DeclOrStmt**
- const **ktc_treeType_t tid_FuncBody**
- const **ktc_treeType_t tid_FuncTryBlock**
- const **ktc_treeType_t tid_PromisedFuncBody**
- const **ktc_treeType_t tid_NoHandler**
- const **ktc_treeType_t tid_Handler**
- const **ktc_treeType_t tid_NoException**
- const **ktc_treeType_t tid_Exception**
- const **ktc_treeType_t tid_DefaultException**
- const **ktc_treeType_t tid_UnparsedException**
- const **ktc_treeType_t tid_NoTemplateParam**
- const **ktc_treeType_t tid_TemplateParam**
- const **ktc_treeType_t tid_NoDeclSpec**
- const **ktc_treeType_t tid_DeclSpec**
- const **ktc_treeType_t tid_NoAttributeSpec**
- const **ktc_treeType_t tid_AttributeSpec**
- const **ktc_treeType_t tid_NoAttribute**

- const **ktc_treeType_t tid_AnyAttribute**
- const **ktc_treeType_t tid_NoPropertyFunc**
- const **ktc_treeType_t tid_AnyPropertyFunc**
- const **ktc_treeType_t tid_NoMemberDecl**
- const **ktc_treeType_t tid_AnyMemberDecl**
- const **ktc_treeType_t tid_NoEnumerator**
- const **ktc_treeType_t tid_AnyEnumerator**
- const **ktc_treeType_t tid_NoDeclarator**
- const **ktc_treeType_t tid_AnyDeclarator**
- const **ktc_treeType_t tid_NoParamName**
- const **ktc_treeType_t tid_AnyParamName**
- const **ktc_treeType_t tid_NoInitializer**
- const **ktc_treeType_t tid_AnyInitializer**
- const **ktc_treeType_t tid_NoDesignator**
- const **ktc_treeType_t tid_AnyDesignator**
- const **ktc_treeType_t tid_Label**
- const **ktc_treeType_t tid_CaseLabel**
- const **ktc_treeType_t tid_DefaultLabel**
- const **ktc_treeType_t tid_CaseRangeLabel**
- const **ktc_treeType_t tid_UnparsedLabel**
- const **ktc_treeType_t tid_NoExpr**
- const **ktc_treeType_t tid_AnyExpr**
- const **ktc_treeType_t tid_NoNameQualifier**
- const **ktc_treeType_t tid_GlobalScope**
- const **ktc_treeType_t tid_SuperScope**
- const **ktc_treeType_t tid_AnyNameSpec**
- const **ktc_treeType_t tid_NoName**
- const **ktc_treeType_t tid_AnyName**
- const **ktc_treeType_t tid_NoTemplateArg**
- const **ktc_treeType_t tid_AnyTemplateArg**
- const **ktc_treeType_t tid_NoBaseSpec**
- const **ktc_treeType_t tid_BaseSpec**
- const **ktc_treeType_t tid_NoLambdaDeclarator**
- const **ktc_treeType_t tid_LambdaDeclarator**
- const **ktc_treeType_t tid_CaptureDefault**
- const **ktc_treeType_t tid_Capture**
- const **ktc_treeType_t tid_NoCapture**
- const **ktc_treeType_t tid_NoTypeId**
- const **ktc_treeType_t tid_TypeId**
- const **ktc_treeType_t tid_NoNewInitializer**
- const **ktc_treeType_t tid_NewInitializer**
- const **ktc_treeType_t tid_NoMemberInitializer**
- const **ktc_treeType_t tid_MemberInitializer**
- const **ktc_treeType_t tid_NoCtorInitializer**
- const **ktc_treeType_t tid_CtorInitializer**
- const **ktc_treeType_t tid_NoExceptionSpec**
- const **ktc_treeType_t tid_ExceptionSpec**
- const **ktc_treeType_t tid_DenyThrowSpec**
- const **ktc_treeType_t tid_AnyDecl**
- const **ktc_treeType_t tid_AnyStmt**
- const **ktc_treeType_t tid_AnyTypeParam**
- const **ktc_treeType_t tid_Param**
- const **ktc_treeType_t tid_AutoType**
- const **ktc_treeType_t tid_ConstExpr**
- const **ktc_treeType_t tid_StorageClass**

