TITLE

AUTHOR
Version 1
CREATEDATE
Table of Contents

Table of contents
PDT Ductape API Hierarchical Index

PDT Ductape API Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:
PDB .................................................. 2
pdbBounds ........................................................................................................... 8
pdbComment ......................................................................................................... 12
pdbEnum ............................................................................................................... 14
pdbLoc .................................................................................................................... 29
  pdbArg ................................................................................................................. 5
  pdbBase ................................................................................................................. 6
  pdbCallee ............................................................................................................. 9
pdbSimpleItem ...................................................................................................... 35
pdbItem ................................................................................................................... 22
  pdbFatItem .......................................................................................................... 15
    pdbNamespace .................................................................................................... 33
  pdbTemplate ....................................................................................................... 38
  pdbTemplateItem ............................................................................................... 42
    pdbGroup ......................................................................................................... 17
      pdbClass ...................................................................................................... 10
      pdbModule ..................................................................................................... 31
    pdbGroupField ............................................................................................... 20
  pdbType .............................................................................................................. 44
pdbTemplateArg ................................................................................................. 40
PDT Ductape API Data Structure Index

PDT Ductape API Data Structures

Here are the data structures with brief descriptions:

- **PDB** (A class to control the reading and writing of pdb files) .......................................................... 2
- **pdbArg** (A class for arguments for routines) .......................................................................................... 5
- **pdbBase** (A class to define a super class) ............................................................................................. 6
- **pdbBounds** (A class to represent Fortran bounds) ............................................................................... 8
- **pdbCallee** (A class to represent a routine call) .................................................................................. 9
- **pdbClass** (This class defines the attributes for a class) ..................................................................... 10
- **pdbComment** ....................................................................................................................................... 12
- **pdbEnum** (A class to represent C/C++ enumerations) ....................................................................... 14
- **pdbFatItem** (A class for items spanning several lines of code) ......................................................... 15
- **pdbGroup** (PdbGroups representing abstract data types) ................................................................. 17
- **pdbGroupField** (A class to define field within a group) .................................................................... 20
- **pdbItem** (An item class with more complex members) ..................................................................... 22
- **pdbLoc** (Class that stores the location of pdbItems in the source code) .......................................... 29
- **pdbModule** (A class to define modules) ............................................................................................. 31
- **pdbNamespace** (A class to define the namespace) .............................................................................. 33
- **pdbSimpleItem** (The Root class is the pdb hierarchy) ........................................................................ 35
- **pdbTemplate** (Template Items class) .................................................................................................. 38
- **pdbTemplateArg** (A class to define argument in a template definitions) .......................................... 40
- **pdbTemplateItem** (A class to record templates) .................................................................................. 42
- **pdbType** (A class to contain the abstract Type information) .............................................................. 44
PDT Ductape API Data Structure Documentation

PDB Class Reference

A class to control the reading and writing of pdb files.

```cpp
#include <pdb.h>
```

Public Types

- `enum lang_t`

Public Member Functions

- `PDB (char *fname)`
- `typevec & getTypeVec()`
- `filevec & getFieVec()`
- `classvec & getClassVec()`
- `modulevec & getModuleVec()`
- `croutinevec & getCRoutineVec()`
- `froutinevec & getFRoutineVec()`
- `templatevec & getTemplateVec()`
- `macrovec & getMacroVec()`
- `pragmavec & getPragmaVec()`
- `namespacevec & getNamespaceVec()`
- `itemvec & getItemVec()`

Data Structures

- `class classTag`
- `class croutineTag`
- `class fileTag`
- `class froutineTag`
- `struct ltstr`
- `class macroTag`
- `class moduleTag`
- `class namespaceTag`
- `class pragmaTag`
- `class templateTag`
- `class typeTag`

Detailed Description

A class to control the reading and writing of pdb files.

In addition, there is a class PDB that represents an entire PDB file. It provides methods to read, write, and merge PDB files, to get the version of the PDB file format and the programming language it got generated from.

Examples:
Member Enumeration Documentation

enum PDB::lang_t
    the language of the source files.

Constructor & Destructor Documentation

PDB::PDB (char * fname)
    A PDB class constructor
    Parameters:
        *fname the name of the source file.

Member Function Documentation

PDB::classvec & PDB::getClassVec () [inline]
    a vector of the classes within the pdb.

PDB::croutinevec & PDB::getCRoutineVec () [inline]
    a vector of the c/c++ routines within the pdb.

PDB::filevec & PDB::getFileVec () [inline]
    a vector of the files within the pdb.

