$=1$

Lab 2

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 ComplexForecast Class Reference

Class for managing a collection of daily weather forecasts.

`#include <ComplexForecast.h>`

Public Member Functions

- **ComplexForecast** ()

  is the default constructor. Initializes an object with an initial capacity for storing forecasts.
- ComplexForecast (const SimpleForecast *forecastsArray, std::size_t count)
- ComplexForecast (const ComplexForecast &other)
- ComplexForecast (ComplexForecast &&other) noexcept
- ComplexForecast & operator= (const ComplexForecast &other)

  Assignment operator for copying.
- ComplexForecast & operator= (ComplexForecast &&other) noexcept
- ~ComplexForecast ()
- void addForecast (const SimpleForecast &forecast)
- SimpleForecast & operator[ ] (std::size_t index)
- void removeForecast (std::size_t index)
- SimpleForecast findColdestDay () const
- SimpleForecast findNearestSunnyDay (long currentTimestamp) const
- void **removeInvalidForecasts** ()

  Delete all erroneous predictions.
- void **sortForecasts** ()

  Sort forecasts by date.

Friends

- std::ostream & operator<< (std::ostream &out, const ComplexForecast &forecast)

  The output statement to the stream.
- std::istream & operator>> (std::istream &in, ComplexForecast &forecast)

3.1.1 Detailed Description

Class for managing a collection of daily weather forecasts.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ComplexForecast() [1/3]

```
ComplexForecast::ComplexForecast (
            const SimpleForecast * forecastsArray,
            std::size_t count )
```

```
@brief Constructor with initialization of the specified number of forecasts.
```

Parameters

forecastsArray	Array of daily forecasts.
count	The number of forecasts.

Exceptions

std::invalid_argument	If the number of predictions is less than 1.

3.1.2.2 ComplexForecast() [2/3]

```
ComplexForecast::ComplexForecast (
            const ComplexForecast & other )
```

```
@brief Copying constructor.
```

Creates a copy of another 'ComplexForecast` object.

Parameters

other	Another `ComplexForecast` object to copy.

3.1.2.3 ComplexForecast() [3/3]

```
ComplexForecast::ComplexForecast (
            ComplexForecast && other )  [noexcept]
```

```
@brief Moving constructor.
```

Moves data from another 'ComplexForecast` object.

Parameters

other	Another `ComplexForecast` object to move.

3.1.2.4 ~ComplexForecast()

```
ComplexForecast::~ComplexForecast ( )
```

```
@brief Destructor.
```

Frees up dynamically allocated memory.

3.1.3 Member Function Documentation

3.1.3.1 addForecast()

```
void ComplexForecast::addForecast (
            const SimpleForecast & forecast )
```

```
@brief Add a new daily forecast to the collection.
```

Parameters

forecast	New daily forecast. `@throw std::bad_alloc If memory could not be allocated for the new forecast.`

3.1.3.2 findColdestDay()

```
SimpleForecast ComplexForecast::findColdestDay ( ) const
```

@brief Find the coldest day.

Returns

Forecast with the lowest average temperature.

Exceptions

std::runtime_error	If the forecast collection is empty.

3.1.3.3 findNearestSunnyDay()

```
SimpleForecast ComplexForecast::findNearestSunnyDay (
            long currentTimestamp ) const
```

@brief Find the nearest sunny day from the current date.

Parameters

currentTimestamp	Current date (unix timestamp).

Returns

The nearest forecast with sunny weather.

Exceptions

std::runtime_error	If there is no nearest sunny day.

3.1.3.4 operator=() [1/2]

```
ComplexForecast & ComplexForecast::operator= (
            ComplexForecast && other )  [noexcept]
```

@brief Assignment operator to move.

Parameters

other	Another `ComplexForecast` object to move.

Returns

Link to the current `ComplexForecast` object.

3.1.3.5 operator=() [2/2]

```
ComplexForecast & ComplexForecast::operator= (
            const ComplexForecast & other )
```

Assignment operator for copying.

Parameters

other	Another `ComplexForecast` object to copy.

Returns

Link to the current `ComplexForecast` object.

3.1.3.6 operator[]()

```
SimpleForecast & ComplexForecast::operator[] (
            std::size_t index )
```

```
 @brief Get the forecast for the index.
```

Parameters

index	The index of the forecast in the collection.

Returns

A reference to the `SimpleForecast` object.

Exceptions

std::out_of_range	If the index goes beyond the bounds of the array.

3.1.3.7 removeForecast()

```
void ComplexForecast::removeForecast (
            std::size_t index )
```

@brief Delete the index forecast.

