Combinatorial Functions

Walter Brown
Richard Dosselmann
Larry Lewis
Phil Ratzloff
Jorge Silva
Michael Wong

Document Number: P...
Date: Month #, 2021 (Pre-XYZ mailing): 10 AM ET
Project: ISO JTC1/SC22/WG21: Programming Language C++
Audience: SG6, SG19, WG21, LEWG
Emails: webrown.cpp@gmail.com,
dosselmr@cs.uregina.ca (corresponding author),
Larry.Lewis@sas.com,
Phil.Ratzloff@sas.com,
Jorge.Silva@sas.com,
michael@codeplay.com

Contents

1 Introduction

This document proposes an extension to the C++ library to support combinatorial functions. Such functions are found in Calc [1], Excel [2], Julia [3], Maple [4, 5], MATLAB [6], Mathematica [7], Python [8], R [9], SAS [10] and Scilab [11]. These functions are also found in the Boost mathematics package [12]. Alongside of combinatorics [13], applications of combinatorial functions are found in domains as varied as biology [14, 15], chemistry [16, 17], coding theory [18], cryptography [19], graph theory [20], ... **machine learning** [21, 22], operations research [23], pattern classification [24], physics [17, 25], probability [26, 27, 28] and recurrence relations [29]. The proposed functions would complement the special mathematical functions [30], which include the associated beta and gamma functions [31].

TBA - existing proposals. The importance of combinatorics in the realm of **machine learning** is noted in P1415 [32].

2 Impact on the Standard

This proposal is a pure **library** extension.

3 Functions

Three functions are defined in this proposal.

3.1 Factorial

The *factorial* [33] of a **non-negative integer** n, denoted $n!$, is defined as

$$n! = n \cdot (n-1) \cdot (n-2) \cdots 3 \cdot 2 \cdot 1, \tag{1}$$

or, equivalently,

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{if } n \geq 1. \end{cases}$$

Methods of computing factorials are given in [34, 35, 36].

3.2 Permutation

A *permutation* [33] is an **ordered** arrangement of $k \geq 0$ objects of a collection of $n \geq k \geq 0$ objects. Denoted $P(n, k)$, it is defined as

$$P(n, k) = n \cdot (n-1) \cdot (n-2) \cdots (n-k+1), \tag{2}$$

or, equivalently,

$$P(n, k) = \frac{n!}{(n-k)!}. \tag{3}$$

3.3 Combination

A *combination* [33], a key element of the binary theorem [29], is an **unordered** selection of $k \geq 0$ objects from a collection of $n \geq k \geq 0$ objects. Denoted $C(n, k)$ or $\binom{n}{k}$, it is defined as

$$C(n, k) = \frac{n!}{k!(n-k)!} = \frac{P(n, r)}{k!}. \tag{4}$$

In the special case that n is **negative** [29],

$$C(-n, k) = (-1)^k \cdot C(n+k-1, k). \tag{5}$$

4 Proposal

This document proposes the inclusion of the **factorial**, **permutation** and **combination** functions in the C++ `<cmath>` library.

4.1 Factorial

The proposed form of the factorial function is given as follows.

```
template <typename T = unsigned>
constexpr T factorial(unsigned n);
```

Parameters

`n` - non-negative integer value

Return Value

If no errors occur, the factorial of `n` is returned.

Complexity

$O(N)$, where `N = ...`.

Error Handling

- TBA

Example

```
std::cout << "5! = " << std::factorial(5) << "\n"; // 120
```

4.2 Permutation

The proposed form of the permutation function is given as follows.

```
template<typename T = unsigned>
constexpr T P(unsigned n, unsigned k);
```

Parameters

- `n` - non-negative integer value
- `k` - non-negative integer value

Return Value

If no errors occur, the number of (ordered) arrangements of `k` objects of a collection of `n` objects is returned.

Complexity

$O(N)$, where `N = ...`.

Error Handling

- TBA

Example

```
std::cout << "P(10, 4) = " << std::P(10, 4) << "\n"; // 5,040
```

4.3 Combination

The proposed form of the combination function is given as follows.

```
template<typename T = unsigned>
constexpr T C(int n, unsigned k);
```

Parameters

- `n` - integer value
- `k` - non-negative integer value

Return Value

If no errors occur, the numbers of (unordered) ways to select `k` objects from a collection of `n` objects is returned.

