Національний університет “Львівська політехніка”

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Об’єктно-орієнтоване програмування»**

**На тему:  
«**Середовище для роботи з погодними показниками**»**

Студента групи   *ПЗ-23*

спеціальності 6.121

“Інженерія програмного забезпечення”

*Юрги В. В.*

Керівник: доцент кафедри ПЗ,

к.т.н., доцент Коротєєва Т. О.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів\_\_\_Оцінка ECTS\_\_\_

Члени комісії          \_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів 20*21*

**ЗАВДАННЯ**

до курсової роботи з дисципліни

«Об’єктно-орієнтоване програмування»

студента групи ПЗ-23 Юрги Володимира

Тема: « Середовище для роботи з погодними показниками »

**Завдання** (Варіант 6)

Створити таблицю у візуальному середовищі

Рік | Місяць | Число | t | Тиск | Вологість | Напрям вітру

1) Визначити середню вологість за місяць.

2) Визначити дні з найвищою t та найменшим тиском.

3) Визначити дні, протягом яких, напрям вітру не змінювався.

4) За заданим періодом, визначити середню температуру протягом цього періоду та дні з найвищою вологістю.

5) Знайти період, в якому тиск змінювався в межах ±1,43%, а t – 5,673%.

6) Побудувати засобами візуального середовища графіки t, тиску та вологості

7) Спрогнозувати погоду на наступний місяць з врахуванням зміни сезону року.

Для класу створити: 1) Конструктор за замовчуванням; 2) Конструктор з параметрами; 3) конструктор копій; 4) перевизначити операції >>, << для зчитування та запису у файл.

|  |  |  |
| --- | --- | --- |
| **№з/п** | **Зміст завдання** | **Дата** |
| 1 | Здійснити аналiтичний огляд лiтератури за заданою темою та обгрунтувати вибір інструментальних засобів реалізації. | **13.09** |
| 2 | Побудова UML діаграм | **14.11** |
| 3 | Розробка алгоритмів реалізації | **15.11** |
| 4 | Реалізація завдання (кодування) | **15-17.11** |
| 5 | Формування інструкції користувача | **20.11** |
| 6 | Оформлення звіту до курсової роботи згідно з вимогами Міжнародних стандартів, дотримуючись такої структури:  ·       зміст;  ·        алгоритм розв‘язку задачі у покроковому представленні;  ·        діаграми UML класів, прецедентів, послідовності виконання;  ·        код розробленої програми з коментарями;  ·        протокол роботи програми для кожного пункту завдання  ·        інструкція користувача та системні вимоги;  ·        опис виняткових ситуацій;  ·        структура файлу вхідних даних;  ·        висновки;  ·        список використаних джерел. | **21.11 22.11** |

Завдання прийнято до виконання: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Юрга В.В.)

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Коротєєва Т. О./

**Зміст**

[**Алгоритм розв‘язку задачі у покроковому представленні** 4](#_Toc88483380)

[**Діаграми UML класів, Діаграми UML класів, прецедентів, послідовності виконання.** 9](#_Toc88483381)

[**Код розробленої програми** 11](#_Toc88483382)

[**Протокол роботи програми** 21](#_Toc88483383)

[**Інструкція користувача** 28](#_Toc88483384)

[**Опис виняткових ситуацій** 33](#_Toc88483385)

[**Структура файлу вхідних даних** 34](#_Toc88483386)

[**Висновки** 35](#_Toc88483387)

[**Список використаної літератури** 35](#_Toc88483388)

# **Алгоритм розв‘язку задачі у покроковому представленні**

Aлгоритм розв'язку задачі загалом:

STEP1. Зчитати дані з файлу або форми.  
STEP2. Виконати необхідні дії відповідно завдання.  
STEP3. Зберегти роботу в тому самому або новому файлі.

**Алгоритм Q.**

Алгоритм швидкого сортування для впорядкування за датою вектора погоди.

Дано X = { x1, …, xn }; start\_pointer - індекс початку масиву X, end\_pointer - індекс кінця масиву Х, k – число для порівняння, i – тимчасовий індекс для проходження по масиву з початку, j – тимчасовий індекс для проходження по масиву з кінця.

Q1. Якщо start\_pointer > end\_pointer то закінчити алгоритм;

Q2. Ініціалізовуємо змінні i, j, compare\_value значеннями start\_pointer, end\_pointer, X[(start\_pointer+end\_pointer)/2].

Q3. Поки i <= j повторювати кроки Q4-Q6

Q4. Поки Х[i] < compare\_value збільшувати і на 1.

Q5. Поки Х[j] > compare\_value зменшувати j на 1.

Q6. Якщо i <= j міняємо місцями значення Х[i] <-> Х[j] та проводимо операції i += 1 та j -= 1

Q7. Виконаємо алгоритм ще раз з параметрами X = { x1 ,…, xn }; start\_pointer = start\_pointer, end\_pointer = j

Q8. Виконаємо алгоритм ще раз з параметрами X = { x1 ,…, xn }; start\_pointer = i, end\_pointer = end\_pointer

Q9. Кінець. Вихід.

**Алгоритм Rev.**

Алгоритм для перевпорядкування вектора в протилежному порядку.

Дано vector = {v1, …, vn } start\_pointer - індекс початку вектора V, end\_pointer - індекс кінця вектора V, i – тимчасовий індекс для проходження по масиву з початку, j – тимчасовий індекс для проходження по масиву з кінця.

Rev1. Ініціалізовуємо змінні i, j значеннями start\_pointer та end\_pointer.

Rev2. Поки i<=j повторювати крок Rev3.

Rev3. Міняємо місцями елементи елементи V[i] та V[j]. Після цього I зменшуємо, а j збільшуємо на одиницю.

Rev4. Кінець. Вихід.

**Алгоритм Ave.**

Алгоритм пошуку середньої вологості за місяць.

V-вектор елементів погоди. Month – шуканий місяць.

Ave1. Ініціалізовуємо початкові змінні. Result, Sum=0, count=0, i=0, змінній end присвоюємо останній індекс масиву.

Ave2. Виконувати крок Ave3 поки i<end.

Ave3. Якщо місяць в елементі дорівнює Month, то до змінної Sum додаємо значення вологості, а змінні i та count збільшуємо на одиницю. Якщо ж ні, то збільшуємо на одиницю тільки і.

Ave4. Повертаємо змінну Res=(Sum/count). Кінець. Вихід.

**Алгоритм Mm.**

Алгоритм пошуку мінімальної температури та найвищого тиску.

V-вектор елементів погоди.

Mm1. Ініціалізовуємо початкові змінні. min, max яким присвоємо температуру та тиск першого елементу вектора відповідно, i=0, змінній end присвоюємо останній індекс масиву.

Mm2. Виконувати кроки Mm3 та Mm4 поки i<end.

Mm3. Якщо температура V[i] < min, то змінній min присвоюємо цю температуру.

Якщо тиск V[i]>max, то змінній max присвоюємо цей тиск.

Mm4. Збільшуємо і на одиницю.

Mm5. Повертаємо змінну Res=(Sum/count).

Mm6. Кінець. Вихід.

**Алгоритм Stat.**

Визначити період, протягом якого напрям вітру не змінювався.

V-вектор елементів погоди.

Stat1. Ініціалізовуємо початкові змінні. Wind, i=1, змінній end присвоюємо останній індекс масиву.

Stat2. Повторювати кроки Stat3 – Stat7 поки i<end.

Stat3. Порівнюємо дні V[i] та V[i-1]. Якщо V[i] – наступний день після V[i-1], то переходимо до кроку Stat4, якщо ні, то до Stat7.

Stat4. Порівнюємо напрям вітру V[i] та V[i-1], якщо рівні, то зберігаємо значення вітру в змінну Wind і переходимо до кроку Stat 5. Якщо ні то до кроку Stat7.

Stat5. Виводимо змінну Wind та V[i-1]. Виконуємо крок Stat6 поки напрям вітру V[i] i V[i-1] не будуть різними і поки і не перевищуватиме end.

Stat6. Виводимо напрям вітру V[i], збільшуємо і на 1.

Stat7. Збільшуємо і на 1.

Stat8. Кінець. Вихід.

**Алгоритм TP.**

За заданим періодом визначити середню температуру портягом цього періоду та найвищу вологість.

V-вектор елементів погоди.

TP1. Ініціалізовуємо початкові змінні. Average та max = 0, from присвоюємо початок періоду, змінній to присвоюємо кінець періоду. Використовуючи алгоритм Q сортуємо вектор V.

TP2. Використовуємо алгоритм Ave але для пошуку середньої температури серед елементів вектору в яких день є в межах між from та to. Результат зберігаємо в змінну Res.

TP3. Використовуємо алгоритм Mm але тільки для пошуку максимальної вологості серед елементів вектору в яких день є в межах між from та to. Результат зберігаємо в змінну max

TP4. Виводимо змінні max та Res.

TP5. Кінець. Вихід.

**Алгоритм TP.**

Знайти період, в якому тиск змінювався в межах ±1,43%, а t – 5,673%.

V-вектор елементів погоди.

TP1. Ініціалізовуємо початкові змінні from, to. Використовуючи алгоритм Q сортуємо вектор V.

TP2. Використовуємо аналогічний алгоритм до Stat, проте перевіряємо не рівність вітрів, а чи між елементами V[i-1] та V[i] тиск змінився не більш як на 1.43%, а температура на 5.673%.

TP3. Кінець. Вихід.

**Алгоритм G.**

Побудувати засобами візуального середовища графіки t, тиску та вологості.

V-вектор елементів погоди.

G1. Ініціалізовуємо змінні:

x-вектор для координат осі ОХ.

y1-вектор для координат температури по осі Оy.

y2-вектор для координат тиску по осі Оy.

y3-вектор для координат вологості по осі Оy.

Year – зберігаємо значення потрібного року для показу погоди. i=0, end присвоюємо останній індекс масиву.

G2. Повторювати крок G3 поки і<end.

G3. Якщо рік V[i] дорівнює Year, то додаємо в масиви x, y1, y2, y3 місяць, температуру, тиск та вологість V[i] відповідно. Збільшуємо і на 1.

G4. Вихід. Кінець.

**Алгоритм F.**

Спрогнозувати погоду на наступний місяць з врахуванням зміни сезону року.

V-вектор елементів погоди.

F1. Ініціалізовуємо змінні: temperature, pressure, humidity. Month та Year змінні місяця та року шуканого попереднього місяця відповідно.

F2. Шукаємо у векторі елемент погоди з роком Year та місяцем Month (алгоритм прямого пошуку) Якщо знайдено, то крок F3, якщо ні, то вивід помилки про відсутність такого місяця і переходимо до кроку F6.

F3. Зчитуємо у змінні temperature, pressure, humidity температуру тиск та вологість з знайденого елемента погоди.

F4. Якщо місяць є в межах [1;2] або [8;12], то зменшуємо показники на 10%. Якщо в інших межах ([3;7]), то збільшуємо показники на 12%.

F5. Виводимо змінні temperature, pressure, humidity.

F6. Кінець. Вихід.

# **Діаграми UML класів,** **Діаграми UML класів, прецедентів, послідовності виконання.**

Для цієї програми було використано власний клас Weather на основі, якого побудовано клас вектор. Останній ж в свою чергу вже використовується в самій програмі (оголошений в класі MainWindow), власне це головний клас програми у якому є всі слоти кнопок, сам вектор та інші потрібні змінні. (MainWindow наслідується від класу QMainWindow) (Рис 2.1)

На Рис. 2.2 зображена діаграма прецедентів з основними можливостями програми, а також зв’язками між діями.

Рис 2.3 – звичайна діаграма послідовності для користувача програми.

