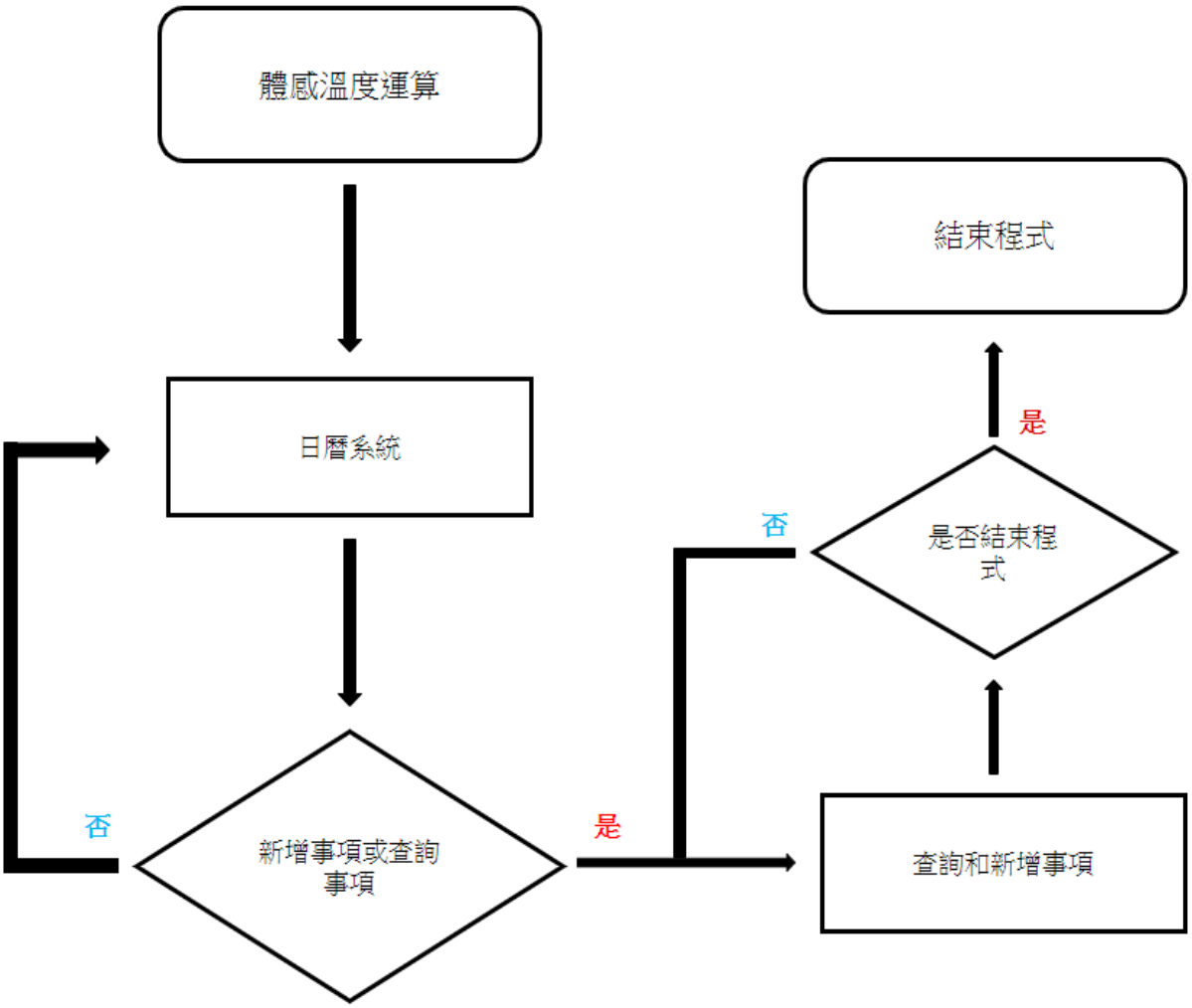
台北科技大學
109 學年度第二學期
物件導向課程
期末報告

指導教授：黃柏鈞
109360723　電子一丙　王裕誠

題目動機

由於台灣氣候潮濕多變，因此體感溫度與日常氣溫會有些差異存在，藉由原本天氣及時預報系統的資料建立，並搭配此課堂所學的繼承和封裝概念建構程式，實現體感溫度的功能提供給使用者快速提醒當天天氣以及溫差差異。