- const **ktc_treeType_t tid_AlignAsExpr**
- const **ktc_treeType_t tid_AlignAsType**
- const **ktc_treeType_t tid_AnyTypeName**
- const **ktc_treeType_t tid_FuncSpec**
- const **ktc_treeType_t tid_AttributeDeclSpec**
- const **ktc_treeType_t tid_UnparsedDeclSpec**
- const **ktc_treeType_t tid_Attribute**
- const **ktc_treeType_t tid_AttributeWithArgs**
- const **ktc_treeType_t tid_PropertyAttribute**
- const **ktc_treeType_t tid_PropertyPutFunc**
- const **ktc_treeType_t tid_PropertyGetFunc**
- const **ktc_treeType_t tid_UnparsedPropertyFunc**
- const **ktc_treeType_t tid_MemberDecl**
- const **ktc_treeType_t tid_MemberFunc**
- const **ktc_treeType_t tid_AccessSpecification**
- const **ktc_treeType_t tid_MemberTemplate**
- const **ktc_treeType_t tid_MemberUsingDecl**
- const **ktc_treeType_t tid_UnparsedMemberDecl**
- const **ktc_treeType_t tid_PromisedMemberDecl**
- const **ktc_treeType_t tid_Enumerator**
- const **ktc_treeType_t tid_UnparsedEnumerator**
- const **ktc_treeType_t tid_AnyNonPtrDeclarator**
- const **ktc_treeType_t tid_PtrDeclarator**
- const **ktc_treeType_t tid_BitFieldDeclarator**
- const **ktc_treeType_t tid_AttributedDeclarator**
- const **ktc_treeType_t tid_InitializedDeclarator**
- const **ktc_treeType_t tid_UnparsedDeclarator**
- const **ktc_treeType_t tid_ParamName**
- const **ktc_treeType_t tid_UnparsedParamName**
- const **ktc_treeType_t tid_CopyInitializer**
- const **ktc_treeType_t tid_InitClause**
- const **ktc_treeType_t tid_TruncatedInitClause**
- const **ktc_treeType_t tid_DirectInitializer**
- const **ktc_treeType_t tid_UnparsedInitializer**
- const **ktc_treeType_t tid_FieldDesignator**
- const **ktc_treeType_t tid_MemberDesignator**
- const **ktc_treeType_t tid_IndexDesignator**
- const **ktc_treeType_t tid_RangeDesignator**
- const **ktc_treeType_t tid_IdExpr**
- const **ktc_treeType_t tid_BoolLiteralExpr**
- const **ktc_treeType_t tid_LiteralExpr**
- const **ktc_treeType_t tid_UserLiteralExpr**
- const **ktc_treeType_t tid_StringLiteralExpr**
- const **ktc_treeType_t tid_UserStringLiteralExpr**
- const **ktc_treeType_t tid_NullptrLiteralExpr**
- const **ktc_treeType_t tid_MemberExpr**
- const **ktc_treeType_t tid_CallExpr**
- const **ktc_treeType_t tid_TypeConvExpr**
- const **ktc_treeType_t tid_IndexExpr**
- const **ktc_treeType_t tid_UnaryExpr**
- const **ktc_treeType_t tid_ThrowExpr**
- const **ktc_treeType_t tid_BinaryExpr**
- const **ktc_treeType_t tid_ConditionalExpr**
- const **ktc_treeType_t tid_SizeOfExpr**
- const **ktc_treeType_t tid_AlignOfExpr**

- const **ktc_treeType_t tid_CastExpr**
- const **ktc_treeType_t tid_SpecialCastExpr**
- const **ktc_treeType_t tid_ParensExpr**
- const **ktc_treeType_t tid_ThisExpr**
- const **ktc_treeType_t tid_NewExpr**
- const **ktc_treeType_t tid_DeleteExpr**
- const **ktc_treeType_t tid_TypeTypeIdExpr**
- const **ktc_treeType_t tid_ExprTypeIdExpr**
- const **ktc_treeType_t tid_StmtExpr**
- const **ktc_treeType_t tid_InitializerExpr**
- const **ktc_treeType_t tid_LambdaExpr**
- const **ktc_treeType_t tid_UnparsedExpr**
- const **ktc_treeType_t tid_NameSpec**
- const **ktc_treeType_t tid_TemplateSpec**
- const **ktc_treeType_t tid_TypeOfSpec**
- const **ktc_treeType_t tid_UnparsedNameQualifier**
- const **ktc_treeType_t tid_QualifiedName**
- const **ktc_treeType_t tid_UnqualifiedName**
- const **ktc_treeType_t tid_ExprArg**
- const **ktc_treeType_t tid_TypeArg**
- const **ktc_treeType_t tid_TemplateTypeArg**
- const **ktc_treeType_t tid_FuncDef**
- const **ktc_treeType_t tid_Decl**
- const **ktc_treeType_t tid_TemplateDecl**
- const **ktc_treeType_t tid_LinkageSpec**
- const **ktc_treeType_t tid_ExplicitInstantiation**
- const **ktc_treeType_t tid_NamespaceDecl**
- const **ktc_treeType_t tid_NamespaceAlias**
- const **ktc_treeType_t tid_AnyUsing**
- const **ktc_treeType_t tid_AliasDecl**
- const **ktc_treeType_t tid_AsmDef**
- const **ktc_treeType_t tid_StaticAssertDecl**
- const **ktc_treeType_t tid_UnparsedDecl**
- const **ktc_treeType_t tid_LabeledStmt**
- const **ktc_treeType_t tid_ExprStmt**
- const **ktc_treeType_t tid_CompoundStmt**
- const **ktc_treeType_t tid_IfStmt**
- const **ktc_treeType_t tid_IfDeclStmt**
- const **ktc_treeType_t tid_SwitchStmt**
- const **ktc_treeType_t tid_SwitchDeclStmt**
- const **ktc_treeType_t tid_WhileStmt**
- const **ktc_treeType_t tid_WhileDeclStmt**
- const **ktc_treeType_t tid_DoStmt**
- const **ktc_treeType_t tid_DoDeclStmt**
- const **ktc_treeType_t tid_ForStmt**
- const **ktc_treeType_t tid_ForEachStmt**
- const **ktc_treeType_t tid_ForRangeStmt**
- const **ktc_treeType_t tid_GotoStmt**
- const **ktc_treeType_t tid_ContinueStmt**
- const **ktc_treeType_t tid_BreakStmt**
- const **ktc_treeType_t tid_ReturnStmt**
- const **ktc_treeType_t tid_TryStmt**
- const **ktc_treeType_t tid_TryExceptStmt**
- const **ktc_treeType_t tid_TryFinallyStmt**
- const **ktc_treeType_t tid_LeaveStmt**

