C++ static code analysis: Dynamic heap memory allocation should not be used

2 minutes

The use of dynamic memory can lead to out-of-storage run-time failures, which are undesirable.

The built-in `new` and `delete` operators, other than the placement versions, use dynamic heap memory. The functions `calloc`, `malloc`, `realloc` and `free` also use dynamic heap memory.

There is a range of unspecified, undefined and implementation-defined behaviour associated with dynamic memory allocation, as well as a number of other potential pitfalls. Dynamic heap memory allocation may lead to memory leaks, data inconsistency, memory exhaustion, non-deterministic behaviour, etc.

Note that some implementations may use dynamic heap memory allocation to implement other functions (for example, functions in the library `cstring`). If this is the case, then these functions shall also be avoided.

Noncompliant Code Example

```
int *b;
void initialize()
{
  b = (int*) malloc(1024 * sizeof(int)); // Noncompliant, could lead to an out-of-storage run-time failure.
  if (b == 0)
  {
    // handle case when dynamic allocation failed.
  }
}
```

Compliant Solution

```
int b[1024]; // Compliant solution.
```

See

- MISRA C:2004, 20.4 - Dynamic heap memory allocation shall not be used.
- MISRA C++ 2008, 18-4-1 - Dynamic heap memory allocation shall not be used.
- MISRA C:2012, 21.3 The memory allocation and deallocation functions of <stdlib.h> shall not be used