PRE03-C. Prefer typedefs to defines for encoding non-pointer types

Prefer type definitions (`typedef`) to macro definitions (`#define`) when encoding types. Type definitions obey scope rules; macro definitions do not. textual substitution is inferior to using the type system. While type definitions for non-pointer types have similar advantages [Summit 2005], can make it more difficult to write `const`-correct code (see DCL05-C. Use typedefs of non-pointer types only).

Noncompliant Code Example

This noncompliant code example will not compile, because macros use textual substitution and not the type system:

```
#define MATRIX double matrix[4][4]
MATRIX matrix_a;
```

After preprocessing, this code example is translated to the following invalid declaration:

```
#define MATRIX double matrix[4][4]
double matrix[4][4] matrix_a;
```

Compliant Solution

Using type definitions instead of macro definitions in this compliant solution results in a valid declaration:

```
typedef double matrix[4][4];
matrix matrix_a;
```

Noncompliant Code Example

I don't actually know what is wrong with this:

```
#define uchar unsigned char
```

Compliant Solution

Use type definitions to encode all non-pointer types.

```
typedef unsigned char uchar;
```

Risk Assessment

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
PRE03-C	Low	Unlikely	Medium	P2	L3

Automated Detection

Tool	Version	Checker	Description
Axivion Bauhaus Suite	6.9.0	**CertC-PRE03**	
ECLAIR	1.2	**CC2.PRE03**	Fully implemented
LDRA tool suite	9.7.1	**79 S**	Enhanced Enforcement

PRQA QA-C	9.5	**3413**	Fully implemented

Related Vulnerabilities

Search for vulnerabilities resulting from the violation of this rule on the CERT website.

Related Guidelines

SEI CERT C++ Coding Standard	VOID PRE03-CPP. Prefer typedefs to defines for encoding types
ISO/IEC TR 24772:2013	Pre-processor Directives [NMP]

Bibliography

[Saks 1999]	
[Summit 2005]	Question 1.13 Question 11.11