- const **ktc_treeType_t tid_AsmStmt**
- const **ktc_treeType_t tid_UnparsedStmt**
- const **ktc_treeType_t tid_TypeParam**
- const **ktc_treeType_t tid_TemplateTypeParam**
- const **ktc_treeType_t tid_CVQualifier**
- const **ktc_treeType_t tid_ReservedTypeSpec**
- const **ktc_treeType_t tid_TypeName**
- const **ktc_treeType_t tid_EnumType**
- const **ktc_treeType_t tid_ClassType**
- const **ktc_treeType_t tid_AnyTypeOf**
- const **ktc_treeType_t tid_NameDeclarator**
- const **ktc_treeType_t tid_ParensDeclarator**
- const **ktc_treeType_t tid_ArrayDeclarator**
- const **ktc_treeType_t tid_FuncDeclarator**
- const **ktc_treeType_t tid_KRFuncDeclarator**
- const **ktc_treeType_t tid_Name**
- const **ktc_treeType_t tid_Dtor**
- const **ktc_treeType_t tid_SuffixFunc**
- const **ktc_treeType_t tid_OpFunc**
- const **ktc_treeType_t tid_ConvFunc**
- const **ktc_treeType_t tid_TemplateName**
- const **ktc_treeType_t tid_AnyPseudoDtor**
- const **ktc_treeType_t tid_UnparsedName**
- const **ktc_treeType_t tid_UsingDecl**
- const **ktc_treeType_t tid_UsingDirective**
- const **ktc_treeType_t tid_TypeAdjective**
- const **ktc_treeType_t tid_BuiltinType**
- const **ktc_treeType_t tid_TypeOfExpr**
- const **ktc_treeType_t tid_TypeOfType**
- const **ktc_treeType_t tid_PseudoDtor**
- const **ktc_treeType_t tid_QualifiedPseudoDtor**
- const **ktc_childId_t cid_Next**
- const **ktc_childId_t cid_Decls**
- const **ktc_childId_t cid_Expr**
- const **ktc_childId_t cid_Stmt**
- const **ktc_childId_t cid_LambdaCapture**
- const **ktc_childId_t cid_CtorInit**
- const **ktc_childId_t cid_Handlers**
- const **ktc_childId_t cid_Exception**
- const **ktc_childId_t cid_DeclSpecs**
- const **ktc_childId_t cid_Declarator**
- const **ktc_childId_t cid_Attributes**
- const **ktc_childId_t cid_Designators**
- const **ktc_childId_t cid_Lower**
- const **ktc_childId_t cid_Upper**
- const **ktc_childId_t cid_Qualifier**
- const **ktc_childId_t cid_Name**
- const **ktc_childId_t cid_Params**
- const **ktc_childId_t cid_Throw**
- const **ktc_childId_t cid_TrailingReturnType**
- const **ktc_childId_t cid_Args**
- const **ktc_childId_t cid_MemberInitializers**
- const **ktc_childId_t cid_TypeIds**
- const **ktc_childId_t cid_Decl**
- const **ktc_childId_t cid_Type**

- const **ktc_childId_t cid_AttributeSpec**
- const **ktc_childId_t cid_PropertyFuncs**
- const **ktc_childId_t cid_Declarators**
- const **ktc_childId_t cid_FuncBody**
- const **ktc_childId_t cid_TemplateParams**
- const **ktc_childId_t cid_MemberDecl**
- const **ktc_childId_t cid_NameSpec**
- const **ktc_childId_t cid_CVQualifiers**
- const **ktc_childId_t cid_Bits**
- const **ktc_childId_t cid_Init**
- const **ktc_childId_t cid_Inits**
- const **ktc_childId_t cid_Index**
- const **ktc_childId_t cid_Adjacent**
- const **ktc_childId_t cid_Literal**
- const **ktc_childId_t cid_Func**
- const **ktc_childId_t cid_Left**
- const **ktc_childId_t cid_Right**
- const **ktc_childId_t cid_Cond**
- const **ktc_childId_t cid_Then**
- const **ktc_childId_t cid_Else**
- const **ktc_childId_t cid_Placement**
- const **ktc_childId_t cid_Initializer**
- const **ktc_childId_t cid_Introducer**
- const **ktc_childId_t cid_TemplateName**
- const **ktc_childId_t cid_KRParams**
- const **ktc_childId_t cid_Label**
- const **ktc_childId_t cid_Stmts**
- const **ktc_childId_t cid_Handler**
- const **ktc_childId_t cid_Default**
- const **ktc_childId_t cid_Base**
- const **ktc_childId_t cid_Enumerators**
- const **ktc_childId_t cid_AttributeSpecs**
- const **ktc_childId_t cid_BaseSpecs**
- const **ktc_childId_t cid_MemberDecls**
- const **ktc_childId_t cid_Size**
- const **ktc_childId_t cid_ConversionType**
- const int **KTC_OPCODE_NONE**
- const int **KTC_OPCODE_ASSIGN**
- const int **KTC_OPCODE_MULASSIGN**
- const int **KTC_OPCODE_DIVASSIGN**
- const int **KTC_OPCODE_MODASSIGN**
- const int **KTC_OPCODE_ADDASSIGN**
- const int **KTC_OPCODE_SUBASSIGN**
- const int **KTC_OPCODE_ASLASSIGN**
- const int **KTC_OPCODE_ASRASSIGN**
- const int **KTC_OPCODE_ANDASSIGN**
- const int **KTC_OPCODE_XORASSIGN**
- const int **KTC_OPCODE_ORASSIGN**
- const int **KTC_OPCODE_COMMA**
- const int **KTC_OPCODE_COND**
- const int **KTC_OPCODE_LOGOR**
- const int **KTC_OPCODE_LOGAND**
- const int **KTC_OPCODE_BITOR**
- const int **KTC_OPCODE_BITXOR**
- const int **KTC_OPCODE_BITAND**