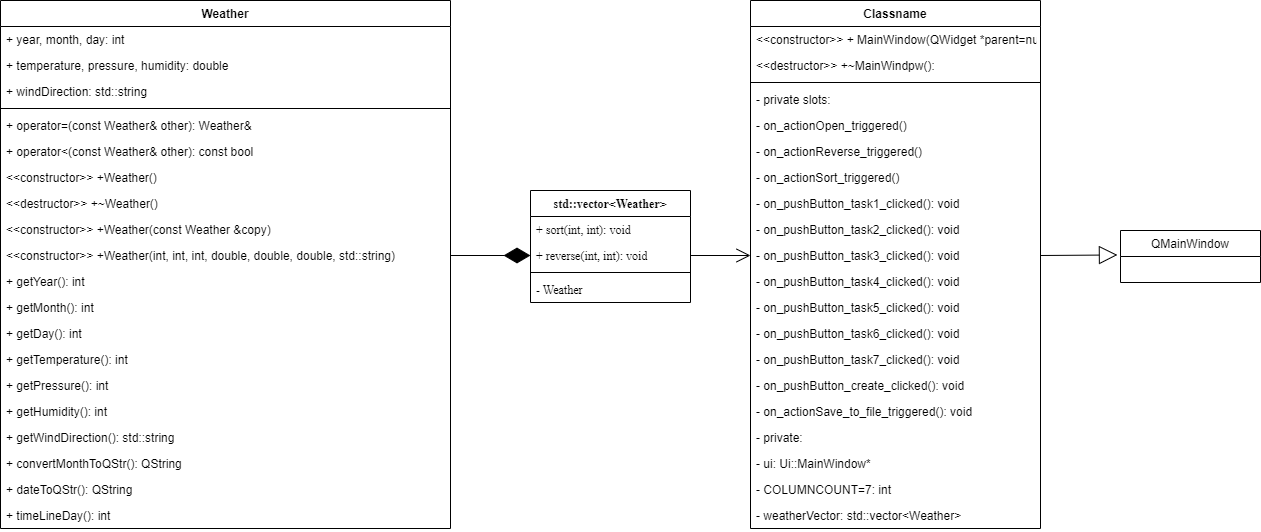


Рис 2.1 UML Діаграма класів

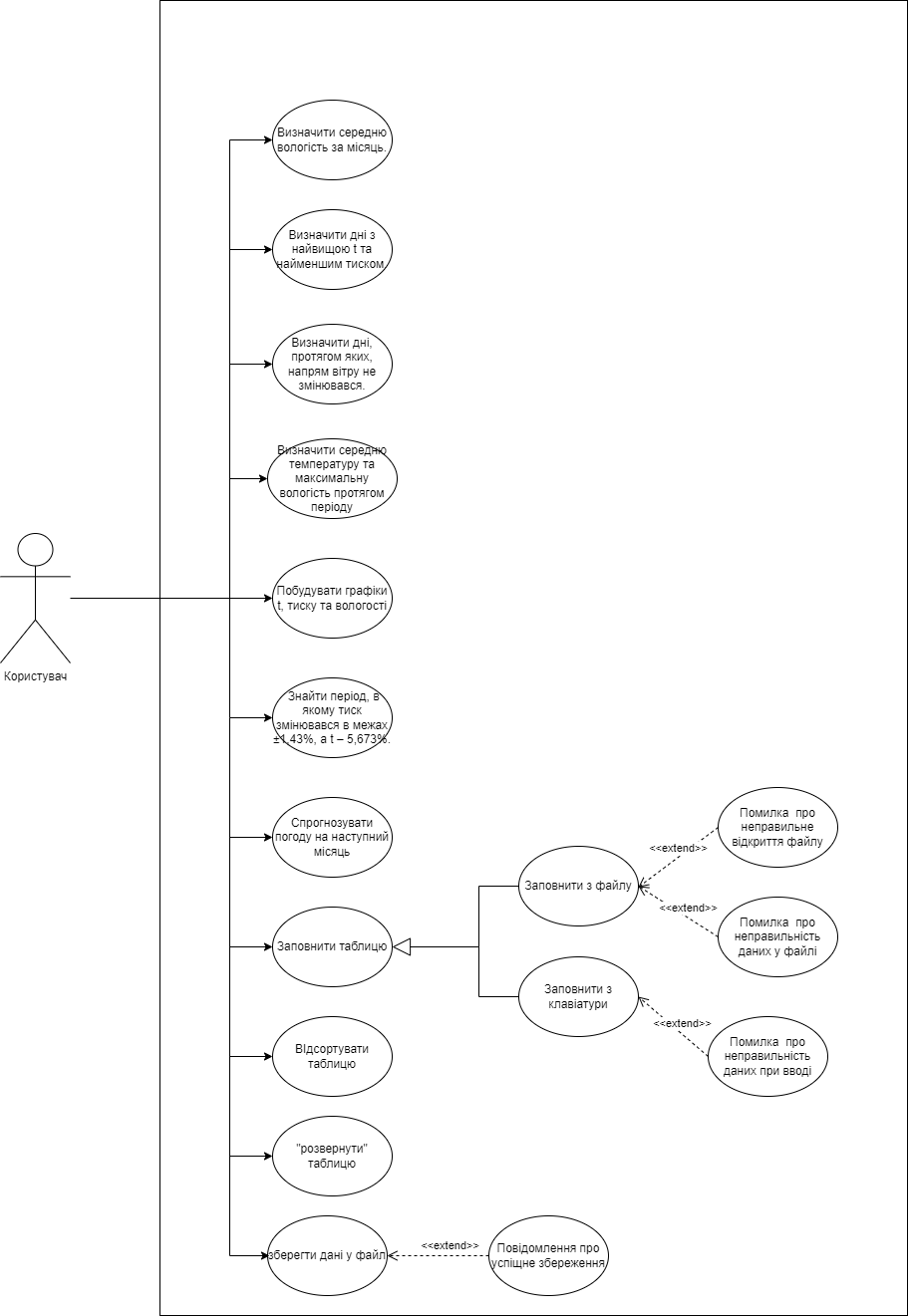


Рис 2.2 UML Діаграма прецедентів

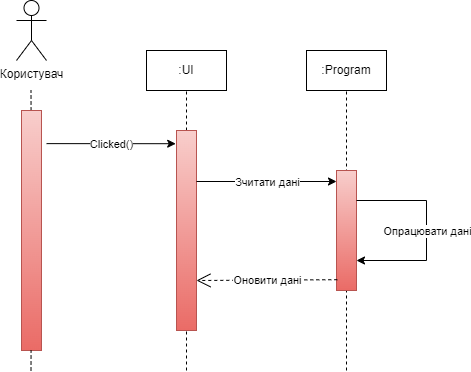


Рис 2.3 UML Діаграма послідовності

# **Код розробленої програми**

weather.h

#ifndef WEATHER\_H

#define WEATHER\_H

#include <QString>

*//using* *namespace* *std;*

*class* **Weather**

{

*public*:

**Weather**();*//коструктор* *без* *параметрів*

**Weather**(int y, int m, int d, double t, double p, double h, std::string wd);*//конструктор* *з* *параметрами*

**Weather**(*const* Weather &copy);*//конструктор* *копіювання*

~**Weather**(){};*//деструктор*

Weather& *operator*=(*const* Weather& other){*//оператор* *присвоювання*

*this*->year=other.year;

*this*->month=other.month;

*this*->day=other.day;

*this*->temperature=other.temperature;

*this*->pressure=other.pressure;

*this*->humidity=other.humidity;

*this*->windDirection=other.windDirection;

*return* \**this*;

}

bool *operator*<(*const* Weather &other) *const*;*//оператор* *порівняння* *(потрібний* *для* *виклику* *функції* *sort* *у* *векторі)*

*//* *GETERS*

int **getYear**(){*return* year;}

int **getMonth**(){*return* month;}

int **getDay**(){*return* day;}

double **getTemperature**(){*return* temperature;}

double **getPressure**(){ *return* pressure;}

double **getHumidity**(){*return* humidity;}

std::string **getWindDirection**(){*return* windDirection;}

*FUNCTIONS*

int **timeLineDay**() *const*; *//дозволяє* *дізнатись* *до* *розміщується* *цей* *день* *в* *часі*

QString **convertMonthToQStr**(); *//повертає* *перетворений* *місяць* *з* *числа* *в* *слово*

QString **dateToQStr**();*//повератє* *дати* *у* *вигляді* *[yyyy.mm.dd]*

*private*:

int year, month, day;

double temperature, pressure, humidity;

std::string windDirection;

};

#endif *//* *WEATHER\_H*

weather.cpp

#include "weather.h"

Weather::**Weather**()

{

year=0;

month=0;

day=0;

temperature = 0;

pressure=0;

humidity=0;

windDirection="none";

}

Weather::**Weather**(int y, int m, int d, double t, double p, double h, std::string wd)

{

*this*->year=y;

*this*->month=m;

*this*->day=d;

*this*->temperature = t;

*this*->pressure=p;

*this*->humidity=h;

*this*->windDirection=wd;

}

Weather::**Weather**(*const* Weather& other)

{

*this*->year=other.year;

*this*->month=other.month;

*this*->day=other.day;

*this*->humidity=other.humidity;

*this*->pressure=other.pressure;

*this*->temperature=other.temperature;

*this*->windDirection=other.windDirection;

}

QString Weather::**convertMonthToQStr**(){

*switch* (month) {

*case* 1: *return* "January";*//31*

*case* 2: *return* "February";

*case* 3: *return* "March";

*case* 4: *return* "April";

*case* 5: *return* "May";

*case* 6: *return* "June";

*case* 7: *return* "July";

*case* 8: *return* "August";

*case* 9: *return* "September";

*case* 10: *return* "October";

*case* 11: *return* "November";

*default*: *return* "December";

}

}

QString Weather::**dateToQStr**(){

QString result = "["+QString::number(year) +"." + QString::number(month)+'.'+QString::number(day)+']';

*return* result;

}

int Weather::**timeLineDay**() *const*{

int monthDays;

*switch* (month) {

*//кожен* *місяць* *буде* *повертати* *кількість* *днів* *від* *1* *січня* *до* *1* *дня* *цього* *місяця*

*case* 1: monthDays = 0; *break*;

*case* 2: monthDays = 31; *break*;

*case* 3: monthDays = 31+28; *break*;

*case* 4: monthDays = 31+28+31; *break*;

*case* 5: monthDays = 31+28+31+30; *break*;

*case* 6: monthDays = 31+28+31+30+31; *break*;

*case* 7: monthDays = 31+28+31+30+31+30; *break*;

*case* 8: monthDays = 31+28+31+30+31+30+31; *break*;

*case* 9: monthDays = 31+28+31+30+31+30+31+31; *break*;

*case* 10: monthDays = 31+28+31+30+31+30+31+31+30; *break*;

*case* 11: monthDays = 31+28+31+30+31+30+31+31+30+31; *break*;

*default*: monthDays = 31+28+31+30+31+30+31+31+30+31+30; *break*;

}

*return* year\*365 + monthDays + day;*//кількість* *днів* *від* *початку* *Ери*

}

*//оператор* *порівняння* *на* *основі* *розміщення* *дня* *в* *часовій* *лінії*

bool Weather::*operator*<(*const* Weather &other) *const*{

*return* *this*->timeLineDay()<other.timeLineDay();

}

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QString>

#include "weather.h"

*//using* *namespace* *std;*

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*//Головний* *інтерфейсний* *клас* *для* *вікна* *з* *основними* *функціями*

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **on\_actionOpen\_triggered**();

void **on\_pushButton\_task1\_clicked**();

void **on\_actionSort\_triggered**();

void **on\_actionReverse\_triggered**();

void **on\_pushButton\_task2\_clicked**();

void **on\_pushButton\_task3\_clicked**();

void **on\_pushButton\_find4\_clicked**();

void **on\_pushButton\_find5\_clicked**();

void **on\_pushButton\_find7\_clicked**();

void **on\_pushButton\_draw6\_clicked**();

void **on\_pushButton\_create\_clicked**();

void **on\_actionSave\_to\_file\_triggered**();

*private*:

Ui::MainWindow \*ui;

int COLUMNCOUNT = 7;

std::vector<Weather> weatherVector;

*//void* *operator>>(string* *filePath,* *vector<Weather>&myarr);*

};

#endif *//* *MAINWINDOW\_H*

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "weather.h"

#include <iostream>

#include <stdio.h>

#include <QString>

#include <string.h>

#include <fstream>

#include <sstream>

#include <QMessageBox>

#include <QFileDialog>

*//#include* *<fileapi.h>*

*//#include* *<stringstream>*

*//#include* *<system>*

*//#include* *<vector>*

*//using* *namespace* *std;*

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

ui->tableWidget->setColumnCount(COLUMNCOUNT);

*//ui->tableWidget->setRowCount(1);*

ui->tableWidget->setHorizontalHeaderLabels(QStringList() << "Year" << "Mounth" << "Day" << "Temperature" << "Pressure" << "Humidity" << "Wind dir.");

ui->tableWidget->horizontalHeader()->setSectionResizeMode(QHeaderView::*Stretch*);

}

std::vector<Weather> weatherVector;

bool isSortedVector = *false*;

void *operator*>>(std::string filePath, std::vector<Weather>&weatherVec){

std::ifstream file;

file.exceptions(std::ifstream::badbit | std::ifstream::failbit);

bool isOpen=*false*;

*try*{

file.open(filePath);

isOpen=*true*;

*while*(!file.eof()){

*//Subscription* *mySubscription;*

*//Person* *person;*

*//std::string* *subscriptionType;*

std::string line;

std::getline(file, *line*);

*if*(line == ""){

*continue*;

}

std::stringstream strstream(line);

std::vector<std::string> tokenVect;

std::string token;

*while*(std::getline(strstream, *token*, ' ')){

*//RemoveSpaces(token);*

tokenVect.push\_back(token);

}

Weather wthItem(stoi(tokenVect[0]),stoi(tokenVect[1]),stoi(tokenVect[2]),stod(tokenVect[3]),stod(tokenVect[4]),stod(tokenVect[5]),tokenVect[6]);

*//string* *str* *="hi";*

*//Weather* *wthItem(1,2,3,4.0,5.0,6.0,str);*

weatherVec.push\_back(wthItem);

}

}

*catch*(*const* std::ifstream::failure & ex){

*if*(!isOpen) QMessageBox::warning(NULL,"File", "Problem with file opening!\n"+QString::fromStdString(ex.*what*()));

}

*catch*(...){

QMessageBox::critical(NULL,"File", "Problem with file data!");

weatherVector.clear();

}

}

void *operator*<<(std::string filePath, std::vector<Weather>&weatherVector){

QFile file(filePath.c\_str());

*if*(file.*open*(QIODevice::*WriteOnly* | QIODevice::*Text*))

{

QTextStream stream(*&file*);

QString strYear, strMonth, strDay, strTemperature, strPressure, strHumidity, strWind;

*for*(int i=0; i<weatherVector.size(); i++){

strYear=QString::number(weatherVector[i].getYear());

strMonth=QString::number(weatherVector[i].getMonth());

strDay=QString::number(weatherVector[i].getDay());

strTemperature=QString::number(weatherVector[i].getTemperature());

strPressure=QString::number(weatherVector[i].getPressure());

strHumidity=QString::number(weatherVector[i].getHumidity());

strWind=QString::fromStdString(weatherVector[i].getWindDirection());

stream<<strYear+' '+strMonth+' '+strDay+' '+strTemperature+' '+strPressure+' '+strHumidity+' '+strWind+'\n';

}

file.*close*();

}

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

*//кнопка* *відкриття* *файлу*

void MainWindow::**on\_actionOpen\_triggered**()

{

weatherVector.clear();

std::string filePath = (QFileDialog::getOpenFileName(*this*,"Open file to read", "D://QT//Projects//ASD//CourseWork//weatherData")).toStdString();

filePath>>weatherVector;

int row=weatherVector.size();

ui->tableWidget->setRowCount(row);

*for*(int i=0; i<row; i++){

QTableWidgetItem \*item = *new* QTableWidgetItem(QString::number(weatherVector[i].getYear()));*//YEAR*

ui->tableWidget->setItem(i,0,*item*);

*//item* *=* *new* *QTableWidgetItem(QString::number(weatherVector[i].getMonth()));//MONTH*

item = *new* QTableWidgetItem(weatherVector[i].convertMonthToQStr());

ui->tableWidget->setItem(i,1,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getDay()));*//DAY*

ui->tableWidget->setItem(i,2,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getTemperature()) + " °C");*//TEMPERATURE*

ui->tableWidget->setItem(i,3,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getPressure()) + " hPA");*//PRESSURE*

ui->tableWidget->setItem(i,4,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getHumidity()) + " %");*//HUMIDITY*

ui->tableWidget->setItem(i,5,*item*);

item = *new* QTableWidgetItem(QString::fromStdString(weatherVector[i].getWindDirection()));*//WIND* *DIRECTION*

ui->tableWidget->setItem(i,6,*item*);

}

}

*//кнопка* *зберігання* *у* *файл*

void MainWindow::**on\_actionSave\_to\_file\_triggered**()

{

*//try{*

std::string filePath = (QFileDialog::getOpenFileName(*this*,"Open file to save", "D://QT//Projects//ASD//CourseWork//weatherData")).toStdString();

*//QFile* *file(file\_name);*

*//file.open(QIODevice::ReadOnly);*

filePath<<weatherVector;

QMessageBox::information(*this*,"Saving", "Saved successfully!");

}

void MainWindow::**on\_pushButton\_task1\_clicked**()*//Пощук* *середньої* *вологості*

{

int tYear=ui->spinBox\_year1->value();

int tMonth=ui->spinBox\_month1->value();

double Sum=0;

int count=0;

bool exist=*false*;

*for*(int i=0; i<weatherVector.size(); i++){

*//якщо* *потрібний* *місяць* *та* *рік* *то* *додаємо* *температуру* *до* *суми* *і* *збільшуємо* *лічильник*

*if*( (weatherVector[i].getYear() == tYear) && (weatherVector[i].getMonth() == tMonth) ){

Sum+=weatherVector[i].getHumidity();

count++;

*if*(!exist) exist = *true*;

}

}

*if*(exist) ui->plainTextEdit\_task1result->setPlainText("Finded:\t" + QString::number(Sum/count) + " %");

*else* ui->plainTextEdit\_task1result->setPlainText("There are no data for this month");

}

void MainWindow::**on\_actionSort\_triggered**()*//кнопка* *сортування*

{

sort(weatherVector.begin(), weatherVector.end());

isSortedVector=*true*;

*//обновленя* *таблиці*

*for*(int i=0; i<weatherVector.size(); i++){

QTableWidgetItem \*item = *new* QTableWidgetItem(QString::number(weatherVector[i].getYear()));*//YEAR*

ui->tableWidget->setItem(i,0,*item*);

item = *new* QTableWidgetItem(weatherVector[i].convertMonthToQStr());

ui->tableWidget->setItem(i,1,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getDay()));*//DAY*

ui->tableWidget->setItem(i,2,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getTemperature()) + " °C");*//TEMPERATURE*

ui->tableWidget->setItem(i,3,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getPressure()) + " hPA");*//PRESSURE*

ui->tableWidget->setItem(i,4,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getHumidity()) + " %");*//HUMIDITY*

ui->tableWidget->setItem(i,5,*item*);

item = *new* QTableWidgetItem(QString::fromStdString(weatherVector[i].getWindDirection()));*//WIND* *DIRECTION*

ui->tableWidget->setItem(i,6,*item*);

}

*//weatherVector.*

}

void MainWindow::**on\_actionReverse\_triggered**()*//кнопка* *розвертання* *таблиці*

{

reverse(weatherVector.begin(), weatherVector.end());

*if*(isSortedVector) isSortedVector=*false*;

*else* isSortedVector=*true*;

