rules.sonarsource.com

C++ static code analysis: Array type function arguments should not decay to pointers

2 minutes

When a variable with array type decays to a pointer, its bounds are lost.

If a design requires arrays of different lengths, then a class should be used to encapsulate the array objects and so ensure that the size is maintained. Note that in C++20, the class `std::span` is designed for this purpose, and you can find implementation of `span` that works with earlier versions of the standard.

Noncompliant Code Example

```
void f1( int p[ 10 ] ); // Non-compliant, function called with
arguments of array type
void f2( int *p ); // Non-compliant, function is called with arguments
of array type
void functionWorkingOnSingleInt(int * p); // Non-compliant, function
is called with arguments of array type

void b ()
{
  int a[ 10 ];
  f1( a ); // The size is lost
  f2( a ); // The size is lost
  functionWorkingOnSingleInt( a ); // Not clear that the function acts
on only one element
}
```

Compliant Solution

```
void f3( int ( &p )[ 10 ] ); // Compliant
void f4(span<int> s); // Compliant
void functionWorkingOnSingleInt(int * p); // Compliant

void b ()
{
 int a[ 10 ];
  f3( a ); // size preserved.
  f4( a ); // size captured by the span class
  functionWorkingOnSingleInt( &a[0] ); // explicit about what
happens
}
```

See

- MISRA C++:2008, 5-2-12 - An identifier with array type passed as a function argument shall not decay to a pointer.
- C++ Core Guidelines I.13 - Do not pass an array as a single pointer