Parameters

index	The forecast index to delete.

Exceptions

std::out_of_range	If the index goes beyond the bounds of the array.

3.1.4 Friends And Related Symbol Documentation

3.1.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            const ComplexForecast & forecast )  [friend]
```

The output statement to the stream.

Parameters

out	Output stream.
forecast	The ComplexForecast object.

Returns

Link to the output stream.

3.1.4.2 operator>>

```
std::istream & operator>> (
            std::istream & in,
            ComplexForecast & forecast )  [friend]
```

@brief is an input statement from a stream.

Parameters

in	Input stream.
forecast	The ComplexForecast object.

Returns

Link to the input stream.

The documentation for this class was generated from the following files:

- WeatherReport/include/WeatherReport/ComplexForecast.h
- WeatherReport/source/ComplexForecast.cpp

3.2 SimpleForecast Class Reference

Class for presenting the daily weather forecast.

```
#include <SimpleForecast.h>
```

Public Member Functions

- SimpleForecast ()
- SimpleForecast (long timestamp, float morningTemp, float dayTemp, float eveningTemp, std::string weather, float precipitation)
- SimpleForecast (long timestamp, float morningTemp, float dayTemp, float eveningTemp, float precipitation)
- long getTimestamp () const
- float getMorningTemp () const
- float getDayTemp () const
- float getEveningTemp () const
- std::string getWeather () const
- float getPrecipitation () const
- void setTimestamp (long timestamp)
- void setMorningTemp (float temp)
- void setDayTemp (float temp)

  Set the temperature during the day.
- void setEveningTemp (float temp)

Set the temperature in the evening.

- void setWeather (const std::string &weather)

  Set the weather phenomenon.
- void setPrecipitation (float precipitation)

  Set the amount of precipitation.
- bool isInvalidForecast () const

  Check if the forecast is incorrect.
- float getAverageTemp () const

  Calculate the average temperature per day.
- SimpleForecast & operator+= (const SimpleForecast &other)

  Operator += to combine two forecasts for the same date.
- bool operator< (const SimpleForecast &other) const

  Operator < for comparing forecasts by date.
- bool operator== (const SimpleForecast &other) const

Static Public Member Functions

- static std::string getReadableDate (long timestamp)

  Get the date in a readable format.

Friends

- std::istream & operator>> (std::istream &in, SimpleForecast &forecast)

  is the input operator for the forecast.
- std::ostream & operator<< (std::ostream &out, const SimpleForecast &forecast)

  is the output operator for the forecast.

3.2.1 Detailed Description

Class for presenting the daily weather forecast.

The class stores weather data for one day, including the date, temperatures in the morning, afternoon and evening, weather phenomenon and precipitation. It also includes methods for verifying the correctness of the data.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 SimpleForecast() [1/3]

```
SimpleForecast::SimpleForecast ( )
```

```
 @brief is the default constructor.
```

Initializes the object with the initial values: zero date, zero temperature, sunny and no precipitation.

3.2.2.2 SimpleForecast() [2/3]

```
SimpleForecast::SimpleForecast (
            long timestamp,
            float morningTemp,
            float dayTemp,
            float eveningTemp,
            std::string weather,
            float precipitation )
```

@brief Constructor with parameters.

Initializes the object with the specified values of date, temperature, weather phenomenon and precipitation.

Parameters

timestamp	Forecast date (unix timestamp).
morningTemp	Temperature in the morning.
dayTemp	Daytime temperature.
eveningTemp	Temperature in the evening.
weather	Weather phenomenon (sunny, cloudy, rain, snow).
precipitation	Precipitation amount (in mm).

Exceptions

std::invalid_argument	If temperatures exceed the permissible limits of [-273, +60] degrees or precipitation exceeds 1500 mm.

3.2.2.3 SimpleForecast() [3/3]

```
SimpleForecast::SimpleForecast (
            long timestamp,
            float morningTemp,
            float dayTemp,
            float eveningTemp,
            float precipitation )
```

```
 @brief Constructor with automatic weather phenomenon selection.
```

If precipitation is greater than 0, rain or snow is selected depending on the temperature.

Parameters

timestamp	Forecast date (unix timestamp).
morningTemp	Temperature in the morning.
dayTemp	Daytime temperature.
eveningTemp	Temperature in the evening.
precipitation	Precipitation amount (in mm).

Exceptions

std::invalid_argument	If temperatures exceed the permissible limits of [-273, +60] degrees or precipitation exceeds 1500 mm.

3.2.3 Member Function Documentation

3.2.3.1 getAverageTemp()