*for*(int i=0; i<weatherVector.size(); i++){

QTableWidgetItem \*item = *new* QTableWidgetItem(QString::number(weatherVector[i].getYear()));*//YEAR*

ui->tableWidget->setItem(i,0,*item*);

*//item* *=* *new* *QTableWidgetItem(QString::number(weatherVector[i].getMonth()));//MONTH*

item = *new* QTableWidgetItem(weatherVector[i].convertMonthToQStr());

ui->tableWidget->setItem(i,1,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getDay()));*//DAY*

ui->tableWidget->setItem(i,2,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getTemperature()) + " °C");*//TEMPERATURE*

ui->tableWidget->setItem(i,3,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getPressure()) + " hPA");*//PRESSURE*

ui->tableWidget->setItem(i,4,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getHumidity()) + " %");*//HUMIDITY*

ui->tableWidget->setItem(i,5,*item*);

item = *new* QTableWidgetItem(QString::fromStdString(weatherVector[i].getWindDirection()));*//WIND* *DIRECTION*

ui->tableWidget->setItem(i,6,*item*);

}

}

void MainWindow::**on\_pushButton\_task2\_clicked**()*//пошук* *дня* *з* *максимальною* *температурою* *та* *мінімальним* *тиском*

{

double temperatureMax = weatherVector[0].getTemperature();

double pressureMin=weatherVector[0].getPressure();

*//Finding* *numbers*

*for*(int i=0; i<weatherVector.size(); i++){

*if*(weatherVector[i].getTemperature()>temperatureMax) temperatureMax = weatherVector[i].getTemperature();

*if*(weatherVector[i].getPressure()<pressureMin) pressureMin = weatherVector[i].getPressure();

}

ui->plainTextEdit\_task2result->setPlainText("Max. temperature:\t" + QString::number(temperatureMax) + " °C\nMin. pressure:\t\t" + QString::number(pressureMin) + " hPA\n");

ui->plainTextEdit\_task2result->moveCursor(QTextCursor::*End*);

*//Finding* *this* *days*

bool findedOnce = *false*;

*for*(int i=0; i<weatherVector.size(); i++){

*if*(weatherVector[i].getTemperature()==temperatureMax && weatherVector[i].getPressure()==pressureMin){

*if*(!findedOnce){

ui->plainTextEdit\_task2result->insertPlainText("Existing date:\n");

findedOnce=*true*;

}

ui->plainTextEdit\_task2result->insertPlainText("[" + QString::number(weatherVector[i].getYear())+'.'+QString::number(weatherVector[i].getMonth())+'.'+QString::number(weatherVector[i].getDay())+"]\n");

ui->plainTextEdit\_task2result->moveCursor(QTextCursor::*End*);

}

}

*if*(!findedOnce) ui->plainTextEdit\_task2result->insertPlainText("There are no date with Max. temperature and Min. pressure");

}

void MainWindow::**on\_pushButton\_task3\_clicked**()*//пошук* *періоду* *коли* *вітер* *не* *міняв* *напрямок*

{

*if*(isSortedVector){

*//ui->plainTextEdit\_task3result*

ui->plainTextEdit\_task3result->clear();

int nowDay = 0;

bool existConsistentDays=*false*;

bool existNeedDays=*false*;

*for*(int i=1; i<weatherVector.size(); i++){

*if*(weatherVector[i].timeLineDay()==weatherVector[i-1].timeLineDay()+1){*//перевіряєм* *чи* *два* *дні* *послідовні*

existConsistentDays=*true*;

*if*(weatherVector[i].getWindDirection()==weatherVector[i-1].getWindDirection()){*//перевіряємо* *чи* *ці* *дні* *мають* *однаковий* *напрям* *вітру*

*if*(!existNeedDays){

existNeedDays=*true*;

ui->plainTextEdit\_task3result->insertPlainText("Finded same wind! ("+ QString::fromStdString(weatherVector[i-1].getWindDirection())+ ")\nThis date:\n[" + QString::number(weatherVector[i-1].getYear())+'.'+QString::number(weatherVector[i-1].getMonth())+'.'+QString::number(weatherVector[i-1].getDay())+"]\n");

ui->plainTextEdit\_task3result->moveCursor(QTextCursor::*End*);

}

ui->plainTextEdit\_task3result->insertPlainText("[" + QString::number(weatherVector[i].getYear())+'.'+QString::number(weatherVector[i].getMonth())+'.'+QString::number(weatherVector[i].getDay())+"]\n");

ui->plainTextEdit\_task3result->moveCursor(QTextCursor::*End*);

}

}

}

*if*(!existNeedDays) ui->plainTextEdit\_task3result->setPlainText("There are no one same winds");

*if*(!existConsistentDays) ui->plainTextEdit\_task3result->setPlainText("There are no cosistent days");

}

*else* ui->plainTextEdit\_task3result->setPlainText("This task need a sorted input vector.\nPlease sort by date!\n");

}

void MainWindow::**on\_pushButton\_find4\_clicked**()*//пошук* *середньої* *температури* *та* *днів* *з* *максимальною* *вологістю*

{

int pos\_from, pos\_to;

pos\_from=ui->spinBox\_from4->value()-1;

pos\_to=ui->spinBox\_to4->value()-1;

double tempSum=0, maxHumidity=0;

int tempCount=pos\_to-pos\_from+1;

*for*(int i=pos\_from; i<pos\_to; i++){

tempSum+=weatherVector[i].getTemperature();

*//tempCount++;*

*if*(weatherVector[i].getHumidity()>maxHumidity) maxHumidity=weatherVector[i].getHumidity();

}

ui->plainTextEdit\_temperature4->setPlainText("Finded av. Temperature: " + QString::number(tempSum/tempCount) + " °C");

*//пошук* *УСІХ* *днів* *з* *максимальною* *вологістю*

ui->plainTextEdit\_humidity4->setPlainText("Finded, date:\n");

*for*(int i=pos\_from; i<pos\_to; i++){

*if*(weatherVector[i].getHumidity()==maxHumidity){

ui->plainTextEdit\_humidity4->moveCursor(QTextCursor::*End*);

ui->plainTextEdit\_humidity4->insertPlainText(weatherVector[i].dateToQStr()+" with:\t" + QString::number(maxHumidity) + " %;\n");

}

}

}

void MainWindow::**on\_pushButton\_find5\_clicked**()*//пошук* *періоду* *коли* *тиск* *та* *температура* *мінялись* *на* *певну* *кількість* *відсотків*

{

int findedPeriod=*false*;

double perPressure, perTemperature;

perPressure=ui->doubleSpinBox\_pressure5->value()/100;

perTemperature=ui->doubleSpinBox\_temperature5->value()/100;

double prevPressure, prevTemperature;

prevPressure=weatherVector[0].getPressure();

prevTemperature=weatherVector[0].getTemperature();

*for*(int i=1; i<weatherVector.size(); i++){

int curPressure, curTemperature;

curPressure = weatherVector[i].getPressure();

curTemperature=weatherVector[i].getTemperature();

*//Необхідна* *умова*

*if*( (curPressure>(1-perPressure)\*prevPressure && curPressure<(1+perPressure)\*prevPressure) && (curTemperature>(1-perTemperature )\*prevTemperature && curTemperature<(1+perTemperature)\*prevTemperature) && (weatherVector[i-1].timeLineDay()==weatherVector[i].timeLineDay()-1)){

*if*(!findedPeriod){

findedPeriod=*true*;

ui->plainTextEdit\_task5result->setPlainText("Finded period!\nDate:\n" + weatherVector[i-1].dateToQStr() + ";\n");

}

ui->plainTextEdit\_task5result->moveCursor(QTextCursor::*End*);

ui->plainTextEdit\_task5result->insertPlainText(weatherVector[i].dateToQStr() + ";\n");

}

prevPressure=curPressure;

prevTemperature=curTemperature;

}

*if*(!findedPeriod) ui->plainTextEdit\_task5result->setPlainText("There are not this period in table");

}

*//прогноз* *погоди*

*//якщо* *місяць* *від* *серпня* *до* *січня* *то* *температура* *зменшується* *на* *4.3* *градуси,* *тиск* *збільшується* *на* *1%* *а* *вологість* *на* *10%*

*//якщо* *ж* *місяць* *в* *інших* *межах* *то* *температура* *буде* *збільшуватись* *на* *5* *градусів,* *а* *тиск* *та* *вологість* *зменшуватись* *на* *2%* *та* *20%* *відповідно*

void MainWindow::**on\_pushButton\_find7\_clicked**()

{

bool exist=*false*;

int yearToFind,monthToFind;

yearToFind=ui->spinBox\_year7->value();

monthToFind=ui->spinBox\_month7->value();

*for*(int i=0; i<weatherVector.size(); i++){

*if*(weatherVector[i].getMonth()==monthToFind && weatherVector[i].getYear()==yearToFind){

*if*(!exist) exist=*true*;

ui->plainTextEdit\_task7result->setPlainText("Finded!\n");

*//month* *from* *Sep* *to* *March*

*if*(monthToFind>7||monthToFind<2){

ui->plainTextEdit\_task7result->moveCursor(QTextCursor::*End*);

ui->plainTextEdit\_task7result->insertPlainText("Forecast for month №"+QString::number(weatherVector[i].getMonth()+1)+"\n");

ui->plainTextEdit\_task7result->moveCursor(QTextCursor::*End*);

*//* *v1* *ui->plainTextEdit\_task7result->insertPlainText("Temperature:\t"+QString::number(weatherVector[i].getTemperature()\*0.65)+"* *°C\n");*

ui->plainTextEdit\_task7result->insertPlainText("Temperature:\t"+QString::number(weatherVector[i].getTemperature()-4.3)+" °C\n");

ui->plainTextEdit\_task7result->moveCursor(QTextCursor::*End*);

ui->plainTextEdit\_task7result->insertPlainText("Pressure:\t"+QString::number(weatherVector[i].getPressure()\*1.01)+" hPA\n");

ui->plainTextEdit\_task7result->moveCursor(QTextCursor::*End*);

ui->plainTextEdit\_task7result->insertPlainText("Humidity:\t"+QString::number(weatherVector[i].getHumidity()\*1.1)+" %\n");

}

*else*{

ui->plainTextEdit\_task7result->moveCursor(QTextCursor::*End*);

ui->plainTextEdit\_task7result->insertPlainText("Forecast for month №"+QString::number(weatherVector[i].getMonth()+1)+"\n");

ui->plainTextEdit\_task7result->moveCursor(QTextCursor::*End*);

*//ui->plainTextEdit\_task7result->insertPlainText("Temperature:\t"+QString::number(weatherVector[i].getTemperature()\*1.4)+"* *°C\n");*

ui->plainTextEdit\_task7result->insertPlainText("Temperature:\t"+QString::number(weatherVector[i].getTemperature()+5)+" °C\n");

ui->plainTextEdit\_task7result->moveCursor(QTextCursor::*End*);

ui->plainTextEdit\_task7result->insertPlainText("Pressure:\t"+QString::number(weatherVector[i].getPressure()\*0.98)+" hPA\n");

ui->plainTextEdit\_task7result->moveCursor(QTextCursor::*End*);

ui->plainTextEdit\_task7result->insertPlainText("Humidity:\t"+QString::number(weatherVector[i].getHumidity()\*0.8)+" %\n");

}

*break*;

}

}

*if*(!exist) ui->plainTextEdit\_task7result->setPlainText("This month doesn't exist");

}

bool createdRow=*false*;

*//кнопка* *для* *малювання* *графіків*

void MainWindow::**on\_pushButton\_draw6\_clicked**()

{

QVector<double> y1,y2,y3;

QVector<double> x;

ui->plotTemperature->clearItems();

ui->plotPressure->clearItems();

ui->plotHumidity->clearItems();

ui->plotTemperature->xAxis->setRange(1,12);

ui->plotPressure->xAxis->setRange(1,12);

ui->plotHumidity->xAxis->setRange(1,12);

double minP,maxP,minT,maxT,minH,maxH;

minP=weatherVector[0].getPressure();

minT=weatherVector[0].getTemperature();

minH=weatherVector[0].getHumidity();

maxP=minP;

maxT=minT;

maxH=minH;

*//ui->plotTemperature->yAxis->setRange(0,100);*

*//ui->plotTemperature->plotLayout()->clear();*

*if*(!createdRow){

createdRow=*true*;

ui->plotTemperature->plotLayout()->insertRow(0);

ui->plotHumidity->plotLayout()->insertRow(0);

ui->plotPressure->plotLayout()->insertRow(0);

ui->plotTemperature->plotLayout()->addElement(0, 0, *new* QCPTextElement*(ui->plotTemperature,* "Temperature"*,* QFont*(*"sans"*,* 12*,* QFont*::Bold))*);

ui->plotHumidity->plotLayout()->addElement(0, 0, *new* QCPTextElement*(ui->plotHumidity,* "Humidity"*,* QFont*(*"sans"*,* 12*,* QFont*::Bold))*);

ui->plotPressure->plotLayout()->addElement(0, 0, *new* QCPTextElement*(ui->plotPressure,* "Pressure"*,* QFont*(*"sans"*,* 12*,* QFont*::Bold))*);

}

ui->plotTemperature->xAxis->setLabel("Month (number)");

ui->plotTemperature->yAxis->setLabel("Temperature (°C)");

ui->plotPressure->xAxis->setLabel("Month (number)");

ui->plotPressure->yAxis->setLabel("Temperature (hPA)");

ui->plotHumidity->xAxis->setLabel("Month (number)");

ui->plotHumidity->yAxis->setLabel("Temperature (%)");

*//ui->plotTemperature->xAxis->set*

int needYear=ui->spinBox\_year6->value();

*for*(int i=0; i<weatherVector.size(); i++){

*if*(weatherVector[i].getYear()==needYear){

y1.push\_back(weatherVector[i].getTemperature());

y2.push\_back(weatherVector[i].getPressure());

y3.push\_back(weatherVector[i].getHumidity());

x.push\_back(weatherVector[i].getMonth() + 1/(365/12.0)\*weatherVector[i].getDay());

*if*(weatherVector[i].getHumidity()>maxH) maxH=weatherVector[i].getHumidity();

*if*(weatherVector[i].getHumidity()<minH) minH=weatherVector[i].getHumidity();

*if*(weatherVector[i].getTemperature()>maxT) maxT=weatherVector[i].getTemperature();

*if*(weatherVector[i].getTemperature()<minT) minT=weatherVector[i].getTemperature();

*if*(weatherVector[i].getPressure()>maxP) maxP=weatherVector[i].getPressure();

*if*(weatherVector[i].getPressure()<minP) minP=weatherVector[i].getPressure();

}

}

ui->plotTemperature->yAxis->setRange(minT-abs(minT\*0.1),maxT+abs(maxT\*0.1));

ui->plotPressure->yAxis->setRange(minP-abs(minP\*0.1),maxP+abs(maxP\*0.1));

ui->plotHumidity->yAxis->setRange(minH-abs(minH\*0.1),maxH+abs(maxH\*0.1));

ui->plotTemperature->addGraph();

ui->plotTemperature->graph(0)->setData(x,y1);

ui->plotTemperature->replot();

ui->plotPressure->addGraph();

ui->plotPressure->graph(0)->setData(x,y2);

ui->plotPressure->replot();

ui->plotHumidity->addGraph();

ui->plotHumidity->graph(0)->setData(x,y3);

ui->plotHumidity->replot();

*//ui->plotTemperature->setInteractions(QCP::iRangeDrag* *|* *QCP::iRangeZoom* *|* *QCP::iSelectPlottables);*

}

*//функція* *для* *перевірки* *коректності* *даних* *для* *створення* *(присутня* *перевірка* *на* *30* *лютого)*

bool **checkCreatingData**(int month, int day, std::string windDirection){

*if*(month==1||month==3||month==5||month==7||month==8||month==10||month==12){

*if*(day>31){

*throw* std::exception("incorrect day (>31)!");