Complexity

$O(N)$, where `N = ...`.

Error Handling

- TBA

Example

```
std::cout << "C(49,6) = " << std::C(49, 6) << "\n"; // 13,983,816
std::cout << "C(-5, 2) = " << std::C(-5, 2) << "\n"; // 15
```

5 Implementation Issues

This section addresses issues related to the implementations of the proposed functions. As a rule, factorials, and therefore permutations and combinations, grow **rapidly** [37], resulting in **enormous** values. Indeed, "[a] programming language that packs integers into 32 binary digits cannot reach beyond 12!, and even 64-bit arithmetic runs out of room at 20!. To go further requires a language or a program library capable of handling arbitrarily large integers" [37]. Even an 8-byte `double`, having a maximum value of $1.7976931348623157 \cdot 10^{308}$ [38], overflows after $n = 170$, that is, 170!. This is also the upper bound given by MATLAB [39]. Experiments in Python (version 3.8.2) on a 3.60 GHz Intel (CPU) system with 64GB of RAM show the ability to handle much **larger** values, such as 65000! (in about 3 seconds). This is unsurprising, given that integers in Python "represent numbers in an unlimited range, subject to available (virtual) memory only" [40]. Fortunately, such support is proposed for C++, namely the `wide_int` data

type [41], which would allow for the calculation of much larger values of $n!$. By employing **templates**, the proposed functions can accommodate new number types. A **lookup table** can be used for "small" n, such as $n \leq 170$, as is done in Boost [12]. The beta and gamma functions [31] are another option for "small" n.

TBA: show plot of Python number of digits and computation time?

6 Acknowledgements

Michael Wong's work is made possible by Codeplay Software Ltd., ISOCPP Foundation, Khronos and the Standards Council of Canada.

References

[1] "Documentation/How Tos/Calc: Mathematical functions", *Apache OpenOffice*. Web. 23 Oct. 2020
https://wiki.openoffice.org/wiki/Documentation/How_Tos/Calc:_Mathematical_functions

[2] "Excel functions (alphabetical)", *Microsoft*. Web. 23 Oct. 2020
https://support.microsoft.com/en-us/office/excel-functions-alphabetical-b3944572-255d-4efb-bb96-c6d90033e188

[3] "Mathematics", *Julia*. Web. 23 Oct. 2020
https://docs.julialang.org/en/v1/base/math/

[4] "!", *Maplesoft*. Web. 6 Nov. 2020
https://www.maplesoft.com/support/help/Maple/view.aspx?path=factorial

[5] "binomial", *Maplesoft*. Web. 6 Nov. 2020
https://www.maplesoft.com/support/help/Maple/view.aspx?path=binomial

[6] "MATLAB - Functions", *MathWorks*. Web. 23 Oct. 2020
https://www.mathworks.com/help/matlab/referencelist.html?type=function

[7] "Factorial (!)", *Wolfram*. Web. 6 Nov. 2020
https://reference.wolfram.com/language/ref/Factorial.html

[8] "math - Mathematical functions", *Python*. Web. 23 Oct. 2020
https://docs.python.org/3/library/math.html

[9] "Special", *RDocumentation*. Web. 25 Oct. 2020
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/Special

[10] "Functions and CALL Routines by Category", *sas*. Web. 23 Oct. 2020
https://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000245860.htm

[11] "factorial", *Scilab*. Web. 6 Nov. 2020
https://help.scilab.org/docs/6.0.2/en_US/factorial.html

[12] Agrawal, Nikhar et al. "Factorials and Binomial Coefficients", *Boost: C++ Libraries*. Web. 23 Oct. 2020.
https://www.boost.org/doc/libs/1_74_0/libs/math/doc/html/math_toolkit/factorials.html

[13] Wilson, Robin. Combinatorics: A Very Short Introduction, *Oxford University Press*, 2016.

[14] Shonkwiler, Ronald W. and Herod, James. Mathematical Biology: An Introduction with Maple and Matlab - Second Edition, *Springer*, 2009.