PDB::froutinevec & PDB::getFRoutineVec () [inline]
    a vector of the fortran routines within the pdb.

PDB::itemvec & PDB::getItemVec () [inline]
    a vector of the items within the pdb.

PDB::macrovec & PDB::getMacroVec () [inline]
    a vector of the macros within the pdb.

PDB::modulevec & PDB::getModuleVec () [inline]
    a vector of the modules within the pdb.

PDB::namespacevec & PDB::getNamespaceVec () [inline]
    a vector of the namespaces within the pdb.
PDB::pragmavec & PDB::getPragmaVec () [inline]
a vector of the pragmas within the pdb.

PDB::templatevec & PDB::getTemplateVec () [inline]
a vector of the templates within the pdb.

PDB::typevec & PDB::getTypeVec () [inline]
a vector of the types within the pdb.

The documentation for this class was generated from the following files:
- pdb.h
- pdb.inl
- pdbTdefs.h
pdbArg Class Reference
A class for arguments for routines.
#include <pdbType.h>

Inheritance diagram for pdbArg:

```
+-----------------------
<table>
<thead>
<tr>
<th>pdbLoc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
|    +------+
|    |      |
|    |      |
|    |      |
|    |      +-----------------------
|    |      |       pdbArg               |
|    |      +-----------------------
|    |      |                +--------
|    |      |                |     |
|    |      |                |     |
|    +------+
```

Public Member Functions
- const pdbType * type () const
- const string & name () const
- bool hasDefault () const

Detailed Description
A class for arguments for routines.

This class describes arguments given to pbdRoutines. It hold information about these arguments and how they are given to Routines as parameters.

Member Function Documentation

```
bool pdbArg::hasDefault () const  [inline]
    if the argument of this routine has a default parameter for this argument.

const string & pdbArg::name () const  [inline]
    the name of the argument.

const pdbType * pdbArg::type () const  [inline]
    the abstract type of the argument.
```

The documentation for this class was generated from the following files:
- pdbType.h
- pdbType.inl
pdbBase Class Reference

A class to define a super class.
#include <pdbClass.h>

Inheritance diagram for pdbBase:

```
pdbLoc
   
pdbBase
```

Public Member Functions

- `pdbBase()`
- `pdbBase(pdbItem::virt_t v, pdbItem::access_t a, const pdbClass *base, pdbFile *file, int line, int col)`
- `pdbItem::virt_t virtuality() const`
- `pdbItem::access_t access() const`
- `const pdbClass * base() const`
- `bool isVirtual() const`

Detailed Description

A class to define a super class.

pdbBase describes a base class (super class) of a pdbClass. It provides methods to ask for the base class, its access mode (e.g., public or private), and whether the derivation was virtual.

Constructor & Destructor Documentation

```
const pdbBase::pdbBase () [inline]
   
   A constructor with no arguments
```

```
const pdbBase::pdbBase (pdbItem::virt_t v, pdbItem::access_t a, const pdbClass * base, pdbFile * file, int line, int col) [inline]
   
   A constructor

Parameters:
   - `v` the virtual type of this class.
   - `a` the access type of this class.
   - `*base` a pointer to the base class.
   - `*file` a pointer to the source file.
   - `line` the line number where this class began.
```
Member Function Documentation

**pdbItem::access_t pdbBase::access () const [inline]**
the access type of this class

**const pdbClass * pdbBase::base () const [inline]**
a pointer to the base class

**bool pdbBase::isVirtual () const [inline]**
is this class derivation virtual?

**pdbItem::virt_t pdbBase::virtuality () const [inline]**
the virtual type of this class

The documentation for this class was generated from the following files:
- pdbClass.h
- pdbClass.inl
pdbBounds Class Reference
A class to represent Fortran bounds.
#include <pdbType.h>

Public Member Functions
• pdbBounds (int low, int upp)
• int lower () const
• int upper () const

Detailed Description
A class to represent Fortran bounds.

pdbBounds is used to describe the bounds of one dimension of a Fortran array.

Constructor & Destructor Documentation

pdbBounds::pdbBounds (int low, int upp) [inline]
A constructor

Parameters:
low lower bound.
upp upper bound.

Member Function Documentation

int pdbBounds::lower () const [inline]
lower bound.

int pdbBounds::upper () const [inline]
upper bound.