*return* *false*;

}

}

*else* *if*(month == 2 && day>28){

*throw* std::exception("incorrect day on Fabruary! (only 28 days)");

*return* *false*;

}

*else* *if*((month==4||month==6||month==9||month==11)&& (day>30)){

*throw* std::exception("incorrect day (only 30 days exist in this month)");

*return* *false*;

}

*if*( !(windDirection=="North" || windDirection=="NorthEast" || windDirection=="NorthWest" || windDirection=="South" || windDirection=="SouthEast" || windDirection=="SouthWest" ||windDirection=="West" || windDirection=="East")){

*throw* std::exception("incorrect wind directions");

*return* *false*;

}

*return* *true*;

}

*//кнопка* *створення* *нового* *обєкту* *погоди*

void MainWindow::**on\_pushButton\_create\_clicked**()

{

int year,month,day;

double temperature, pressure, humidity;

std::string windDir;

year=ui->spinBox\_yearCreate->value();

month=ui->spinBox\_monthCreate->value();

day=ui->spinBox\_dayCreate->value();

temperature=ui->doubleSpinBox\_temperatureCreate->value();

pressure=ui->doubleSpinBox\_pressureCreate->value();

humidity=ui->doubleSpinBox\_humidityCreate->value();

windDir= ui->plainTextEdit\_windCreate->toPlainText().toStdString();

bool isOk=*false*;

*try* {

isOk = checkCreatingData(month,day,windDir);

} *catch* (*const* std::exception &ex) {

QMessageBox::critical(*this*,"Data", QString::fromStdString( ex.*what*() ));

}

*if*(isOk){

Weather newWeather(year, month, day, temperature, pressure, humidity, windDir);

weatherVector.push\_back(newWeather);

ui->tableWidget->setRowCount(weatherVector.size());

int i = weatherVector.size()-1;

QTableWidgetItem \*item = *new* QTableWidgetItem(QString::number(weatherVector[i].getYear()));*//YEAR*

ui->tableWidget->setItem(i,0,*item*);

*//item* *=* *new* *QTableWidgetItem(QString::number(weatherVector[i].getMonth()));//MONTH*

item = *new* QTableWidgetItem(weatherVector[i].convertMonthToQStr());

ui->tableWidget->setItem(i,1,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getDay()));*//DAY*

ui->tableWidget->setItem(i,2,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getTemperature()) + " °C");*//TEMPERATURE*

ui->tableWidget->setItem(i,3,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getPressure()) + " hPA");*//PRESSURE*

ui->tableWidget->setItem(i,4,*item*);

item = *new* QTableWidgetItem(QString::number(weatherVector[i].getHumidity()) + " %");*//HUMIDITY*

ui->tableWidget->setItem(i,5,*item*);

item = *new* QTableWidgetItem(QString::fromStdString(weatherVector[i].getWindDirection()));*//WIND* *DIRECTION*

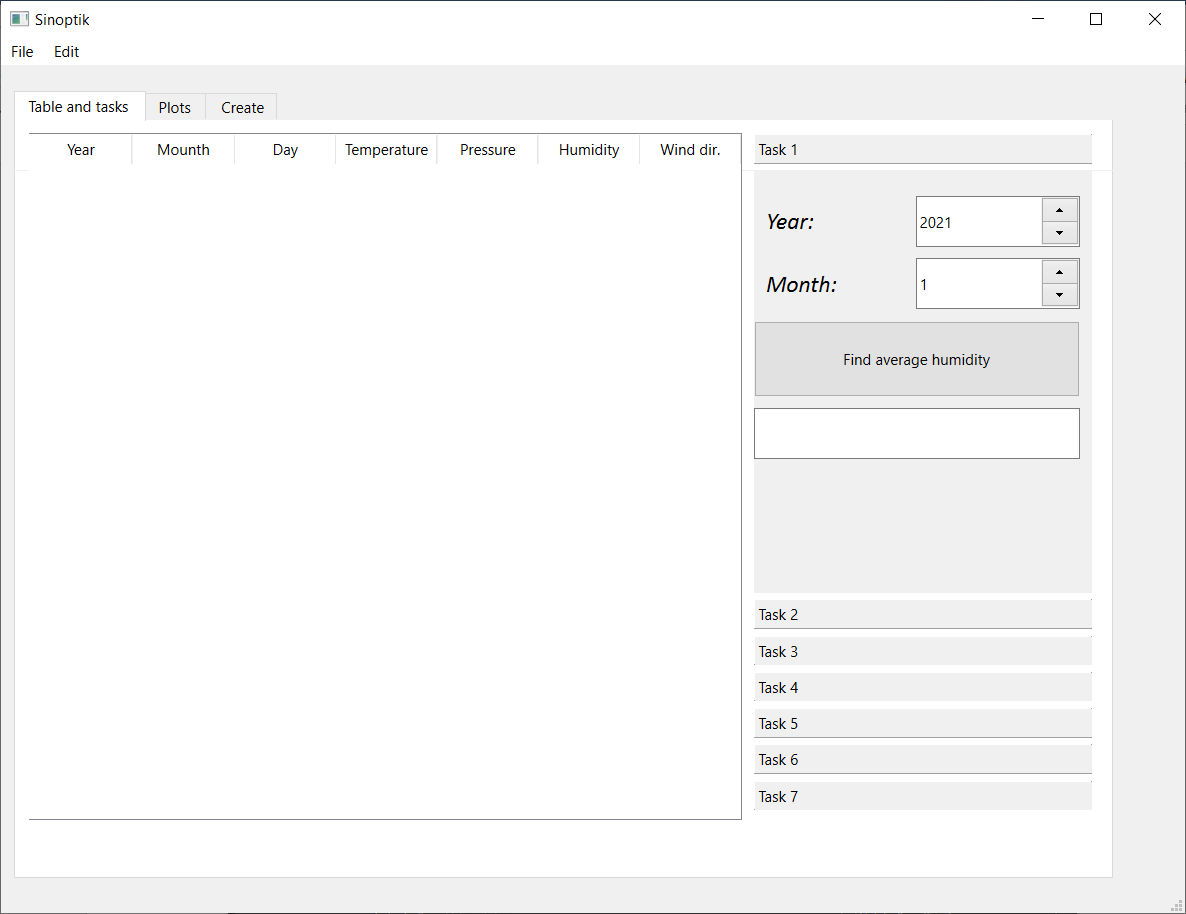
ui->tableWidget->setItem(i,6,*item*);

QMessageBox::information(*this*,"Data", "New weather item created successfully!");

