TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN ĐIỆN TỬ

TIỂU LUẬN CUỐI KỲ

Môn học: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

TÊN TIỂU LUẬN:

THIẾT KẾ CẤU TRÚC DỮ LIỆU CHO HỆ THÔNG QUẢN LÝ NHÀ KÍNH BẰNG VI XỬ LÝ

Giảng viên: PGS.TS. Hoàng Văn Dũng

Danh sách sinh viên thực hiện

Mã số SV	Họ và tên	Mức độ đóng góp (%)
20139015	Trần Toàn Thắng	100%
20139096	Đinh Thanh Tùng	100%
20139059	Mai Văn Anh	100%
20139089	Nguyễn Bá Quốc Tài	100%

TP. Hồ Chí Minh, tháng 05 năm 2022

MỤC LỤC

DANH MỤC HÌNH ẢNH

PHẦN 1: MỞ ĐẦU

1. Lý do chọn đề tài:

Để phát triển mạnh mẽ ngành nông nghiệp, cũng như đưa vị thế nghành xuất khẩu nông nghiệp và sản xuất nông nghiệp sạch và chất lượng là nghành chủ lực cho Việt Nam, những vấn đề như biến đổi khí hậu, thay đổi giống làm cho nghành nông nghiệp gặp không ít khó khăn trong nông nghiệp, hậu quả là các bệnh dịch nghành nông nghiệp chết hàng loạt. Chính vì vậy các hệ thống nông nghiệp thông minh ra đời.

Theo truyền thống nông nghiệp được thực hiện bằng cách thực hiện một nhiệm vụ cụ thể, chẳng hạn như trồng, chăm sóc, tưới tiêu hoặc thu hoạch với một lịch trình định trước. Nhưng bằng cách thu thập dữ liệu thời gian thực về thời tiết, đất và chất lượng không khí v.v.. nhằm theo dõi sự trưởng thành của cây trồng và thậm chí cả trang thiết bị và chi phí lao động, các phân tích có thể được sử dụng để đưa ra quyết định thông minh hơn. Đây được gọi là nông nghiệp chính xác (hoặc canh tác chính xác - precision agriculture). Một định nghĩa của nông nghiệp chính xác là: kỹ thuật áp dụng đúng số lượng đầu vào(nước, phân bón, thuốc trừ sâu, v..v..) vào đúng vị trí và vào đúng thời điểm để tăng cường sản xuất và nâng cao chất lượng sản phẩm nông nghiệp.

Để thúc đẩy sự thông minh và tự động hoá trong nông nghiệp, nó là một xu hướng áp dụng công nghệ IoT. Hệ thống giám sát môi trường dựa trên IoT có thể đáp ứng được các phép đo nhanh, chính xác và liên tục yêu cầu trong nông nghiệp chính xác. Các kỹ thuật mới của IoT cung cấp mới phương pháp tiếp cận với kỹ thuật thông tin môi trường nông nghiệp. Ứng dụng mạng cảm biến không dây cho nông nghiệp chính xác là hướng đi mới đáp ứng các yêu cầu quan trọng trong lĩnh vực hiện đại hóa nông nghiệp. Mạng cảm biến không dây cho phép giám sát quản lý và điều khiển các thông số liên quan đến quy trình sản xuất nông nghiệp như các thông số liên quan đến môi trường (nhiệt độ không khí, độ ẩm không khí, ánh sáng, v..v..), các thông số liên quan đến điều kiện đất trồng (độ ẩm đất, độ pH, 2 v..v..) và sức khỏe của cây trồng (độ xanh của lá, côn trùng và các bệnh tật, cỏ dại, v..v..). Các dữ liệu này được thu thập, lưu trữ và truyền tải không dây đến người dùng để xử lý, qua đó họ có thể điều khiển và đưa ra các biện pháp thích hợp trong sản xuất nông nghiệp nhằm tăng năng xuất, nâng cao chất lượng và sự đồng đều trong sản phẩm thu hoạch. Vì thế cần có một ứng dụng có thể lưu trữ và quản lý các dữ liệu trên một cách đơn giản và hiệu quả.

2. Mục đích, yêu cầu cần thực hiện:

Ứng dụng cần quản lý được đầy đủ các thông tin của các khu vực nhà kính, các dữ liệu sau khi được gởi về sẽ được lưu trữ một cách có hệ thống để có thể truy xuất thông tin một cách rõ ràng và trực quan nhất. Vì vậy cần thiết kế một cấu trúc dữ liệu

có thể quản lý được các thông tin về nhiệt độ, độ ẩm, độ pH,…của từng nhà kính. Ngoài ra còn phải quản lý các khu vực nhà kính khác nhau, quản lý thời gian của các dữ liệu để có thể đưa ra các giải pháp chính xác và kịp thời cho hệ thống nhà kính của mình.

Yêu cầu cần thực hiện: Xây dựng được cấu trúc dữ liệu có thể:

+ Lưu trữ dữ liệu nhiệt độ, độ ẩm của đất và không khí, độ pH đất, nồng độ CO_2 trong không khí, quang thông.

+ Lưu trữ các dữ liệu trên kèm thời gian nhận được dữ liệu để quản lý.

+ Quản lý số lượng khu vực nhà kính hiện có, thêm, xóa các khu vực.

+ Quản lý các bộ dữ liệu theo từng ngày.

+ Truy xuất dữ liệu của ngày hiện hành và các ngày trước.

+ Tự giải phóng dữ liệu khi thời gian lưu trữ quá giới hạn đặt ra.

3. Phạm vi và đối tượng:

Ứng dụng được thiết kế để lưu trữ các dữ liệu của môi trường trồng trọt, cụ thể ở đây là nhà kính, ngoài ra vẫn phù hợp để quản lý các mô hình nông nghiệp khác.

Đối tượng sử dụng:

- Là những người thực hiện nghiên cứu, quan sát môi trường sinh trưởng và quá trình phát triển của thực vật như các kĩ sư nông nghiệp, nhà thực vật học,…
- Những người giám sát cây trồng như chủ vườn, chủ nhà kính,…

PHẦN 2: CƠ SỞ THỰC HIỆN DỰ ÁN

1.Công cụ, môi trường cho lập trình và phát triển dự án:

Các phần mềm được các thành viên trong nhóm sử dụng để viết chương trình là: DEV-C++, Visual Studio Code, Microsoft Visual Studio,, Eclipse, Sublime Text. Ngoại trừ Sublime Text không có trình biên dịch sẵn để biên dịch chương trình, buộc phải dùng Command Prompt để biên dịch và chạy chương trình bằng file *.exe, còn các công cụ soạn thảo, công cụ tìm kiếm và sửa sửa lỗi, công cụ nhắc lệnh trong quá trình lập trình thì các ứng dụng trên đều đáp ứng tốt.

Biểu đồ thể hiện thông tin của các lớp(Class Diagram) được nhóm sử dụng phần mềm Enterprise Architect để vẽ, đây là một công cụ thiết kế và xây dựng hệ thống phần mềm; mô hình hóa quy trình kinh doanh và các tên miền dựa trên ngành công nghiệp mô hình hóa. Enterprise Architect hỗ trợ các công cụ vẽ mạnh mẽ để thiết kế và trình bày các Class Diagram một cách trực quan, tự động tạo Class Diagram bằng đường dẫn của mã nguồn.

Sơ đồ khối thể hiện ý tưởng hoạt động của chương trình, cách nhập dữ liệu, xuất dữ liệu và xóa dữ liệu được nhóm sử dụng ứng dụng online có tên miền là “Draw.io”. Đây là một ứng dụng hỗ trợ vẽ các thuật toán, lưu đồ, sơ đồ khối, biểu đồ, cơ sở dữ liệu, mạch điện,… rất mạnh mẽ.

2. Ngôn ngữ lập trình, thư viện hỗ trợ và cơ sở lý thuyết:

Project được nhóm lập trình bằng ngôn ngữ C++, một ngôn ngữ lập trình bậc cao với các câu lệnh trực quan, dễ dàng thể hiện những ý tưởng hoạt động của chương trình từ những suy nghĩ và ngôn ngữ của con người, có rất nhiều thư viện hỗ trợ cho việc lập trình và phù hợp với lý thuyết được dạy trên lớp.

Danh sách các thư viện được sử dụng cho project:

- iostream: thư viện hỗ trợ nhập xuất dữ liệu.
- iomainip: thư viện hỗ trợ định dạng xuất dữ liệu.
- ctime: thư viện để lấy và thao tác các thông tin ngày giờ cho dữ liệu.
- vector: thư viện hỗ trợ các thao tác quản lý dữ liệu bằng cấu trúc dữ liệu vector.
- string: thư viện hỗ trợ tạo ra một đối tượng string để lưu trữ chuỗi ký tự.
- random: thư viện hỗ trợ tạo ra các giá trị ngẫu nhiên (cho các thông số môi trường thay thế cho ngoại vi để chạy giả lập ứng dụng).
- math.h: tập tin tiêu đề định nghĩa các hàm toán học đa dạng và một macro. Tất cả các hàm có sẵn trong thư viện này nhận double như là một tham số và trả về kết quả ở kiểu double.

- bits/stdc++.h: tập tin tiêu đề bao gồm mọi thư viện tiêu chuẩn.
- fstream: thư viện đọc/ghi file (dùng để đọc và in file logo của nhóm).
- unistd.h: là tên của tệp tiêu đề cung cấp quyền truy cập vào API hệ điều hành POSIX (chủ yếu dùng lệnh tạm ngưng chương trình cho giao diện giả lập).
- windows.h: là một tập tin tiêu đề của Windows dành riêng cho ngôn ngữ lập trình C và C++. Trong đó chứa các khai báo cho tất cả các hàm (function) trong Windows API, tất cả các macro thường dùng bởi các lập trình viên Windows, và tất cả các kiểu dữ liệu (data type) sử dụng cho nhiều hàm và hệ thống con (subsystem).

Cơ sở lý thuyết:

- Lý thuyết về lớp (Class), hướng đối tượng(OOP): đóng gói (Encapsulation), tính đa hình (Polymorphism), bản mẫu (Template).
- Lý thuyết cấu trúc dữ liệu: cấu trúc dữ liệu Vector và cấu trúc cây nhị phân tìm kiếm (Binary Search Tree).
- Lý thuyết con trỏ, cấp phát động.

PHẦN 3: PHÂN TÍCH VÀ THIẾT KẾ GIẢI PHÁP

1. Phân tích bài toán cần thực hiện:

Đây là một cấu trúc dữ liệu dùng để quản lý dữ liệu môi trường của khu vực trồng trọt, cụ thể là nhà kính. Yêu cầu cơ bản của cấu trúc này là cần phải lưu được các giá trị:

- Nhiệt độ của đất và của không khí.
- Độ ẩm của đất và của không khí.
- Độ pH của đất.
- Nồng độ CO2 trong không khí.
- Giá trị quang thông(lumen) của khu vực.

Bên cạnh đó, các dữ liệu ở trên cần có thêm giá trị thời gian đi kèm, giá trị ngày của từng bộ dữ liệu để dễ dàng kiểm soát các mốc thời gian sự kiện, các giá trị đột biến, sự thay đổi giá trị theo thời gian. Từ đó có thể đưa ra giải pháp dự kiến cho các vấn đề hiện tại và rút kinh nghiệm cho tương lai.

Ngoài ra, với một hệ thống nhà kính hoặc vườn có diện tích đất trồng lớn, cần phải chia nhỏ các khu vực ra để quản lý vì vậy cấu trúc dữ liệu này cần có thêm khả năng quản lý một cách linh động các khu vực, có thể thêm bớt khu vực không giới hạn.

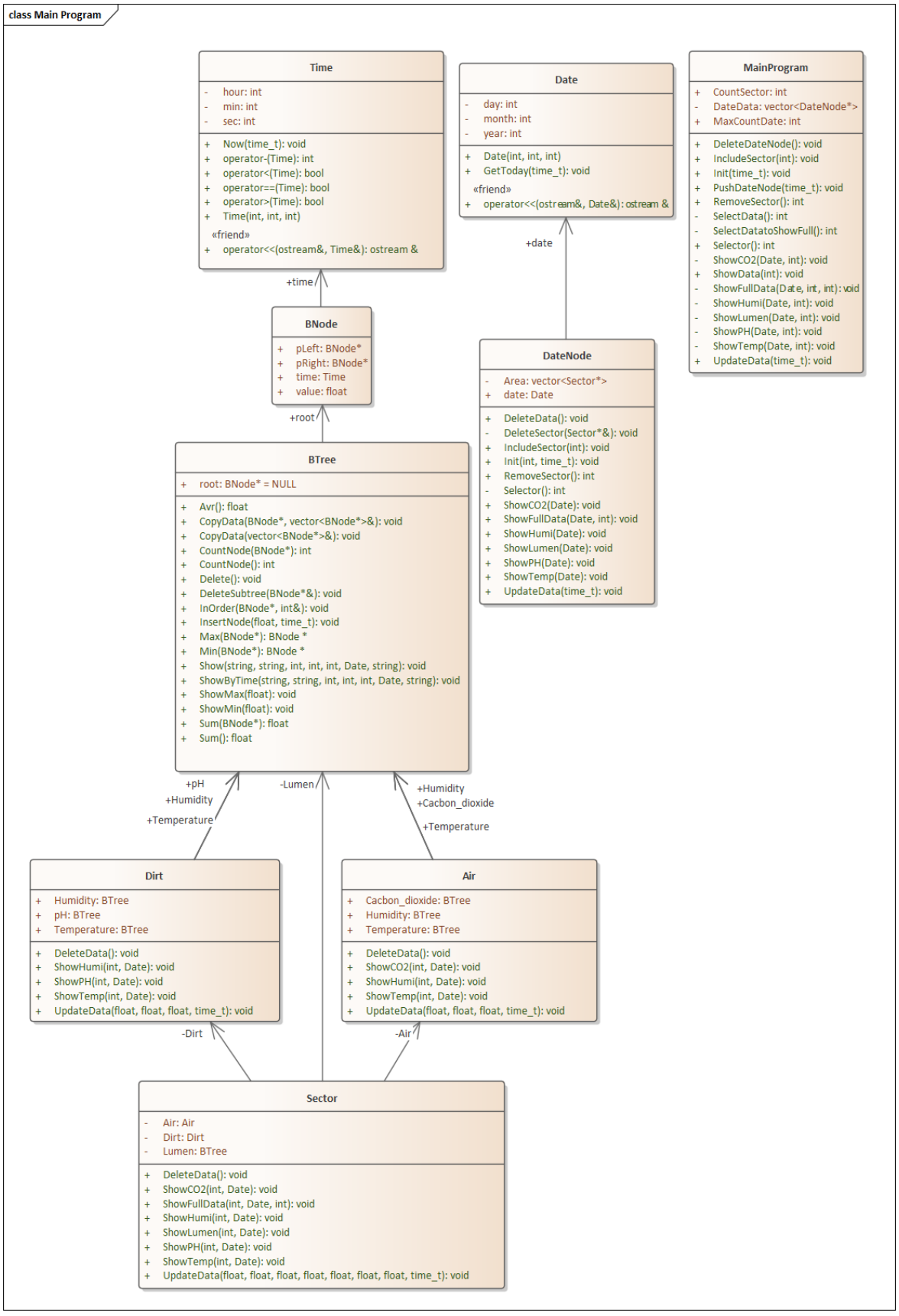
Cuối cùng, cần thiết kế giao diện cho chương trình quản lý này một cách trực quan và gọn gàng, có đầy đủ banner thông tin tên chương trình và các sinh viên thực hiện, có các thao tác và hướng dẫn sử dụng. Vì đây là một chương trình hoạt động trên ý tưởng tự động cập nhật dữ liệu theo thời gian, nên cần thiết kế bộ đếm thời gian giả lập giúp cho mọi người có thể nhìn rõ hơn về cách hoạt động của chương trình.

