Rec. 08. Declarations and Initialization (DCL)

Contents

- Android Only
- C
- C++
- Java

Android Only

Content by label

There is no content with the specified labels

C

- DCL00-C. Const-qualify immutable objects
- DCL01-C. Do not reuse variable names in subscopes
- DCL02-C. Use visually distinct identifiers
- DCL03-C. Use a static assertion to test the value of a constant expression
- DCL04-C. Do not declare more than one variable per declaration
- DCL05-C. Use typedefs of non-pointer types only
- DCL06-C. Use meaningful symbolic constants to represent literal values
- DCL07-C. Include the appropriate type information in function declarators
- DCL08-C. Properly encode relationships in constant definitions
- DCL09-C. Declare functions that return errno with a return type of errno_t
- DCL10-C. Maintain the contract between the writer and caller of variadic functions
- DCL11-C. Understand the type issues associated with variadic functions
- DCL12-C. Implement abstract data types using opaque types
- DCL13-C. Declare function parameters that are pointers to values not changed by the function as const
- DCL15-C. Declare file-scope objects or functions that do not need external linkage as static
- DCL16-C. Use "L," not "l," to indicate a long value
- DCL17-C. Beware of miscompiled volatile-qualified variables
- DCL18-C. Do not begin integer constants with 0 when specifying a decimal value
- DCL19-C. Minimize the scope of variables and functions
- DCL20-C. Explicitly specify void when a function accepts no arguments
- DCL21-C. Understand the storage of compound literals
- DCL22-C. Use volatile for data that cannot be cached
- DCL23-C. Guarantee that mutually visible identifiers are unique

C++

Content by label

There is no content with the specified labels

Java

- DCL50-J. Use visually distinct identifiers
- DCL51-J. Do not shadow or obscure identifiers in subscopes
- DCL52-J. Do not declare more than one variable per declaration
- DCL53-J. Minimize the scope of variables
- DCL54-J. Use meaningful symbolic constants to represent literal values in program logic
• DCL55-J. Properly encode relationships in constant definitions
• DCL56-J. Do not attach significance to the ordinal associated with an enum
• DCL57-J. Avoid ambiguous overloading of variable arity methods
• DCL58-J. Enable compile-time type checking of variable arity parameter types
• DCL59-J. Do not apply public final to constants whose value might change in later releases
• DCL60-J. Avoid cyclic dependencies between packages