流程圖



程式介紹

標頭檔：

```
#include <iostream>
#include<string>
#include<ctime>
#include"weather.h"
#include"event.h"
using namespace std;
```

本次的程式會使用 weather.h 和 event.h 標頭檔

主程式：

```
//feels_like_temperature
bool* temperature_scale = new bool;
double* main_humidity = new double;
double* main_temperature = new double;
string temperature_scale℉℃[2] = { "℉" ,"℃" };
weather test;
int weather_chose;

//calendar
int calendar_chose;
time_t variable1 = time(NULL);
struct tm* calendar_day;
int Event_i[12][32] = { 0 };
string find_event;
size_t find_event_temp_i;

//Event
event test_1;
bool* things_to_do_chose = new bool;
bool* thiings_to_do_a = new bool;
int* things_to_do_chose_add = new int;
```

宣告變數

feels_like_temperature

```
feels_temperature:
    {
        cout << "請問要使用什麼溫標？\n\n0)攝氏℉   1)華氏℃" << endl;
        cin >> *temperature_scale;
        test.set_temperature_scale(*temperature_scale);
        if (*temperature_scale == 0)cout << "已設定為攝氏℉\n" << endl;
        else cout << "已設定為華氏℃\n" << endl;

        cout << "輸入相對濕度(%)和溫度" << temperature_scale℉℃[*temperature_scale] << "\n\n相對濕度(%)：";
        cin >> *main_humidity;
        cout << "溫度" << temperature_scale℉℃[*temperature_scale] << "：";
        cin >> *main_temperature;
    temperature_value:
        {
            test.set_All_value(*main_humidity, *main_temperature);
            cout << "\n體感溫度為" << test.feels_like_temperature() << temperature_scale℉℃[*temperature_scale] << endl;
            cout << "\n小提醒：" << test.dectect() << endl;
        }
        cout << "\n0)將溫標轉換為" << temperature_scale℉℃[!*temperature_scale] << "  1)執行日曆" << "  2)結束程式" << "  3)重新執行" << endl;
        cin >> weather_chose;
```

feels_temperature 到 feels_temperature_value 使用者先決定使用什麼類型的溫標，並且將結果記錄到 temperature_scale 以便之後輸出，接著使用者輸入當時濕度和氣溫，透過使用者輸入的參數，我們可以透過 weather.h 標頭檔裡面的 weather 類別提供的 feels_like_temperature 成員函式來進行運算，進而得到體感溫度的值。

Calendar

```
calender:
    {
        calendar_day = localtime(&variable1);
        cout << "\n現在日期" << calendar_day->tm_year + 1900 << "年" << calendar_day->tm_mon + 1 << "月" <<
            calendar_day->tm_mday << "日" << calendar_day->tm_hour << "時" << calendar_day->tm_min << "分\n" << endl;
        cout << "0)查看今日待辦事項  1)回到體感溫度  2)查看其他日期待辦事項  3)尋找事項  4)結束程式\n\n";
        cin >> calendar_chose;
```