2. Ý tưởng thiết kế cấu trúc và sơ đồ khối hoạt động:

2.1 Ý tưởng thiết kế:

Yêu cầu cơ bản cho ứng dụng này là phải lưu trữ được các giá trị dữ liệu đọc được từ môi trường. Tùy vào khoảng cách giữa 2 lần cập nhật dữ liệu mà số lượng giá trị được lưu sẽ khác nhau, ngoài ra ứng dụng cần tính toán và xuất ra được các giá trị lớn nhất, giá trị nhỏ nhất, giá trị trung bình trong ngày, sắp xếp các giá trị theo thứ tự tăng dần. Nhóm quyết định sử dụng “Cây nhị phân tìm kiếm” để lưu trữ các dữ liệu môi trường, một cấu trúc dữ liệu có thể thực hiện tốt các vấn đề trên. Bên cạnh đó, nhóm còn thiết kế các lớp (class) để thực hiện quản lý từng môi trường, khu vực, ngày và giờ.

Sơ đồ các lớp cấu trúc dữ liệu:



Hình 2.1 Class diagram của chương trình

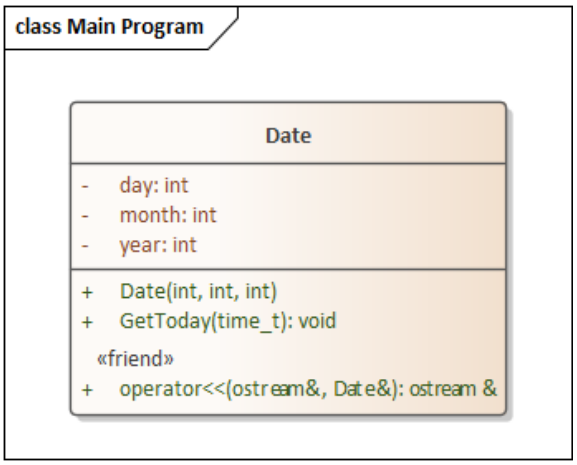
2.2 Thông tin về các class trong CTDL:

2.2.1 Class Date:

Đây là lớp dùng để định nghĩa dữ liệu ngày/tháng/năm, lớp hày hỗ trợ việc quản lý thời gian các dữ liệu và thời gian của phầm mềm giả lập.

Lớp Date gồm có 3 phần tử chính có kiểu dữ liệu là số nguyên, bao gồm:

- Ngày (day: int).
- Tháng (month: int).
- Năm (year: int).



Hình 2.2.1.1 Thông tin cơ bản của class Date

```
class Date{
private:
    int day,month,year;
public:

    Date(int day=0, int month=0, int year=0){
        this->day = day;
        this->month = month;
        this->year = year;
    }

    void GetToday(time_t now = time(0)){
        tm *ltm = localtime(&now);
        this->day = ltm->tm_mday;
        this->month = 1 + ltm->tm_mon;
        this->year = 1900 + ltm->tm_year;
    }

    friend ostream &operator<<(ostream &out, Date &tmp){
        out<<setfill('0');
        out<<setw(2)<<right<<tmp.day<<"/"<<setw(2)<<right<<tmp.month<<"/"<<tmp.year;
        out<<setfill(' ');
        return out;
    }
};
```

Hình 2.2.1.2 Nội dung đầy đủ của class Date

Và có 3 hàm con:

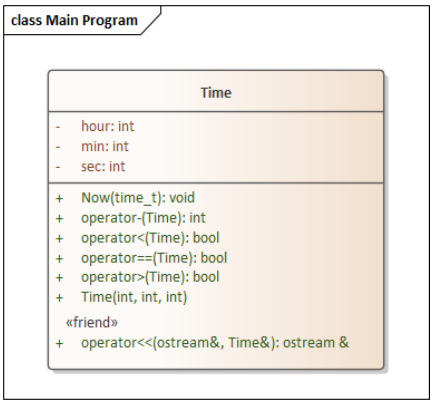
- Hàm khởi tạo Date(int, int, int) sẽ tạo ra biến có kiểu Date và có các giá trị ngày tháng năm bằng với đối số truyền vào, nếu không có đối số thì mặc định các giá trị đó bằng 0.
- Hàm GetToday(time_t) là hàm đặt các giá trị các phần tử dựa vào tham số "now", "now" là tham số mang giá trị số giây tính từ mốc thời gian gốc:01/01/1970, mặc định thì "now" sẽ có giá trị bằng "time(0)", time(0) là hàm trả về giá trị số giây hiện tại của hệ thống so với thời gian gốc, tức là mặc định thì hàm này sẽ đặt giá trị của các phần tử là ngày tháng năm của hệ thống.
- Hàm operator<<(ostream, Date) là hàm nạp chồng toán tử xuất "<<" dùng để đơn giản hóa cách xuất class Date ra màn hình.

2.2.2 Class Time:

Đây là lớp dùng để định nghĩa dữ liệu giờ/phút/giây, lớp hày hỗ trợ việc quản lý thời gian các dữ liệu và thời gian của phầm mềm giả lập.

Lớp Time gồm có 3 phần tử chính có kiểu dữ liệu là số nguyên, bao gồm:

- Giờ (hour: int).
- Phút (min: int).
- Giây (sec: int).



Hình 2.2.2.1 Thông tin cơ bản của class Time

Và có 6 hàm con:

- Hàm khởi tạo Time(int, int, int) sẽ tạo ra biến có kiểu Time và có các giá trị giờ phút giây bằng với đối số truyền vào, nếu không có đối số thì mặc định các giá trị đó bằng 0.

```
Time(int hour=0, int min=0, int sec=0){
    this->hour = hour;
    this->min = min;
    this->sec = sec;
}
```

Hình 2.2.2.2 Nội dung chi tiết của hàm Time

-

- Hàm Now(time_t) tương tự như hàm GetToday(time_t) thuộc class Date, hàm này sẽ đặt các giá trị giờ phút giây dựa vào số giây được truyền vào, mặc định là số giây hiện tại của hệ thống.

```
void Now(time_t now = time(0)){
    tm *ltm = localtime(&now);
    this->hour = ltm->tm_hour;
    this->min = ltm->tm_min;
    this->sec = ltm->tm_sec;
}
```

Hình 2.2.2.3 Nội dung chi tiết của hàm Now

- Hàm operator<<(ostream, Time) là hàm nạp chồng toán tử xuất "<<" dùng để đơn giản hóa cách xuất class Date ra màn hình.

```
friend ostream &operator<<(ostream &out, Time &tmp){
    out<<setfill('0');
    out<<setw(2)<<right<<tmp.hour<<":"<<setw(2)<<right<<tmp.min<<":"<<setw(2)<<tmp.sec;
    out<<setfill(' ');
    return out;
}
```

Hình 2.2.2.4 Nội dung chi tiết của hàm operator<<

- Hàm operator>(Time) là hàm nạp chồng toán tử so sánh lớn hơn cho kiểu dữ liệu Time, hàm này hoạt động bằng cách so sánh tổng giá trị giây của 2 biến Time và đưa ra kết quả 0(false) hoặc 1(true).

```
bool operator>(Time a){
    int tmp1 = hour*3600 + min*60 + sec;
    int tmp2 = a.hour*3600 + a.min*60 + a.sec;
    if(tmp1 <= tmp2)
        return 0;
    return 1;
}
```

Hình 2.2.2.5 Nội dung chi tiết của hàm operator>

- Hàm operator<(Time) là hàm nạp chồng toán tử so sánh bé hơn cho kiểu dữ liệu Time, hàm này hoạt động bằng cách so sánh tổng giá trị giây của 2 biến

```
bool operator<(Time a){
    int tmp1 = hour*3600 + min*60 + sec;
    int tmp2 = a.hour*3600 + a.min*60 + a.sec;
    if(tmp1 >= tmp2)
        return 0;
    return 1;
}
```

Time và đưa ra kết quả 0(false) hoặc 1(true).

Hình 2.2.2.5 Nội dung chi tiết của hàm operator<

-
- Hàm operator==(Time) là hàm nạp chồng toán tử so sánh bằng cho kiểu dữ liệu Time, hàm này hoạt động bằng cách lần lượt so sánh các phần tử tương ứng của 2 biến Time và đưa ra kết quả 0(false) hoặc 1(true).

```
bool operator==(Time a){
    if(this->sec==a.sec)
        if(this->min==a.min)
            if(this->hour == a.hour)
                return 1;
    return 0;
}
```

Hình 2.2.2.6 Nội dung chi tiết của hàm operator==

- Hàm operator-(Time) là hàm nạp chồng toán tử phép trừ với giá trị trả về là số giây trôi qua giữa 2 biến kiểu Time.

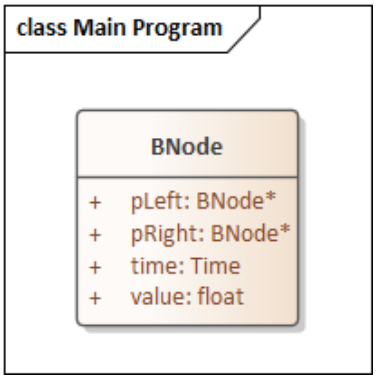
```
int operator-(Time a){
    return this->sec - a.sec + (this->min-a.min)*60 + (this->hour-a.hour)*3600;
}
```

Hình 2.2.2.7 Nội dung chi tiết của hàm operator-

2.2.3 Class BNode:

Đây là lớp định nghĩa kiểu dữ liệu 1 nút của cây nhị phân tìm kiếm, dùng để lưu 1 giá trị dữ liệu môi trường. Lớp này chỉ có 4 phần tử:

- Giá trị (value: float).
- Thời gian nhận giá trị (time: Time).
- Con trỏ lưu địa chỉ nút trái (pLeft: BNode*)
- Con trỏ lưu địa chỉ nút phải (pRight: BNode*)

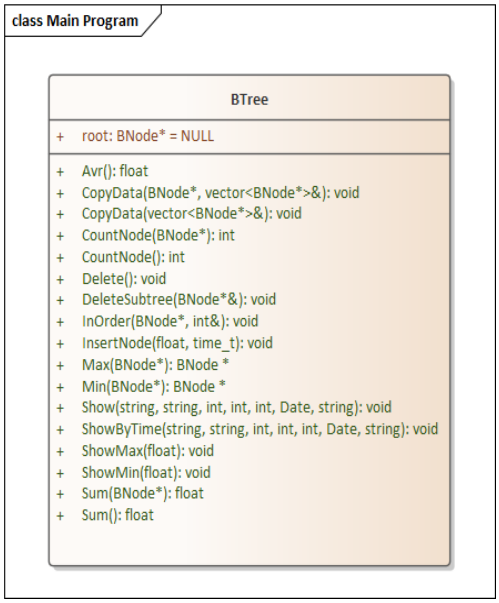


Hình 2.2.3 Thông tin cơ bản của class Bnode

2.2.4 Class BTree:

Đây là lớp định nghĩa kiểu cấu trúc dữ liệu cây nhị phân tìm kiếm, dùng để quản lý các giá trị bằng cách quản lý các BNode. Lớp này có duy nhất một phần tử là:

- nút gốc (root: BNode*).