[15] Meirovitch, Hagai. Entropy and Free Energy in Structural Biology: From Thermodynamics to Statistical Mechanics to Computer Simulation, *CRC Press*, 2020.

[16] Steiner, Erich. The Chemistry Maths Book - 2nd Edition, *Oxford University Press*, 2008.

[17] Mellor, Joseph W. Higher Mathematics for Students of Chemistry and Physics, *Cosmo*, 2007.

[18] Joyner, David (editor). Coding Theory and Cryptography: From Enigma and Geheimschreiber to Quantum Theory, *Springer*, 2000.

[19] Hershey, John E. Cryptography Demystified, *McGraw-Hill*, 2003.

[20] Harris, John M., Hirst, Jeffry L. and Mossinghoff, Michael J. Combinatorics and Graph Theory - Second Edition, *Springer*, 2008.

[21] Dulhare, Uma N., Ahmad, Khaleel, Ahmad, Khairol Amali Bin. Machine Learning and Big Data: Concepts, Algorithms, Tools and Applications, *John Wiley & Sons*, 2020.

[22] Wichert, Andreas Miroslaus. Principles Of Quantum Artificial Intelligence: Quantum Problem Solving And Machine Learning - Second Edition, *World Scientific*, 2020.

[23] Ignizio, James P., Gupta, Jatinder N. D. and McNichols, Gerald R. Operations Research in Decision Making, *Crane, Russak*, 1975.

[24] Duda, Richard O., Hart, Peter E. and Stork, David G. Pattern Classification - Second Edition, *John Wiley & Sons*, 2001.

[25] Goldstein, Melvin. Physics Foibles, *Bonum*, 2003.

[26] Apte, D. P. Probability and Combinatorics, *Excel Books*, 2007.

[27] Hill, Tim. Essential Permutations & Combinations: A Self-Teaching Guide, *Questing Vole Press*, 2018.

[28] Borovkov, Alexandr A. Probability Theory, *Springer*, 2013.

[29] Grimaldi, Ralph P. Discrete and Combinatorial Mathematics: An Applied Introduction, *Addison-Wesley Publishing Company*, 1985.

[30] Brown, Walter E., Naumann, Axel and Smith-Rowland, Edward. "Mathematical Special Functions for C++17, v4", *P0226R0*. Web. 12 Jun. 2020
`www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0226r0.pdf`

[31] Farrell, Orin J. and Ross, Bertram. Solved Problems in Analysis: As Applied to Gamma, Beta, Legendre and Bessel Functions, *Dover Publications*, 1991.

[32] Wong, Michael et al. "P1415R1: SG19 Machine Learning Layered List", *ISO JTC1/SC22/WG21: Programming Language C++*. Web. 9 Aug. 2020
`http://open-std.org/JTC1/SC22/WG21/docs/papers/2019/p1415r1.pdf`

[33] Epp, Susanna S. Discrete Mathematics: Introduction to Mathematical Reasoning, *Brooks/Cole*, 2011.

[34] Borwein, Peter B. "On the complexity of calculating factorials", *Journal of Algorithms*, **6**(3), Sept. 1985, p. 376-380.

[35] "FastFactorialFunctions", *http://www.luschny.de/*. Web. 24 Oct. 2020
`http://www.luschny.de/math/factorial/FastFactorialFunctions.htm`

[36] "Efficient Factorials Calculation !", *hackerearth*. Web. 24 Oct. 2020
`https://www.hackerearth.com/practice/notes/efficient-factorials-calculation/`

[37] Hayes, Brian. Foolproof, and Other Mathematical Meditations, *MIT Press*, 2018.

[38] "Fundamental types", *cppreference.com*. Web. 24 Oct. 2020
`https://en.cppreference.com/w/cpp/language/types`

[39] "factorial", *MathWorks*. Web. 25 Oct. 2020
`https://www.mathworks.com/help/matlab/ref/factorial.html`

[40] "Data model", *Python*. Web. 25 Oct. 2020
`https://docs.python.org/3/reference/datamodel.html#objects-values-and-types`

[41] Klevanets, Igor and Polukhin, Antony. "A Proposal to add wide_int Template Class", *P0539R5*. Web. 24 Oct. 2020
`http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2020/p0539r5.pdf`