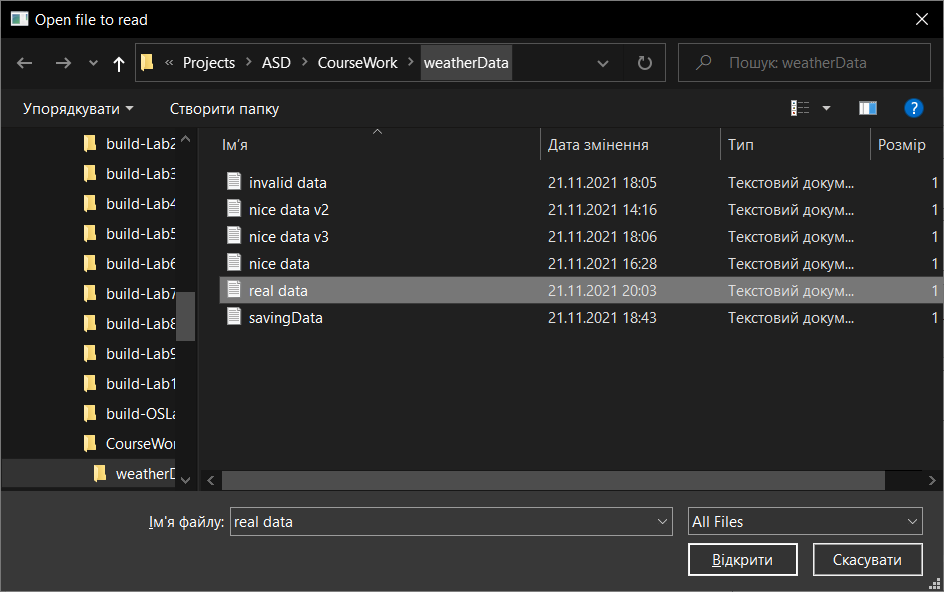
}

}

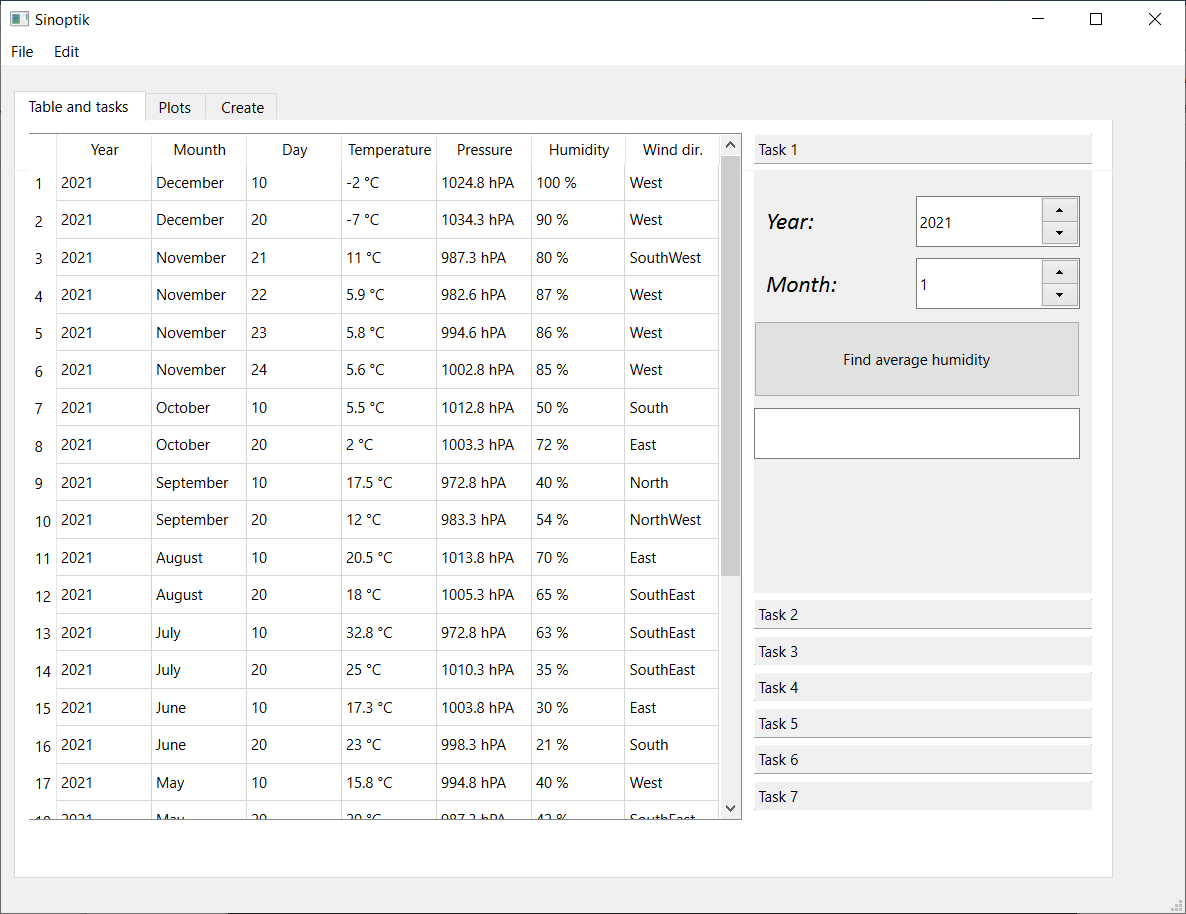
# **Протокол роботи програми**



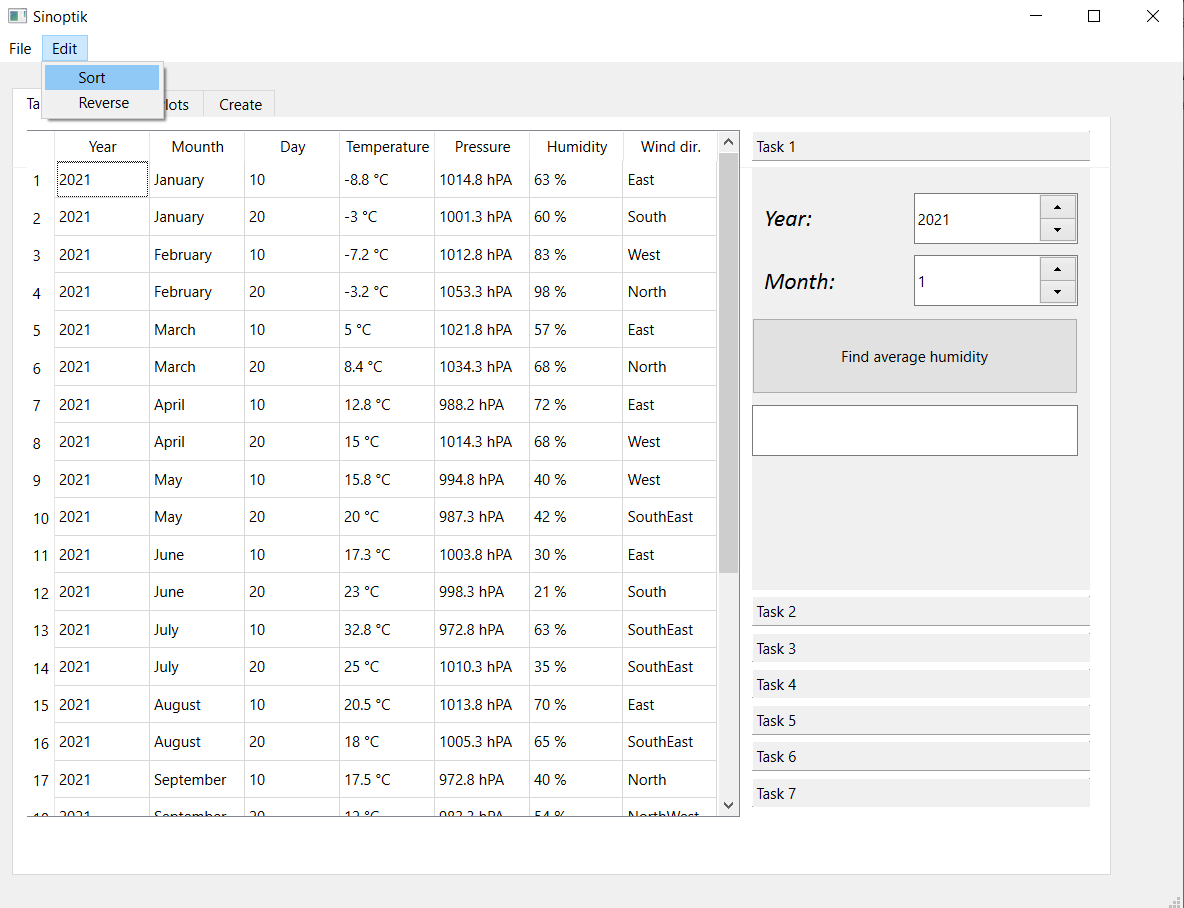
**Рис.4.1.** Головний екран



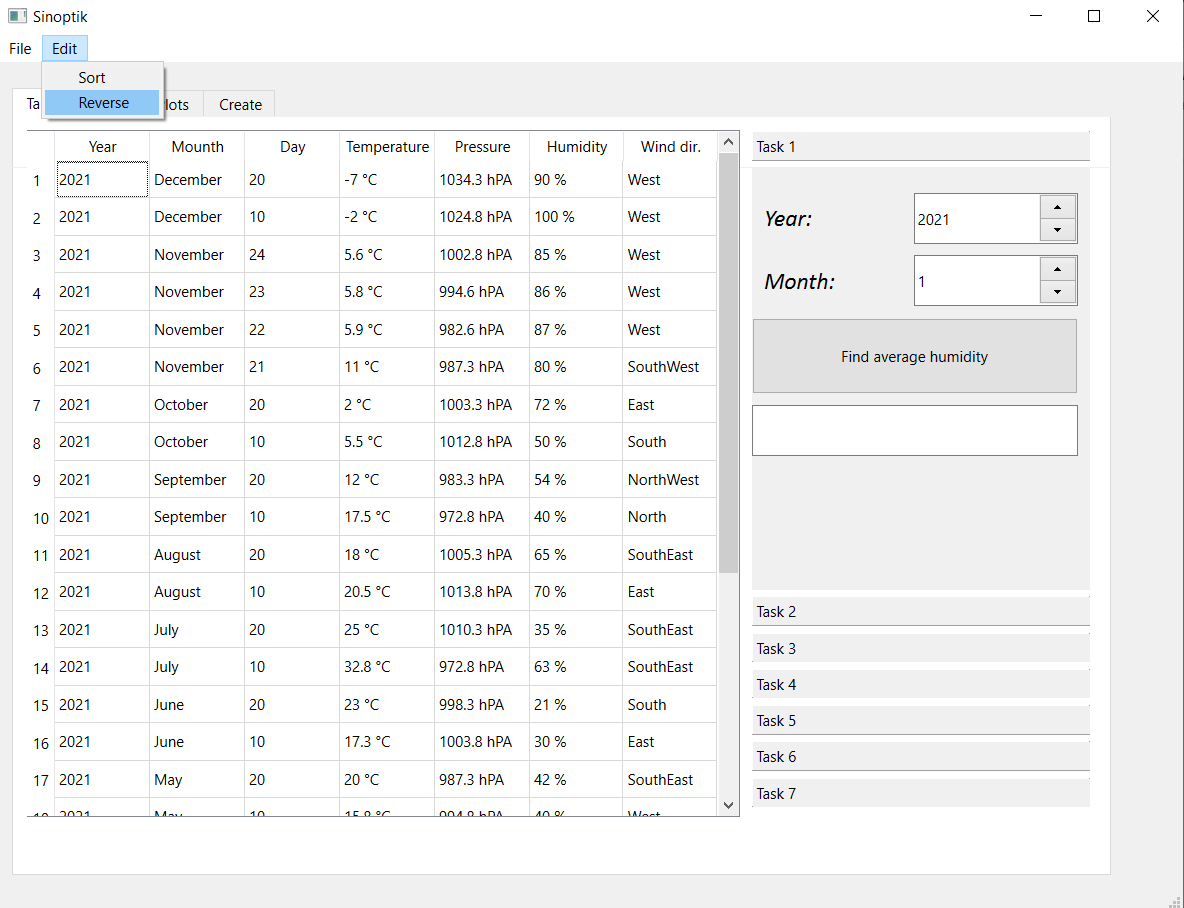
**Рис.4.2.** Екран вибору файлу для зчитування показників



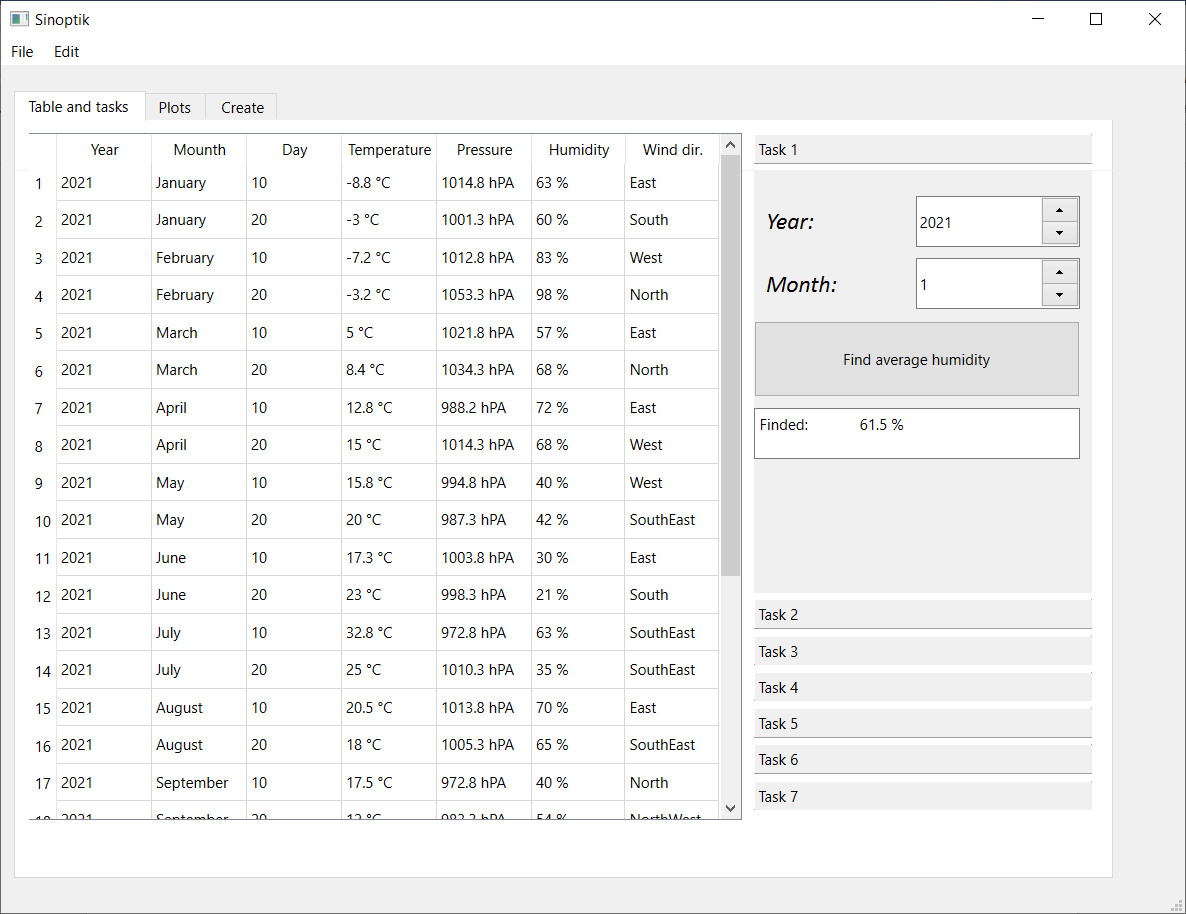
**Рис.4.3.** Вигляд програми після вибору файлу з вхідними даними

****

**Рис.4.4.** Результат сортування погоди за хронологією

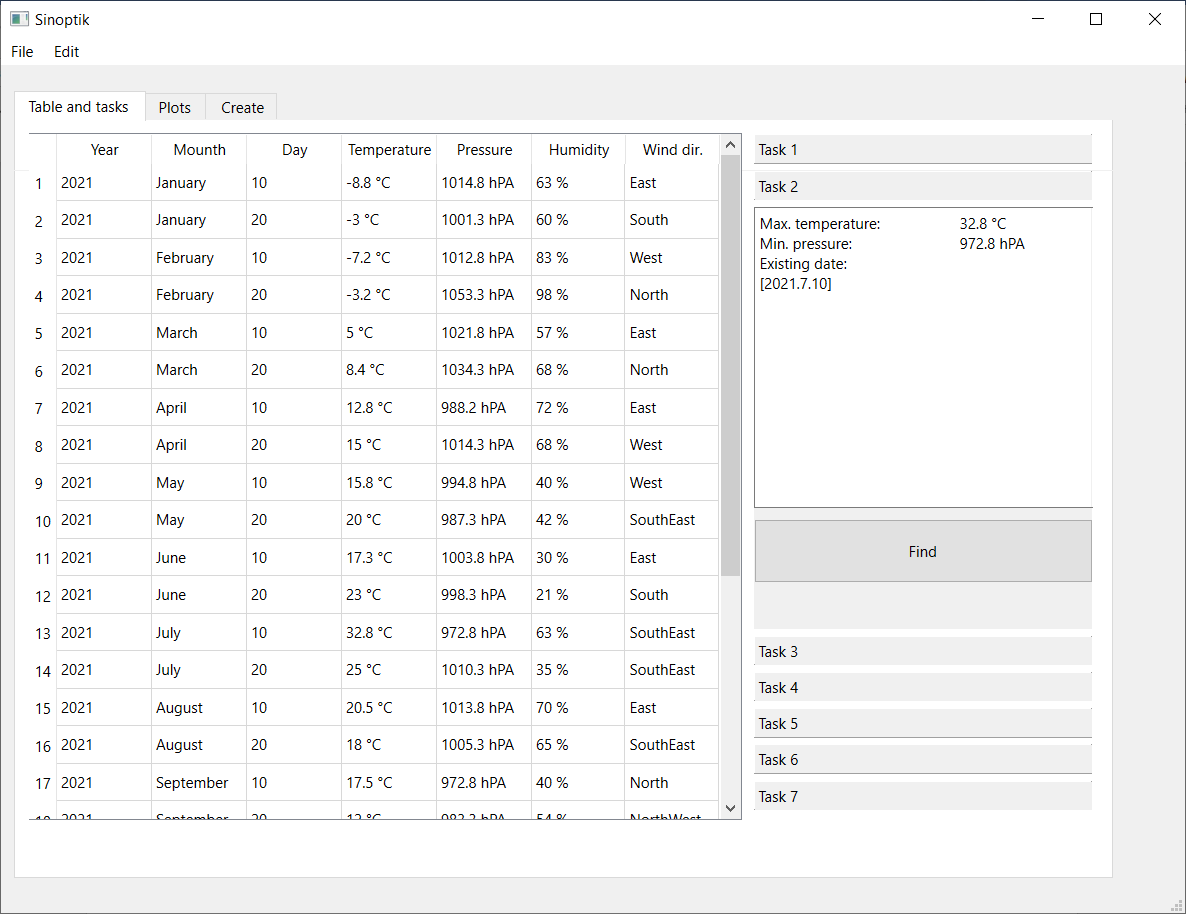


**Рис.4.5.** “Розвернення” таблиці



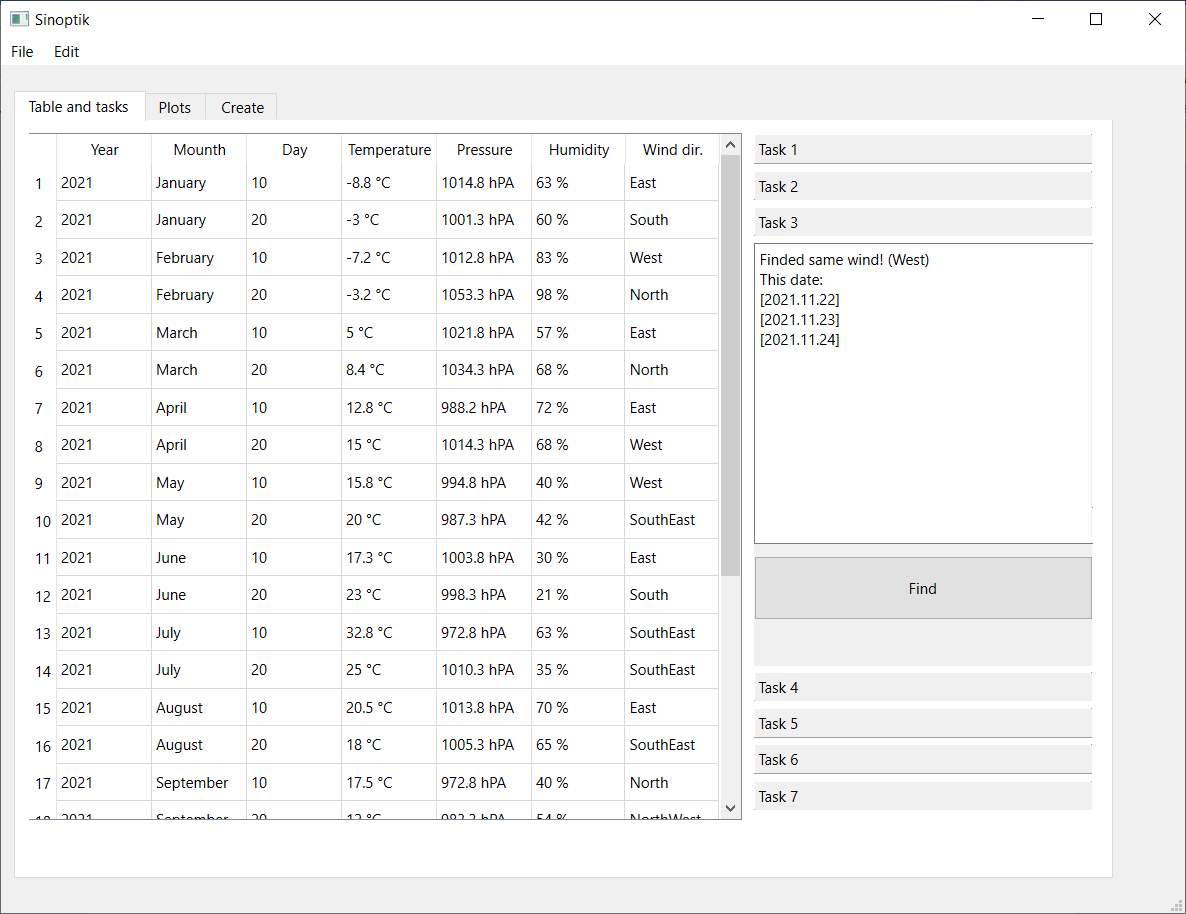
**Рис.4.6.** Результат виконання завдання 1

(Визначити середню вологість за місяць)



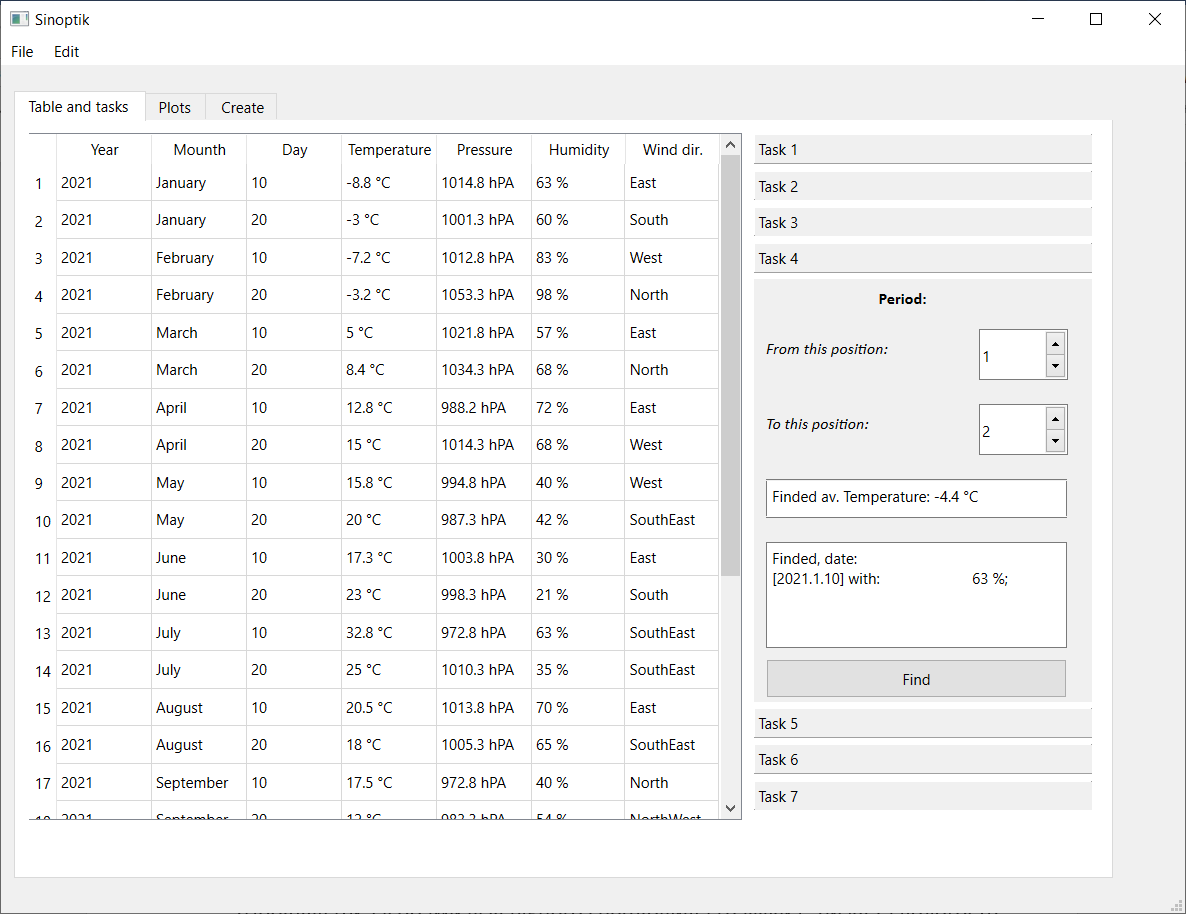
**Рис.4.7.** Результат виконання завдання 2

