Physical Constants

Name	Symbol	Value (SI)
Speed of light in vacuum	c	2.99792458×10^{8} m/s (exact)
Newtonian constant of gravitation	G	$6.67430(15) \times 10^{-11}$ $\mathrm{m^3 \cdot kg^{-1} \cdot s^{-2}}$
Planck constant	h	$6.62607015 \times 10^{-34}$ J · s (exact)
Reduced Planck constant $h/2\pi$	$\hbar$	$1.054571817... \times 10^{-34}$ J · s (exact)
Elementary charge	e	$1.602176634 \times 10^{-19}$ C (exact)
Vacuum permittivity	ε_0	$8.8541878188(14) \times 10^{-12}$ F/m
Vacuum permeability $1/c^2\epsilon_0$	μ_0	$1.25663706127(20) \times 10^{-7}$ H/m
Electron mass	m_e	$9.109383(28) \times 10^{-31}$ kg
Proton mass	m_p	$1.67262192595(52) \times 10^{-27}$ kg
Proton-electron mass ratio	m_p/m_e	1836.152673426(32)
Boltzmann constant	k_B	1.380649×10^{-23} J/K (exact)
Avogadro constant	N_A	$6.02214076 \times 10^{23}$ $\mathrm{mol^{-1}}$ (exact)
Electron volt (e/C) J	eV	$1.602176634 \times 10^{-19}$ J (exact)
(Unified) atomic mass unit $m(^{12}\mathrm{C})/12$	u, m_u	$1.66053906892(52) \times 10^{-27}$ kg
Temperature 1 eV	–	$1.160451812... \times 10^{4}$ K (exact)

To use the above physical constants in C++, #include the following header file:

```cpp
// physical_constants.hpp
#if !defined(PHYSICAL_CONSTANTS_HPP)
#define PHYSICAL_CONSTANTS_HPP

#include <numbers> // namespace std::numbers 包含 e, egamma, pi, 以及一些常用的 log 和 sqrt
#include <type_traits> // for std::is_arithmetic_v<> and std::is_floating_point_v<>

// SI since 2019
namespace physical_constants {
    // Favor double over float unless there is no enough space, as the lack of precision in a float will often lead to inaccuracies.

    // Note that constexpr functions are implicitly inline, but constexpr variables are not implicitly inline.

    // Inline variables have external linkage by default, so that they are visible to the linker. This is necessary so the linker can de-duplicate the definitions.
    // Non-inline constexpr variables have internal linkage. If included into multiple translation units, each translation unit will get its own copy of the variable. This is not an ODR violation because they are not exposed to the linker.

    // Since we are using inline variables, changing anything in this header file requires recompiling those files who include it.
    // Use constexpr std::string_view for constexpr strings.

    // ============================================================
    //  Exact constants
    // ============================================================

    // Speed of light in vacuum: c = 299 792 458 m/s  (exact)
    template<typename T>
        requires std::is_arithmetic_v<T> // 要求 T 是算术类型（整数或浮点数）
    inline constexpr T speed_of_light_v { static_cast<T>(299'792'458.0L) }; // static_cast 是静态类型转换，在编译期完成
    inline constexpr double speed_of_light { speed_of_light_v<double> };

    template<typename T>
        requires std::is_arithmetic_v<T>
    inline constexpr T sqrt_c2_v { speed_of_light_v<T> };
    inline constexpr double sqrt_c2 { speed_of_light };

    template<typename T>
        requires std::is_arithmetic_v<T>
    inline constexpr T c_v { speed_of_light_v<T> };
    inline constexpr double c { speed_of_light };


    template<typename T>
        requires std::is_arithmetic_v<T> // 要求 T 是算术类型（整数或浮点数）
    inline constexpr T speed_of_light_square_v { static_cast<T>(89'875'517'873'681'764.0L) }; // static_cast 是静态类型转换，在编译期完成
```

```
44  inline constexpr double speed_of_light_square { speed_of_light_square_v<double> };
45
46  template<typename T>
47      requires std::is_arithmetic_v<T>
48  inline constexpr T c2_v { speed_of_light_square_v<T> };
49  inline constexpr double c2 = speed_of_light_square;
50
51
52
53  // Planck constant: h = 6.626 070 15e-34 J·s  (exact)
54  template<typename T>