Hình 2.2.4.1 Thông tin cơ bản của class Btree

Và có 17 hàm con:

- Hàm CountNode(BNode*) là hàm sử dụng đệ quy để đếm số lượng giá trị được lưu tính từ đối số nút truyền vào.

```
int CountNode(BNode *tmp){
    if(tmp==NULL)
        return 0;
    else
        return 1 + CountNode(tmp->pLeft) + CountNode(tmp->pRight);
}
```

Hình 2.2.4.2 Nội dung chi tiết của hàm CountNode

- Hàm CountNode() là hàm được định nghĩa để dùng cho việc đếm số giá trị được lưu trên cả cây.

```
int CountNode(){
    return CountNode(root);
}
```

Hình 2.2.4.3 Nội dung chi tiết của hàm CountNode()

- Hàm Sum(BNode*) là hàm trả về tổng các giá trị được lưu tính từ đối số nút được truyền vào.

```
float Sum(BNode *tmp){
    if(tmp==NULL)
        return 0;
    else
        return tmp->value + Sum(tmp->pLeft) + Sum(tmp->pRight);
}
```

Hình 2.2.4.4 Nội dung chi tiết của hàm Sum

- Hàm Sum() là hàm trả về tổng các giá trị được lưu trên cả cây.

```
float Sum(){
    return Sum(root);
}
```

Hình 2.2.4.5 Nội dung chi tiết của hàm Sum

- Hàm Avr() là hàm trả về giá trị trung bình của tất cả cá giá trị được lưu trong cây nhị phân.

```
float Avr(){
    return Sum()/float(CountNode());
}
```

Hình 2.2.4.6 Nội dung chi tiết của hàm Avr

- Hàm Min(BNode*) là hàm trả về con trỏ trỏ đến nút có giá trị nhỏ nhất tính từ đối số truyền vào, chính là giá trị nằm ở nút cây con trái nhất.

```
BNode *Min(BNode *tmp){
    if(tmp->pLeft==NULL)
        return tmp;
    else
        return Min(tmp->pLeft);
}
```

Hình 2.2.4.7 Nội dung chi tiết của hàm Min

- Hàm ShowMin(float) là hàm in ra màn hình giá trị nhỏ nhất của cả cây nhị phân tìm kiếm, khi giá trị nhỏ nhất của cây nhị phân nhỏ hơn tham số cảnh báo được truyền vào thì thông tin sẽ được in ra với màu đỏ kèm với dòng cảnh báo. Hàm SetConsoleTextAttribute là hàm thuộc thư viện windows.h, dùng để chỉnh màu khi in dữ liệu lên màn hình console.

```
void ShowMin(float Warning){
    cout<<"Gia tri nho nhat: ";
    if(Min(root)->value<=Warning){
        SetConsoleTextAttribute(hConsole, 4+15*16);
        cout<<setw(6)<<right<<Min(root)->value<<"\t"<<Min(root)->time<<"  (Canh bao!!!)"<<endl;
        SetConsoleTextAttribute(hConsole, 0+15*16);
    }
    else{
        cout<<setw(6)<<right<<Min(root)->value<<"\t"<<Min(root)->time<<endl;
    }
}
```

Hình 2.2.4.8 Nội dung chi tiết của hàm ShowMin

- Hàm Max(BNode*) là hàm trả về con trỏ trỏ đến nút có giá trị lớn nhất tính từ đối số truyền vào, chính là giá trị nằm ở nút cây con phải nhất.

```
BNode *Max(BNode *tmp){
    if(tmp->pRight==NULL)
        return tmp;
    else
        return Max(tmp->pRight);
}
```

Hình 2.2.4.9 Nội dung chi tiết của hàm Max

- Hàm ShowMax(float) là hàm in ra màn hình giá trị lớn nhất của cả cây nhị phân tìm kiếm, khi giá trị lớn nhất của cây nhị phân lớn hơn tham số cảnh báo được truyền vào thì thông tin sẽ được in ra với màu đỏ kèm với dòng cảnh báo.

```
void ShowMax(float Warning){
    cout<<"Gia tri lon nhat: ";
    if(Max(root)->value>=Warning){
        SetConsoleTextAttribute(hConsole, 4+15*16);
        cout<<setw(6)<<right<<Max(root)->value<<"\t"<<Max(root)->time<<"  (Canh bao!!!)"<<endl;
        SetConsoleTextAttribute(hConsole, 0+15*16);
    }
    else{
        cout<<setw(6)<<right<<Max(root)->value<<"\t"<<Max(root)->time<<endl;
    }
}
```

Hình 2.2.4.10 Nội dung chi tiết của hàm ShowMax

- Hàm DeleteSubtree(BNode*) là hàm xóa toàn bộ các giá trị được thêm vào sau nút đối số truyền vào hàm, thu hồi lại bộ nhớ được cấp phát cho các nút.

```
void DeleteSubtree(BNode *&tmp){
    if(tmp==NULL){
        return;
    }
    DeleteSubtree(tmp->pLeft);
    DeleteSubtree(tmp->pRight);
    delete tmp;
    tmp = NULL;
}
```

Hình 2.2.4.11 Nội dung chi tiết của hàm DeleteSubtree

- Hàm Delete() là hàm dùng để xóa toàn bộ cây nhị phân tìm kiếm.

```
void Delete(){
    DeleteSubtree(root);
}
```

Hình 2.2.4.12 Nội dung chi tiết của hàm Delete

- Hàm InOrder(BNode *) là hàm duyệt cây nhị phân theo thứ tự giữa và in chúng ra màn hình, từ đó ta được thứ tự sắp xếp của các giá trị đã lưu tăng dần.

```
void InOrder(BNode *root, int &color){
    if(root!=NULL){
        InOrder(root->pLeft,color);
        SetConsoleTextAttribute(hConsole, (color+1)+15*16);
        cout<<setw(10)<<right<<root->value<<"\t "<<root->time<<endl;
        color++;
        if(color>5)
            color=1;
        InOrder(root->pRight,color);
    }
}
```

Hình 2.2.4.13 Nội dung chi tiết của hàm InOrder

- Hàm InsertNode(float, time_t) là hàm tạo thêm nút mang giá trị môi trường được gởi về và thêm nút đó vào cây nhị phân tìm kiếm bằng cách tìm kiếm vị trí phù hợp với nó.

```
void InsertNode(float data, time_t now){
    BNode *node = CreateNode(data, now);
    if(root == NULL){
        root = node;
    }
    else{
        BNode *Tmp = root;
        BNode *Parent = NULL;
        while(Tmp){
            Parent = Tmp;
            if(Tmp->value>node->value)
                Tmp = Tmp->pLeft;
            else
                Tmp = Tmp->pRight;
        }
        if(Parent->value>node->value)
            Parent->pLeft = node;
        else
            Parent->pRight = node;
    }
}
```

Hình 2.2.4.14 Nội dung chi tiết của hàm InsertNode

- Hàm Show(string, string, int, int, Date, string) là hàm in tất cả các giá trị được lưu vào cây nhị phân bao gồm cả thông tin về giá trị lớn nhất, giá trị nhỏ nhất, giá trị trung bình và các giá trị được lưu vào cây sẽ in ra theo thứ tự sắp xếp tăng dần theo giá trị, nếu như giá trị trung bình nằm ngoài giới hạn giữa 2 tham số min và max thì sẽ in ra một dòng cảnh báo và gợi ý hành động cần thực hiện. Ngoài ra sau khi dùng phím Tab sẽ gọi đến hàm ShowByTime() để in ra theo thứ tự thời gian của các giá trị. Hàm GetKeyState chính là hàm kiểm tra trạng thái phím Tab.

```
    void Show(string message1, string message2, int min, int max, int i, Date time, string Name){
        system("CLS");
        cout << " Khu vuc: " << i << endl << " " << time << endl <<  " " << Name << endl;
        if(root==NULL){
            cout<<"Khong co gia tri duoc cap nhat!"<<endl;
            cout<<"Nhap phim bat ki de tiep tuc chuong trinh!"<<endl;
            system("pause >nul");
            return;
        }
        cout<<"Gia tri trung binh: ";
        float avr = Avr();
        if(avr<=min){
            SetConsoleTextAttribute(hConsole, 4+15*16);
            cout<<avr<<endl;
            cout<<message1<<endl;
            SetConsoleTextAttribute(hConsole, 0+15*16);
        }
        else if(avr>=max){
            SetConsoleTextAttribute(hConsole, 4+15*16);
            cout<<avr<<endl;
            cout<<message2<<endl;
            SetConsoleTextAttribute(hConsole, 0+15*16);
        }
        else
            cout<<Avr()<<endl;
        ShowMin(min);
        ShowMax(max);
        cout<<endl<<setw(10)<<right<<"Gia tri"<<"\tThoi gian"<<endl;
        cout<<setfill('-');
        cout<<setw(25)<<"-"<<endl;
        cout<<setfill(' ');
        int color = 1;
        InOrder(root, color);
        cout<<endl;
        SetConsoleTextAttribute(hConsole, 0+15*16);
        cout<<"Nhap phim TAB de sap xep theo thoi gian!"<<endl;
        cout<<"Nhap phim bat ki de tiep tuc chuong trinh!"<<endl;
        system("pause >nul");
        if ((GetKeyState(VK_TAB) & 0x8000))
            {
                ShowByTime(message1, message2, min, max, i, time, Name);
            }
    }
};
```

Hình 2.2.4.15 Nội dung chi tiết của hàm Show

- Hàm ShowByTime(string, string, int, int, Date, string) tương tự như hàm Show, là hàm in tất cả các giá trị được lưu vào cây nhị phân bao gồm cả thông tin về giá trị lớn nhất, giá trị nhỏ nhất, giá trị trung bình, nếu như giá trị trung bình nằm ngoài giới hạn giữa 2 tham số min và max thì sẽ in ra một dòng cảnh báo và gợi ý hành động cần thực hiện, tuy nhiên các giá trị được lưu vào cây sẽ in ra theo thứ tự sắp xếp tăng dần theo thời gian, sau khi dùng

phím Tab sẽ gọi đến hàm Show() để in ra theo thứ tự sắp xếp tăng dần theo giá trị.

```
void ShowByTime(string message1, string message2, int min, int max, int i, Date time, string Name){
    system("CLS");
    cout << " Khu vuc: " << i << endl << " " << time << endl <<  " " << Name << endl;
    if(root==NULL){
        cout<<"Khong co gia tri duoc cap nhat!"<<endl;
        cout<<"Nhap phim bat ki de tiep tuc chuong trinh!"<<endl;
        system("pause >nul");
        return;
    }
    cout<<"Gia tri trung binh: ";
    float avr = Avr();
    if(avr<=min){
        SetConsoleTextAttribute(hConsole, 4+15*16);
        cout<<avr<<endl;
        cout<<message1<<endl;
        SetConsoleTextAttribute(hConsole, 0+15*16);
    }
    else if(avr>=max){
        SetConsoleTextAttribute(hConsole, 4+15*16);
        cout<<avr<<endl;
        cout<<message2<<endl;
        SetConsoleTextAttribute(hConsole, 0+15*16);
    }
    else
        cout<<avr<<endl;
    ShowMin(min);
    ShowMax(max);
    cout<<endl<<setw(10)<<right<<"Gia tri"<<"\tThoi gian"<<endl;
    cout<<setfill('-');
    cout<<setw(25)<<"-"<<endl;
    cout<<setfill(' ');
    int countnode = CountNode();
    vector<BNode*> tmp;
    CopyData(root, tmp);
    sort(tmp.begin(),tmp.end(),compareBNode);
    int color=0;
    for(int i=0;i<countnode;i++){
        color++;
        if(color>5)
            color=1;
        SetConsoleTextAttribute(hConsole, (color+1)+15*16);
        cout<<setw(10)<<right<<tmp[i]->value<<"\t "<<tmp[i]->time<<endl;
    }
    SetConsoleTextAttribute(hConsole, 0+15*16);
    cout<<endl<<"Nhap phim TAB de sap xep theo gia tri!"<<endl;
    cout<<"Nhap phim bat ki de tiep tuc chuong trinh!"<<endl;
    system("pause >nul");
    if ((GetKeyState(VK_TAB) & 0x8000))
        {
            Show(message1,message2, min, max, i, time, Name);
        }
}
```

Hình 2.2.4.16 Nội dung chi tiết của hàm ShowByTime

- Hàm CopyData(Bnode*, vector<Bnode*>) là hàm dùng duyệt giữa để sao chép toàn bộ dữ liệu của tham số nút được truyền vào và toàn bộ cây con của nó vào một vector.

```
void CopyData(BNode *root, vector<BNode*> &M){
    if(root!=NULL){
        CopyData(root->pLeft,M);
        M.push_back(root);
        CopyData(root->pRight,M);
    }
}
```

Hình 2.2.4.17 Nội dung chi tiết của hàm CopyData

- Hàm CopyData(vector<Bnode*>) là hàm dùng để sao chép toàn bộ dữ liệu của cây nhị phân vào tham số vector.

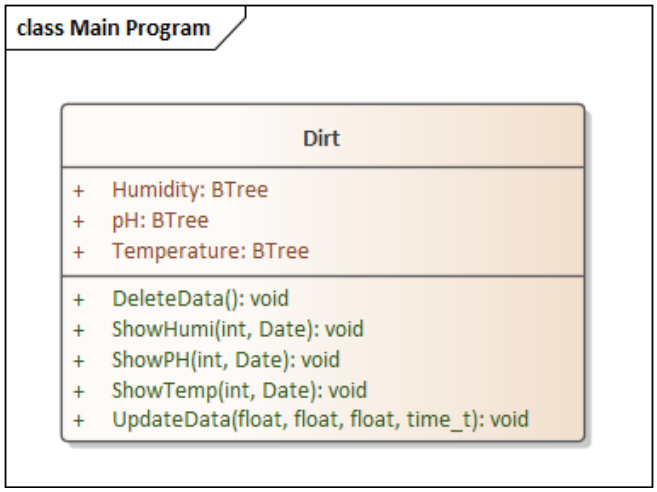
```
void CopyData(vector<BNode*> &M){
    CopyData(root,M);
}
```

Hình 2.2.4.18 Nội dung chi tiết của hàm CopyData

2.2.5 Class Dirt

Đây là lớp quản lý 3 dữ liệu nhiệt độ, độ ẩm, độ pH của đất. Lớp này bao gồm 3 phần tử:

- Nhiệt độ (Temperature: BTree)
- Độ ẩm (Humidity: BTree)
- Độ pH (pH: BTree)



Hình 2.2.5.1 Thông tin cơ bản của class Dirt

Và có 5 hàm con:

- Hàm UpdateData(float, float, float, time_t) là hàm cập nhật thêm giá trị cho các dữ liệu của đất.

```
void UpdateData(float Temp, float Humi, float pH, time_t now){
    this->Temperature.InsertNode(Temp, now);
    this->Humidity.InsertNode(Humi, now);
    this->pH.InsertNode(pH, now);
}
```

Hình 2.2.5.2 Nội dung chi tiết của hàm UpdateData

- Hàm ShowHumi(int, Date) là hàm in dữ liệu độ ẩm và các cảnh báo của đất ra màn hình.

```
void ShowHumi(int i, Date time){
    string message1 = "Do am cua dat kha thap, can bat he thong phun suong hoac giam nhiet do cua dat xuong!!";
    string message2 = "Do am cua dat kha cao, can giam luong nuoc tuoi hoac tang nhiet do cua dat len!!";
    Humidity.Show(message1, message2, 50, 80, i, time, "Do am cua dat");
}
```

Hình 2.2.5.3 Nội dung chi tiết của hàm ShowHumi

- Hàm ShowPH(int, Date) là hàm in dữ liệu độ pH và các cảnh báo của đất ra màn hình.

```
void ShowPH(int i, Date time){
    string message1 = "Do pH cua dat kha thap, nen bon them phan lan hoac voi de tang len!!";
    string message2 = "Do pH cua dat kha cao, can bon cac loai phan huu co de giam xuong!!";
    pH.Show(message1, message2, 6, 8, i, time, "Do pH cua dat");
}
```

Hình 2.2.5.4 Nội dung chi tiết của hàm ShowPH

- Hàm ShowTemp(int, Date) là hàm in dữ liệu nhiệt độ của đất ra màn hình.

```
void ShowTemp(int i, Date time){
    string message1 = "Nhiet do trung binh cua dat kha thap, can don trong dat va thu gon mai che de dat tiep xuc anh sang tot!!";
    string message2 = "Nhiet do trung binh cua dat kha cao, can bat he thong phun suong, nen che phu dat hoac bat mai che!!";
    Temperature.Show(message1, message2, 20, 35, i, time, "Nhiet do cua dat");
}
```

Hình 2.2.5.5 Nội dung chi tiết của hàm ShowTemp

- Hàm DeleteData() là hàm xóa toàn bộ giá trị được lưu trong các dữ liệu của đất.