calandar 透過 ctime 這個 API 的函式我們可以得到現在使用者的時間，並且將使用者現在的年、月、日....等資料個別放入 tm 的結構成員裡面。

things_to_do

```
things_to_do:
    {
        if (test_1.check_event(calendar_day->tm_mon,calendar_day->tm_mday)) {
            cout << "本日未偵測到事項，是否要新增？\n\n0)否，回到日曆系統  1)是(最大新增數量10個)\n";
            cin >> *things_to_do_chose;
            if (!*things_to_do_chose) {
                goto calender;
            }
            else {
                goto add_event;
            add_event:
                {
                    if (Event_i[calendar_day->tm_mon][calendar_day->tm_mday] == 10) {
                        cout << "事件已達10個，回到日曆系統" << endl;
                        goto things_to_do;
                    }
                    do {
                        cout << "\n請輸入事件：" << endl;
                        //
                        cout << Event_i[calendar_day->tm_mon][calendar_day->tm_mday] + 1 << ".";
                        test_1.add_event(calendar_day->tm_mon, calendar_day->tm_mday, Event_i[calendar_day->tm_mon][calendar_day->tm_mday]);
                        Event_i[calendar_day->tm_mon][calendar_day->tm_mday] += 1;
                        if (Event_i[calendar_day->tm_mon][calendar_day->tm_mday] == 10)break;
                        cout << "\n\n0)結束新增  1)繼續新增\n";
                        cin >> *thiings_to_do_a;
                    } while (*thiings_to_do_a);
                }
                goto things_to_do;
            }
        }
        else {
            cout << "\n\n事項共有" << Event_i[calendar_day->tm_mon][calendar_day->tm_mday] << "項" << endl;
            test_1.printf_event(calendar_day->tm_mon, calendar_day->tm_mday, Event_i[calendar_day->tm_mon][calendar_day->tm_mday]);
            cout << "\n\n0)結束程式  1)回到日曆系統  2)查看其他日期待辦事項  3)新增事項\n";
```

things_to_do 使用這在這裡可以檢查自己是否有代辦事項，如果沒有程式就會詢問是否要新增，使用者也可以在這裡查詢或是新增其他日期的代辦事項，透過修改 tm 的結構成員來讓程式得知使用者要新增或是查詢哪個日期的事項。

weather.h

成員變數：

```
class weather {
private:
    double humidity;
    double temperature;
    double temperature_scale;
    int feels_like_temperature_value;
    string hint;
```

humidity 用來存放濕度，temperature 用來存放溫度，temperature_scale 用來存放目前溫標，feels_like_temperature_value 用來存放體感溫度，hint 用來存放提醒。

成員函式：

set_value

```
void weather::set_temperature_scale(bool main_temperature_scale) {
    temperature_scale = main_temperature_scale;
}
void weather::set_All_value(double main_humidity, double main_temperature) {
    humidity = main_humidity;
    temperature = main_temperature;
}
```

透過參數來指派資料成員的值。

溫標轉換

```
void weather::C_tranform_F(double& temp) {
    //set_temperature(temperature * 9 / 5 + 32);
    temp = temp * 9 / 5 + 32;
}
void weather::F_tranform_C(double& temp) {
    //set_temperature((temperature - 32) * 5 / 9);
    temp = (temp - 32) * 5 / 9;
}
```

將溫度進行攝氏華氏轉換。

體感溫度計算

```
int weather::feels_like_temperature() {
    if (!temperature_scale) {
        F_tranform_C(temperature);
        feels_like_temperature_value = (int)(1.07 * temperature) +
            (0.2 * ((humidity / 100) * 6.105 * 2.17 * ((17.2 * temperature) / (237.7 + temperature))))
            - (0.65 * 2.5) - 2.7 + 1;
        C_tranform_F(temperature);
        return feels_like_temperature_value * 9 / 5 + 32;
    }
    else feels_like_temperature_value = (int)(1.07 * temperature) +
        (0.2 * ((humidity / 100) * 6.105 * 2.17 * ((17.2 * temperature) / (237.7 + temperature))))
        - (0.65 * 2.5) - 2.7 + 1;
    return feels_like_temperature_value;
}
```

根據攝氏或是華氏來進行計算(如果為攝氏會先經過溫標轉換)。

event.h

成員變數：

```
class event {
private:
    string Event[12][32][10];
```

Event 變數用於存放事件。

成員函式：

檢測事件

```
bool event::check_event(time_t mon, time_t day) {
    if (Event[mon][day][0] == "")return 1;
    else return 0;
}
```

