CS 3460

Introduction to Date & Time Handling

Date & Time Handling

- Starting with C++ 11 very good support for time was added
- Starting with C++ 20 very good support for dates is added
- Concepts
  - **duration** : a span of time (hours, minutes, etc)
  - **time point** : specific point in time (4th of July)
  - **clock** : used to obtain a time point
  - **calendar** : represents a year, month, day
- Header file: `<chrono>`

Duration

- Composed of two pieces of information
  - a period
  - a count
- Period
  - unit of time expressed as a fraction of a second
    - Can be less than, equal to, or greater than 1
- Count
  - Number of periods in the duration

Duration Examples

- Example A
  - Period = 1
  - Count = 10
  - Represents 10 seconds
- Example B
  - Period = 0.001 (1 / 1000); 1 millisecond
  - Count = 10,000
  - Represents 10,000 ms or 10 seconds

Duration Code Examples

- Type : `std::chrono::duration`
- Two parts
  - underlying data type
    - Can be either integral or floating point
  - ratio

```
std::chrono::duration<std::uint32_t, std::ratio<1, 1>> seconds(10);
std::chrono::duration<std::uint32_t, std::ratio<1, 1000>> milliseconds(10000);
std::chrono::duration<std::uint32_t, std::ratio<60, 1>> minutes(4);
```

Duration – Type Aliases

- The declaration can be quite long
- Use type aliases to shorten

```
using seconds = std::chrono::duration<std::uint32_t, std::ratio<1, 1>>;
using milliseconds = std::chrono::duration<std::uint32_t, std::ratio<1, 1000>>;
using minutes = std::chrono::duration<std::uint32_t, std::ratio<60, 1>>;


seconds s(10);
milliseconds ms(10000);
minutes min(4);
```

Duration – Type Aliases

- The declaration can be quite long
- Use type aliases to shorten

```
using seconds = std::chrono::duration<std::uint32_t, std::ratio<1, 1>>;
using milliseconds = std::chrono::duration<std::uint32_t, std::ratio<1, 1000>>;
using minutes = std::chrono::duration<std::uint32_t, std::ratio<60, 1>>;


seconds s(10);
milliseconds ms(10000);
minutes min(4);
```

- The library provides types for common durations...

```
std::chrono::seconds s(10);
std::chrono::milliseconds ms(10000);
std::chrono::minutes min(4);
.. others ...
```

Duration – Conversions

- Non-narrowing conversions happen automatically

```
std::chrono::seconds seconds(30);
std::chrono::minutes minutes(1);
std::chrono::seconds minutesToSeconds = minutes;
```

Duration – Conversions

- Non-narrowing conversions happen automatically

```
std::chrono::seconds seconds(30);
std::chrono::minutes minutes(1);
std::chrono::seconds minutesToSeconds = minutes;
```

- Narrowing conversions require a cast
  - e.g. from seconds to minutes

```
std::chrono::seconds seconds(30);
std::chrono::minutes minutes(1);

minutes += std::chrono::duration_cast<std::chrono::minutes>(seconds);
```

Code Demo of Duration Conversions

Clocks

- A clock is defined by two items
  - epoch : some starting point (often 1/1/1970)
  - tick rate : rate at which time passes
- Many clocks provided in the `std::chrono` namespace
  - `system_clock` : System real-time clock (wall clock)
  - `steady_clock` : Monotonic clock
  - `high_resolution_clock` : Shortest possible tick
  - `utc_clock` : Coordinated Universal Time
  - `tai_clock` : International Atomic Time
  - `gps_clock` : GPS
  - `file_clock` : Alias for the clock used to represent filesystem times
- These clocks may or may not be the same thing, can vary by system

Clocks

- Exposes various types
  - `rep` : type representing # of tics
  - `period` : `std::ratio` of the tick period; relative to 1s
  - `duration` : duration type for the clock
  - `time_point` : time point type for the clock
- Method
  - `now` : returns a `time_point` of the current time

Code Demo of Clocks

Time Points

- Represents a point in time
  - A duration from the epoch of a clock
  - Therefore, based on a specific clock
  - Possible to convert time points between clocks, but not covered in this class
- All clocks have a `::now` method that returns a `time_point`
  - The `time_point` type for the system clock is

```
std::chrono::time_point<std::chrono::system_clock>
```

Code Demo of Time Points

Code Demo of Timing Things

Calendars

- A calendar is used to represent a specific year, month, and day
  - A number of variations like last day of month, weekday, month, etc.
- Some example types
  - `std::chrono::day`
  - `std::chrono::month` (e.g., `std::chrono::January`)
  - `std::chrono::weekday` (e.g., `std::chrono::Sunday`)
  - `std::chrono::year_month_day`
  - many others…

Code Demo of Calendars

Adding Days to `year_month_day`

- For some reason, not possible to directly add days to a `year_month_day`, so here is how to deal with it
  - `auto ymd = std::chrono::January/1/2001;`
  - Step 1: Convert the `year_month_day` to a number of days
    - `auto days = std::chrono::sys_days{ymd};`
  - Step 2: Add the desired number of days to this
    - `days += std::chrono::days{7};`
  - Step 3: Convert this back to a `year_month_day`
    - `ymd = std::chrono::year_month_day{days};`