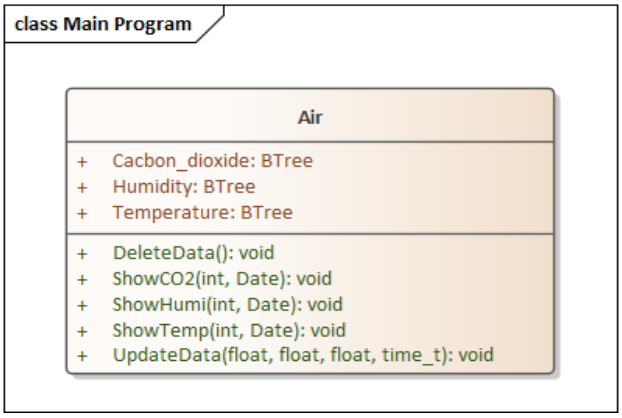
```
void DeleteData(){
    Temperature.Delete();
    Humidity.Delete();
    pH.Delete();
}
```

Hình 2.2.5.6 Nội dung chi tiết của hàm DeleteData

2.2.6 Class Air

Đây là lớp quản lý 3 dữ liệu nhiệt độ, độ ẩm, nồng độ khí CO_2 của không khí. Lớp này bao gồm 3 phần tử:

- Nhiệt độ (Temperature: BTree)
- Độ ẩm (Humidity: BTree)
- Nồng độ CO_2 (Cacbon_dioxide: BTree)



Hình 2.2.6.1 Class diagram của class Air

Và có 5 hàm con:

- Hàm UpdateData(float, float, float, time_t) là hàm cập nhật thêm giá trị cho các dữ liệu của không khí.

```
void UpdateData(float Temp, float Humi, float pH, time_t now){
    this->Temperature.InsertNode(Temp, now);
    this->Humidity.InsertNode(Humi, now);
    this->pH.InsertNode(pH, now);
}
```

Hình 2.2.6.2 Nội dung chi tiết của hàm UpdateData

- Hàm ShowHumi(int, Date) là hàm in dữ liệu độ ẩm của không khí ra màn hình.

```
void ShowHumi(int i, Date time){
    string message1 = "Do am trung binh cua khong khi kha thap, can bat he thong phun suong, giam hoat dong cua he thong thong khi!!";
    string message2 = "Do am trung binh cua khong khi kha cao, can bat he thong thong khi, tang nhiet do khong khi len!!";
    Humidity.Show(message1, message2, 50, 80, i, time, "Do am cua khong khi");
}
```

Hình 2.2.6.3 Nội dung chi tiết của hàm ShowHumi

- Hàm ShowCO2(int, Date) là hàm in dữ liệu nồng độ khí CO_2 của không khí ra màn hình.

```
void ShowCO2(int i, Date time){
    string message1 = "Do am trung binh cua khong khi kha thap, can bat he thong phun suong, giam hoat dong cua he thong thong khi!!";
    string message2 = "Do am trung binh cua khong khi kha cao, can bat he thong thong khi, tang nhiet do khong khi len!!";
    Cacbon_dioxide.Show(message1, message2, 0.025, 0.035, i, time, "Nong do CO2 cua khong khi");
}
```

Hình 2.2.6.4 Nội dung chi tiết của hàm ShowCO2

- Hàm ShowTemp(int, Date) là hàm in dữ liệu nhiệt độ của không khí ra màn hình.

```
void ShowTemp(int i, Date time){
    string message1 = "Nhiet do trung binh cua khong khi kha thap, can thu gon mai che hoac tang do sang!!";
    string message2 = "Nhiet do trung binh cua khong khi kha cao, nen che bot sang, bat he thong phun suong hoac bat he thong thong khi!!";
    Temperature.Show(message1, message2, 20, 35, i, time, "Nhiet do cua khong khi");
}
```

Hình 2.2.6.5 Nội dung chi tiết của hàm ShowTemp

- Hàm DeleteData() là hàm xóa toàn bộ giá trị được lưu trong các dữ liệu của không khí.

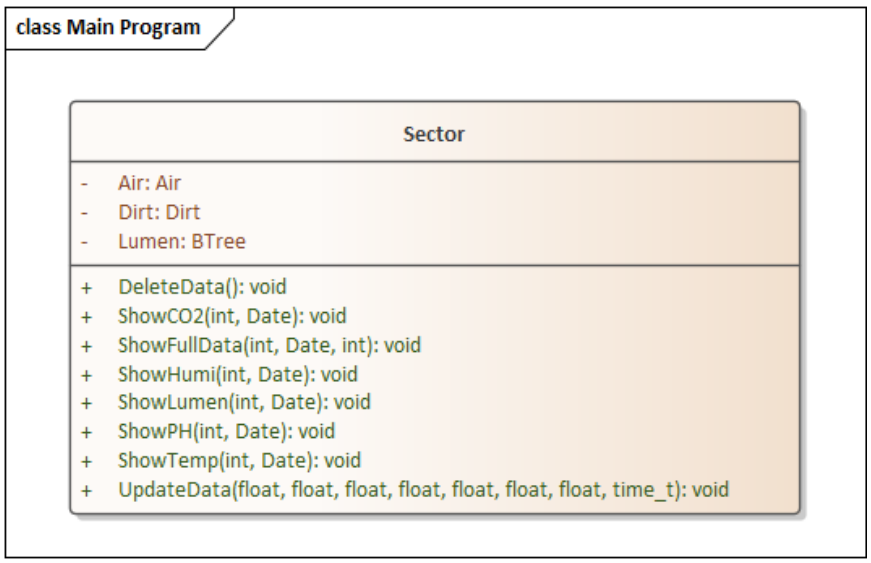
```
void DeleteData(){
    Temperature.Delete();
    Humidity.Delete();
    pH.Delete();
}
```

Hình 2.2.5.6 Nội dung chi tiết của hàm DeleteData

2.2.7 Class Sector

Đây là lớp đại diện cho một khu vực, quản lý tất cả dữ liệu nhiệt độ, độ ẩm, độ pH đất, nồng độ khí CO_2 trong không khí và quang thông của cả một khu vực đó, cụ thể là của một nhà kính. Lớp này có 3 phần tử:

- Môi trường đất (Dirt: Dirt)
- Môi trường không khí (Air: Air)
- Quang thông (Lumen: Btree)



Hình 2.2.7.1 Thông tin cơ bản của class Sector

Và lớp này có 8 hàm con:

- Hàm DeleteData() là hàm xóa toàn bộ các giá trị dữ liệu được lưu trong một khu vực.

```
void DeleteData(){
    Dirt.DeleteData();
    Air.DeleteData();
    Lumen.Delete();
}
```

Hình 2.2.7.2 Hình ảnh chi tiết hàm DeleteData

- Hàm ShowCO2(int, Date) là hàm in giá trị của nồng độ CO2 trong không khí của chính khu vực này.

```
void ShowCO2(int i, Date time){
    Air.ShowCO2(i, time);
}
```

Hình 2.2.7.3 Hình ảnh chi tiết hàm ShowCO2

- Hàm ShowHumi(int, Date) là hàm cho phép lựa chọn in giá trị độ ẩm của môi trường đất hay môi trường không khí, sau đó sẽ thực hiện in giá trị tương ứng lên màn hình

```
void ShowHumi(int i, Date time){
    bool tmp;
    system("CLS");
    cout<<"Ban muon xem nhiet do cua:"<<endl;
    cout<<"[0]: Dat"<<endl<<"[1]: Khong khi"<<endl;
    cout<<"Nhap lua chon cua ban: ";cin>>tmp;
    if(tmp)
        Air.ShowHumi(i, time);
    else
        Dirt.ShowHumi(i, time);
}
```

Hình 2.2.7.4 Hình ảnh chi tiết hàm ShowHumi

- Hàm ShowLumen(int, Date) là hàm là hàm in giá trị của quang thông đo được của chính khu vực này.

```
void ShowLumen(int i, Date time){
    string message1 = "Do am trung binh cua khong khi kha thap, can bat he thong phun suong, giam hoat dong cua he thong thong khi!!";
    string message2 = "Do am trung binh cua khong khi kha cao, can bat he thong thong khi, tang nhiet do khong khi len!!";
    Lumen.Show(message1, message2, 80, 400, i, time, "Quang thong");
}
```

Hình 2.2.7.5 Hình ảnh chi tiết hàm ShowLumen

- Hàm ShowPH(int, Date) là hàm in giá trị của độ pH trong đất của chính khu vực này.

```
void ShowPH(int i, Date time){
    Dirt.ShowPH(i, time);
}
```

Hình 2.2.7.6 Hình ảnh chi tiết hàm ShowPH

- Hàm ShowTemp(int, Date) là hàm cho phép lựa chọn in giá trị nhiệt độ của môi trường đất hay môi trường không khí, sau đó sẽ thực hiện in giá trị tương ứng lên màn hình.

```
void ShowTemp(int i, Date time){
    bool tmp;
    system("CLS");
    cout<<"Ban muon xem nhiet do cua:"<<endl;
    cout<<"[0]: Dat"<<endl<<"[1]: Khong khi"<<endl;
    cout<<"Nhap lua chon cua ban: ";cin>>tmp;
    if(tmp)
        Air.ShowTemp(i, time);
    else
        Dirt.ShowTemp(i, time);
}
```

Hình 2.2.7.7 Hình ảnh chi tiết hàm ShowTemp

- Hàm UpdateData(float,float,float,float,float,float,float,time_t) là hàm cập nhật thêm giá trị cho các dữ liệu của cả khu vực.

```
void UpdateData(float Temp1, float Humi1, float pH, float Temp2, float Humi2, float CO2, float Lumen, time_t now){
    this->Dirt.UpdateData(Temp1, Humi1, pH, now);
    this->Air.UpdateData(Temp2, Humi2, CO2, now);
    this->Lumen.InsertNode(Lumen, now);
}
```

Hình 2.2.7.8 Hình ảnh chi tiết hàm UpdateData

- Hàm ShowFullData(int, Date, int) là hàm cho phép lựa chọn và in hàng loạt những dữ liệu mà mình mong muốn trên một màn hình, các giá trị sẽ được sắp xếp theo thứ tự tăng dần của thời gian.

```
void ShowFullData(int i,Date time,int a){
    system("CLS");
    cout << setw(105) << "" << "Khu vuc: " << i << endl;
    cout << setw(105) << "" << time << endl;
    bool Select[7] = {};
    int surplus;
    int tmp = a;
    int count=0;
    while(tmp!=1){
        surplus = tmp % 10;
        Select[surplus] = 1;
        tmp=tmp/10;
        count++;
    }
    vector<BNode*> TempDirt, HumiDirt, PH;
    vector<BNode*> TempAir, HumiAir, CO2, lumen;
    string Name[7]={"Nhiet do cua dat","Do am cua dat","Do pH cua dat","Nhiet do cua khong khi",
                    "Do am cua khong khi","Nong do khi CO2","Quang thong"};
    BNode *Min[7], *Max[7];
    float Avr[7];
    if(Lumen.root==NULL){
        cout<<setw(89)<<""<<"Hien tai chua co du lieu cap nhat!"<<endl;
        cout<<setw(85)<<""<<"Nhap phim bat ki de tiep tuc chuong trinh!";
        system("pause >nul");
        return;
    }
```

Hình 2.2.7.9.1 Hình ảnh phần kiểm tra yêu cầu của người dùng và khởi tạo các vector để sao chép dữ liệu

```
for(int i=0;i<7;i++){
    if(Select[i]==0)
        continue;
    switch(i){
        case(0):{
            if(Select[i]){
                Dirt.Temperature.CopyData(TempDirt);
                sort(TempDirt.begin(),TempDirt.end(),compareNode);
                Min[0] = Dirt.Temperature.Min(Dirt.Temperature.root);
                Max[0] = Dirt.Temperature.Max(Dirt.Temperature.root);
                Avr[0] = Dirt.Temperature.Avr();
                break;
            }
        }
        case(1):{
            if(Select[i]){
                Dirt.Humidity.CopyData(HumiDirt);
                sort(HumiDirt.begin(),HumiDirt.end(),compareNode);
                Min[1] = Dirt.Humidity.Min(Dirt.Humidity.root);
                Max[1] = Dirt.Humidity.Max(Dirt.Humidity.root);
                Avr[1] = Dirt.Humidity.Avr();
                break;
            }
        }
        case(2):{
            if(Select[i]){
                Dirt.pH.CopyData(PH);
                sort(PH.begin(),PH.end(),compareNode);
                Min[2] = Dirt.pH.Min(Dirt.pH.root);
                Max[2] = Dirt.pH.Max(Dirt.pH.root);
                Avr[2] = Dirt.pH.Avr();
                break;
            }
        }
        case(3):{
```

Hình 2.2.7.9.2 Hình ảnh minh họa phần kiểm tra yêu cầu của người dùng và sao chép các dữ liệu theo yêu cầu

```
vector<vector<BNode*> > Data{TempDirt, HumiDirt, PH, TempAir, HumiAir, CO2, lumen};
int align = (211 - 30*count)/2;
cout<<setw(align)<<"";
for(int i=0;i<7;i++){
    if(Select[i]){
        for(int j=1;j<=30;j++)
            cout<<"_";
    }
}
cout<<endl;
cout<<setw(align-1)<<""<<"|";
for(int i=0;i<7;i++){
    if(Select[i])
        cout<<setw(29)<<left<<Name[i]<<"|";
}
cout<<endl;
cout<<setw(align-1)<<""<<"|";
for(int i=0;i<7;i++){
    if(Select[i]){
        cout<<left<<"Gia tri trung binh: "<<setw(9)<<left<<Avr[i]<<"|";
    }
}
cout<<endl;
cout<<setw(align-1)<<""<<"|";
for(int i=0;i<7;i++){
    //cout<<Select[i];
    if(Select[i]){
        cout<<setw(29)<<left<<"Gia tri nho nhat: "<<"|";
    }
}
cout<<endl;
cout<<setw(align-1)<<""<<"|";
for(int i=0;i<7;i++){
    if(Select[i]){
        cout<<setw(7)<<left<<Min[i]->value<<"    "<<Min[i]->time<<setw(11)<<right<<"|";
    }
}
cout<<endl;
cout<<setw(align-1)<<""<<"|";
for(int i=0;i<7;i++){
    if(Select[i]){
        cout<<setw(29)<<left<<"Gia tri lon nhat: "<<"|";
    }
}
cout<<endl;
cout<<setw(align-1)<<""<<"|";
for(int i=0;i<7;i++){
    if(Select[i]){
        cout<<setw(7)<<left<<Max[i]->value<<"    "<<Max[i]->time<<setw(11)<<right<<"|";
    }
}
```