The documentation for this class was generated from the following files:
• pdbType.h
• pdbType.inl
pdbCallee Class Reference
A class to represent a routine call.
#include <pdbRoutine.h>

Inheritance diagram for pdbCallee:

```
pdbLoc
    
    pdbCallee
```

Public Member Functions
- const pdbRoutine * call () const
- bool isVirtual () const

Detailed Description
A class to represent a routine call.

pdbCallee is used to represent a routine call (i.e., a call site). Attributes are the routine called, whether it is was called virtually, and the location of the call site.

Member Function Documentation

```cpp
class pdbCallee
{
public:

    const pdbRoutine * call () const [inline]
        A pointer to the routine called.

    bool isVirtual () const [inline]
        Is this a virtual call.
}
```

The documentation for this class was generated from the following files:
- pdbRoutine.h
- pdbRoutine.inl
**pdbClass Class Reference**

This class defines the attributes for a class.

```cpp
#include <pdbClass.h>
```

Inheritance diagram for pdbClass:

![Inheritance Diagram for pdbClass](image)

### Public Member Functions

- `const classvec & derivedClasses () const`
- `const methodvec & methods () const`
- `const friendclassvec & friendClasses () const`
- `const friendfuncvec & friendRoutines () const`

### Detailed Description

This class defines the attributes for a class.

This class defines the generic class within a class hierarchy, and its associated friend classes.
Member Function Documentation

const pdbClass::classvec & pdbClass::derivedClasses () const [inline]
   a vector listing the classes derived from this one.

const pdbClass::friendclassvec & pdbClass::friendClasses () const [inline]
   a vector of friendly classes.

const pdbClass::friendfuncvec & pdbClass::friendRoutines () const [inline]
   a vector of friendly Routines.

const pdbClass::methodvec & pdbClass::methods () const [inline]
   a vector of methods.

The documentation for this class was generated from the following files:
   • pdbClass.h
   • pdbClass.inl
pdbComment Class Reference

#include <pdbFile.h>

Public Member Functions

- pdbComment (int id)
- PDB::lang_t kind () const
- const pdbLoc & cmtBegin () const
- const pdbLoc & cmtEnd () const
- const string & text () const

Detailed Description

A class to represent comment in the source file.
pdbComment represents a comment in a source file. Comments are numbered 0 to N inside one file. The kind, its exact location, and the comment text is available.

Constructor & Destructor Documentation

pdbComment::pdbComment (int id) [inline]

A constructor

Parameters:

- id: unique identifier.

Member Function Documentation

const pdbLoc & pdbComment::cmtBegin () const [inline]

location in the source file where the comment began.

const pdbLoc & pdbComment::cmtEnd () const [inline]

location in the source file where the comment ends.

PDB::lang_t pdbComment::kind () const [inline]

returns the language of this source file.

const string & pdbComment::text () const [inline]

the contents of the comment.

The documentation for this class was generated from the following files:

- pdbFile.h
• pdbFile.inl
pdbEnum Class Reference
A class to represent C/C++ enumerations.
#include <pdbType.h>

Public Member Functions
• pdbEnum (const char *id, int val)

Detailed Description
A class to represent C/C++ enumerations.

pdbEnum describes one element of an C / C++ enumeration type by its name (identifier) and the corresponding integer value.

Constructor & Destructor Documentation

pdbEnum::pdbEnum (const char * id, int val) [inline]
A constructor

Parameters:
*id unique identifier.
val integer value of the enumeration.

The documentation for this class was generated from the following files:
• pdbType.h
• pdbType.inl
**pdbFatItem Class Reference**

A class for items spanning several lines of code.

```
#include <pdbFatItem.h>
```

Inheritance diagram for pdbFatItem:

```
pdbSimpleItem  
   |   |
   |   v
pdbItem
   |   |
   |   v
pdbFatItem
   |   |   |
   |   |   v
pdbNamespace pdbTemplate pdbTemplateItem
   |   |   |   |   |   |
   |   |   |   |   |   v
pdbGroup pdbGroupField
   |   |   |
   |   |   v
pdbClass pdbModule
```

### Public Member Functions

- `pdbFatItem (int id)`
- `pdbFatItem (const string &name, int id)`
- `const pdbLoc &headBegin () const`
- `const pdbLoc &headEnd () const`
- `const pdbLoc &bodyBegin () const`
- `const pdbLoc &bodyEnd () const`

### Detailed Description

A class for items spanning several lines of code.

pdbItems are pdbPragmas, pdbMacros, pdbTypes, or so-called fat items. pdbFatItems have a header and a body, and attributes describing the source location of these parts.
Constructor & Destructor Documentation

pdbFatItem::pdbFatItem (int id) [inline]

pdbFatItem constructor

Parameters:
id an unique identifier.

pdbFatItem::pdbFatItem (const string & name, int id) [inline]

pdbFatItem constructor

Parameters:
name the name of the item.
id an unique identifier.

