Doc. No.: P2879R0
Project: Programming Language - C++ (WG21)
Audience: SG18 LEWG Incubator, SG20 Education
Author: Andrew Tomazos <andrewtomazos@gmail.com>
Date: 2024-09-17

Proposal of std::dump

Introduction

We propose adding to the <print> header a standard library function **std::dump** that prints its arguments space-separated with a new-line. A call to std::dump(arg1, arg2, …, argn) is equivalent to std::println("{} {} … {}", arg1, arg2, …, argn)

Examples

```
std::dump("Hello, World!"); // output: Hello, World!
std::dump(2+2); // output: 4
std::dump(1,2,3,4,5); // output: 1 2 3 4 5
int x = 10, y = 20, z = 30;
std::dump(x,y,z); // output: 10 20 30
```

Motivation

- The semantics of the std::dump function have existing practice in numerous other programming languages. For example, the built-in print function of Python has the same semantics. `print(a,b,c)` in Python does the same thing as `std::dump(a,b,c)`
- std::dump is useful for writing small throw-away programs such as quick tests, demos and small experiments.
- std::dump is useful for writing code examples that are unobfuscated by the format string "{} {} … {}" of std::print or the std::cout / std::endl / `<<` of iostreams. `std::dump(expr)` simply prints out the value that expr yields.
- std::dump is useful as quick temporary code during development to get a quick readout of variable values at runtime without hooking up a debugger
- std::dump is useful for doing temporary "printf-debugging" for environments where connecting a debugger is not possible due to technical limitations of the environment, or not practical due to realtime constraints (pausing execution would change the behavior)

- std::dump is useful for beginners to have a function that simply outputs its arguments, without having to first learn about format strings / formatters or the “streaming” iostreams library interface, or first learning how to use a debugger.
- In scientific computing a commonly portable matrix/table notation is space-separated numbers with rows separated by new-lines - which is the same format that std::dump produces.
- There are a number of traditional unix “token-based” command-line tools that work with the simple format std::dump produces.

Informal Specification

```
namespace std {

  template< class... Args >
    void dump( Args&&... args );
  }

} // namespace std
```

A function call `std::dump()` with no arguments shall have the same effects as std::println(“”). A function call to dump with N arguments arg1, arg2 through argN, shall have the same effects as a function call to println with arguments S, arg1, arg2 through argN - where S has the same value as an ordinary string literal formed by concatenating N string literal tokens “{}” separated by N-1 string literal tokens of “ “. [Example: When N is 1, S is “{}”. When N is 2, S is “{} {}”. When N is 3, S is “{} {} {}”, and so on. - end example]

Alternative Names

The function needs a short intuitive name in order to satisfy the motivation (particularly for beginners), so we wanted a name that is a single English word that “says what it does”. Alternatives that could be considered to the name std::dump include (in no particular order):

std::display
std::show
std::output
std::trace
std::echo
std::report
std::put

Acknowledgements

Thank you to Sebastian Wittmeier, Ville Voutilainen, Jonathan Wakely, Victor Zverovich, Barry Revzin and Arthur O'Dwyer for providing input to this proposal.