Hình 2.2.7.9.3 Hình ảnh minh họa phần kiểm tra yêu cầu của người dùng và in các giá trị lớn nhất nhỏ nhất, trung bình của dữ liệu được yêu cầu

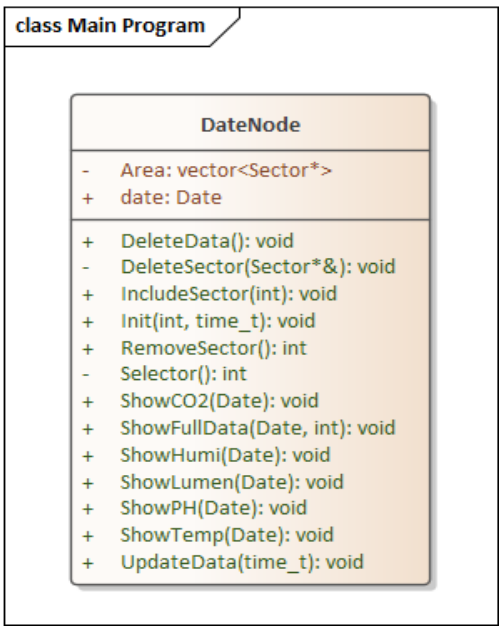
```
    cout<<setw(align-1)<<""<<"|";
    for(int i=0;i<7;i++){
        //cout<<Select[i];
        if(Select[i]){
            cout<<setw(10)<<right<<"Gia tri"<<setw(19)<<left<<"    Thoi gian"<<"|";
        }
    }
    cout<<endl;
    int color;
    for(int line=0;line<Lumen.CountNode();line++){
        cout<<setw(align-1)<<""<<"|";
        color++;
        if(color>5){
            color=1;
        }
        for(int type=0;type<7;type++){
            if(Select[type]){
                SetConsoleTextAttribute(hConsole, (color+1)+15*16);
                cout<<setw(10)<<right<<Data[type][line]->value<<"    "<<Data[type][line]->time<<setw(8)<<right;
                SetConsoleTextAttribute(hConsole, 0+15*16);
                cout<<"|";
            }
        }
        cout<<endl;
    }
    cout<<setw(align-1)<<""<<"|";
    for(int i=0;i<7;i++){
        if(Select[i]){
            for(int j=1;j<=29;j++)
                cout<<"_";
            cout<<"|";
        }
    }
    cout<<endl;
    cout<<setw(85)<<""<<"Nhap phim bat ki de tiep tuc chuong trinh!";
    system("pause >nul");
}
```

Hình 2.2.7.9.4 Hình ảnh minh họa phần kiểm tra yêu cầu của người dùng và in toàn bộ giá trị được lưu của dữ liệu được yêu cầu

2.2.8 Class DateNode

Đây là lớp đại diện cho dữ liệu của 1 ngày, lớp nãy sẽ quản lí dữ liệu của tất cả các khu vực hiện đang có. Lớp này có 2 phần tử:

- Giá trị ngày của bộ dữ liệu (date: Date)
- Danh sách các khu vực quản lý (Area: vector<Sector*>)



Hình 2.2.8.1 Thông tin cơ bản của class DateNode

Và có 13 hàm con:

- Hàm Seclector() là hàm cho phép lựa chọn khu vực mình muốn thực hiện các thao tác in dữ liệu hoặc xóa khu vực quản lý.

```
int Selector(){
    int tmp;
    for(int i=0;i<Area.size();i++){
        cout<<"["<<i<<"]: Khu "<<i+1<<endl;
    }
    cout<<"["<<Area.size()<<"]: Huy thao tac"<<endl;
    cout<<"Nhap lua chon cua ban: ";cin>>tmp;
    return tmp;
}
```

Hình 2.2.8.2 Hình ảnh chi tiết hàm Selector

- Hàm DeleteSector(Sector *&) là hàm hủy dữ liệu của một khu vực được chọn.

```
void DeleteSector(Sector *&tmp){
    tmp->DeleteData();
    delete tmp;
    tmp = NULL;
}
```

Hình 2.2.8.3 Hình ảnh chi tiết hàm DeleteSector

- Hàm khởi tạo Init(int,time_t) là hàm tạo ra và quản lý dữ liệu các khu vực cho ngày mới dựa vào số lượng khu vực hiện tại đang quản lý.

```
void Init(int count, time_t now){
    this->date.GetToday(now);
    for(int i=0;i<count;i++){
        Sector *Stmp = CreateSector();
        this->Area.push_back(Stmp);
    }
}
```

Hình 2.2.8.4 Hình ảnh chi tiết hàm Init

- Hàm UpdateData(time_now) là hàm cập nhật dữ liệu cho tất cả các khu vực hiện đang quản lý trong ngày đó.

```
void UpdateData(time_t now){
    for(int i=0;i<Area.size();i++){
        this->Area[i]->UpdateData(Random(0.1,39.9),Random(0.1,49.9),Random(0.1,13.9),Random(0.1,59.9),Random(0.1,49.9),Random(0.001,0.0399),Random(1,100), now);
    }
}
```

Hình 2.2.8.5 Hình ảnh chi tiết hàm UpdateDate

- Các hàm ShowTemp(Date), ShowHumi(Date), ShowPH(Date), ShowCO2(Date), ShowLumen(Date) là các hàm cho phép lựa chọn khu vực và in ra màn hình thông tin dữ liệu tương ứng của khu vực đó. Các hàm này

có cấu trúc tương tự nhau, chỉ khác duy nhất ở lệnh gọi hàm xuất dữ liệu của class Sector.

```
void ShowTemp(Date time){
    system("CLS");
    cout<<"Ban muon xem nhiet do cua khu vuc nao:"<<endl;
    int tmp = Selector();
    if(tmp==Area.size())
        return;
    Sector *Stmp;
    Stmp = this->Area[tmp];
    Stmp->ShowTemp(tmp+1, time);
}
```

Hình 2.2.8.5 Hình ảnh chi tiết hàm ShowTemp

- Hàm IncludeSector(int) là hàm thêm một số lượng khu vực vào ngày hiện tại dựa vào đối số truyền vào.

```
void IncludeSector(int tmp){
    for(int i=0;i<tmp;i++){
        Sector *Stmp = CreateSector();
        this->Area.push_back(Stmp);
    }
}
```

Hình 2.2.8.6 Hình ảnh chi tiết hàm IncludeSector

- Hàm RemoveSector() là hàm cho phép lựa chọn và xóa dữ liệu của các khu vực hiện đang quản lý.

```
int RemoveSector() {
    int n = 0, tmp = 0 ;
    while (tmp < Area.size()) {
        system("CLS");
        if (Area.size() != 0){
            cout << " Lua chon khu vuc ban muon xoa:" << endl;
            tmp = Selector();
            if (tmp >= Area.size())
                break;
            n++;
            DeleteSector(this->Area[tmp]);
            Area.erase(Area.begin() + tmp);
        }
        else {
            tmp = 1;
            cout << " Hien khong con khu vuc quan ly nao!" << endl;
            cout << " Nhap phim bat ki de tiep tuc chuong trinh!";
            system("pause >nul");
        }
    }
    return n;
}
```

Hình 2.2.8.7 Hình ảnh chi tiết hàm RemoveSector

- Hàm DeleteData() là hàm xóa tất cả dữ liệu của ngày đó.

```
void DeleteData(){
    for(int i=0;i<Area.size();i++){
        DeleteSector(this->Area[i]);
    }
    Area.clear();
}
```

Hình 2.2.8.8 Hình ảnh chi tiết hàm DeleteData

- Hàm ShowFullData() là hàm cho phép lựa chọn khu vực muốn in dữ liệu hàng loạt và gọi đến hàm in tương ứng của khu vực đó.

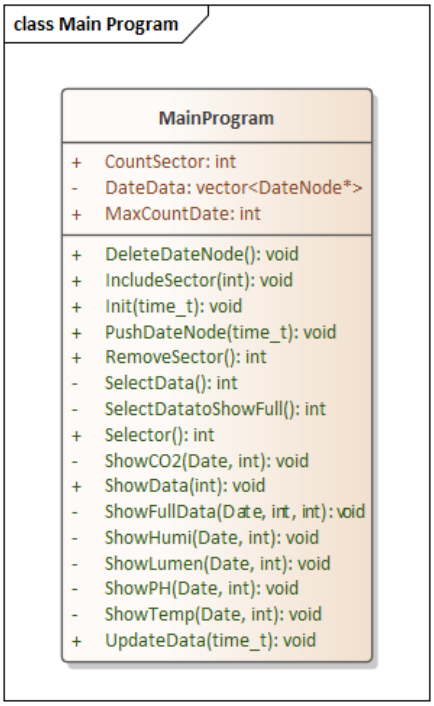
```
void ShowFullData(Date time,int select){
    system("CLS");
    cout<<"Ban muon xem du lieu cua khu vuc nao:"<<endl;
    int tmp = Selector();
    if(tmp==Area.size())
        return;
    Sector *Stmp;
    Stmp = this->Area[tmp];
    Stmp->ShowFullData(tmp+1,time,select);
}
```

Hình 2.2.8.9 Hình ảnh chi tiết hàm ShowFullData

2.2.9 Class MainProgram

Đây là lớp đại diện cho cấu trúc chính của dự án này, lớp này sẽ quản lý tất cả các dữ liệu của từng ngày, mọi thao tác muốn thực hiện lên trên các dữ liệu đều sẽ do lớp này quản lý. Lớp này có 3 phần tử:

- Số lượng khu vực hiện đang quản lý (CountSector: int)
- Thời gian tối đa lưu các dữ liệu (MaxCountDate: int)
- Danh sách các ngày đã lưu (DateData: vector<DateNode*>)



Hình 2.2.9.1 Thông tin cơ bản của class Main Program

Và lớp này có 16 hàm con:

- Hàm SelectData() là hàm cho phép lựa chọn loại dữ liệu sẽ in ra màn hình.

```
int SelectData(){
    system("CLS");
    int tmp;
    cout<<"Ban muon xem du lieu nao:"<<endl;
    cout<<"\t[0] Nhiet do."<<endl;
    cout<<"\t[1] Do am."<<endl;
    cout<<"\t[2] Do pH."<<endl;
    cout<<"\t[3] Nong do CO2 trong khong khi."<<endl;
    cout<<"\t[4] Quang thong(lumen)."<<endl;
    cout<<"\t[5] Kiem tra hang loat."<<endl;
    cout<<"\t[6] Huy thao tac."<<endl;
    cout<<"Hay nhap lua chon cua ban: ";
    cin>>tmp;
    return tmp;
}
```

Hình 2.2.9.2 Hình ảnh chi tiết hàm SelectData

- Các hàm ShowTemp(Date, int), ShowHumi(Date, int), ShowPH(Date, int), ShowCO2(Date, int), ShowLumen(Date, int) là các hàm có cấu trúc tương tự nhau và thực hiện in ra màn hình dữ liệu tương ứng của một ngày, tham số truyền vào các hàm này sẽ quyết định in dữ liệu của ngày nào trong danh sách.

```
void ShowTemp(Date time, int tmp = 0) {
    if (tmp == DateData.size())
        return;
    DateData[tmp]->ShowTemp(time);
}
```

Hình 2.2.9.3 Hình ảnh chi tiết hàm ShowTemp

- Hàm Init(time_t) là hàm khởi tạo cho lớp này, cho phép người dùng nhập số lượng khu vực muốn quản lý và số ngày tối đa muốn lưu dữ liệu sau đó sẽ

```
void Init(time_t now){
    cout<<"Vui long nhap so khu vuc ban can quan ly: ";cin>>CountSector;
    cout<<"Vui long nhap so ngay toi da can luu du lieu: ";cin>>MaxCountDate;
    DateData.push_back(CreateDateNode(CountSector, now));
}
```

khởi tạo cấu trúc lưu trữ dữ liệu của ngày đầu tiên.

Hình 2.2.9.4 Hình ảnh chi tiết hàm Init

- Hàm DeleteDateNode() là hàm dùng để xóa dữ liệu của ngày cũ nhất trong danh sách các ngày.

```
void DeleteDateNode(){
    DateNode *tmp = DateData[DateData.size()-1];
    tmp->DeleteData();
    delete tmp;
    tmp = NULL;
    DateData.pop_back();
}
```

Hình 2.2.9.5 Hình ảnh chi tiết hàm DeleteDateNode

- Hàm PushDateNode(time_t) là hàm tạo và thêm cấu trúc dữ liệu của ngày mới vào danh sách các ngày, hàm sẽ tự động kiểm tra số lượng ngày đã lưu để xác định đã đến giới hạn lưu trữ các ngày chưa và tự động xóa dữ liệu ngày cũ nhất khi đã đến giới hạn MaxCountDate.

```
void PushDateNode(time_t now){
    if(DateData.size()>=MaxCountDate){
        DeleteDateNode();
    }
    DateData.insert(DateData.begin(), CreateDateNode(CountSector, now));
}
```

Hình 2.2.9.6 Hình ảnh chi tiết hàm PushDateNode

- Hai hàm IncludeSector(int) và RemoveSector() là 2 hàm dùng để thêm hoặc xóa khu vực quản lý.

```
void IncludeSector(int tmp){
    DateData[0]->IncludeSector(tmp);
}

int RemoveSector() {
    return DateData[0]->RemoveSector();
}
```

Hình 2.2.9.7 Hình ảnh chi tiết hàm IncludeSector và RemoveSector

- Hàm Selector() là hàm cho phép lựa chọn những ngày cũ trong danh sách các ngày để in dữ liệu ra màn hình.

```
int Selector(){
    system("CLS");
    int tmp;
    for(int i=1;i<DateData.size();i++){
        cout<<"["<<i<<"]: Ngay "<<DateData[i]->date<<endl;
    }
    cout<<"["<<DateData.size()<<"]: Huy thao tac"<<endl;
    cout<<"Hay nhap lua chon cua ban: ";cin>>tmp;
    return tmp;
}
```

Hình 2.2.9.8 Hình ảnh chi tiết hàm Seclector

- Hàm UpdateData(time_t) là hàm cập nhật tất cả dữ liệu môi trường cho ngày hiện tại.

```
void UpdateData(time_t now){
    DateData[0]->UpdateData(now);
}
```