```
float SimpleForecast::getAverageTemp ( ) const
```

Calculate the average temperature per day.

Returns

Average temperature per day (in degrees Celsius).

3.2.3.2 getDayTemp()

```
float SimpleForecast::getDayTemp ( ) const
```

```
 @brief Get the temperature in the afternoon.
```

Returns

Daytime temperature (in degrees Celsius).

3.2.3.3 getEveningTemp()

```
float SimpleForecast::getEveningTemp ( ) const
```

```
 @brief Get the temperature in the evening.
```

Returns

Temperature in the evening (in degrees Celsius).

3.2.3.4 getMorningTemp()

```
float SimpleForecast::getMorningTemp ( ) const
```

```
 @brief Get the temperature in the morning.
```

Returns

Temperature in the morning (in degrees Celsius).

3.2.3.5 getPrecipitation()

```
float SimpleForecast::getPrecipitation ( ) const
```

```
 @brief Get the amount of precipitation.
```

Returns

Precipitation amount (in mm).

3.2.3.6 getReadableDate()

```
std::string SimpleForecast::getReadableDate (
            long timestamp )  [static]
```

Get the date in a readable format.

Returns

Date in string format.("%Y-%m-%d %H:%M:%S")

3.2.3.7 getTimestamp()

```
long SimpleForecast::getTimestamp ( ) const
```

```
 @brief Get the forecast date.
```

Returns

Forecast date (unix timestamp).

3.2.3.8 getWeather()

```
std::string SimpleForecast::getWeather ( ) const
```

```
 @brief Get a weather phenomenon.
```

Returns

Weather phenomenon (sunny, cloudy, rain or snow).

3.2.3.9 isInvalidForecast()

```
bool SimpleForecast::isInvalidForecast ( ) const
```

Check if the forecast is incorrect.

An erroneous forecast is considered if there is precipitation in sunny or cloudy weather, or if the temperature is above 0 in snow.

Returns

true If the forecast is incorrect.

false If the forecast is correct.

3.2.3.10 operator+=()

```
SimpleForecast & SimpleForecast::operator+= (
            const SimpleForecast & other )
```

Operator += to combine two forecasts for the same date.

When combining, the average value of temperatures and precipitation is taken, and the worst of the two is selected for a weather phenomenon.

Parameters

other	Another forecast for the union.

Returns

Link to the updated object of this forecast.

Exceptions

std::invalid_argument	If the forecasts relate to different dates.

3.2.3.11 operator<()

```
bool SimpleForecast::operator< (
            const SimpleForecast & other ) const
```

Operator < for comparing forecasts by date.

Parameters

other	Another forecast for comparison.

Returns

true If the date of the current forecast is less than the date of another forecast.

3.2.3.12 operator==()

```
bool SimpleForecast::operator== (
            const SimpleForecast & other ) const
```

```
 @brief Operator == for comparing forecasts by date.
```

Parameters

other	Another forecast for comparison.

Returns

true If the date of both forecasts is the same.

3.2.3.13 setDayTemp()

```
void SimpleForecast::setDayTemp (
            float temp )
```

Set the temperature during the day.

Parameters

temp	Daytime temperature (in degrees Celsius).

3.2.3.14 setEveningTemp()

```
void SimpleForecast::setEveningTemp (
            float temp )
```

Set the temperature in the evening.

Parameters

temp	Temperature in the evening (in degrees Celsius).

3.2.3.15 setMorningTemp()

```
void SimpleForecast::setMorningTemp (
            float temp )

 @brief Set the temperature in the morning.
```

Parameters

temp	Temperature in the morning (in degrees Celsius).

3.2.3.16 setPrecipitation()

```
void SimpleForecast::setPrecipitation (
            float precipitation )
```

Set the amount of precipitation.

Parameters

precipitation	Precipitation amount (in mm).

3.2.3.17 setTimestamp()

```
void SimpleForecast::setTimestamp (
            long timestamp )

 @brief Set the forecast date.
```

Parameters

timestamp	Forecast date (unix timestamp).

3.2.3.18 setWeather()

```
void SimpleForecast::setWeather (
            const std::string & weather )
```

Set the weather phenomenon.

Parameters

weather	Weather phenomenon (sunny, cloudy, rain, snow).

3.2.4 Friends And Related Symbol Documentation

3.2.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            const SimpleForecast & forecast )  [friend]
```

is the output operator for the forecast.