(Визначити дні з найвищою t та найменшим тиском)



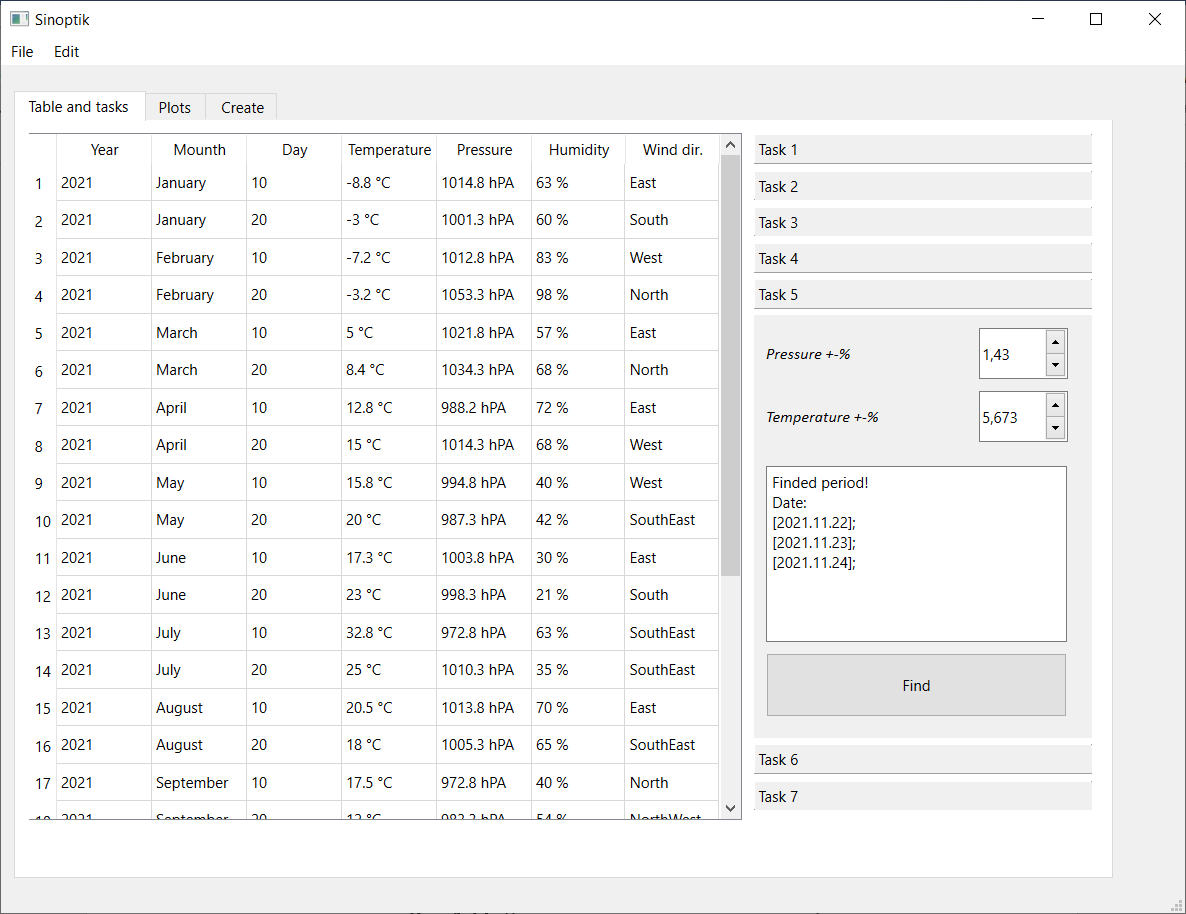
**Рис.4.8.** Результат виконання завдання 3

(Визначити дні, протягом яких, напрям вітру не змінювався)



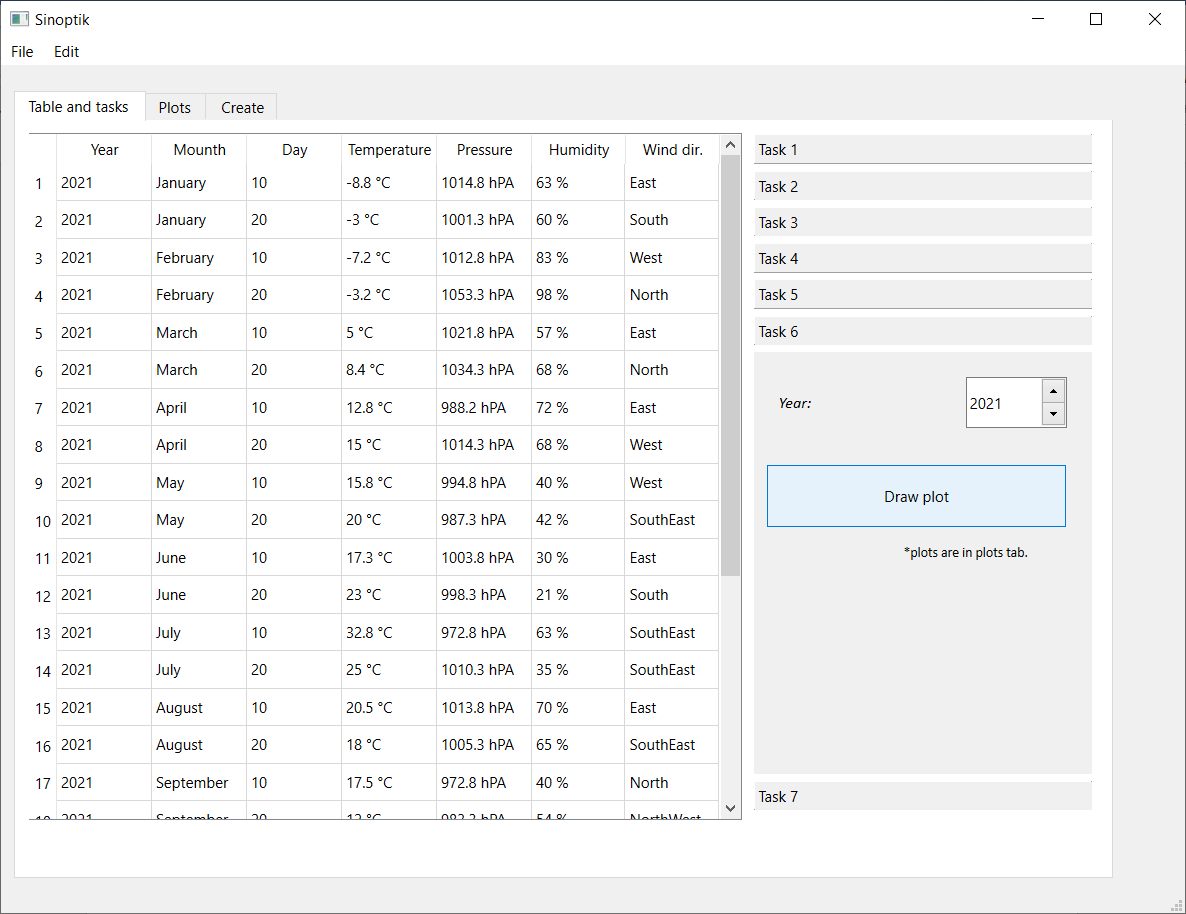
**Рис.4.9.** Результат виконання завдання 4

(За заданим періодом, визначити середню температуру протягом цього періоду та дні з найвищою вологістю)

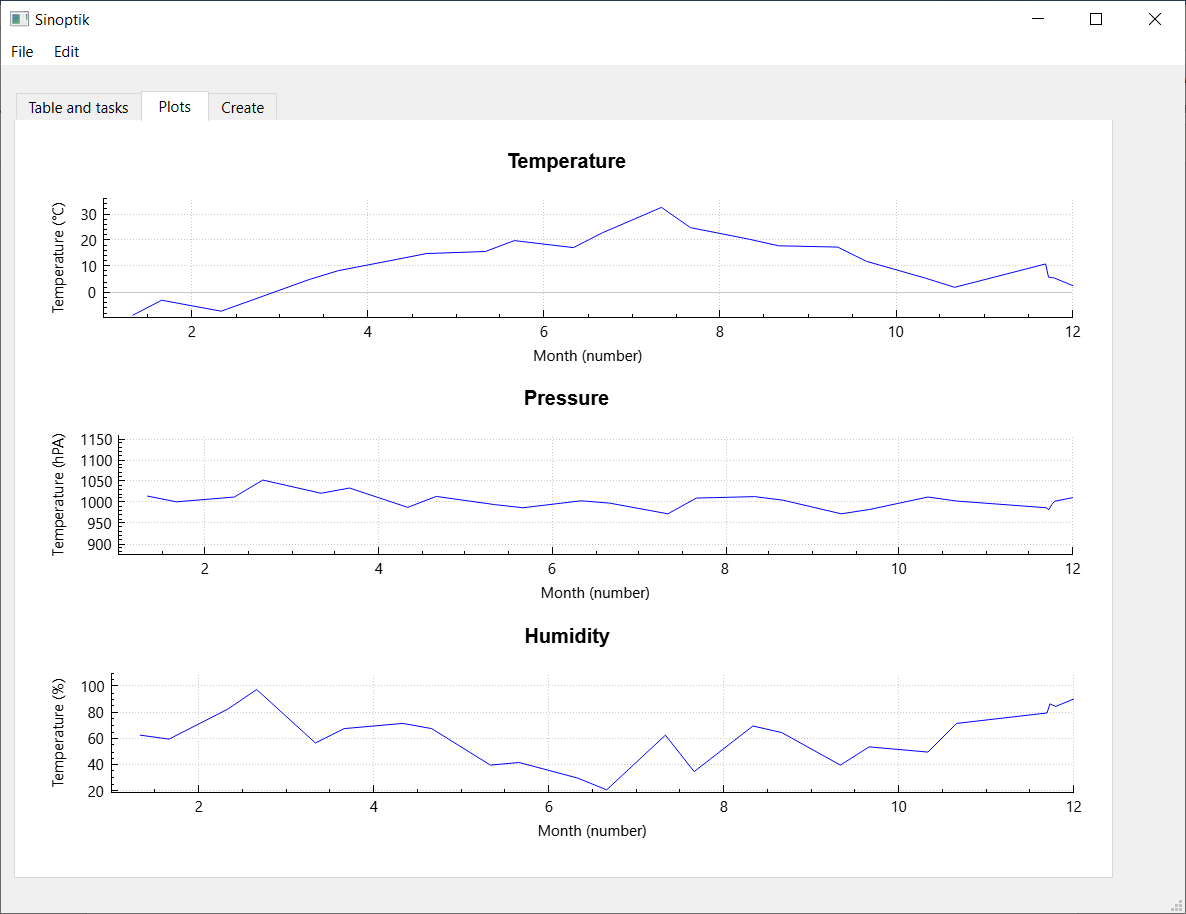


**Рис.4.10.** Результат виконання завдання 5

(Знайти період, в якому тиск змінювався в межах ±1,43%, а t – 5,673%)