Hình 2.2.9.9 Hình ảnh chi tiết hàm UpdateData

- Hàm SelectDatatoShowFull() là hàm cho phép lựa chọn các thông tin muốn in ra đồng loạt.

```
int SelectDatatoShowFull(){
    system("CLS");
    string tmp;
    cout<<"Ban muon xem du lieu nao:"<<endl;
    cout<<"\t[0] Nhiet do cua Dat."<<endl;
    cout<<"\t[1] Do am cua Dat."<<endl;
    cout<<"\t[2] Do pH cua Dat."<<endl;
    cout<<"\t[3] Nhiet do cua Khong khi."<<endl;
    cout<<"\t[4] Do am cua khong khi."<<endl;
    cout<<"\t[5] Nong do CO2 trong khong khi."<<endl;
    cout<<"\t[6] Quang thong(lumen)."<<endl;
    cout<<"Hay nhap cac lua chon cua ban(VD: 0423): ";
    cin>>tmp;
    tmp = '1' + tmp;
    return atoi(tmp.c_str());
}
```

Hình 2.2.9.10 Hình ảnh chi tiết hàm SelectDatatoShowFull

- Hàm ShowData(int) là hàm cho phép lựa chọn và in ra màn hình loại dữ liệu muốn kiểm tra của ngày được chọn, ngày được chọn sẽ phụ thuộc vào đối số được truyền vào hàm, tham số mặc định cho hàm này là bằng 0, lúc này hàm sẽ chọn dữ liệu của ngày hiện tại.

```
void ShowData(int tmp = 0){
    if(tmp==DateData.size())
        return;
    int data = SelectData();
    switch(data){
        case(0):
        {
            ShowTemp(DateData[tmp]->date, tmp);
            break;
        }
        case(1):
        {
            ShowHumi(DateData[tmp]->date, tmp);
            break;
        }
        case(2):
        {
            ShowPH(DateData[tmp]->date, tmp);
            break;
        }
        case(3):
        {
            ShowCO2(DateData[tmp]->date, tmp);
            break;
        }
        case(4):
        {
            ShowLumen(DateData[tmp]->date, tmp);
            break;
        }
        case(5):
        {
            ShowFullData(DateData[tmp]->date, SelectDatatoShowFull(), tmp);
            break;
        }
    }
}
```

Hình 2.2.9.11 Hình ảnh chi tiết hàm ShowData

2.2.10 Một số hàm được định nghĩa để hỗ trợ cho hoạt động của chương trình

Hàm CheckDay là hàm trả về giá trị số giây cần có để giá trị thời gian tăng lên đúng số giây của 0h ngày hôm sau.

```
int CheckDay(time_t time) {
    Time a;
    a.Now(time);
    Time b(23, 59, 59);
    int c = b - a + 1;
    return c;
}
```

Hình 2.2.10.1 Hình ảnh chi tiết hàm CheckDay

Hàm NextDay(time_t &) là hàm tăng giá trị của tham số đầu vào đến đúng số giây của 0h ngày hôm sau.

```
void NextDay(time_t& time) {
    int a = CheckDay(time);
    time = time + a;
}
```

Hình 2.2.10.2 Hình ảnh chi tiết hàm NextDay

Hàm *CreateNode(float, time_t) là hàm tạo ra một nút con trỏ có kiểu BNode dùng để lưu 1 giá trị của dữ liệu từ môi trường gởi về và thời gian nhận dữ liệu, hàm này trả về dữ liệu con trỏ kiểu BNode.

```
BNode *CreateNode(float data, time_t now){
    BNode *node = new BNode;
    node->value = data;
    node->time.Now(now);
    node->pLeft = node->pRight = NULL;
    return node;
}
```

Hình 2.2.10.3 Hình ảnh chi tiết hàm CreateNode

Hàm *CreateSector() là hàm tạo ra một biến con trỏ có kiểu Sector dùng để quản lý các dữ liệu của một khu vực, hàm này trả về dữ liệu con trỏ kiểu Sector.

```
Sector *CreateSector(){
    Sector *sector = new Sector;
    return sector;
}
```

Hình 2.2.10.4 Hình ảnh chi tiết hàm CreateSector

Hàm Random(Type,Type) là hàm dược định nghĩa theo khuôn mẫu(template), hàm sẽ trả về giá trị random giữa 2 tham số, kiểu dữ liệu trả về sẽ phụ thuộc vào kiểu dữ liệu của 2 đối số đưa vào hàm.

```
template<class Type>
Type Random(Type HI, Type LO){
    Type r = LO + static_cast <Type> (rand()) /( static_cast <Type> (RAND_MAX/(HI-LO)));
    return round(r*1000)/1000;
}
```

Hình 2.2.10.5 Hình ảnh chi tiết hàm Random

Hàm *CreateDateNode(int, time_t) là hàm tạo ra một biến con trỏ kiểu DateNode lưu dữ liệu của 1 ngày, dùng để quản lý các khu vực có trong ngày đó.

```
DateNode *CreateDateNode(int count, time_t now){
    DateNode *daynode = new DateNode;
    daynode->Init(count,now);
    return daynode;
}
```

Hình 2.2.10.6 Hình ảnh chi tiết hàm CreateDateNode

Hàm getFileContents(ifstream&) là hàm dùng để đọc dữ liệu của một file, hàm này sẽ lưu các dòng dữ liệu trong file vào chung 1 chuỗi kiểu string và trả về chuỗi đó, cụ thể hàm này dùng để đọc file logo của nhóm.

```
string getFileContents (std::ifstream& File){
    string Lines = "";
    if(File){
        while (File.good ()){
            string TempLine;
            getline (File , TempLine);
            TempLine += "\n";
            for(int i=0;i<73;i++){
                TempLine += " ";
            }
            Lines += TempLine;
        }
        return Lines;
    }
}
```

Hình 2.2.10.7 Hình ảnh chi tiết hàm getFileContents

Hàm LogoStartup() là hàm in giao diện khởi động của chương trình.

```
void LogoStartup(){
    ::SendMessage(::GetConsoleWindow(), WM_SYSKEYDOWN, VK_RETURN, 0x20000000);
    cout<<setw(80)<<""<<"CHAO MUNG DEN VOI CHUONG TRINH QUAN LY NHA KINH 4 TY"<<endl;
    ifstream Reader ("Logo");
    string Art = getFileContents (Reader);
    cout<< Art << std::endl;
    Reader.close ();
    cout<<setw(87)<<""<<"Nhap phim bat ki de vao chuong trinh!!!";
    system("pause >nul");
    system("CLS");
}
```

Hình 2.2.10.8 Hình ảnh chi tiết hàm LogoStartup

Hàm MainScreen(time_t) là hàm in giao diện chính của chương trình, giao diện này có đầy đủ thông tin của chương trình và các thao tác có thể thực hiện.

```
int MainScreen(time_t time){
    system("CLS");
    Date SimulationDate;
    SimulationDate.GetToday(time);
    Time SimulationTime;
    SimulationTime.Now(time);
    cout<<" Thoi gian khoi dong chuong trinh: "<<StartupTime<<"_"<<StartupDate<<endl<<endl;
    cout<<setw(71)<<""<<"*************************************************************************"<<endl; //69 char
    cout<<setw(71)<<""<<"*             _CHUONG TRINH QUAN LY HE THONG NHA KINH_              *"<<endl;
    cout<<setw(71)<<""<<"* Lap trinh boi Nhom 12:                                            *"<<endl;
    cout<<setw(71)<<""<<"*     ::.  :                                                        *"<<endl;
    cout<<setw(71)<<""<<"*    .--- -= ..          Ho va ten          Ma so sinh vien       *"<<endl;
    cout<<setw(71)<<""<<"*     ---. #- ---::..                                              *"<<endl;
    cout<<setw(71)<<""<<"*   :--: +*= ..:-----.    Tran Toan Thang     20139015           *"<<endl;
    cout<<setw(71)<<""<<"*  .--- .*##*=-. ::::                                            *"<<endl;
    cout<<setw(71)<<""<<"*  :::.. .-+**. :..      Dinh Thanh Tung     20139096           *"<<endl;
    cout<<setw(71)<<""<<"*  ....:... :*: :...                                             *"<<endl;
    cout<<setw(71)<<""<<"*      ...: :+ ....      Mai Van Anh         20139059           *"<<endl;
    cout<<setw(71)<<""<<"*          :: :.:                                                *"<<endl;
    cout<<setw(71)<<""<<"*           : .:         Nguyen Ba Quoc Tai  20139089           *"<<endl;
    cout<<setw(71)<<""<<"*   Thoi gian gia lap: "<<SimulationTime<<" "<<SimulationDate<<"                        *"<<endl;
    cout<<setw(71)<<""<<"*   So khu vuc dang quan ly: "<<setw(3)<<left<<Main.CountSector<<"                          *"<<end
    cout<<setw(71)<<""<<"*   Thoi gian cap nhat du lieu(phut): "<<setw(3)<<left<<TimerUpdate<<"                  *"<<endl;
    cout<<setw(71)<<""<<"*************************************************************************"<<endl;
    cout<<setw(71)<<""<<"\tHay chon thao tac muon thuc hien:"<<endl;
    cout<<setw(71)<<""<<"\t[0] Gia lap thoi gian cap nhat du lieu."<<endl;
    cout<<setw(71)<<""<<"\t[1] Gia lap thoi gian qua ngay hom sau."<<endl;
    cout<<setw(71)<<""<<"\t[2] Kiem tra du lieu cua ngay hom nay."<<endl;
    cout<<setw(71)<<""<<"\t[3] Kiem tra du lieu cua cac ngay truoc."<<endl;
    cout<<setw(71)<<""<<"\t[4] Them khu vuc quan ly."<<endl;
    cout<<setw(71)<<""<<"\t[5] Xoa khu vuc quan ly."<<endl;
    cout<<setw(71)<<""<<"\t[6] Thoat chuong trinh."<<endl<<endl;
    cout<<setw(71)<<""<<"\tNhap thao tac muon thuc hien: ";
    int tmp;cin>>tmp;
    return tmp;
}
```

Hình 2.2.10.9 Hình ảnh chi tiết hàm MainScreen

Hàm ExitScreen(int) là hàm giao diện kết thúc chương trình, chương trình sẽ tự động tắt sau một khoảng thời gian nhất định phụ thuộc vào đối số truyền vào hàm.

```
void ExitScreen(int ExitTime) {
    for (int i = ExitTime; i >= 1; i--) {
        system("CLS");
        ifstream Reader("Logo");
        string Art = getFileContents(Reader);
        cout << Art << std::endl;
        Reader.close();
        cout << setw(87) << "" << "CAM ON BAN DA SU DUNG CHUONG TRINH!!!" << endl;
        cout << setw(83) << "" << "Chuong trinh se tu dong thoat sau " << i << " giay...!";
        Sleep(i * 500);
    }
}
```

Hình 2.2.10.10 Hình ảnh chi tiết hàm ExitScreen

Hàm compareBNode(Bnode*,Bnode*) là hàm trả về kết quả so sánh phần tử thời gian của 2 nút.

```
bool compareBNode(BNode *a, BNode *b){
    return (a->time<b->time);
}
```

Hình 2.2.10.11 Hình ảnh chi tiết hàm compareBNode

2.3 Hoạt động của chương trình chính

- Khởi tạo chương trình:

```
#include <iostream>
using namespace std;
#include <iomanip>
#include <windows.h>
#include <ctime>
#include <vector>
#include <random>
#include <math.h>
#include <fstream>
#include <bits/stdc++.h>
#include <string>
#include "MainProgram.hpp"

MainProgram Main;
Date StartupDate;
Time StartupTime;
int TimerUpdate;

#include "Interface.hpp"
```

Hình 2.3.1 Hình ảnh phần khởi tạo chương trình

1. Các file header chứa cấu trúc dữ liệu và thư viện cần thiết được thêm vào chương trình chính.
2. Biến Main có kiểu dữ liệu MainProgram được khai báo, đây là biến đại diện cho cấu trúc dữ liệu, quản lý mọi hoạt động chính của chương trình.
3. Các biến StartupDate, StartupTime, TimerUpdate được khai báo, đây là các biến hỗ trợ cho việc chạy giả lập các chức năng của chương trình.
4. File header chứa các hàm liên quan đến giao diện được thêm vào.

➢ Chương trình chính:

```
system("Color F0");
LogoStartup();
StartupDate.GetToday();
StartupTime.Now();
time_t SimulationNow = time(0);
Main.Init(SimulationNow);
cout << " Nhap thoi gian cap nhat du lieu (don vi phut): "; cin >> TimerUpdate;
int quit = 0;
int tmp;
```