- const int **KTC_OPCODE_EQ**
- const int **KTC_OPCODE_NE**
- const int **KTC_OPCODE_LT**
- const int **KTC_OPCODE_GT**
- const int **KTC_OPCODE_LE**
- const int **KTC_OPCODE_GE**
- const int **KTC_OPCODE_ASR**
- const int **KTC_OPCODE_ASL**
- const int **KTC_OPCODE_ADD**
- const int **KTC_OPCODE_SUB**
- const int **KTC_OPCODE_MUL**
- const int **KTC_OPCODE_DIV**
- const int **KTC_OPCODE_MOD**
- const int **KTC_OPCODE_PREINC**
- const int **KTC_OPCODE_PREDEC**
- const int **KTC_OPCODE_SIZEOF**
- const int **KTC_OPCODE_DEREF**
- const int **KTC_OPCODE_ADDRESS**
- const int **KTC_OPCODE_PLUS**
- const int **KTC_OPCODE_MINUS**
- const int **KTC_OPCODE_BITNOT**
- const int **KTC_OPCODE_LOGNOT**
- const int **KTC_OPCODE_POSTINC**
- const int **KTC_OPCODE_POSTDEC**
- const int **KTC_OPCODE_FIELD**
- const int **KTC_OPCODE_FIELDREF**
- const int **KTC_OPCODE_DOTAST**
- const int **KTC_OPCODE_DEREFAST**
- const int **KTC_OPCODE_ROUND_BRACKETS**
- const int **KTC_OPCODE_SQUARE_BRACKETS**
- const int **KTC_OPCODE_THROW**
- const int **KTC_OPCODE_MIN**
- const int **KTC_OPCODE_MAX**
- const int **KTC_STORAGECLASS_NONE**
- const int **KTC_STORAGECLASS_TYPEDEF**
- const int **KTC_STORAGECLASS_EXTERN**
- const int **KTC_STORAGECLASS_STATIC**
- const int **KTC_STORAGECLASS_AUTO**
- const int **KTC_STORAGECLASS_REGISTER**
- const int **KTC_STORAGECLASS_MUTABLE**
- const int **KTC_STORAGECLASS_THREADLOCAL**
- const int **KTC_CVQUALIFIER_NONE**
- const int **KTC_CVQUALIFIER_CONST**
- const int **KTC_CVQUALIFIER_VOLATILE**
- const int **KTC_CVQUALIFIER_RESTRICT**
- const int **KTC_CLASSTAG_NONE**
- const int **KTC_CLASSTAG_STRUCT**
- const int **KTC_CLASSTAG_UNION**
- const int **KTC_CLASSTAG_CLASS**
- const int **KTC_BUILTINTYPE_NONE**
- const int **KTC_BUILTINTYPE_VOID**
- const int **KTC_BUILTINTYPE_BOOL**
- const int **KTC_BUILTINTYPE_WCHAR_T**
- const int **KTC_BUILTINTYPE_CHAR**
- const int **KTC_BUILTINTYPE_SIGNEDCHAR**

- const int **KTC_BUILTINTYPE_UNSIGNEDCHAR**
- const int **KTC_BUILTINTYPE_SHORTINT**
- const int **KTC_BUILTINTYPE_SIGNEDSHORTINT**
- const int **KTC_BUILTINTYPE_UNSIGNEDSHORTINT**
- const int **KTC_BUILTINTYPE_INT**
- const int **KTC_BUILTINTYPE_SIGNEDINT**
- const int **KTC_BUILTINTYPE_UNSIGNEDINT**
- const int **KTC_BUILTINTYPE_LONGINT**
- const int **KTC_BUILTINTYPE_SIGNEDLONGINT**
- const int **KTC_BUILTINTYPE_UNSIGNEDLONGINT**
- const int **KTC_BUILTINTYPE_LONGLONGINT**
- const int **KTC_BUILTINTYPE_SIGNEDLONGLONGINT**
- const int **KTC_BUILTINTYPE_UNSIGNEDLONGLONGINT**
- const int **KTC_BUILTINTYPE_FLOAT**
- const int **KTC_BUILTINTYPE_DOUBLE**
- const int **KTC_BUILTINTYPE_LONGDOUBLE**
- const int **KTC_FUNCSPECIFIER_NONE**
- const int **KTC_FUNCSPECIFIER_INLINE**
- const int **KTC_FUNCSPECIFIER_VIRTUAL**
- const int **KTC_FUNCSPECIFIER_EXPLICIT**
- const int **KTC_FUNCSPECIFIER_FRIEND**
- const int **KTC_CASTSPECIFIER_DYNAMIC**
- const int **KTC_CASTSPECIFIER_STATIC**
- const int **KTC_CASTSPECIFIER_REINTERPRET**
- const int **KTC_CASTSPECIFIER_CONST**
- const int **KTC_POINTEROPERATOR_NONE**
- const int **KTC_POINTEROPERATOR_POINTER**
- const int **KTC_POINTEROPERATOR_REFERENCE**
- const int **KTC_POINTEROPERATOR_RVALUE**