用於檢測當日使否有事件。

尋找事件

```
bool event::find_event(string key_word) {
    int flag = 0;
    for (size_t i0 = 0; i0 < 12; i0++) {
        for (size_t i1 = 0; i1 < 32; i1++) {
            for (size_t i2 = 0; i2 < 10; i2++) {
                if (Event[i0][i1][i2].find(key_word) == string::npos);
                else {
                    cout << i0 + 1 << "月" << i1 << "日" << Event[i0][i1][i2] << endl;
                    flag = 1;
                }
            }
        }
    }
    cout << "\n\n";
    return flag;
}
```

用於尋找事件，如果 find 函式找到該日的事件符合關鍵字就會印出。

ctime

ctime 是一個從 C 語言傳遞下來的 API(time.h)，所以這個函基本都是使用結構。

型別

type
time_t

Time type

Alias of a fundamental arithmetic type capable of representing times, as those returned by function time.

在這個 API 裡面是用來存放秒數的的型別，時常和 time 搭配使用。

type
struct tm

Time structure

Structure containing a calendar date and time broken down into its components.

The structure contains nine members of type int (in any order), which are:

這是一個結構的型別，裡面有很多的資料成員，像是 tm_day、tm_mon、tm_sec.....經常搭配 localtime 來使用。

函式

function
time

<ctime>

```
time_t time (time_t* timer);
```

Get current time

Get the current calendar time as a value of type time_t.

計算 1970 年 1 月 1 號凌晨 00：00 到現在時間的秒數，回傳型別是 time_t，這裡的參數可以使用 NULL。

localtime

<ctime>

```
struct tm * localtime (const time_t * timer);
```

Convert time_t to tm as local time

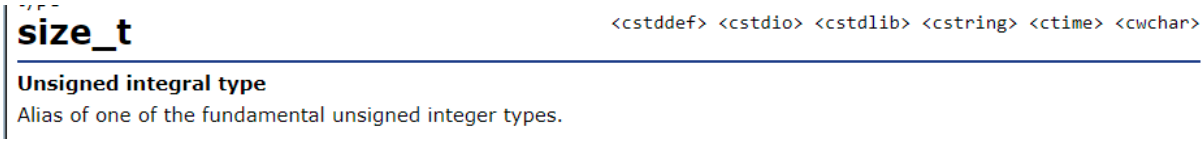
Uses the value pointed by *timer* to fill a tm structure with the values that represent the corresponding time, expressed for the local timezone.