Member Function Documentation

const pdbLoc & pdbFatItem::bodyBegin () const [inline]

the line number in the source code that begins the body.

const pdbLoc & pdbFatItem::bodyEnd () const [inline]

the line number in the source code that ends the body.

const pdbLoc & pdbFatItem::headBegin () const [inline]

the line number in the source code that begins the header.

const pdbLoc & pdbFatItem::headEnd () const [inline]

the line number in the source code that ends the header.

The documentation for this class was generated from the following files:

- pdbFatItem.h
- pdbFatItem.inl
pdbGroup Class Reference

pdbGroups represent abstract data types.

#include <pdbGroup.h>

Inheritance diagram for pdbGroup:

Public Member Functions

- **pdbGroup** (int id)
  
  *pdbGroup constructor.*

- **pdbGroup** (const string &name, int id)
  
  *pdbGroup constructor.*

- const fieldvec & **dataMembers** () const
- **group_t kind** () const
  
  *the type of this group.*
Detailed Description

pdbGroups represent abstract data types.

Groups represent abstract data types, i.e. collections of public and private members. Members are divided into data members (described by pdbGroupFields) and member functions/methods (described by pdbRoutines). The different kinds of groups are Fortran 90 derived types or modules, or C and C++ structs, unions, or classes.

Constructor & Destructor Documentation

pdbGroup::pdbGroup (int id) [inline]

pdbGroup constructor.

Parameters:

- id an unique identifier.

pdbGroup::pdbGroup (const string & name, int id) [inline]

pdbGroup constructor.

Parameters:

- name the name of the item.
- id an unique identifier.

Member Function Documentation

const pdbGroup::fieldvec & pdbGroup::dataMembers () const [inline]

A list of members of this group.

pdbItem::group_t pdbGroup::kind () const [inline]

the type of this group.

kind() specifies the abstract type of this group whether Fortran 90 derived types or modules, C/C++ structs, unions, or classes.

The documentation for this class was generated from the following files:

- pdbGroup.h
- pdbGroup.inl
pdbGroupField Class Reference
A class to define field within a group.
#include <pdbGroupField.h>
Inheritance diagram for pdbGroupField:

Public Member Functions
• mem_t kind () const
• const pdbType * type () const
• bool isBitField () const
• bool isMutable () const
• bool isStaticConst () const

Detailed Description
A class to define field within a group.

Member Function Documentation

bool pdbGroupField::isBitField () const [inline]
   Is this a bit field?

bool pdbGroupField::isMutable () const [inline]
   Is this field mutable?
bool pdbGroupField::isStaticConst () const  [inline]
    Is this a static constant

pdbItem::mem_t pdbGroupField::kind () const  [inline]
    the memory type of this field

const pdbType * pdbGroupField::type () const  [inline]
    the type of this field

The documentation for this class was generated from the following files:
  •  pdbGroupField.h
  •  pdbGroupField.inl
pdbItem Class Reference

An item class with more complex members.

#include <pdbItem.h>

Inheritance diagram for pdbItem:

```
<table>
<thead>
<tr>
<th>pdbSimpleItem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>pdbItem</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>pdbFatItem</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>pdbNamespace</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>pdbGroup</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>pdbClass</td>
</tr>
</tbody>
</table>
```

Public Types

- enum access_t { AC_NA, AC_PRIV, AC_PROT, AC_PUB }
- enum routine_t { RO_NA, RO_EXT, RO_TPROTO, RO_FEXT, RO_FPROG, RO_FBLDAT, RO_FINTRIN, RO_FINT, RO_FSTFN, RO_FMPROC, RO_FUNSPEC, RO_FALIAS }
- enum rspec_t
- enum templ_t { , TE_CLASS, TE_FUNC, TE_MEMCLASS, TE_MEMFUNC }
- enum float_t { FL_NA, FL_FLOAT, FL_DBL, FL_LONGBL }
- enum int_t { I_NA, I_CHAR, I_SCHAR, I_UCHAR, I_SHORT, I_USHORT, I_INT, I_UINT, I_LONG, I_ULONG, I_LONGLONG, I_ULONGLONG }
- enum type_t
- enum group_t { , GR_CLASS, GR_STRUCT, GR_UNION, GR_TPROTO, GR_FDERIVED, GR_FMODULE }
- enum link_t { LK_NA, LK_INTERNAL, LK_CXX, LK_C, LK_FINT, LK_F90 }
- enum shape_t { SH_NA, SH_EXPLICIT, SH_ASIZE, SH_ASHAPE, SH_DEFERRED }
- enum qual_t { QL_NA, QL_CONST, QL_VOLATILE, QL_RESTRICT }