55      requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
56  inline constexpr T planck_constant_v { static_cast<T>(6.626'070'15e-34L) };
57  inline constexpr double planck_constant { planck_constant_v<double> };
58
59  template<typename T>
60      requires std::is_floating_point_v<T>
61  inline constexpr T h_v { planck_constant_v<T> };
62  inline constexpr double h { planck_constant };
63
64  // Reduced Planck constant: ħ = h / (2π)  (exact, derived)
65  template<typename T>
66      requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
67  inline constexpr T planck_constant_reduced_v { static_cast<T>(3.313'035'075e-34L * std::numbers::inv_pi_v<long
    double>) };
68  inline constexpr double planck_constant_reduced { planck_constant_reduced_v<double> };
69
70  template<typename T>
71      requires std::is_floating_point_v<T>
72  inline constexpr T hbar_v { planck_constant_reduced_v<T> };
73  inline constexpr double hbar { planck_constant_reduced };
74
75
76
77  // Elementary charge: e = 1.602 176 634e-19 C  (exact)
78  template<typename T>
79      requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
80  inline constexpr T elementary_charge_v { static_cast<T>(1.602'176'634e-19L) };
81  inline constexpr double elementary_charge { elementary_charge_v<double> };
82
83  template<typename T>
84      requires std::is_floating_point_v<T>
85  inline constexpr T qe_v { elementary_charge_v<T> };
86  inline constexpr double qe { elementary_charge };
87
88  template<typename T>
89      requires std::is_floating_point_v<T>
90  inline constexpr T e_v { elementary_charge_v<T> };
91  inline constexpr double e { elementary_charge };
```

```
92
93
94  // Boltzmann constant: k_B = 1.380 649e-23 J/K  (exact)
95  template<typename T>
96      requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
97  inline constexpr T boltzmann_constant_v { static_cast<T>(1.380'649e-23L) };
98  inline constexpr double boltzmann_constant { boltzmann_constant_v<double> };
99
100 template<typename T>
101     requires std::is_floating_point_v<T>
102 inline constexpr T kB_v { boltzmann_constant_v<T> };
103 inline constexpr double kB { boltzmann_constant };
104
105
106
107 // Avogadro constant: N_A = 6.022 140 76e23 mol⁻¹  (exact)
108 template<typename T>
109     requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
110 inline constexpr T avogadro_constant_v { static_cast<T>(6.022'140'76e23L) };
111 inline constexpr double avogadro_constant { avogadro_constant_v<double> };
112
113 template<typename T>
114     requires std::is_floating_point_v<T>
115 inline constexpr T NA_v { avogadro_constant_v<T> };
116 inline constexpr double NA { avogadro_constant };
117
118
119
120 // Electron volt: eV = 1.602 176 634e-19 J  (exact, same as elementary charge)
121 template<typename T>
122     requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
123 inline constexpr T electron_volt_v { static_cast<T>(1.602'176'634e-19L) };
124 inline constexpr double electron_volt { electron_volt_v<double> };
125
126 template<typename T>
127     requires std::is_floating_point_v<T>
128 inline constexpr T eV_v { electron_volt_v<T> };
129 inline constexpr double eV { electron_volt };
130
131
132
133 // Temperature of 1 eV: e / k_B (exact, derived)
134 template<typename T>
135     requires std::is_arithmetic_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧⌧⌧⌧⌧⌧⌧⌧
136 inline constexpr T electron_volt_temperature_v { static_cast<T>(elementary_charge_v<long double> /
    boltzmann_constant_v<long double>) };
137 inline constexpr double electron_volt_temperature { electron_volt_temperature_v<double> };
138
139 template<typename T>
```

```
140         requires std::is_arithmetic_v<T>
141     inline constexpr T eV_temp_v { electron_volt_temperature_v<T> };