5.1.1 Function Documentation

5.1.1.1 int ktc_get_child_index (ktc_treeType_t *tt*, ktc_childId_t *child_id*)

5.2 ktcAPI.h File Reference

```
#include "ktcMainAPI.h"
#include "gen-ktcAPI.h"
```

5.3 ktcMainAPI.h File Reference

```
#include "kwapi.h"
#include <stdint.h>
```

Macros

- #define **KTC_API_VERSION_MAJOR** 1
- #define **KTC_API_VERSION_MINOR** 9
- #define **KTC_API_VERSION_PATCHLEVEL** 0
- #define **KTC_CUSTOM_TYPES**
- #define **ktc_require**(t, ttype) if (!**ktc_isTreeType**((t),(ttype))) { return 0; }

Typedefs

- typedef union CTree_Node * **ktc_tree_t**
- typedef union rs_element * **ktc_semanticInfo_t**
- typedef union rs_element * **ktc_languageType_t**
- typedef struct ktc_string_t_ * **ktc_string_t**
- typedef int64_t **ktc_long_long_t**
- typedef int **ktc_treeType_t**
- typedef int **ktc_childId_t**
- typedef int(* **ktc_treeHook_t**) (**ktc_tree_t**)
- typedef void(* **ktc_eventHook_t**) (void)
- typedef void * **ktc_position_t**
- typedef **ktc_position_t ktc_position**
- typedef void * **ktc_autofix_t**
- typedef void * **ktc_message_t**
- typedef void * **ktc_message**

Functions

- int **ktc_isTreeType** (**ktc_tree_t** t, **ktc_treeType_t** ttype)
- const char * **ktc_treeType_getName** (**ktc_tree_t** ttype)
- **ktc_tree_t ktc_proceed** (**ktc_tree_t** t, **ktc_childId_t** child_id)
- void **ktc_forAllSubtreeNodes** (**ktc_tree_t** t, int(*callback)(**ktc_tree_t**, void *), void *data)
- void **ktc_sema_forAllSubtreeNodes** (**ktc_semanticInfo_t** si, int(*callback)(**ktc_tree_t**, void *), void *data)
- int **ktc_isMacroExpansion** (**ktc_tree_t** t)
- int **ktc_isMacroExpansion2** (**ktc_tree_t** t)
- int **ktc_nodeStackTop** (void)
- **ktc_tree_t ktc_nodeStackGet** (int n)
- void **ktc_registerTreeHook** (int tree_event, **ktc_treeType_t** tt, **ktc_treeHook_t** p_hook)
- void **ktc_registerStartTraverseHook** (**ktc_eventHook_t** p_hook)
- void **ktc_registerSaveContextHook** (**ktc_eventHook_t** p_hook)
- void **ktc_registerRestoreContextHook** (**ktc_eventHook_t** p_hook)
- void **ktc_registerStopTraverseHook** (**ktc_eventHook_t** p_hook)
- void **ktc_unregisterTreeHook** (int tree_event, **ktc_treeType_t** tt, **ktc_treeHook_t** p_hook)
- **ktc_semanticInfo_t ktc_getSemanticInfo** (**ktc_tree_t** t)
- **ktc_semanticInfo_t ktc_getCalledFunction** (**ktc_tree_t** t)
- **ktc_semanticInfo_t ktc_getAssociatedScope** (**ktc_tree_t** t)
- **ktc_semanticInfo_t ktc_sema_getOverridenMethod** (**ktc_tree_t** t)
- int **ktc_sema_isSameFunctions** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isSameVariables** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_haveSameSignature** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_haveSameFunctionType** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isSameClass** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isSameEnum** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)
- int **ktc_sema_isSameScope** (**ktc_semanticInfo_t** si1, **ktc_semanticInfo_t** si2)

- int **ktc_sema_isNone** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isBitfield** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isScope** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isClass** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isUnion** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isTemplate** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isEnum** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isEnumConstant** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isType** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isVariable** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPointer** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isArray** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isReference** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFunctionType** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFunction** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isBuiltin** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isSpecialization** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isInstantiation** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isInstantiatedFunction** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isIntegerValue** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isObjectValue** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFunctionTemplateSet** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFunctionTemplate** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isNamespaceAlias** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isUsingDirective** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isUsingDeclaration** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isUsing** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isNamespace** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPOD** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isFriend** (**ktc_semanticInfo_t** si)
- int **ktc_sema_hasMethods** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getClassTag** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isAnonymous** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isTypeParameter** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isVirtual** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPrivate** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isProtected** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPublic** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isPureVirtual** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isConstructor** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isDestructor** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isOperatorFunction** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isLocal** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isGlobal** (**ktc_semanticInfo_t** si)
- **ktc_tree_t ktc_sema_getVariableInitializer** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getVariableType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getVariableValue** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getNumberOfArguments** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFormalArgument** (**ktc_semanticInfo_t** si, int n)
- **ktc_semanticInfo_t ktc_sema_getReturnType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFunctionType** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getBuiltinCode** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getCVQualifiers** (**ktc_semanticInfo_t** si)
- int **ktc_sema_isImmutable** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getPointedType** (**ktc_semanticInfo_t** si)