Hình 2.3.2 Hình ảnh phần đầu tiên của chương trình chính

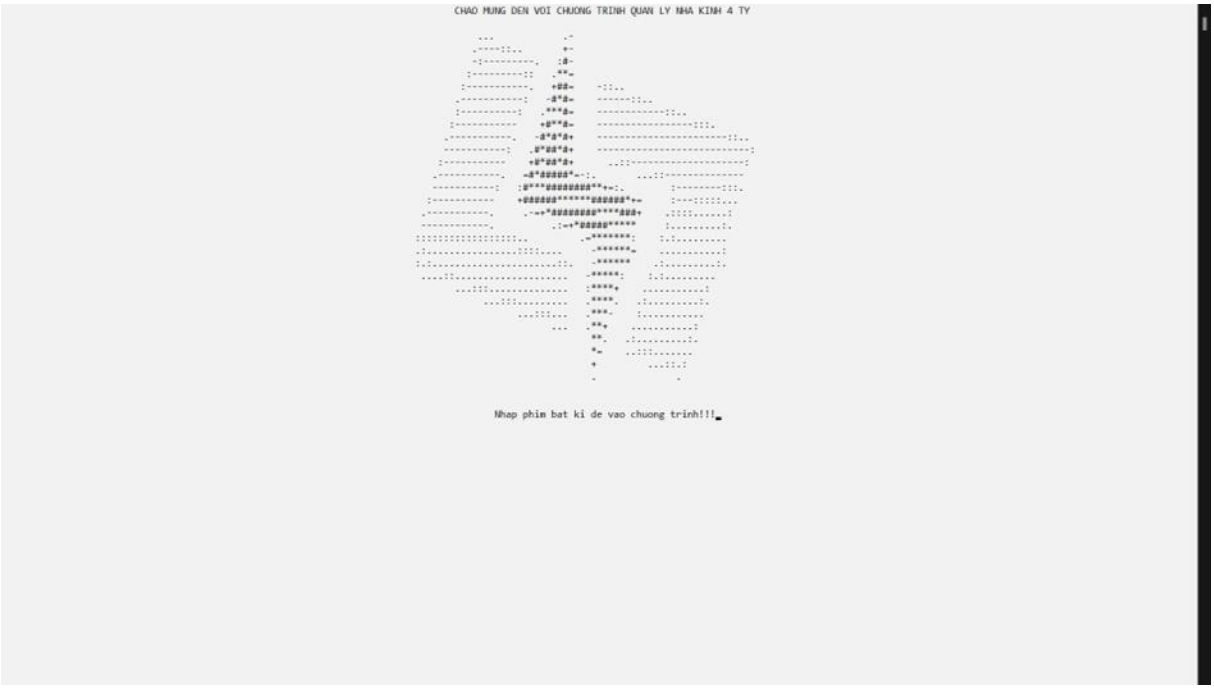
1. Gọi hàm đặt màn hình console thành chữ đen nền trắng.
2. Gọi hàm xuất ra màn hình giao diện khởi động của chương trình
3. Các biến mang giá trị thời gian khởi động chương trình(StartupDate và StartupTime) thực hiện hàm lấy dữ liệu thời gian từ hệ thống.
4. Khai báo một biến thời gian SimulationNow, biến này đại diện cho thời gian hiện tại được giả lập của chương trình, giá trị ban đầu của biến này là thời gian của hệ thống.
5. Biến Main thực hiện hàm khởi tạo dữ liệu, tạo ra bộ dữ liệu cho ngày đầu tiên.
6. Chương trình xuất ra lệnh yêu cầu nhập thời gian cập nhật dữ liệu mong muốn cho việc giả lập theo đơn vị phút.
7. Sau khi nhập xong chương trình sẽ khai báo thêm 2 biến là “quit” dùng để xác định yêu cầu thoát chương trình và “tmp” để lưu lại thao tác muốn thực hiện do người dùng nhập vào.
8. Khi đến vòng lặp while, vòng lặp sẽ kiểm tra “!quit” nếu quit được đặt thành 1 thì !quit sẽ bằng 0 và vòng lặp chương trình kết thúc, chương trình sẽ ngừng.
9. Nếu biến quit vẫn bằng 0, chương trình sẽ thực hiện hàm MainScreen() để cho người dùng lựa chọn thao tác muốn thực hiện, sau đó sẽ lưu vào biến tmp và khối switch case sẽ dựa vào giá trị của biến tmp để thực hiện lệnh tương ứng.

```
    while (!quit) {
        system("Color F0");
        tmp = MainScreen(SimulationNow);
        switch (tmp) {
        case(0):
        {
            SimulationNow += (TimerUpdate * 60);
            Main.UpdateData(SimulationNow);
            break;
        }
        case(1):
        {
            int i = SimulationNow + CheckDay(SimulationNow);
            while (SimulationNow < i ) {
                Main.UpdateData(SimulationNow);
                SimulationNow += 3600;
            }
            SimulationNow -= 86400;
            NextDay(SimulationNow);
            Main.PushDateNode(SimulationNow);
            break;
        }
        case(2):
        {
            Main.ShowData();
            break;
        }
        case(3):
        {
            Main.ShowData(Main.Selector());
            break;
        }
        case(4):
        {
            system("CLS");
            int tmp;
            cout << " Nhap so khu vuc muon them: "; cin >> tmp;
            Main.IncludeSector(tmp);
            Main.CountSector += tmp;
            break;
        }
        case(5):
        {
            Main.CountSector -= Main.RemoveSector() ;
            break;
        }
        case(6):
        {
            ExitScreen(3);
            quit = 1;
            break;
        }
        }
    }
    return 0;
}
```

Hình 2.3.3 Vòng lặp giao diện của chương trình

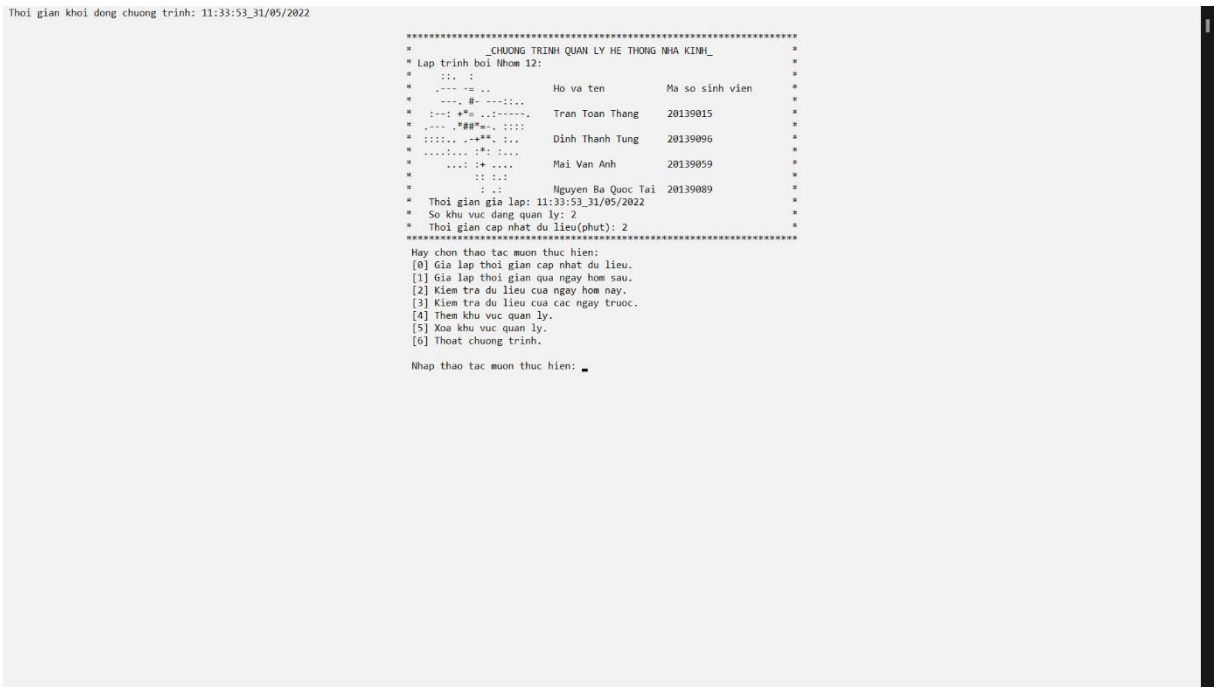
PHẦN 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ CHƯƠNG TRÌNH

1.Thực ngiệm

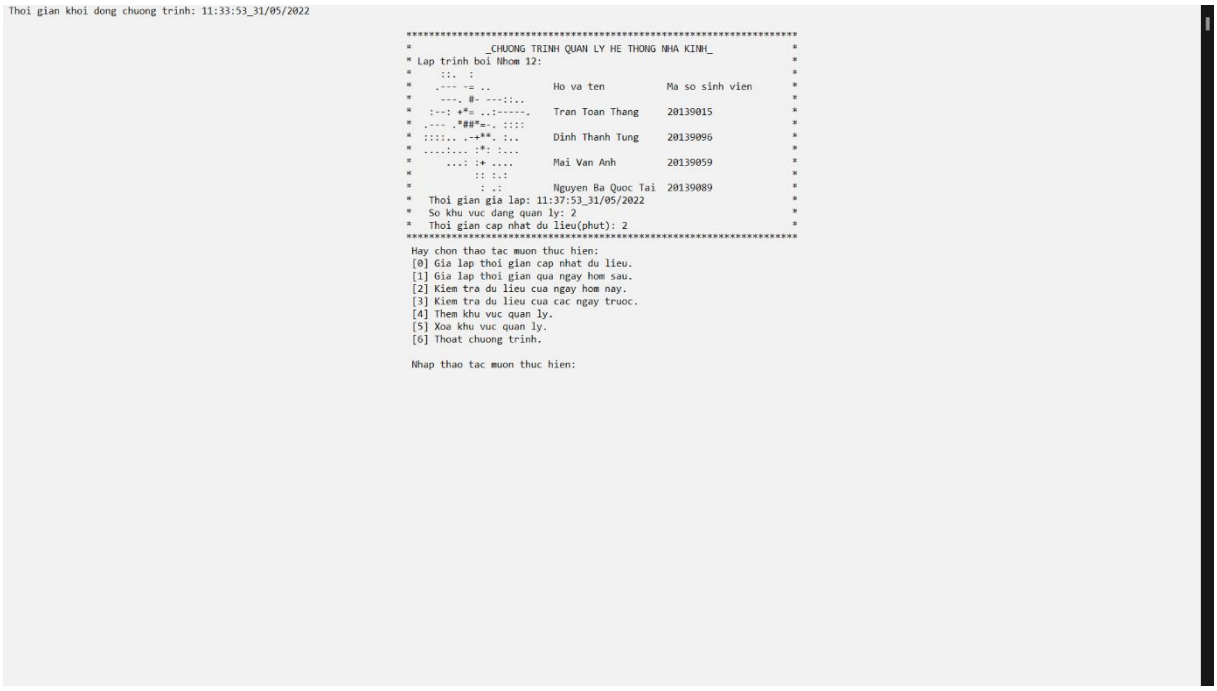


Hình 1.1 Giao diện khởi động của chương trình.

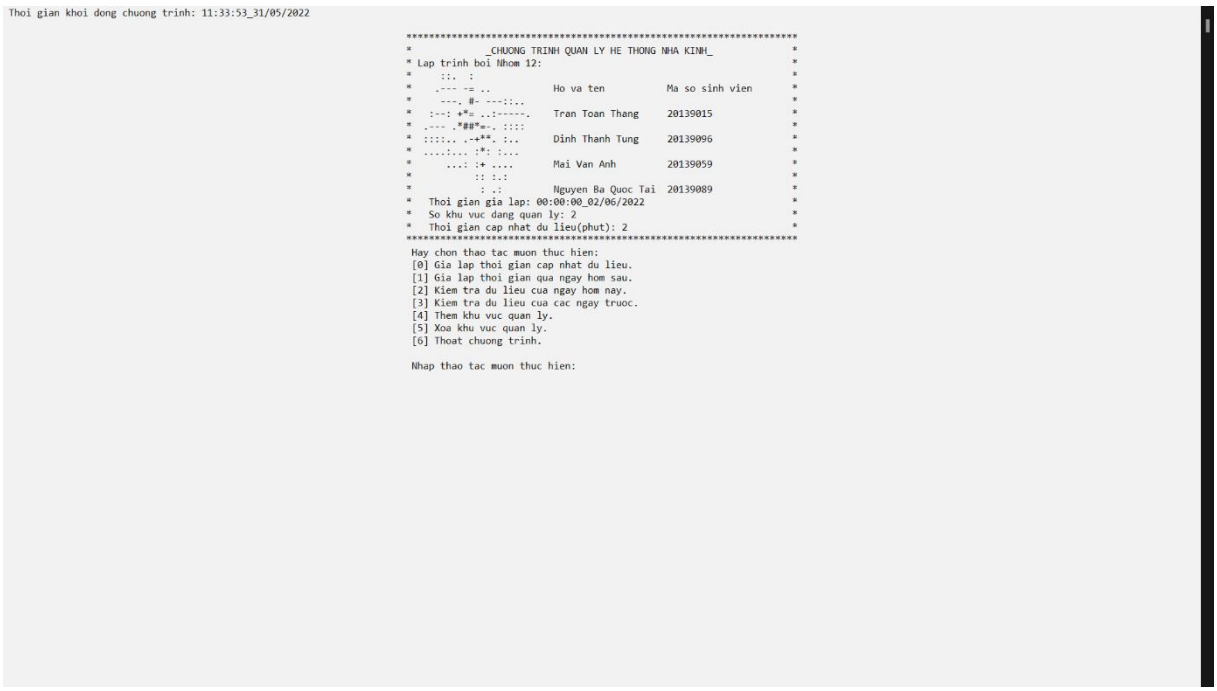
Hình 1.2 Giao diện khởi tạo các giá trị dùng để quản lý dữ liệu



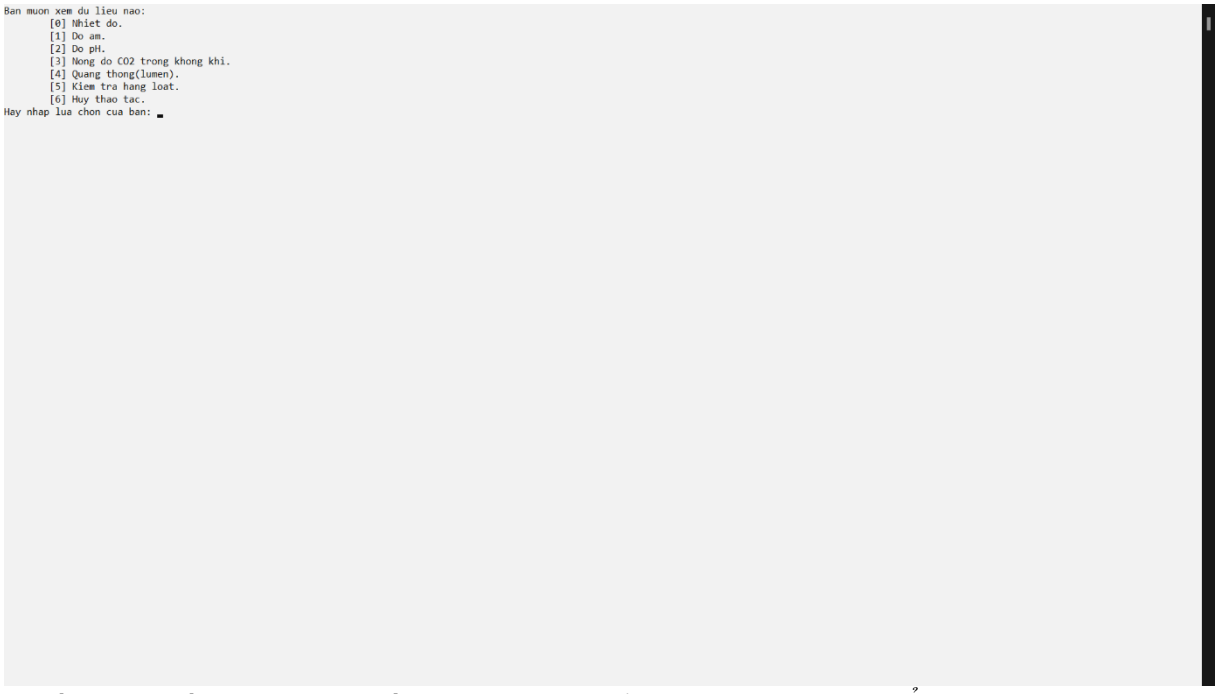
Hình 1.3 Giao diện chính của chương trình.