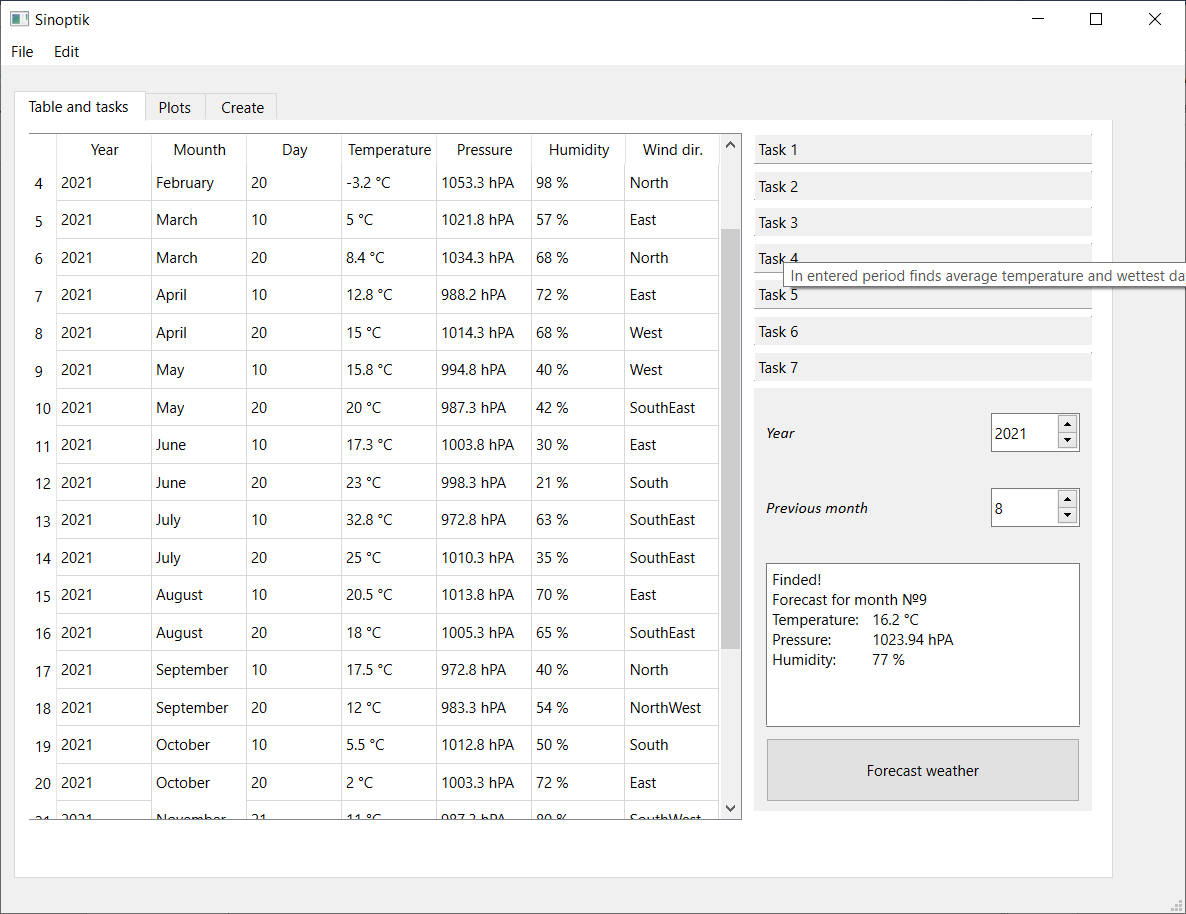


**Рис.4.11.** Виконання завдання 6



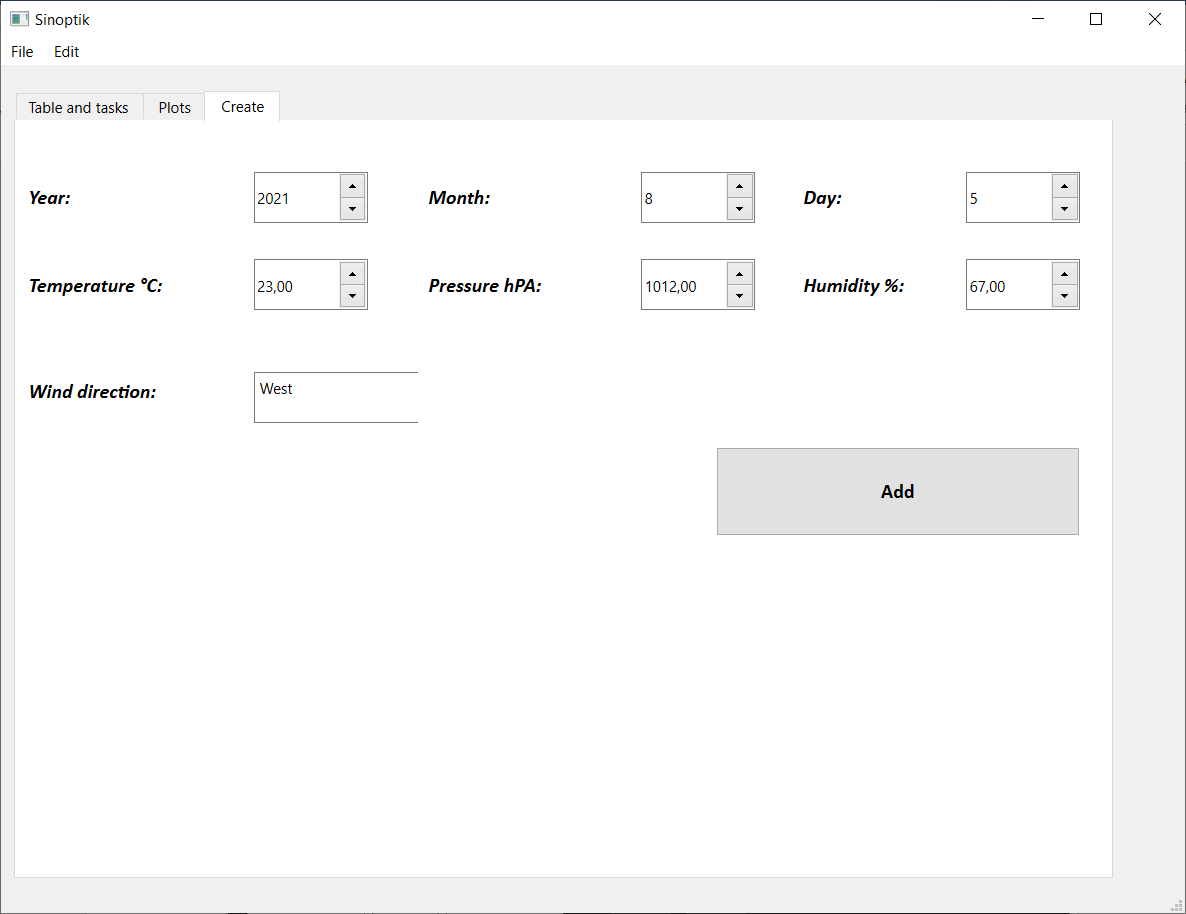
**Рис.4.12.** Результат виконання завдання 6

(Побудувати засобами візуального середовища графіки t, тиску та вологості)

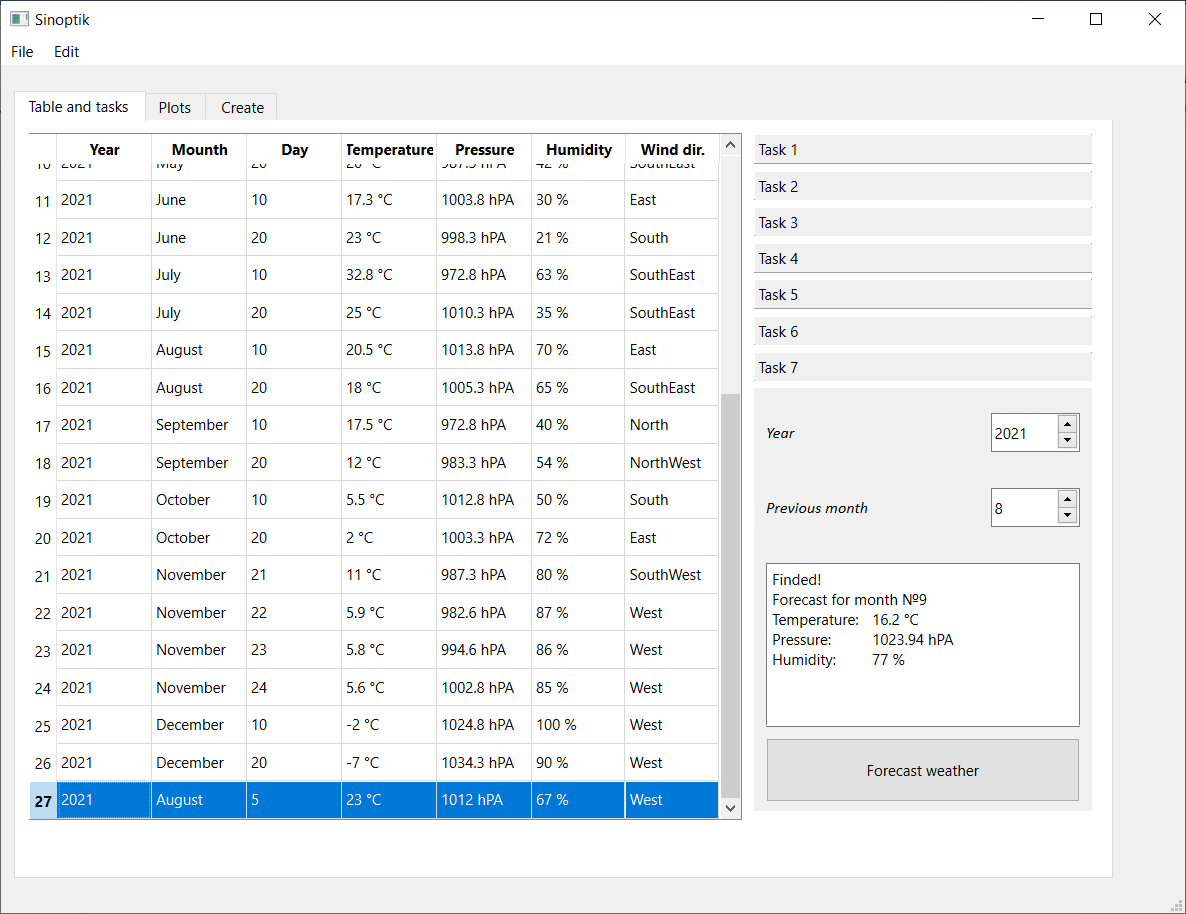


**Рис.4.13.** Результат виконання завдання 7

(Спрогнозувати погоду на наступний місяць з врахуванням зміни сезону року)



**Рис.4.14.** Додавання нового елементу в таблицю



**Рис.4.15.** Результат додавання нового елементу в таблицю

# **Інструкція користувача**

1. **Компоненти ПЗ**

Пакет розроблено на мові програмування С++ за стандартом С++ 11 у середовищі розробки *QT Creator 5.0.0 (Comunity)* і може експлуатуватися під управлінням операційної системи Windows. Програма була побудована з використанням принципів ООП, всі класи задокументувані.

Для коректної роботи пакету необхідна користувацька машина з процесором, тактова частота якого не менше 350 MHz, оперативною пам’яттю не менше 650 Mb.

1. **Встановлення ПЗ**

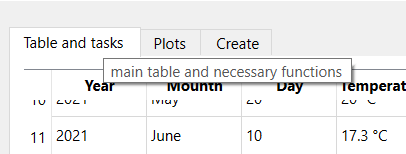
Для роботи пакета потрібно встановити *QT Creator 5.0.0 (Comunity)*. Для початку роботи з програмою необхідно запустити виконавчий файл CourseWork.exe.

1. **Налаштування ПЗ**

Додатково налаштовувати ПЗ не потрібно. Воно є повністю готовим до роботи.

1. **Базові функції ПЗ**
   * **Зручність користування:**

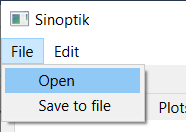
Для того, щоб користуватись програмою було зручно, передбачена система підказок для усього основного функціоналу програми, а також для кожної з робочих вкладок.



*Рис 5.1. Приклад підказки*

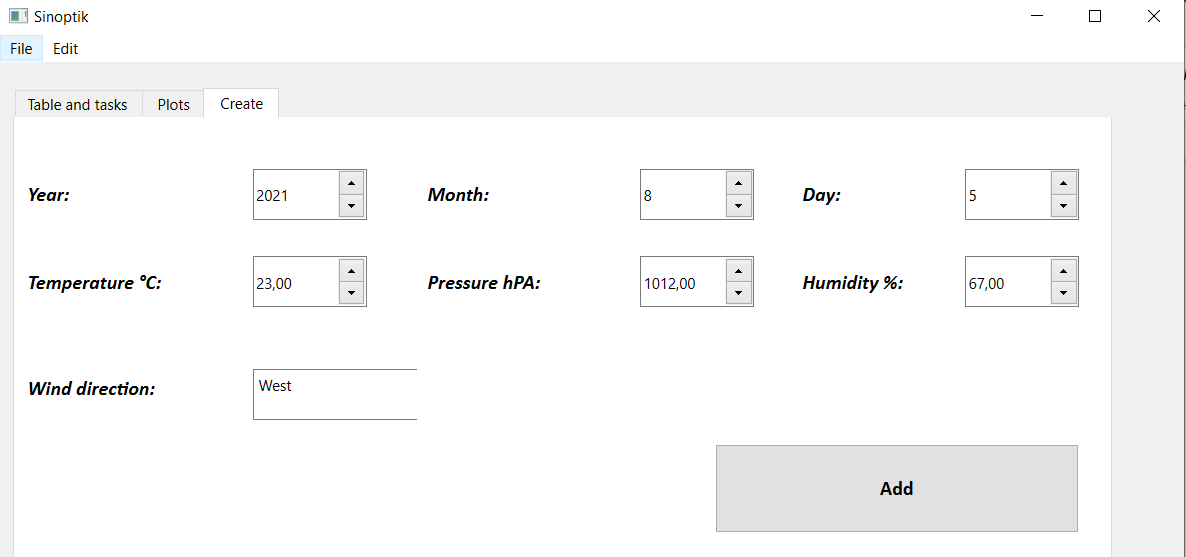
* + **Початок роботи.**

Для того, щоб почати роботу з каталогом, відкрийте файл з даними за допомогою кнопки File->Open в лівому верхньому куті вікна програми.



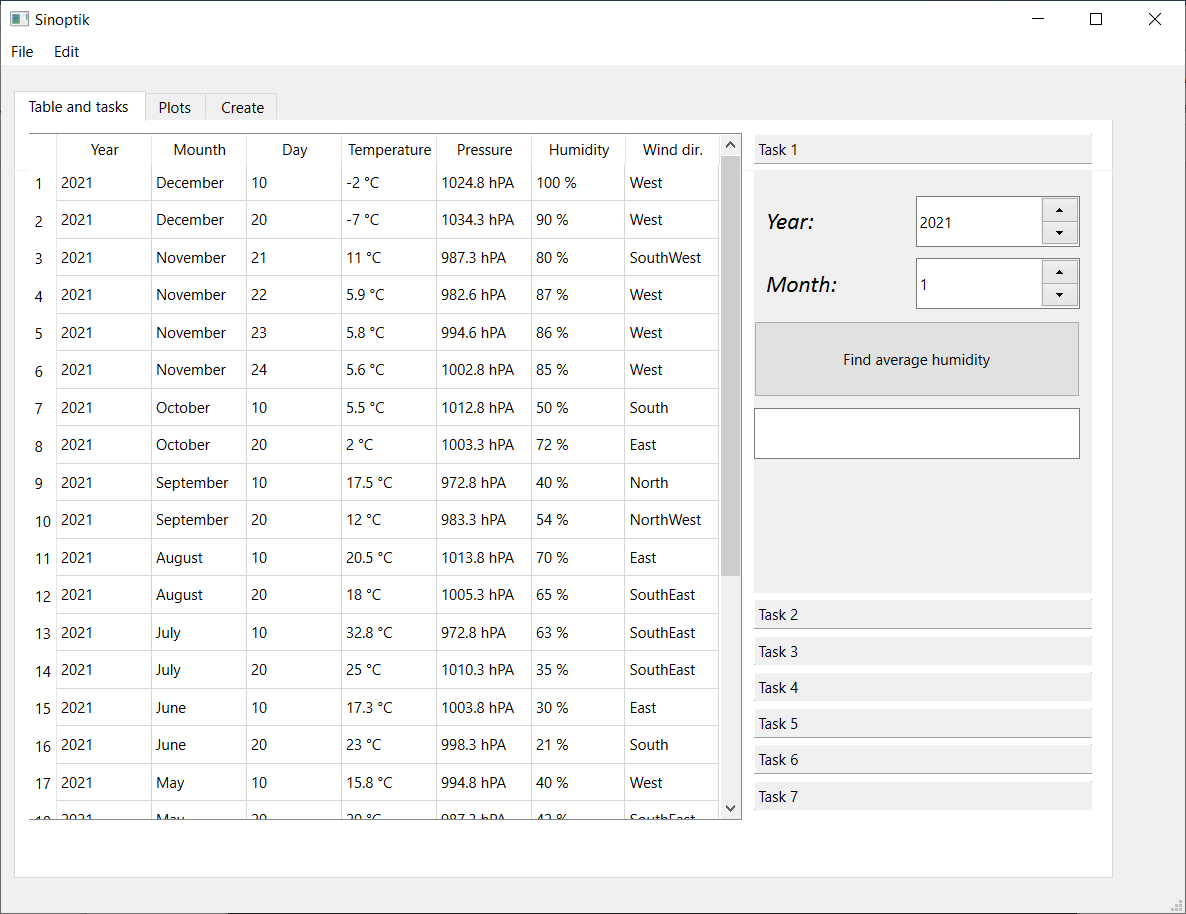
*Рис 5.2. Відкриття файлу з вхідними даними*

Або додайте елементи каталогу вручну. Для цього потрібно перейти у вкладку Create (Рис 5.3).