- **ktc_semanticInfo_t ktc_sema_getArrayElementType** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getArraySize** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getReferencedType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getDefinedType** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_skipTypedefs** (**ktc_semanticInfo_t** si, int ∗cvq)
- const char ∗ **ktc_sema_getIdentifier** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getIdentifierNo** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getNumber** (**ktc_semanticInfo_t** si)
- const char ∗ **ktc_sema_getTypedefedName** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_stripCVQ** (**ktc_semanticInfo_t** si)
- char ∗ **ktc_sema_getQualifiedName** (**ktc_semanticInfo_t** si)
- char ∗ **ktc_sema_getTypeName** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getScope** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getNumberOfBaseInfo** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getBaseInfo** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isBaseVirtual** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isBasePublic** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isBaseProtected** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isBasePrivate** (**ktc_semanticInfo_t** si, int i)
- int **ktc_sema_isConstMethod** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFirstByName** (**ktc_semanticInfo_t** scope, const char ∗name)
- **kw_array_t** ∗ **ktc_sema_getAllByName** (**ktc_semanticInfo_t** scope, const char ∗name)
- **ktc_semanticInfo_t ktc_sema_findFirstByName** (**ktc_semanticInfo_t** scope, const char ∗name)
- **ktc_semanticInfo_t ktc_sema_getGlobalScope** ()
- **kw_array_t** ∗ **ktc_sema_findAllByName** (**ktc_semanticInfo_t** scope, const char ∗name)
- void **ktc_sema_forAllClassDeclarations** (**ktc_semanticInfo_t** class_info, void(∗callback)(**ktc_semantic**↩**Info_t**))
- void **ktc_sema_forAllScopeDeclarations** (**ktc_semanticInfo_t** scope_info, void(∗callback)(**ktc_**↩**semanticInfo_t**))
- int **ktc_sema_functionOverloadsFunction** (**ktc_semanticInfo_t** func1, **ktc_semanticInfo_t** func2)
- int **ktc_sema_RelatedClasses** (**ktc_semanticInfo_t** class1, **ktc_semanticInfo_t** class2)
- **ktc_languageType_t ktc_getLanguageType** (**ktc_tree_t** node)
- int **ktc_languageTypeSize** (**ktc_languageType_t** lang_type)
- int **ktc_languageTypeIsPointer** (**ktc_languageType_t** type)
- **ktc_languageType_t ktc_getPointedType** (**ktc_languageType_t** ptr_type)
- int **ktc_languageTypeIsBuiltin** (**ktc_languageType_t** type, int builtin_code)
- int **ktc_languageTypeSignedness** (**ktc_languageType_t** ctypeinfo)
- int **ktc_languageTypeEffectiveSignedness** (**ktc_languageType_t** ctypeinfo)
- int **ktc_isCallTo** (**ktc_tree_t** node, const char ∗fn_name)
- **ktc_tree_t ktc_getCallArgument** (**ktc_tree_t** node, int n_arg)
- int **ktc_getNumberOfCallArguments** (**ktc_tree_t** node)
- int **ktc_isCharLiteral** (**ktc_tree_t** node)
- **ktc_long_long_t ktc_getIntegerValue** (**ktc_tree_t** t, int ∗error_flag)
- **ktc_long_long_t ktc_sema_getIntegerValue** (**ktc_semanticInfo_t** val, int ∗error_flag)
- const char ∗ **ktc_sema_getObjectName** (**ktc_semanticInfo_t** si)
- int **ktc_sema_getIntValueType** (**ktc_semanticInfo_t** si)
- int **ktc_isNullPointerConstant** (**ktc_tree_t** t)
- **ktc_tree_t ktc_getSizeofArgument** (**ktc_tree_t** t)
- int **ktc_assembleStringConstant** (**ktc_tree_t** exprString, char ∗∗pbuf)
- int **ktc_isWideString** (**ktc_tree_t** exprString)
- int **ktc_isUTF16String** (**ktc_tree_t** exprString)
- int **ktc_isUTF32String** (**ktc_tree_t** exprString)
- const char ∗ **ktc_getNoIdent** (void)
- const char ∗ **ktc_getIdentifier** (**ktc_tree_t** t)
- int **ktc_getIdentifierNo** (**ktc_tree_t** t)

