N2905=09-0095
2009-06-18
Bjarne Stroustrup

Aggregation headers

Abstract

I propose two new standard headers **<std-all>** and **<std-checked>** which are aggregations of other headers. The purpose of the first is to simplify the use of many headers and bring such uses into a common form that should simplify pre-compilation. The purpose of the second one is to simplify the safe use of C++ for common and relatively simple uses.

Introduction

Many people, within the committee and without, have talked about "aggregation headers" and various organizations have provided variants of that idea. There are diverse of motivations for having such headers. My primary motivation is "ease of use for non-expert users." If you are an expert, you can just **#include** what you need. Most suggestions fall neatly into two categories:

- "all of the standard library" (with various definitions of "all of").
- "the basic headers, preferably run-time checked, for non-expert uses" (for a wide variety of definitions of "basic," "checked," and "non-expert").

Here, I try a more precise specification.

The <std-all> header

The **<std-all>** header is simply all the headers, excluding the deprecated ones. Fortunately, we don't have standard headers that include facilities that cannot co-exist. The deprecated headers are the C standard library headers with the **.h** suffix and **<strstream>**. Note that all the C standard headers of the **<c???>** form is included; the C standard library is not "second class."

One could argue (i.e., it has been argued) that deprecated features, such as **auto_ptr**, should be eliminated from the aggregation. However, that would be confusing and complicate implementation as it would almost certainly mean that we would get different results from using **<std-all>** than from including its individual headers. I'm not particularly opposed to this idea, but I'm trying to keep things as simple as possible.

Some people, incl. me, have thought that variants with and without **iostream**s and **stdio** would be useful. For example, we could add **<std-all-noio>**, **<std-all-iostream>,** and **<std-all-stdio>** for people who like neither or either of the I/O libraries. This would allow us the set **sync_with_stdio** to false for **<std-all-iostream>**. I'm not particularly opposed to this idea, but I'm trying to keep things as simple as possible. In particular, accepting those would open the door to discussions about an explosion of aggregation headers to suit people's perceived need for "unusual" facilities, such as **<memory>**, **<task>**, **<regexp>**, etc.

The <std-checked> header

This is the header that I personally really see a major and urgent need for. The purpose of **<std-checked>** is to ease the adoption of C++ and to speed up the development of simple programs. The fundamental idea is to provide the most widely used headers that can have their idiomatic uses run-time checked by default. For example, it will provide **<vector>** with a range checked subscript operator (**[]** not just **at()**) and range-checked iterators. Headers for inherently unsafe libraries, notably **gets()** and the **printf** family, are not included.

Suggested components of **<std-checked>**:

- **<algorithm>**
- **<array>**
- **<bitset>**
- **<cmath>**
- **<complex>**
- **<concepts>**
- **<container_concepts>**
- **<cstdlib>**
- **<exception>**
- **<fstream>**
- **<iomanip>**
- **<iostream>**
- **<istream>**
- **<iterator>**
- **<iterator_concepts>**
- **<limits>**
- **<memory>**
- **<numeric>**
- **<ostream>**
- **<random>**
- **<functional>**
- **<tuple>**
- **<initializer_list>**
- **<map>**
- **<queue>**
- **<regexp>**
- **<set>**
- **<sstream>**
- **<stdexcept>**
- **<string>**
- **<type_traits>**
- **<unordered_map>**
- **<unordered_set>**
- **<utility>**
- **<vector>**

I find the combination of low-level facilities and high-level facilities in **<memory>** disturbing. Ideally, expert-level and inherently unsafe facilities, such as allocators and atomics, should be separate from user-level facilities, such as **unique_ptr** and **shared_ptr**, and only the latter included in **<std-checked>**.

As for **<std-all>**, we could get a combinatorial explosion of aggregation headers if we tried to provide specific aggregation headers to serve every need. The guiding principle is "frequently used and relatively to range and access check." The proposal is based on the observation that "everybody already have the checked facilities", but there is no standard way of using them. So I suggest *one* checked header; the exact selections of headers for it is less important as long as the "frequently used and relatively to range and access check" rule is adhered to. For example, I left out **<list>** and **<deque>** because my experience is that they are more often misused by non-experts than used well. This is of course undiluted paternalism and an expression of personal taste. However, anyone who wants more than **<std-checked>** offers can explicitly include it, but not necessarily in a standard checked form.

What do I mean by "range checked and access checked"?

- A subscript operation on a container is range checked and an exception is thrown if the subscript is out of range.
- Any dereference of an iterator is range checked and an exception is thrown if the iterator refers to "end".
- Any increment, decrement, add, or subtract operation of an iterator is checked and an exception is thrown if the result is out of range (not referring to an element or to "end").

I do *not* propose checking for the core language; for example, I do not propose that every pointer dereference should be checked. Programmers explicitly playing with pointers are back in the "good old world." That includes array access. Of course, implementations are allowed to do such checks (as some already do) and may even use the inclusion of **<std-checked>** as their (implementation specific) hint to do so, but that's beyond my proposal (and of the standard). Similarly, I don't propose checking of arithmetic overflow. Again, an implementation is already allowed to do so if it so desires.

What if some part of the program (translation unit) uses <**std-checked**> and another plain **<vector>**? On the face of it, we'd have an ODR violation. On the other hand, this will almost certainly happen as user code (relying on **<std-checked>** calls library code (often optimized to use unchecked **<vector>**). I propose that doing so is considered conforming. On the other hand I, propose to make it an error to **#include** both **<std-checked>** and one of its components in a single translation using.

Why is this important?

C++ has a reputation for being complex, hard to learn, and hard to use. Some of that reputation is well earned and arguably unavoidable given C++'s history and aims. However, much have to do with the difficulty of getting started and with people reinventing the wheel. C++ offers great standard library facilities but many don't use them because they can't find them among the many headers ("I have to include all that stuff?!!"). Having to include many headers is error-prone (for novices), exposes complexity ("I have to include complicated headers to use strings?"), and encourages people to "roll their own" ("I'll just use C-style strings; that's so much easier and more efficient!"). Furthermore, getting run-time checking by default is great (arguably the greatest help we can offer to real novices), but most implementations do not range check by default –and even if most did we'd have a problem with portability of descriptions ("maybe your implementation range-check **vector**s" is a most worrying statement to novices and a red flag to any wannabe security expert).

Suggested WP wording

Add a new section

20.11 Aggregation headers [utility.aggregation]

This clause describes two headers <std-all> and <std-checked> which are aggregation of other headers provided for ease of use.

20.11.12 std-all [utility.aggregation.all]

<std-all> consists of all the C++ standard library headers with the exception of the deprecated headers (<???.h> and <strstream>).

20.11.13 std-checked [utility.aggregation.checked]

<std-checked> consists of

- **<algorithm>**
- **<array>**
- **<bitset>**
- **<cmath>**
- **<complex>**
- **<concepts>**
- **<container_concepts>**
- **<cstdlib>**
- **<exception>**
- **<fstream>**
- **<iomanip>**
- **<iostream>**
- **<istream>**
- **<iterator>**
- **<iterator_concepts>**
- **<limits>**
- **<memory>**
- **<numeric>**
- **<ostream>**
- **<random>**
- **<functional>**
- **<tuple>**
- **<initializer_list>**
- **<map>**
- **<queue>**
- **<regexp>**
- **<set>**
- **<sstream>**
- **<stdexcept>**
- **<string>**
- **<type_traits>**

- **<unordered_map>**
- **<unordered_set>**
- **<utility>**
- **<vector>**

Where run-time checking is done for iterators and substripting. Including both **<std-checked>** and one of its components (e.g. **<vector>**) is ill formed.