localtime 可以將參數 time_t 的經過計算放入一個 tm 的結構裡面。

string

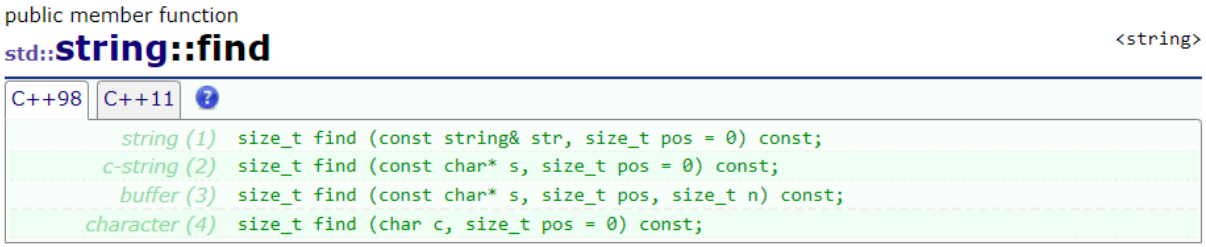
string 是屬於一種類別的 API。

型別：

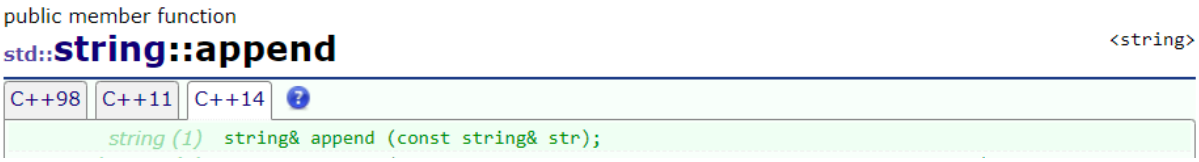


用來存放字串長度的型別。

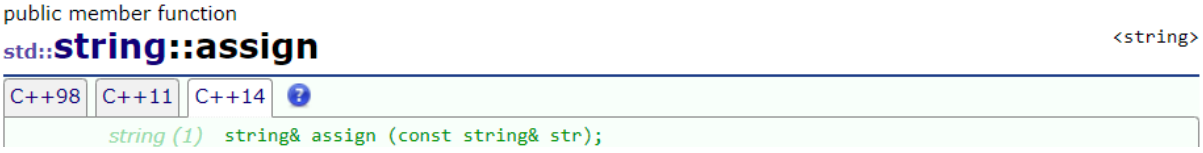
函式：



find 函式用來尋找字串裡面的關鍵字，參數放入要尋找的關鍵字，這個函式會回傳該字串的位置，如果找不到該字串函釋會回傳一個型別為 size_t 的-1(但是根據環境的不同傳回的值不一定是-1，我們可以用 string::npos)。