142     inline constexpr double eV_temp { electron_volt_temperature };
143
144     // ============================================================
145     //  Constants with experimental uncertainty (central values)
146     // ============================================================
147
148     // Newtonian constant of gravitation: G = 6.674 30(15)e-11 m³·kg⁻¹·s⁻²
149     template<typename T>
150         requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
151     inline constexpr T gravitational_constant_v { static_cast<T>(6.674'30e-11L) };
152     inline constexpr double gravitational_constant { gravitational_constant_v<double> };
153
154     template<typename T>
155         requires std::is_floating_point_v<T>
156     inline constexpr T G_v { gravitational_constant_v<T> };
157     inline constexpr double G { gravitational_constant };
158
159
160     // Vacuum permittivity: ε₀ = 8.854 187 818 8(14)e-12 F/m
161     template<typename T>
162         requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
163     inline constexpr T vacuum_permittivity_v { static_cast<T>(8.854'187'818'8e-12L) };
164     inline constexpr double vacuum_permittivity { vacuum_permittivity_v<double> };
165
166     template<typename T>
167         requires std::is_floating_point_v<T>
168     inline constexpr T epsilon0_v { vacuum_permittivity_v<T> };
169     inline constexpr double epsilon0 { vacuum_permittivity };
170
171
172
173     // Vacuum permeability: μ₀ = 1.256 637 061 27(20)e-6 H/m
174     template<typename T>
175         requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
176     inline constexpr T vacuum_permeability_v { static_cast<T>(1.256'637'061'27e-6L) };
177     inline constexpr double vacuum_permeability { vacuum_permeability_v<double> };
178
179     template<typename T>
180         requires std::is_floating_point_v<T>
181     inline constexpr T mu0_v { vacuum_permeability_v<T> };
182     inline constexpr double mu0 { vacuum_permeability };
183
184
185
186     // Electron mass: m_e = 9.109 383(28)e-31 kg
187     template<typename T>
188         requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
```

```
189 inline constexpr T electron_mass_v { static_cast<T>(9.109'383e-31L) };
190 inline constexpr double electron_mass { electron_mass_v<double> };
191
192 template<typename T>
193     requires std::is_floating_point_v<T>
194 inline constexpr T me_v { electron_mass_v<T> };
195 inline constexpr double me { electron_mass };
196
197
198
199 // Electron charge to mass ratio: e / m_e = 1.758 820 1e11 C/kg
200 template<typename T>
201     requires std::is_arithmetic_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
202 inline constexpr T electron_charge_to_mass_ratio_v { static_cast<T>(elementary_charge_v<long double> /
    electron_mass_v<long double>) };
203 inline constexpr double electron_charge_to_mass_ratio { electron_charge_to_mass_ratio_v<double> };
204
205 template<typename T>
206     requires std::is_arithmetic_v<T>
207 inline constexpr T qe_over_me_v { electron_charge_to_mass_ratio_v<T> };
208 inline constexpr double qe_over_me { electron_charge_to_mass_ratio_v<double> };
209
210 template<typename T>
211     requires std::is_arithmetic_v<T>
212 inline constexpr T e_over_me_v { electron_charge_to_mass_ratio_v<T> };
213 inline constexpr double e_over_me { electron_charge_to_mass_ratio_v<double> };
214
215
216
217 // Electron mass to charge ratio: m_e / e = 5.685 629 7e-12 kg/C
218 template<typename T>
219     requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
220 inline constexpr T electron_mass_to_charge_ratio_v { static_cast<T>(electron_mass_v<long double> /
    elementary_charge_v<long double>) };
221 inline constexpr double electron_mass_to_charge_ratio { electron_mass_to_charge_ratio_v<double> };
