**[C++11 std::chrono库详解](https://www.cnblogs.com/jwk000/p/3560086.html)**

https://www.cnblogs.com/jwk000/p/3560086.html

　　所谓的详解只不过是参考www.cplusplus.com的说明整理了一下，因为没发现别人有详细讲解。

　　chrono是一个time library, 源于boost，现在已经是C++标准。话说今年似乎又要出新标准了，好期待啊！

　　要使用chrono库，需要#include<chrono>，其所有实现均在std::chrono namespace下。注意标准库里面的每个命名空间代表了一个独立的概念。所以下文中的概念均以命名空间的名字表示！ chrono是一个模版库，使用简单，功能强大，只需要理解三个概念：duration、time\_point、clock

1.Durations

std::chrono::duration 表示一段时间，比如两个小时，12.88秒，半个时辰，一炷香的时间等等，只要能换算成秒即可。

1 template <class Rep, class Period = ratio<1> > class duration;

其中

Rep表示一种数值类型，用来表示Period的数量，比如int float double

Period是ratio类型，用来表示【用秒表示的时间单位】比如second milisecond

常用的duration<Rep,Period>已经定义好了，在std::chrono::duration下：

ratio<3600, 1>                hours

ratio<60, 1>                    minutes

ratio<1, 1>                      seconds

ratio<1, 1000>               microseconds

ratio<1, 1000000>         microseconds

ratio<1, 1000000000>    nanosecons

这里需要说明一下ratio这个类模版的原型：

1 template <intmax\_t N, intmax\_t D = 1> class ratio;

N代表分子，D代表分母，所以ratio表示一个分数值。

注意，我们自己可以定义Period，比如ratio<1, -2>表示单位时间是-0.5秒。

由于各种duration表示不同，chrono库提供了duration\_cast类型转换函数。

1 template <class ToDuration, class Rep, class Period>

2 constexpr ToDuration duration\_cast (const duration<Rep,Period>& dtn);

典型的用法是表示一段时间：

[复制代码](javascript:void(0);)

1 // duration constructor

2 #include <iostream>

3 #include <ratio>

4 #include <chrono>

5

6 int main ()

7 {

8 typedef std::chrono::duration<int> seconds\_type;

9 typedef std::chrono::duration<int,std::milli> milliseconds\_type;

10 typedef std::chrono::duration<int,std::ratio<60\*60>> hours\_type;

11

12 hours\_type h\_oneday (24); // 24h

13 seconds\_type s\_oneday (60\*60\*24); // 86400s

14 milliseconds\_type ms\_oneday (s\_oneday); // 86400000ms

15

16 seconds\_type s\_onehour (60\*60); // 3600s

17 //hours\_type h\_onehour (s\_onehour); // NOT VALID (type truncates), use:

18 hours\_type h\_onehour (std::chrono::duration\_cast<hours\_type>(s\_onehour));

19 milliseconds\_type ms\_onehour (s\_onehour); // 3600000ms (ok, no type truncation)

20

21 std::cout << ms\_onehour.count() << "ms in 1h" << std::endl;

22

23 return 0;

24 }

25

26 duration还有一个成员函数count()返回Rep类型的Period数量，看代码：

27

28 // duration::count

29 #include <iostream> // std::cout

30 #include <chrono> // std::chrono::seconds, std::chrono::milliseconds

31 // std::chrono::duration\_cast

32

33 int main ()

34 {

35 using namespace std::chrono;

36 // std::chrono::milliseconds is an instatiation of std::chrono::duration:

37 milliseconds foo (1000); // 1 second

38 foo\*=60;

39

40 std::cout << "duration (in periods): ";

41 std::cout << foo.count() << " milliseconds.\n";

42

43 std::cout << "duration (in seconds): ";

44 std::cout << foo.count() \* milliseconds::period::num / milliseconds::period::den;

45 std::cout << " seconds.\n";

46

47 return 0;

48 }

[复制代码](javascript:void(0);)

***2.Time points***

std::chrono::time\_point 表示一个具体时间，如上个世纪80年代、你的生日、今天下午、火车出发时间等，只要它能用计算机时钟表示。鉴于我们使用时间的情景不同，这个time point具体到什么程度，由选用的单位决定。一个time point必须有一个clock计时。参见clock的说明。

1 template <class Clock, class Duration = typename Clock::duration> class time\_point;

下面是构造使用time\_point的例子：

[复制代码](javascript:void(0);)

1 // time\_point constructors

2 #include <iostream>

3 #include <chrono>

4 #include <ctime>

5

6 int main ()