- char * **ktc_getTokens** (**ktc_tree_t** t)
- int **ktc_is_NoToken** (**ktc_tree_t** t)
- int **ktc_is_Token** (**ktc_tree_t** t)
- char * **ktc_getStringConstantValue** (**ktc_tree_t** t)
- **ktc_tree_t ktc_skipBrackets** (**ktc_tree_t** t)
- **ktc_tree_t ktc_getNameDeclarator** (**ktc_tree_t**)
- int **ktc_getOperation** (**ktc_tree_t** t)
- int **ktc_isOperationOverloaded** (**ktc_tree_t** t)
- int **ktc_getStorageClass** (**ktc_tree_t** t)
- int **ktc_getPointerOperator** (**ktc_tree_t** t)
- int **ktc_getTypeQualifiers** (**ktc_tree_t** t)
- int **ktc_getClassTag** (**ktc_tree_t** t)
- int **ktc_getBuiltinType** (**ktc_tree_t** t)
- int **ktc_getBuiltinTypeSize** (const char *type_name)
- int **ktc_getPointerSize** ()
- int **ktc_getCastSpecifier** (**ktc_tree_t** t)
- int **ktc_getFunctionSpecifier** (**ktc_tree_t** t)
- int **ktc_compareSubtrees** (**ktc_tree_t** t1, **ktc_tree_t** t2)
- **ktc_position_t ktc_position_new** (int line, int col, const char *fname)
- **ktc_position_t ktc_position_copy** (**ktc_position_t** pos)
- void **ktc_position_delete** (**ktc_position_t** pos)
- **ktc_position_t ktc_getStartPosition** (**ktc_tree_t** t)
- **ktc_position_t ktc_getEndPosition** (**ktc_tree_t** t)
- int **ktc_position_getLine** (**ktc_position_t** pos)
- void **ktc_position_setLine** (**ktc_position_t** pos, int line)
- int **ktc_position_getColumn** (**ktc_position_t** pos)
- char * **ktc_position_getFileName** (**ktc_position_t** pos)
- **ktc_autofix_t ktc_autofix_new** ()
- void **ktc_autofix_delete** (**ktc_autofix_t** a)
- void **ktc_autofix_addSegment** (**ktc_autofix_t** a, const char *repl, const char *file, int line1, int col1, int line2, int col2)
- int **ktc_error_isEnabled** (const char *error_id)
- const char * **ktc_error_getConfigurationParameter** (const char *error_id, const char *param_name)
- int **ktc_getFrontendLanguage** (void)
- int **ktc_getFrontendDialect** (void)
- int **ktc_hasBuiltinWideChar** (void)
- **ktc_message_t ktc_message_new** (const char *error_id)
- void **ktc_message_setPosition** (**ktc_message_t** msg, **ktc_position_t** pos)
- void **ktc_message_addAttribute** (**ktc_message_t** msg, const char *attr_string)
- void **ktc_message_addAnchorAttribute** (**ktc_message_t** msg, const char *attr_string)
- void **ktc_message_addTraceBySemanticsInfo** (**ktc_message_t** msg, **ktc_semanticInfo_t** sema)
- void **ktc_message_addEvent** (**ktc_message_t** msg, **ktc_position_t** pos, const char *str)
- void * **ktc_event_new** (**ktc_position_t** pos, const char *str)
- void **ktc_event_setParameter** (void *event, const char *name, const char *value)
- void **ktc_message_addEventEx** (void *msg, void *event)
- void **ktc_message_render** (**ktc_message_t** msg)
- void **ktc_message_render_wi_autofix** (**ktc_message_t** msg, **ktc_autofix_t** a)
- void **ktc_message_delete** (**ktc_message_t** msg)
- void **ktc_message_setFunction** (void *function)
- void **ktc_message_unsetFunction** ()
- int **ktc_isStatic** (**ktc_tree_t** node)
- int **ktc_sema_isStatic** (**ktc_semanticInfo_t** info)
- int **ktc_isNameIncluded** (const char *fname)
- int **ktc_isIncluded** (**ktc_tree_t** node)
- int **ktc_sema_isIncluded** (**ktc_semanticInfo_t** si)

- void **ktc_free** (void ∗ptr)
- int **ktc_isConstructor** (**ktc_tree_t** node)
- int **ktc_sema_getFieldNumber** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFieldType** (**ktc_semanticInfo_t** si, int fieldpos)
- **ktc_position_t ktc_sema_getPosition** (**ktc_semanticInfo_t** si)
- **ktc_tree_t ktc_sema_getAST** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getPrimaryTemplate** (**ktc_semanticInfo_t** si)
- **kw_array_t** ∗ **ktc_sema_getInstantiationParameters** (**ktc_semanticInfo_t** si)
- **kw_array_t** ∗ **ktc_sema_getEnumerators** (**ktc_semanticInfo_t** si)
- int **ktc_isDeclaration** (**ktc_tree_t** info)
- int **ktc_isDefinition** (**ktc_tree_t** info)
- int **ktc_sema_isFuncDef** (**ktc_semanticInfo_t** info)
- **ktc_semanticInfo_t ktc_sema_getInstantiatedClass** (**ktc_semanticInfo_t** si)
- **ktc_semanticInfo_t ktc_sema_getFunctionFromTemplate** (**ktc_semanticInfo_t** info)
- **kw_array_t** ∗ **ktc_sema_getFunctionTemplateSpecializations** (**ktc_semanticInfo_t** info)
- **kw_array_t** ∗ **ktc_sema_getFunctionTemplateInstantiations** (**ktc_semanticInfo_t** info)
- **ktc_semanticInfo_t ktc_sema_getInstantiationOrigin** (**ktc_semanticInfo_t** info)
- short **ktc_sema_checkBitField** (**ktc_semanticInfo_t** info)
- int **ktc_sema_isFunctionPointer** (**ktc_semanticInfo_t** info)
- **ktc_semanticInfo_t ktc_sema_getFunctionPointerType** (**ktc_semanticInfo_t** info)
- int **ktc_sema_isInline** (**ktc_semanticInfo_t** info)

- **ktc_string_t ktc_string_new** (const char ∗s)
- void **ktc_string_delete** (**ktc_string_t** ks)
- const char ∗ **ktc_string_get_cstring** (**ktc_string_t** ks)