Public Member Functions

- pdbItem (int id)
  
  * pdbItem constructor.
• **pdbItem** (const string &name, int id)
  *pdbItem constructor.*

• const string & **fullName** () const
  *the full name of the item.*

• **access_t** access () const
  *access mode for this item.*

• const **pdbGroup** *parentGroup () const
  *the groups this item is a member of.*

• const **pdbLoc** & location () const
  *the location of this item in the source file.*

• const **pdbNamespace** *parentNSpace () const
  *the name space this item is in.*

---

**Detailed Description**

An item class with more complex members.

Derived from pdbSimpleItems are pdbFiles and more complex pdbItems, which have a source code location, possibly a parent group or namespace, and an access mode (e.g., public or private) if they are member of a group. The method fullname() returns fully-qualified names (including signatures for routines).

---

**Member Enumeration Documentation**

`enum pdbItem::access_t`

defines the types of access modifiers for template items.
Enumerator:

\texttt{AC\_NA} default

\texttt{AC\_PRIV} private

\texttt{AC\_PROT} protected

\texttt{AC\_PUB} public

\texttt{enum pdbItem::float\_t}
defines the types of floating point numbers.

Enumerator:

\texttt{FL\_NA} not applicable

\texttt{FL\_FLOAT} float type

\texttt{FL\_DBL} double type

\texttt{FL\_LONGDBL} long double type

\texttt{enum pdbItem::group\_t}
defines the types of groups.

Enumerator:

\texttt{GR\_CLASS} class group

\texttt{GR\_STRUCT} structure group

\texttt{GR\_UNION} union group

\texttt{GR\_PROTO} template prototype group

\texttt{GR\_FDERIVED} Fortran derived group

\texttt{GR\_FMODULE} Fortran module group

\texttt{enum pdbItem::int\_t}
defines the types of integer point numbers.
enumerator:

\texttt{I\_NA} not applicable

\texttt{I\_CHAR} character

\texttt{I\_SCHAR} signed character

\texttt{I\_UCHAR} unsigned character

\texttt{I\_SHORT} short

\texttt{I\_USHORT} unsigned short

\texttt{I\_INT} integer

\texttt{I\_UINT} unsigned integer

\texttt{I\_LONG} long

\texttt{I\_ULONG} unsigned long

\texttt{I\_LONGLONG} long long

\texttt{I\_ULONGLONG} unsigned long long

\texttt{enum pdbItem::link\_t}

enumerator:

\texttt{LK\_NA} link not defined

\texttt{LK\_INTERNAL} internal linke

\texttt{LK\_CXX} c++ link

\texttt{LK\_C} c link

\texttt{LK\_FINT} Fortran link

\texttt{LK\_F90} Fortran 90 link

\texttt{enum pdbItem::qual\_t}

enumerator:
**QL NA** unqualified type

**QL CONST** constant type

**QL VOLATILE** volatile type

**QL RESTRICT** restricted type

```cpp
eenum pdbItem::routine_t
    defines the types of routine signatures
```

**Enumerator:**

**RO NA** default routine

**RO EXT** external routines, created by the compiler not explicitly written in the source code

**RO TPROTO** templete routine

**RO FEXT** Fortran external routine

**RO FPROG** Fortran program routine

**RO FBLDAT** Fortran block data

**RO FINTRIN** Fortran intrinsic

**RO FINT** Fortran internal

**RO FSTFN** Fortran statement function

**RO FMPROC** Fortran module procedure

**RO FUNSPEC** Fortran unspecified

**RO FALIAS** Fortran alias

```cpp
eenum pdbItem::rspec_t
    the types of special routines.
```

```cpp
eenum pdbItem::shape_t
    defines the different shapes of fortran arrays.
```
Enumerator:

`SH_NA` initialized value. (for debugging purposes).

`SH_EXPLICIT` set when the rank and extent are defined explicitly.

`SH_ASIZE` set when the extent of one or more dimension is undefined.

`SH_ASHAPE` set when the rank of an array is left undefined.

`SH_DEFERRED` set when an array is allocated but undefined.

```cpp
enum pdbItem::templ_t
```

Enumerator:

`TE_CLASS` template classes

`TE_FUNC` template function

`TE_MEMCLASS` classes that are members of a template class

`TE_MEMFUNC` functions that are members of a template class

```cpp
enum pdbItem::type_t
```

defines other types of primitives.