Parameters

out	Output stream.
forecast	The SimpleForecast object for data output.

Returns

Link to the output stream.

3.2.4.2 operator>>

```
std::istream & operator>> (
            std::istream & in,
            SimpleForecast & forecast )  [friend]
```

is the input operator for the forecast.

Parameters

in	the input stream.
forecast	The SimpleForecast object for data entry.

Returns

Link to the input stream.

The documentation for this class was generated from the following files:

- WeatherReport/include/WeatherReport/SimpleForecast.h
- WeatherReport/source/SimpleForecast.cpp

Chapter 4

File Documentation

4.1 ComplexForecast.h

```
00001 #ifndef COMPLEXFORECAST_H
00002 #define COMPLEXFORECAST_H
00003
00004 #include <WeatherReport/SimpleForecast.h>
00005 #include <stdexcept>
00006
00010 class ComplexForecast {
00011 private:
00012     SimpleForecast* forecasts;
00013     std::size_t size;
00014     std::size_t capacity;
00015
00020     void resize();
00021
00022 public:
00027     ComplexForecast();
00028
00037     ComplexForecast(const SimpleForecast* forecastsArray, std::size_t count);
00038
00046     ComplexForecast(const ComplexForecast& other);
00047
00055     ComplexForecast(ComplexForecast&& other) noexcept;
00056
00063     ComplexForecast& operator=(const ComplexForecast& other);
00064
00071     ComplexForecast& operator=(ComplexForecast&& other) noexcept;
00072
00078     ~ComplexForecast();
00079
00087     void addForecast(const SimpleForecast& forecast);
00088
00097     SimpleForecast& operator[](std::size_t index);
00098
00106     void removeForecast(std::size_t index);
00107
00115     SimpleForecast findColdestDay() const;
00116
00125     SimpleForecast findNearestSunnyDay(long currentTimestamp) const;
00126
00130     void removeInvalidForecasts();
00131
00135     void sortForecasts();
00136
00144     friend std::ostream& operator<<(std::ostream& out, const ComplexForecast& forecast);
00145
00153     friend std::istream& operator>>(std::istream& in, ComplexForecast& forecast);
00154 };
00155
00156 #endif // COMPLEXFORECAST_H
```

4.2 SimpleForecast.h

```
00001 #ifndef SIMPLEFORECAST_H
00002 #define SIMPLEFORECAST_H
```

```
00003
00004 #include <string>
00005 #include <iostream>
00006
00013 class SimpleForecast {
00014 private:
00015     long timestamp;
00016     float morningTemp;
00017     float dayTemp;
00018     float eveningTemp;
00019     std::string weather;
00020     float precipitation;
00021
00022 public:
00028     SimpleForecast();
00029
00044     SimpleForecast(long timestamp, float morningTemp, float dayTemp, float eveningTemp, std::string
      weather,
00045                    float precipitation);
00046
00060     SimpleForecast(long timestamp, float morningTemp, float dayTemp, float eveningTemp, float
      precipitation);
00061
00062     // --- Getters ---
00068     [[nodiscard]] long getTimestamp() const;
00069
00075     [[nodiscard]] float getMorningTemp() const;
00076
00082     [[nodiscard]] float getDayTemp() const;
00083
00089     [[nodiscard]] float getEveningTemp() const;
00090
00096     [[nodiscard]] std::string getWeather() const;
00097
00103     [[nodiscard]] float getPrecipitation() const;
00104
00110     static std::string getReadableDate(long timestamp);
00111
00112     // --- Setters ---
00118     void setTimestamp(long timestamp);
00119
00125     void setMorningTemp(float temp);
00126
00132     void setDayTemp(float temp);
00133
00139     void setEveningTemp(float temp);
00140
00146     void setWeather(const std::string &weather);
00147
00153     void setPrecipitation(float precipitation);
00154
00155     // --- Other methods ---
00164     [[nodiscard]] bool isInvalidForecast() const;
00165
00171     [[nodiscard]] float getAverageTemp() const;
00172
00183     SimpleForecast &operator+=(const SimpleForecast &other);
00184
00185     // --- Input and output operators ---
00193     friend std::istream &operator>>(std::istream &in, SimpleForecast &forecast);
00194
00202     friend std::ostream &operator<<(std::ostream &out, const SimpleForecast &forecast);
00203
00204     // --- Comparison operators ---
00211     bool operator<(const SimpleForecast &other) const;
00212
00219     bool operator==(const SimpleForecast &other) const;
00220 };
00221
00222 #endif // SIMPLEFORECAST_H
```