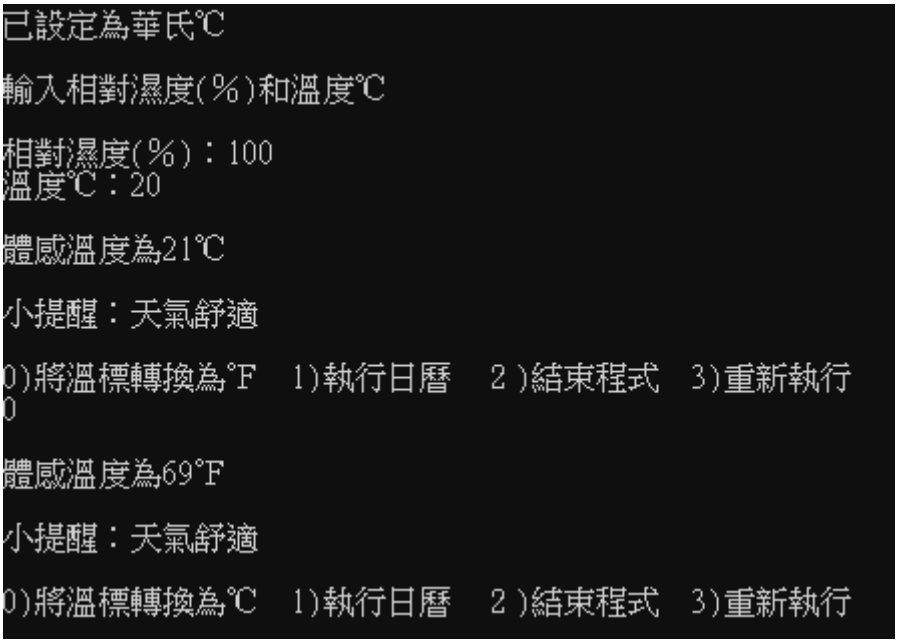
append 可以在字串後面接上作為參數的字串。



assign 可以重新指派新的字串給 string 的變數

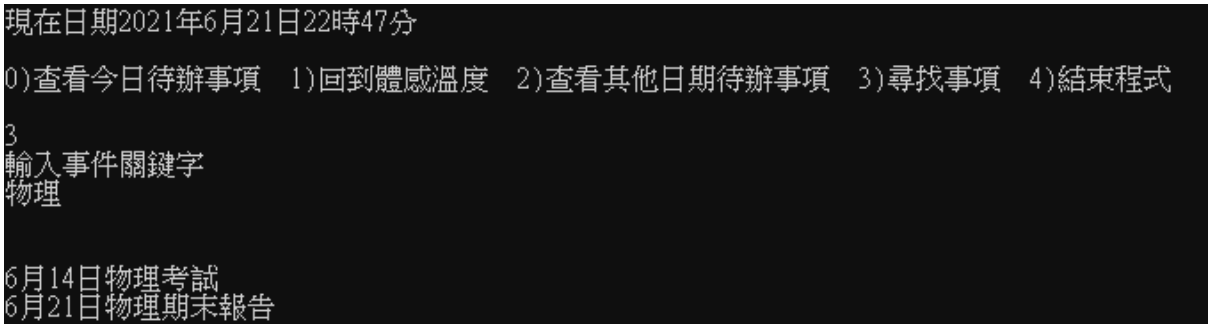
其他功能：

溫標轉換



透過 weather.h 裡面的函式我們可以將使用者輸入的資料轉換成不同溫標的溫度。

事件查詢



使用者可以透過輸入關鍵字來查詢不同日期的事件。

成果展示：

```
請問要使用什麼溫標？

0)攝氏℉    1)華氏℃
1
已設定為華氏℃

輸入相對濕度(%)和溫度℃

相對濕度(%)：100
溫度℃：20

體感溫度為21℃

小提醒：天氣舒適

0)將溫標轉換為℉   1)執行日曆   2 )結束程式   3)重新執行
1

現在日期2021年6月21日21時7分

0)查看今日待辦事項   1)回到體感溫度   2)查看其他日期待辦事項   3)尋找事項   4)結束程式

0
本日未偵測到事項，是否要新增？

0)否，回到日曆系統   1)是(最大新增數量10個)
1

請輸入事件：
1.test


0)結束新增   1)繼續新增
0


事項共有1項
1.test


0)結束程式   1)回到日曆系統   2)查看其他日期待辦事項   3)新增事項
1

現在日期2021年6月21日21時7分

0)查看今日待辦事項   1)回到體感溫度   2)查看其他日期待辦事項   3)尋找事項   4)結束程式

3
輸入事件關鍵字
test


6月21日test



現在日期2021年6月21日21時7分

0)查看今日待辦事項   1)回到體感溫度   2)查看其他日期待辦事項   3)尋找事項   4)結束程式

4
再會!
```

結語

這次的程式我在網路上學找了很多不同類型的程式，也詢問了在這方面有經驗的人，在這個過程中我學會了使用各種不同的 API，也是我第一次看 API 的文件，為了補充自己知識不足的地方我又繼續參考了很多的文章，這一次的專題中有一些遺憾，沒有加入爬蟲來自動抓取資訊，這樣就不用使用者浪費時間慢慢輸入，還有圖形介面，C++ Qt 的介面看起來比 cmd 更加的人性化。

參考資料

體感溫度：

1.https://zh.wikipedia.org/wiki/體感溫度

2.https://taiwan168.timelog.to/a215103082

3.https://health.businessweekly.com.tw/AArticle.aspx?id=ARTL003004286

C++相關參考資料：

1.http://kaiching.org/pydoing/cpp.html

2.http://kaiching.org/pydoing/cpp-guide/unit-11-encapsulation.html

3.http://kaiching.org/pydoing/cpp-guide/unit-9-function.html

4.http://kaiching.org/pydoing/cpp-guide/unit-4-pointer-and-reference.html

5.http://kaiching.org/pydoing/cpp-guide/unit-13-header.html

6.http://kaiching.org/pydoing/cpp/cpp-goto.html

7.https://cpproadadvanced.blogspot.com/

8.https://cpproadadvanced.blogspot.com/2019/12/c-cc-timeh.html

9.https://cpproadadvanced.blogspot.com/2019/10/c-struct.html

10.https://cpproadadvanced.blogspot.com/2020/04/c-c-c.html

API 參考資料：

1.https://www.cplusplus.com/