---

### Constructor & Destructor Documentation

`pdbItem::pdbItem (int id) [inline]`

pdbItem constructor.

**Parameters:**

`id` an unique identifier.

`pdbItem::pdbItem (const string & name, int id) [inline]`

pdbItem constructor.

**Parameters:**

`name` the name of the item.

`id` an unique identifier.
Member Function Documentation

```cpp
const string & pdbItem::fullName () const [inline]
```

the full name of the item.

The full name contains the full signatures for templates.

The documentation for this class was generated from the following files:

- pdbItem.h
- pdbItem.inl
## pdbLoc Class Reference

class that store the location of pdbItems in the source code.

```cpp
#include <pdbSimpleItem.h>
```

Inheritance diagram for pdbLoc:

```
    pdbLoc
   /     \
  /       \
pdbArg -- pdbBase -- pdbCallee
```

### Public Member Functions

- `pdbLoc ()`
- `pdbLoc (const pdbFile *file, int line, int col)`
- `const pdbFile *file () const`
- `int line () const`
- `int col () const`

### Detailed Description

class that store the location of pdbItems in the source code.

pdbLoc describes source code locations which are characterized by a source file, a line number (starting with 1), and a character position within this line (starting with 0).

### Constructor & Destructor Documentation

#### pdbLoc::pdbLoc () [inline]

A constructor without any argument

#### pdbLoc::pdbLoc (const pdbFile * file, int line, int col) [inline]

A constructor

**Parameters:**

- `*file` pointer to the pdbFile.
- `line` the line number of this location in the source file.
- `col` the column number of this location in the source file.
**Member Function Documentation**

```cpp
int pdbLoc::col () const [inline]
   line number in source code.

const pdbFile * pdbLoc::file () const [inline]
   pointer of a pdbFile.

int pdbLoc::line () const [inline]
   line number in source code.
```

The documentation for this class was generated from the following files:

- pdbSimpleItem.h
- pdbSimpleItem.inl
**pdbModule Class Reference**

A class to define modules.

```cpp
#include <pdbModule.h>
```

Inheritance diagram for pdbModule:

```
pdbSimpleItem

pdbItem

pdbFatItem

pdbTemplateItem

pdbGroup

pdbModule
```

**Public Member Functions**

- `pdbModule (int id)`
- `pdbModule (const string &name, int id)`

**Detailed Description**

A class to define modules.

pdbModule hold information about module and their functions.
Constructor & Destructor Documentation

pdbModule::pdbModule (int id)  [inline]
     pdbModule constructor.
     
     Parameters:
     id unique identifier.

pdbModule::pdbModule (const string & name, int id)  [inline]
     pdbModule constructor.
     
     Parameters:
     name the name of this module.
     id unique identifier.

The documentation for this class was generated from the following files:

- pdbModule.h
- pdbModule.inl
pdbNamespace Class Reference

A class to define the namespace.

#include <pdbNamespace.h>

Inheritance diagram for pdbNamespace:

```
    pdbSimpleItem
       \______________________
          |                   |
          |                   |
          |                   |
          |   pdbItem        |
          |__________________|
          |                   |
          |                   |
          |                   |
          |   pdbFatItem     |
          |__________________|
          |                   |
          |                   |
          |                   |
          |   pdbNamespace   |
```

Public Member Functions

- pdbNamespace (int id)
- pdbNamespace (const string &name, int id)
- const memvec & members () const
- const pdbNamespace * isAlias () const

Detailed Description

A class to define the namespace.

This class records the members of each namespace.

Constructor & Destructor Documentation

pdbNamespace::pdbNamespace (int id) [inline]

pdbNamespace constructor

Parameters:

id unique idenfier.
pdbNamespace::pdbNamespace (const string & name, int id) [inline]
    pdbNamespace constructor

    Parameters:
    name this namespace's name.
    id unique identifier.

Member Function Documentation

const pdbNamespace * pdbNamespace::isAlias () const [inline]
    Pointer to alias namespace.

const pdbNamespace::memvec & pdbNamespace::members () const [inline]
    A vector of members of this namespace.