222
223 template<typename T>
224     requires std::is_floating_point_v<T>
225 inline constexpr T me_over_qe_v { electron_mass_to_charge_ratio_v<T> };
226 inline constexpr double me_over_qe { electron_mass_to_charge_ratio_v<double> };
227
228 template<typename T>
229     requires std::is_floating_point_v<T>
230 inline constexpr T me_over_e_v { electron_mass_to_charge_ratio_v<T> };
231 inline constexpr double me_over_e { electron_mass_to_charge_ratio_v<double> };
232
233
234
235 // Proton mass: m_p = 1.672 621 925 95(52)e-27 kg
```

```
236 template<typename T>
237     requires std::is_floating_point_v<T> // ⊠⊠ T ⊠⊠⊠⊠⊠
238 inline constexpr T proton_mass_v { static_cast<T>(1.672'621'925'95e-27L) };
239 inline constexpr double proton_mass { proton_mass_v<double> };
240
241 template<typename T>
242     requires std::is_floating_point_v<T>
243 inline constexpr T mp_v { proton_mass_v<T> };
244 inline constexpr double mp { proton_mass_v<double> };
245
246
247
248 // Proton charge to mass ratio: e / m_p = 9.578 833 143 0e7 C/kg
249 template<typename T>
250     requires std::is_arithmetic_v<T> // ⊠⊠ T ⊠⊠⊠⊠⊠
251 inline constexpr T proton_charge_to_mass_ratio_v { static_cast<T>(elementary_charge_v<long double> /
    proton_mass_v<long double>) };
252 inline constexpr double proton_charge_to_mass_ratio { proton_charge_to_mass_ratio_v<double> };
253
254 template<typename T>
255     requires std::is_arithmetic_v<T>
256 inline constexpr T qe_over_mp_v { proton_charge_to_mass_ratio_v<T> };
257 inline constexpr double qe_over_mp { proton_charge_to_mass_ratio_v<double> };
258
259 template<typename T>
260     requires std::is_arithmetic_v<T>
261 inline constexpr T e_over_mp_v { proton_charge_to_mass_ratio_v<T> };
262 inline constexpr double e_over_mp { proton_charge_to_mass_ratio_v<double> };
263
264
265
266 // Proton mass to charge ratio: m_p / e = 1.043'968'492'9e-8 kg/C
267 template<typename T>
268     requires std::is_floating_point_v<T> // ⊠⊠ T ⊠⊠⊠⊠⊠
269 inline constexpr T proton_mass_to_charge_ratio_v { static_cast<T>(proton_mass_v<long double> /
    elementary_charge_v<long double>) };
270 inline constexpr double proton_mass_to_charge_ratio { proton_mass_to_charge_ratio_v<double> };
271
272 template<typename T>
273     requires std::is_floating_point_v<T>
274 inline constexpr T mp_over_qe_v { proton_mass_to_charge_ratio_v<T> };
275 inline constexpr double mp_over_qe { proton_mass_to_charge_ratio_v<double> };
276
277 template<typename T>
278     requires std::is_floating_point_v<T>
279 inline constexpr T mp_over_e_v { proton_mass_to_charge_ratio_v<T> };
280 inline constexpr double mp_over_e { proton_mass_to_charge_ratio_v<double> };
281
282
```

```
283
284     // Proton-electron mass ratio: m_p / m_e = 1 836.152 673 426(32)
285     template<typename T>
286         requires std::is_arithmetic_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧⌧⌧⌧⌧⌧⌧⌧⌧
287     inline constexpr T proton_electron_mass_ratio_v { static_cast<T>(1'836.152'673'426L) };
288     inline constexpr double proton_electron_mass_ratio { proton_electron_mass_ratio_v<double> };
289
290     template<typename T>
291         requires std::is_arithmetic_v<T>
292     inline constexpr T mp_over_me_v { proton_electron_mass_ratio_v<T> };
293     inline constexpr double mp_over_me { proton_electron_mass_ratio_v<double> };
294
295
296
297     // (Unified) atomic mass unit: u = 1.660 539 068 92(52)e-27 kg
298     template<typename T>
299         requires std::is_floating_point_v<T> // ⌧⌧ T ⌧⌧⌧⌧⌧
300     inline constexpr T atomic_mass_unit_v { static_cast<T>(1.660'539'068'92e-27L) };
301     inline constexpr double atomic_mass_unit { atomic_mass_unit_v<double> };
302
303 } // namespace physical_constants
304
305 #endif // PHYSICAL_CONSTANTS_HPP
306
```