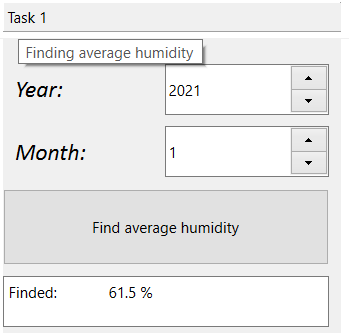


*Рис 5.3. Додавання нового елемента каталогу з клавіатури*

* + **Основні функції.**



*Рис 5.4. Інтерфейс програми після відкриття файлу з вхідними даними*

**

*Рис 5.5. Приклад підказки для “Task’и”*

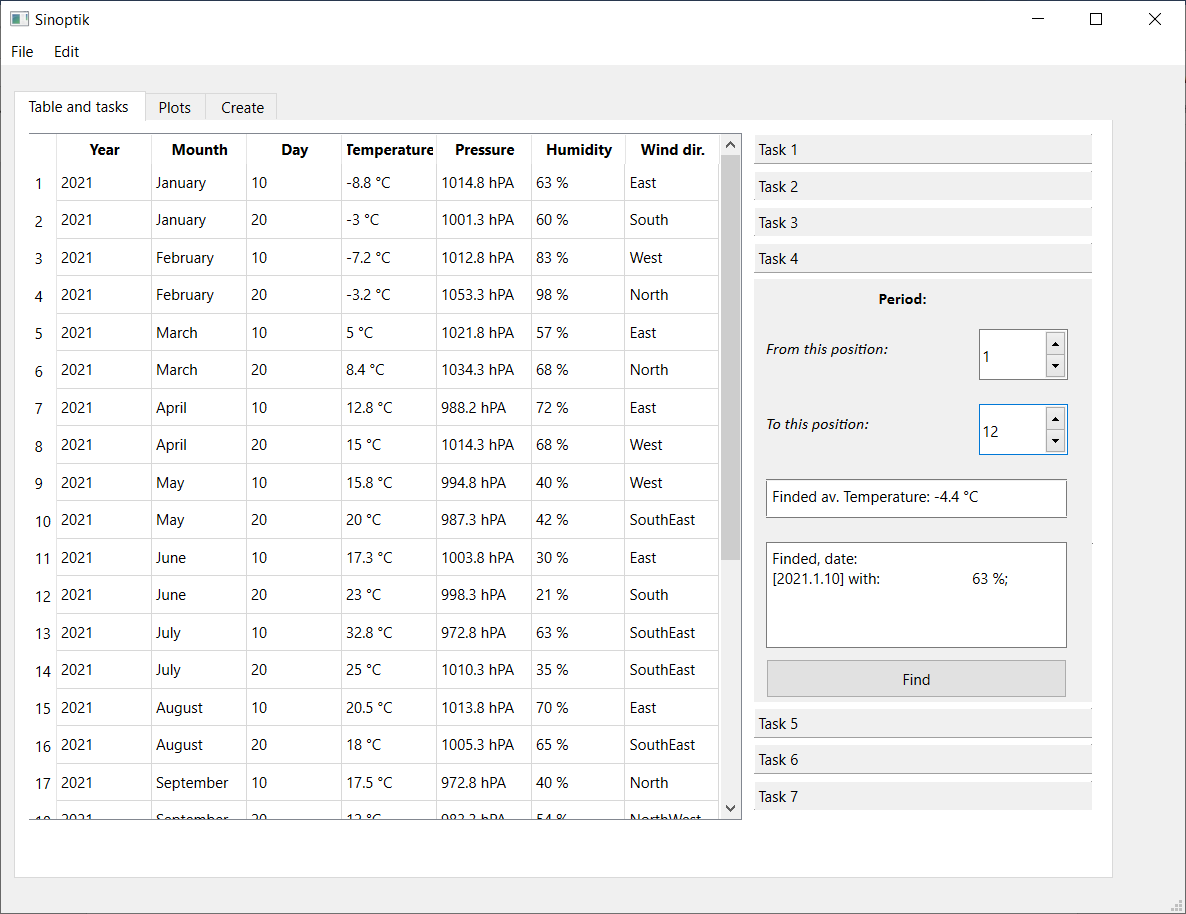
В правій частині вікна є весь необхідний функціонал (виділено червоним). Кожен “Task” – міні-поле для виконання відповідного завдання програми. (Саме завдання можна глянути у підказці до “Task”, навівши на неї курсор миші Рис 5.5 ) Зокрема:

Task 1: (Визначити середню вологість за місяць) Потрібно вибрати рік та місяць, для якого необхідно виконати задачу. Результат отримується натисканням відповідної кнопки.

Task 2: (Визначити дні з найвищою t та найменшим тиском) Не потрібно вводу ніяких даних. При натисканні на кнопку зразу отримуємо результати.

Task 3: (Визначити дні, протягом яких, напрям вітру не змінювався) Для виконання цієї задачі необхідно вісортувати таблицю (меню Edit->Sort). Після цього при натисканні кнопки завдання буде виконано.

Task 4: (За заданим періодом, визначити середню температуру протягом цього періоду та дні з найвищою вологістю) Як і попереднє це завдання також потребує сортування. Після цього потрібно вибрати потрібні (за порядком в таблиці) дні і виконати завдання натисканням кнопки. (Рис 5.6)



*Рис 5.6. Поля вводу для Task 4*

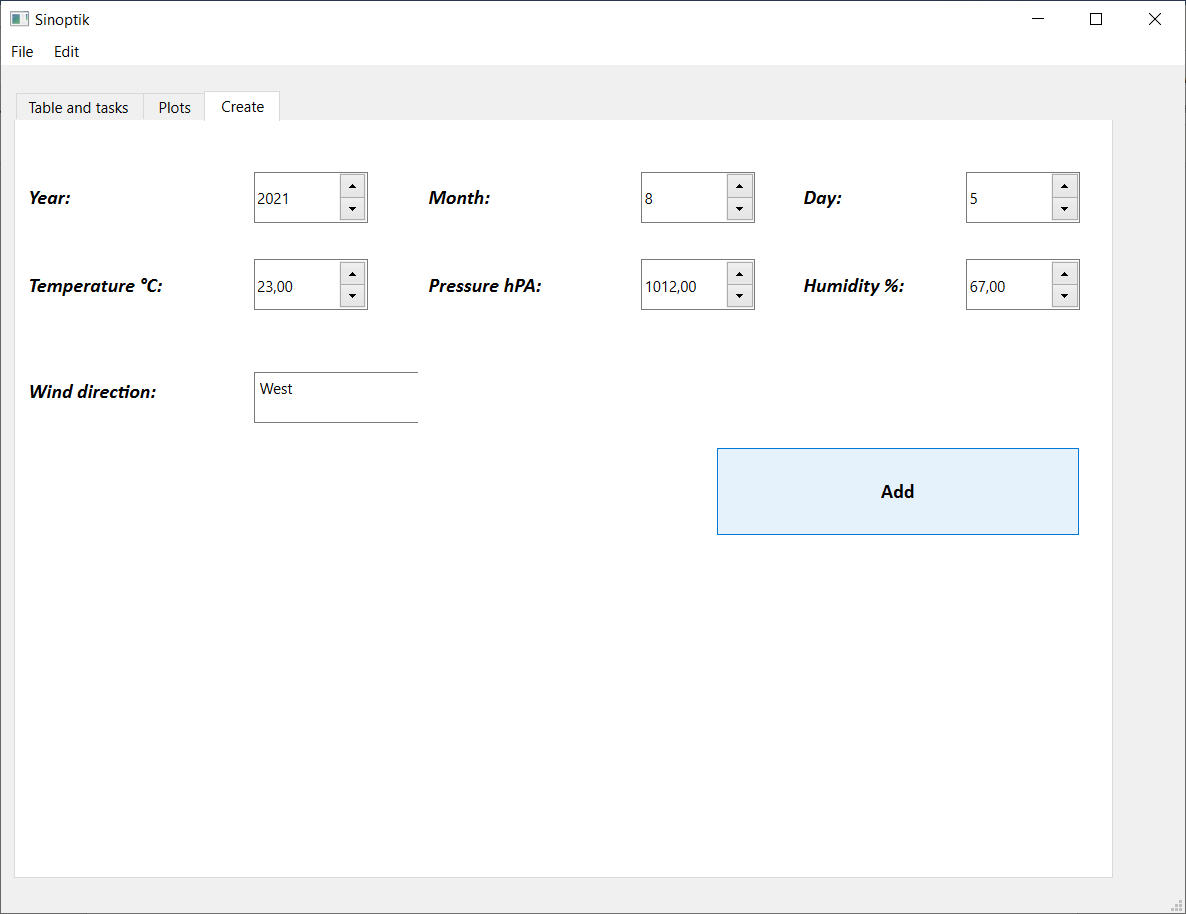
Task 5: (Знайти період, в якому тиск змінювався в межах ±1,43%, а t – 5,673%) Це завдання також потребує сортування. Межі коливання показників можна змінити у відповідних полях. При натисканні на кнопку буде виконаний пошук.

Task 6: (Побудувати засобами візуального середовища графіки t, тиску та вологості) Потрібно вказати рік, для якого слід побудувати графік та натиснути на кнопку. Результат можна побачити у вкладці ***Plots***

Task 7: (Спрогнозувати погоду на наступний місяць з врахуванням зміни сезону року.) У відповідні поля вводиться рік та місяць. Для наступного місяця цього ж року буде складатися прогноз. Завдання виконується при натисканні відповідної кнопки.

* **Додавання нових даних.**

Для того щоб добавити нові дані потрібно перейти у вкладку Create та правильно ввести дані у поля.



*Рис.5.7. Вигляд вкладки програми з полями для вводу даних*

*Для того, щоб добавити нові показники у таблицю натисніть* ***Add*** *внизу-справа.*

Для коректного вводу були використані спеціальні поля, проте помилки все ще можливі, а саме:

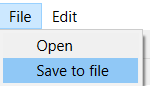
Неправильна кількість днів для відповідного місяця.

Неправильний тест в полі для напрямку вітру. Можна вводити лише 8 наступних:

*Notrh NotrhEast NorthWest South SouthEast SouthWest West East*

* **Збереження у файл.**

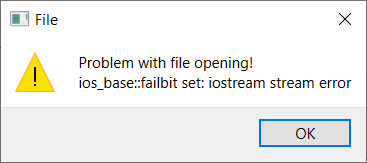
Для того, щоб зберегти дані в існуючий файл, натисніть File-> Save to file.



*Рис.5.8. Кнопка для збереження у існуючий файл*

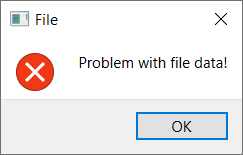
# **Опис виняткових ситуацій**

* Якщо виникне будь-яка помилка при відкритті файлу (не існуючий шлях, некоректно вибрано файл) то буде виведена наступна помилка, та її код:



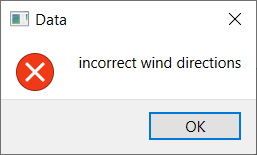
*Рис.6.1. Не вдалося відкрити файл*

* Якщо спробувати зчитати дані з неправильно оформленого текстового документу, то буде наступна помилка.

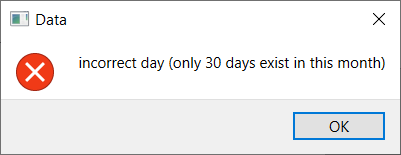


*Рис.6.2. Не коректний файл*

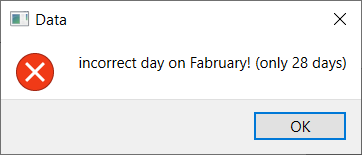
* Якщо вводити некоректні дані з клавіатури, то передбачені наступні винятки:



*Рис.6.3. Введено неправильний напрям вітру*



*Рис.6.4. Спроба в 30-денний місяць записати 31 день*



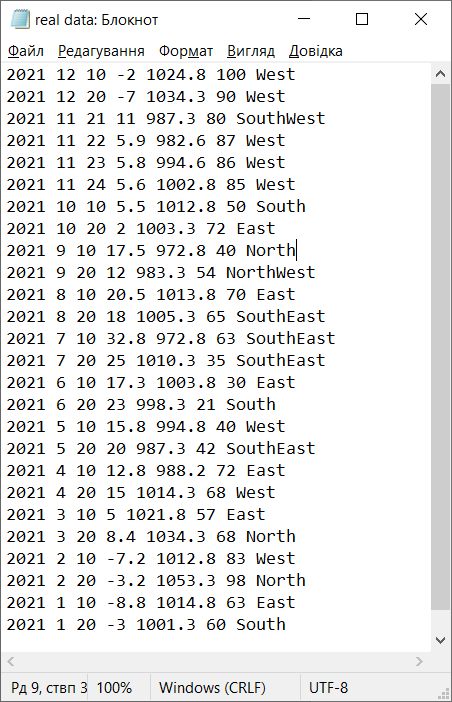
*Рис.6.5. Спроба в Лютий записати більше 28 днів*

# **Структура файлу вхідних даних**

Дані у файлі мають бути записані наступним чином:

<Рік> <Місяць> <День> <Температура> <Тиск> <Вологість> <Напрям Вітру>

Між даними для одного дня ставиться пробіл (має бути 7 “слів” розділених пробілом), а зразу після останнього “слова” перехід на новий рядок.



*Рис 7.1. Приклад вхідного файлу*

Рік, місяць та день – цілі числа.

Температура, тиск та вологість - дробові числа (записуються через крапку).

Напрям вітру – один з восьми можливих варіантів записів (*Notrh NotrhEast NorthWest South SouthEast SouthWest West East*)

# **Висновки**

Виконуючи цю курсову роботу, я покращив свої знання з принципів ООП і завдяки ним склав власне програмне забезпечення для роботи з погодою. Програма вміє зчитувати погодні показники з файлу, обробляти їх (сортувати, модифікувати, прогнозувати, шукати певні періоди з певними умовами та інше), а також зберігати у файл. Програма була реалізована в середовищі розробки QT Creator 5.0.0 (Comunity). Звісно, що по ходу розробки виникали певні труднощі, для яких мені вдалось відшукати рішення. В наступному розділі прикріпив деякі з джерел, якими користувався під час розробки.

# **Список використаної літератури**

**Книги**

1. Є. В. Левус, Т. А. Марусенкова, О. О. Нитребич. Життєвий цикл

програмного забезпечення / Є. В. Левус, Т. А. Марусенкова, О. О.

Нитребич. - Львів : Видавництво Львівської політехніки, 2017.- 208 с.

1. Объектно-ориентированное программирование в С++ 4-е видання / Роберт Лафоре. – Санкт-Петербург Видавництво: Питер, 2013.- 928 с.

**Сайти**

1. *Методичні вказівки до виконання курсової роботи для студентів напряму 6.121 «Інженерія програмного забезпечення» / Укл. Коротєєва Т.О., Дяконюк Л.М.– Львів: Національний університет “Львівська політехніка ” кафедра програмного забезпечення, 2020. – 27с. [Електронний ресурс]* [*https://vns.lpnu.ua/mod/page/view.php?id=71345*](https://vns.lpnu.ua/mod/page/view.php?id=71345)
2. *Офіційна документація QT Creator [Електронний ресурс]* [*https://doc.qt.io/qt-5/*](https://doc.qt.io/qt-5/)
3. *Форум, де знаходив проблеми та рішення, з якими вже зустрічались інші користувачі [Електронний ресурс]* [*https://stackoverflow.com/*](https://stackoverflow.com/)
4. *Офіційна документація Microsoft з с++ [Електронний ресурс]* [*https://docs.microsoft.com/uk-ua/cpp/?view=msvc-170*](https://docs.microsoft.com/uk-ua/cpp/?view=msvc-170)

**Відео**

1. Плей-лист уроків з програмування на с++ Автор #SimpleCode *[Електронний ресурс]* <https://www.youtube.com/watch?v=kRcbYLK3OnQ&list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r>
2. Плей-лист уроків з ООП на с++ Автор #SimpleCode *[Електронний ресурс]* <https://www.youtube.com/watch?v=rZcTaRU7AAw&list=PLQOaTSbfxUtBm7DxblJZShqBQnBAVzlXX>
3. Серія відео Робота з графіками в QT Creator Автор HARSH MITTAL *[Електронний ресурс]* <https://youtu.be/VCx8i2W52TY>