The documentation for this class was generated from the following files:
  • pdbNamespace.h
  • pdbNamespace.inl
**pdbSimpleItem Class Reference**

The Root class is the pdb hierarchy.

```cpp
#include <pdbSimpleItem.h>
```

Inheritance diagram for pdbSimpleItem:

![Inheritance Diagram](image)

**Public Member Functions**

- **pdbSimpleItem** (int id)
  
  *pdbSimpleItem constructor.*

- **pdbSimpleItem** (const string &name, int id)
  
  *pdbSimpleItem constructor.*

- const string & **name** () const
  
  *Item’s Name.*

- int **id** () const
  
  *Unique ID.*
Detailed Description
The Root class is the pdb hierarchy.

The root class of the hierarchy is pdbSimpleItem. pdbSimpleItems, and therefore all items derived from it, have two attributes, their name and PDB ID.

Constructor & Destructor Documentation

pdbSimpleItem::pdbSimpleItem (int id) [inline]
    pdbSimpleItem constructor.

Parameters:
    id an unique identifier.

pdbSimpleItem::pdbSimpleItem (const string & name, int id) [inline]
    pdbSimpleItem constructor.

Parameters:
    name the name of this item.
    id an unique identifier.

Member Function Documentation

int pdbSimpleItem::id () const [inline]
    Unique ID.

    Every pdb Item has a unique identifier.

const string & pdbSimpleItem::name () const [inline]
    Item's Name.

    String to hold the Item's name.
The documentation for this class was generated from the following files:

- pdbSimpleItem.h
- pdbSimpleItem.inl
pdbTemplate Class Reference

Template Items class.

#include <pdbTemplate.h>

Inheritance diagram for pdbTemplate:

Public Member Functions

- pdbTemplate (int id)
- pdbTemplate (const string &name, int id)
- templ_t kind () const
- const targvec & arguments () const
- const targvec & specArguments () const

Detailed Description

Template Items class.

pdbFatItems include pdbTemplates, pdbNamespaces, and pdbTemplateItems. pdbTemplateItems are entities that can be instantiated from templates. Template items are pdbGroups, pdbGroupFields, pdbRoutines.

Constructor & Destructor Documentation

pdbTemplate::pdbTemplate (int id) [inline]

pdbTemplate constructor

Parameters:

id an unique idenifier.
pdbTemplate::pdbTemplate (const string & name, int id) [inline]
    pdbTemplate constructor

    Parameters:
    name the name of the template.
    id an unique identifier.

Member Function Documentation

const pdbTemplate::targvec & pdbTemplate::arguments () const [inline]
    the argument for this template.

pdbTemplate::templ_t pdbTemplate::kind () const [inline]
    the type of this template.

const pdbTemplate::targvec & pdbTemplate::specIArguments () const [inline]
    a vector containing the argument for a routine.

The documentation for this class was generated from the following files:
    • pdbTemplate.h
    • pdbTemplate.inl
pdbTemplateArg Class Reference
A class to define argument in a template definitions.
#include <pdbTemplateArg.h>

Public Types
• enum targ_t { TA_NA, TA_TYPE, TA_NONTYPE, TA_TEMPL }

Public Member Functions
• pdbTemplateArg (targ_t kind, bool specialization=false)
• const string & name () const
• const pdbType * type () const

Detailed Description
A class to define argument in a template definitions.

pdbTemplateArg describes arguments in template definitions and specializations.

Member Enumeration Documentation

enum pdbTemplateArg::targ_t
this enumeration tell the type of template argument Depending on the kind of the template argument, different methods are applicable.

Enumerator:
 TA_NA uninitialized

 TA_TYPE types (type() and defaultType()).

 TA_NONTYPE non types (type(), name() and defaultValue()).

 TA_TEMPL templates (templateArg() and defaultTemplateArg())

Constructor & Destructor Documentation

pdbTemplateArg::pdbTemplateArg (targ_t kind, bool specialization = false) [inline]
A constructor
Parameters:

kind the kind of template argument.
specialization wheather the arguements are specialized.

Member Function Documentation

const string & pdbTemplateArg::name () const [inline]
name of the template arguemnt.

const pdbType * pdbTemplateArg::type () const [inline]
the pointer to the type of the arguement.

The documentation for this class was generated from the following files:

- pdbTemplateArg.h
- pdbTemplateArg.inl
pdbTemplateItem Class Reference

A class to record templates.