Variables

- int **KTC_TREE_EVENT_ON_ENTER**
- int **KTC_TREE_EVENT_ON_NEXT**
- int **KTC_TREE_EVENT_ON_LEAVE**
- const char ∗ **ktc_constructorName**
- const char ∗ **ktc_destructorName**
- int **KTC_TYPESIGNEDNESS_NONE**
- int **KTC_TYPESIGNEDNESS_DEFAULT**
- int **KTC_TYPESIGNEDNESS_SIGNED**
- int **KTC_TYPESIGNEDNESS_UNSIGNED**
- const int **KTC_LANGUAGE_C**
- const int **KTC_LANGUAGE_CXX**
- const int **KTC_DIALECT_STD**
- const int **KTC_DIALECT_GNU**
- const int **KTC_DIALECT_MS**
- const int **KTC_DIALECT_ARM**
- const int **KTC_DIALECT_GHS**
- const int **KTC_DIALECT_SUN**
- const int **KTC_DIALECT_TI**

5.3.1 Macro Definition Documentation

5.3.1.1 #define KTC_API_VERSION_MAJOR 1

5.3.1.2 #define KTC_API_VERSION_MINOR 9

5.3.1.3 #define KTC_API_VERSION_PATCHLEVEL 0

5.3.2 Function Documentation

5.3.2.1 int ktc_languageTypeEffectiveSignedness (ktc_languageType_t *ctypeinfo*)

Get effective 'signedness' of a type

Returns

'signedness' value (see **Attribute values for signedness of types** (p. 38))

5.3.2.2 void ktc_string_delete (ktc_string_t *ks*)

Delete ktc string

Parameters

ks	ktc string

5.3.2.3 const char* ktc_string_get_cstring (ktc_string_t *ks*)

Get character string from ktc string

Parameters

ks	ktc string

Returns

character string

5.3.2.4 ktc_string_t ktc_string_new (const char * *s*)

Create new ktc string from character string

Parameters

s	character string

Returns

newly created ktc string

5.4 kwapi.h File Reference

Macros

- #define **KWAPI_DECLARE**(type) type
- #define **KWAPI_DECLARE_CPP**(type) type
- #define **KWAPI_DECLARE_NONSTD**(type) type
- #define **KWAPI_DECLARE_DATA**

Typedefs

- typedef struct ParameterNode * **kwapi_cfgparam_t**
- typedef unsigned int **kw_size_t**
- typedef struct **kw_array_t** **kw_array_t**

Enumerations

- enum **kwapi_apitypes_t** {
  KWAPI_NONE = 0x0, **KWAPI_TREE** = 0x1, **KWAPI_PATH** = 0x2, **KWAPI_PATTERN** = 0x4,
  KWAPI_PREP = 0x8 }
- enum **kwapi_langtypes_t** { **KWAPI_NOLANG**, **KWAPI_JAVA**, **KWAPI_CXX**, **KWAPI_CSHARP** }

Functions

- **kwapi_cfgparam_t** **kwapi_cfgparam_getRootParameterList** (const char *error)
- **kwapi_cfgparam_t** **kwapi_cfgparam_getListNodeByName** (**kwapi_cfgparam_t**, const char *name)
- const char * **kwapi_cfgparam_getName** (**kwapi_cfgparam_t**)
- const char * **kwapi_cfgparam_getType** (**kwapi_cfgparam_t**)
- unsigned int **kwapi_cfgparam_getListLength** (**kwapi_cfgparam_t**)
- **kwapi_cfgparam_t** **kwapi_cfgparam_getListNodeByIndex** (**kwapi_cfgparam_t**, unsigned int idx)
- int **kwapi_cfgparam_isParameter** (**kwapi_cfgparam_t**)
- const char * **kwapi_cfgparam_getParameterValue** (**kwapi_cfgparam_t**)
- const char * **kwapi_cfgparam_getParameterValueFromList** (**kwapi_cfgparam_t** parent, const char *paramName)
- const char * **kwapi_cfgparam_getConfigurationParameter** (const char *errorId, const char *paramName)
- const char *const * **kwapi_cfgparam_getCheckerErrors** (const char *checker_id)
- const char * **ktc_error_getConfigurationParameter** (const char *errorId, const char *paramName)
- int **kwapi_cfgparam_errorIsEnabled** (const char *error_id)
- **kw_size_t** **kw_array_size** (**kw_array_t** *array)
- void * **kw_array_get** (**kw_array_t** *array, **kw_size_t** index)
- void **kw_array_delete** (**kw_array_t** *array)

5.4.1 Macro Definition Documentation

5.4.1.1 #define KWAPI_DECLARE(*type*) type

5.4.1.2 #define KWAPI_DECLARE_CPP(*type*) type

5.4.1.3 #define KWAPI_DECLARE_DATA

5.4.1.4 #define KWAPI_DECLARE_NONSTD(*type*) type

5.4.2 Typedef Documentation

5.4.2.1 typedef struct kw_array_t kw_array_t

5.4.2.2 typedef unsigned int kw_size_t

5.4.3 Enumeration Type Documentation

5.4.3.1 enum kwapi_apitypes_t

Constants for Klocwork APIs

Enumerator

KWAPI_NONE
KWAPI_TREE
KWAPI_PATH
KWAPI_PATTERN
KWAPI_PREP

5.4.3.2 enum kwapi_langtypes_t

Constants for Klocwork supported languages

Enumerator

KWAPI_NOLANG
KWAPI_JAVA
KWAPI_CXX
KWAPI_CSHARP

5.4.4 Function Documentation

5.4.4.1 void kw_array_delete (kw_array_t * *array*)

5.4.4.2 void* kw_array_get (kw_array_t * *array,* kw_size_t *index*)

5.4.4.3 kw_size_t kw_array_size (kw_array_t * *array*)

Index