7 {

8 using namespace std::chrono;

9

10 system\_clock::time\_point tp\_epoch; // epoch value

11

12 time\_point <system\_clock,duration<int>> tp\_seconds (duration<int>(1));

13

14 system\_clock::time\_point tp (tp\_seconds);

15

16 std::cout << "1 second since system\_clock epoch = ";

17 std::cout << tp.time\_since\_epoch().count();

18 std::cout << " system\_clock periods." << std::endl;

19

20 // display time\_point:

21 std::time\_t tt = system\_clock::to\_time\_t(tp);

22 std::cout << "time\_point tp is: " << ctime(&tt);

23

24 return 0;

25 }

26

[复制代码](javascript:void(0);)

time\_point有一个函数time\_from\_eproch()用来获得1970年1月1日到time\_point时间经过的duration。

举个例子，如果timepoint以天为单位，函数返回的duration就以天为单位。

由于各种time\_point表示方式不同，chrono也提供了相应的转换函数 time\_point\_cast。

1 template <class ToDuration, class Clock, class Duration>

2 time\_point<Clock,ToDuration> time\_point\_cast (const time\_point<Clock,Duration>& tp);

比如计算

/

[复制代码](javascript:void(0);)

1 / time\_point\_cast

2 #include <iostream>

3 #include <ratio>

4 #include <chrono>

5

6 int main ()

7 {

8 using namespace std::chrono;

9

10 typedef duration<int,std::ratio<60\*60\*24>> days\_type;

11

12 time\_point<system\_clock,days\_type> today = time\_point\_cast<days\_type>(system\_clock::now());

13

14 std::cout << today.time\_since\_epoch().count() << " days since epoch" << std::endl;

15

16 return 0;

17 }

[复制代码](javascript:void(0);)

3.Clocks

std::chrono::system\_clock 它表示当前的系统时钟，系统中运行的所有进程使用now()得到的时间是一致的。

每一个clock类中都有确定的time\_point, duration, Rep, Period类型。

操作有：

now() 当前时间time\_point

to\_time\_t() time\_point转换成time\_t秒

from\_time\_t() 从time\_t转换成time\_point

典型的应用是计算时间日期：

[复制代码](javascript:void(0);)

1 // system\_clock example

2 #include <iostream>

3 #include <ctime>

4 #include <ratio>

5 #include <chrono>

6

7 int main ()

8 {

9 using std::chrono::system\_clock;

10

11 std::chrono::duration<int,std::ratio<60\*60\*24> > one\_day (1);

12

13 system\_clock::time\_point today = system\_clock::now();

14 system\_clock::time\_point tomorrow = today + one\_day;

15

16 std::time\_t tt;

17

18 tt = system\_clock::to\_time\_t ( today );

19 std::cout << "today is: " << ctime(&tt);

20

21 tt = system\_clock::to\_time\_t ( tomorrow );

22 std::cout << "tomorrow will be: " << ctime(&tt);

23

24 return 0;

25 }

26

[复制代码](javascript:void(0);)

std::chrono::steady\_clock 为了表示稳定的时间间隔，后一次调用now()得到的时间总是比前一次的值大（这句话的意思其实是，如果中途修改了系统时间，也不影响now()的结果），每次tick都保证过了稳定的时间间隔。

操作有：

now() 获取当前时钟

典型的应用是给算法计时：

[复制代码](javascript:void(0);)

1 // steady\_clock example

2 #include <iostream>

3 #include <ctime>

4 #include <ratio>

5 #include <chrono>

6

7 int main ()

8 {

9 using namespace std::chrono;

10

11 steady\_clock::time\_point t1 = steady\_clock::now();

12

13 std::cout << "printing out 1000 stars...\n";

14 for (int i=0; i<1000; ++i) std::cout << "\*";

15 std::cout << std::endl;

16

17 steady\_clock::time\_point t2 = steady\_clock::now();

18

19 duration<double> time\_span = duration\_cast<duration<double>>(t2 - t1);

20

21 std::cout << "It took me " << time\_span.count() << " seconds.";

22 std::cout << std::endl;

23

24 return 0;

25 }

26

[复制代码](javascript:void(0);)

最后一个时钟，std::chrono::high\_resolution\_clock 顾名思义，这是系统可用的最高精度的时钟。实际上high\_resolution\_clock只不过是system\_clock或者steady\_clock的typedef。

操作有：

now() 获取当前时钟。

chrono库还有几个小特性，但是像这种工具库，本着够用则已的态度，就不求全责备了。

（全文完）