#include <pdbTemplateItem.h>

Inheritance diagram for pdbTemplateItem:

```
  pdbSimpleItem
    
    pdbItem
      
      pdbFatItem
        
        pdbTemplateItem
          
          pdbGroup
          pdbGroupField
            
            pdbClass
            pdbModule
```

Public Member Functions

- const pdbTemplate * isTemplate () const
- bool isSpecialized () const
- const targvec & specArguments () const

Detailed Description

A class to record templates.

pdbTemplateItems are entities that can be instantiated from templates. Template items are pdbGroups, pdbGroupFields, pdbRoutines.
Member Function Documentation

bool pdbTemplateItem::isSpecialized () const [inline]
    Is this item Specialized?

const pdbTemplate * pdbTemplateItem::isTemplate () const [inline]
    pointer to the pdbTemplateItem

const pdbTemplateItem::targvec & pdbTemplateItem::speclArguments () const [inline]
    speclized Arguments

The documentation for this class was generated from the following files:
- pdbTemplateItem.h
- pdbTemplateItem.inl
pdbType Class Reference

A class to contain the abstract Type information.

#include <pdbType.h>

Inheritance diagram for pdbType:

Public Member Functions

- const pdbType * elementType () const
- float_t floatType () const
- int_t integerType () const
- type_t kind () const
- const typevec & exceptionSpec () const

Detailed Description

A class to contain the abstract Type information.

This class hold the information about Fortran and C/C++ types both abstract and basic types.

Member Function Documentation

const pdbType * pdbType::elementType () const [inline]
the abstract type of this argument.

const pdbType::typevec & pdbType::exceptionSpec () const [inline]
for C arrays and f90 characters

pdbItem::float_t pdbType::floatType () const [inline]
the float type of this argument.
\texttt{pdbItem::int\_t pdbType::integerType () const \ [inline]}

the integer type of this argument.

\texttt{pdbItem::type\_t pdbType::kind () const \ [inline]}

the kind of the type.

The documentation for this class was generated from the following files:

- pdbType.h
- pdbType.inl
PDT Ductape API Example Documentation

froutine.cc

Routines describes the common part of global functions, Fortran 90 local and module functions, and C++ class methods. The common attributes are signature, kind (e.g., extern or intrinsic), specialKind (e.g., constructor or operator), a list of routines called from this routine, how often it gets called from other routines, linkage, for C and C++ the statement representing the body and a list of all statements, and for C and Fortran routines the location of the first executable statement and of all return statements.

```cpp
#include "pdbAll.h"
#include "stdio.h"
#include <typeinfo>

int main(int argc, char *argv[]) {
    PDB p(argv[1]); if ( !p ) return 1;

    for (PDB::croutinevec::iterator r = p.getCRoutineVec().begin();
        r!=p.getCRoutineVec().end(); r++)
        {
            cout << (*r)->name() << " ";
            cout << (*r)->specialKind() << endl;
        }
    return 0;
}
```
stmt.cc

This class stores information about the statements (or block of statements) within a source file. This class also keeps track of the next statement within their own context as well as the next statement in child contexts.

```cpp
#include "pdbAll.h"
#include "stdio.h"
#include <typeinfo>

int main(int argc, char *argv[]) {
    PDB p(argv[1]); if ( !p ) return 1;

    // Iterate though the C Routines
    for (PDB::croutinevec::iterator r = p.getCRoutineVec().begin();
         r!=p.getCRoutineVec().end(); r++)
    {
        // Retrive the statement within the routines.
        cout << (**r).signature().name() << endl;
        if ((**r)->kind() == 0)
        {
            const pdbStmt *v = (**r).body();
            cout << typeid(*v).name() << endl;
            // Print out the beginning and ending locations of the
            statement block.
            cout << "statement begins: " << v->stmtBegin() << endl;
            cout << "statement ends: " << v->stmtEnd() << endl;
        }
    }
    return 0;
}
```
vector.cc

An example to show how to iterate through the elements of a PDB file. classvec can be replaced with any other vector type.

/* File to be parsed and analyzed:
   class bar
   |
   |   int foo(int v)
   |   { return v + 2; }
   | class bar2
   |   { int routine(bool t) {return 0;} }
   |
   |
   |
   |
   |
   |
   |
   |
   |
   |
   |
   |
   | To run type:
   | %> g++ -I../inc/ -o vector vector.cc ../lib/libpdb.a
   | %> cxxparse testApp.cc
   | %> ./vector testApp.pdb
   | bar
   | bar2
   */
#include "pdbAll.h"
#include "stdio.h"
int main(int argc, char *argv[]) {
   // Read the pdb file as input for this program.
   PDB p(argv[1]); if ( !p ) return 1;
   // Iterate through each class in the pdb file a print its name.
   for (PDB::classvec::iterator r = p.getClassVec().begin();
    r!=p.getClassVec().end(); r++)
   {
      cout << (*r)->name() << endl;
   }
   return 0;
}