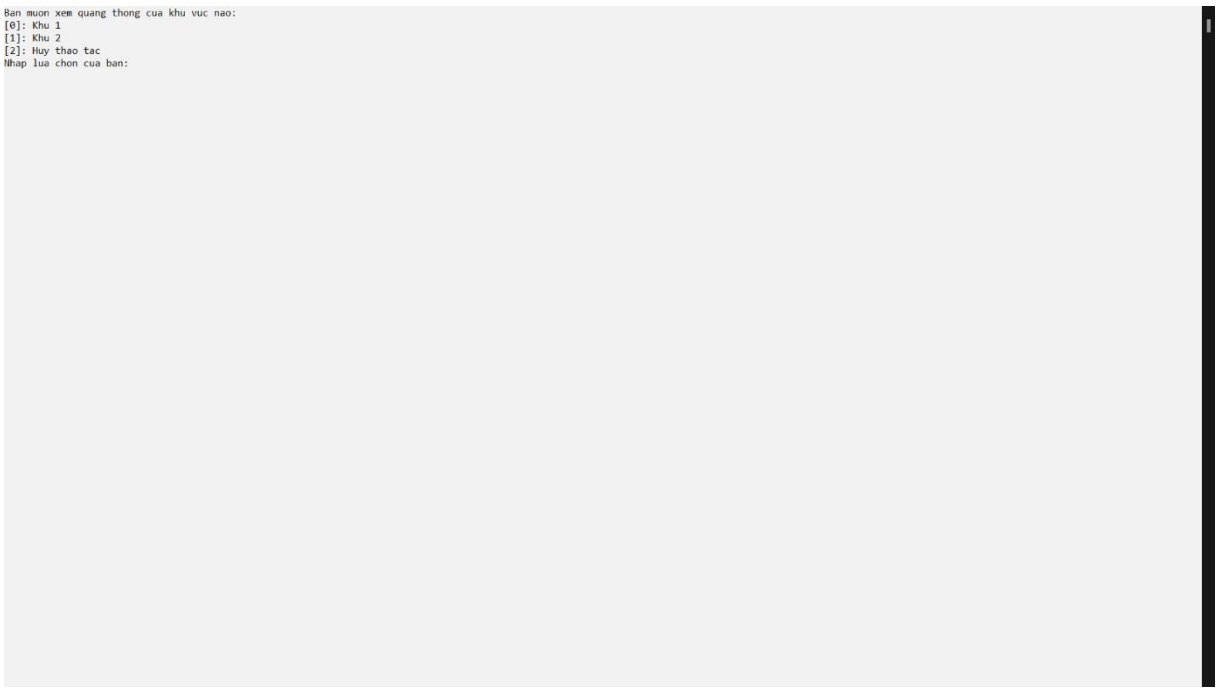
Hình 1.4 Hình ảnh thao tác giả lập đầu tiên được thực hiện 2 lần, thời gian giả lập tăng lên 4 phút.



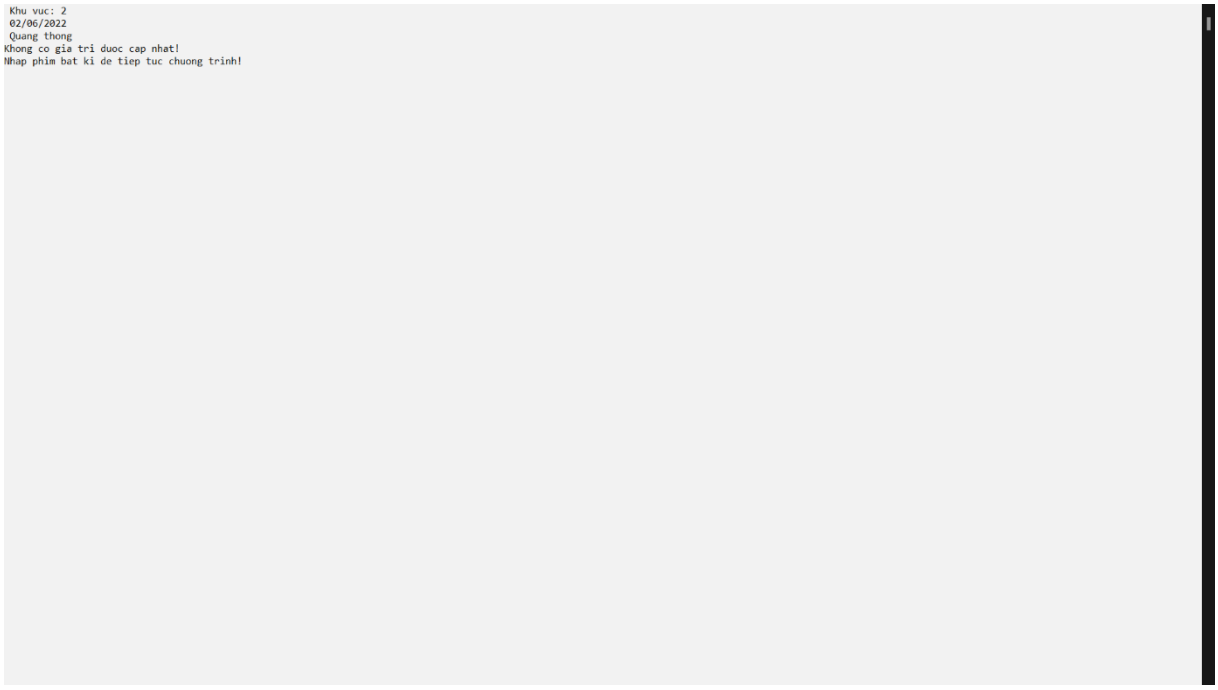
Hình 1.5 Hình ảnh thao tác giả lập thứ 2 được thực hiện 2 lần, thời gian giả lập tăng lên 2 ngày và bắt đầu từ 0h ngày thứ 2.



Hình 1.6 Hình ảnh thao tác giả lập thứ 3 được thực hiện, chuyển sang giao diện lựa chọn dữ liệu muốn kiểm tra.

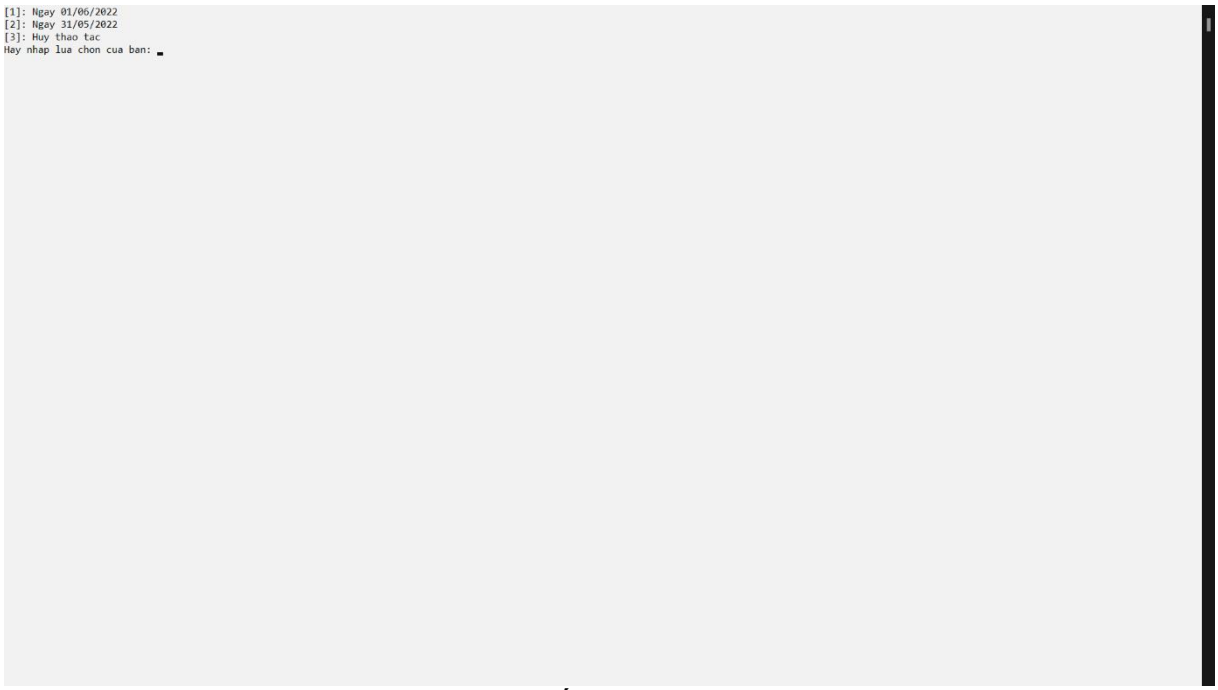


Hình 1.7 Hình ảnh sau khi lựa chọn dữ liệu quang thông, giao diện chuyển sang lựa chọn khu vực muốn kiểm tra.

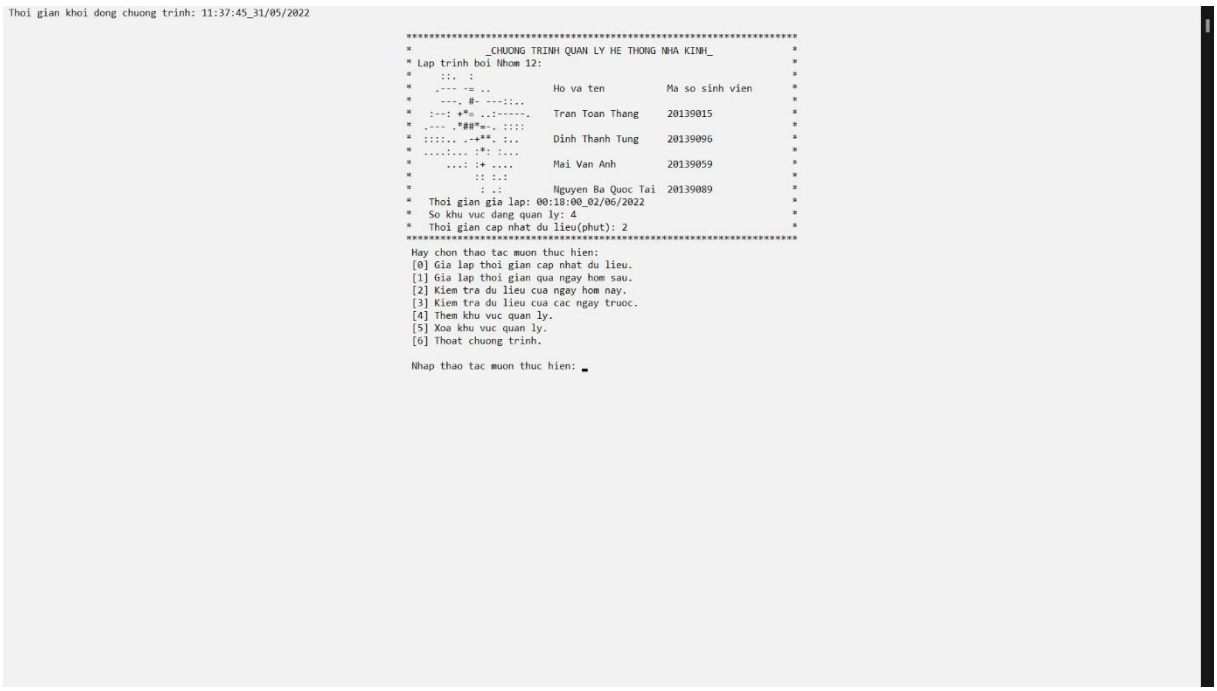


Hình 1.8 Hình ảnh sau khi lựa chọn khu vực muốn xem dữ liệu quang thông, vì vừa mới giả lập qua ngày hôm sau nên vẫn chưa có dữ liệu được cập nhật.

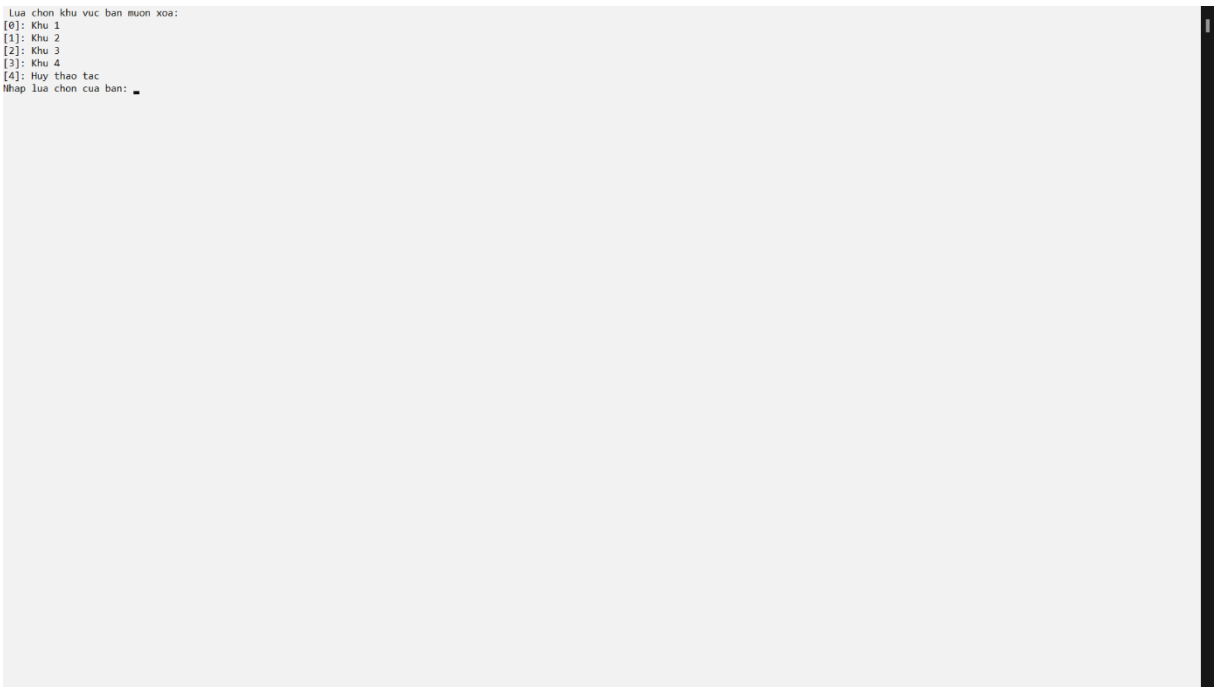
Hình 1.9 Hình ảnh dữ liệu quang thông của khu vực được chọn lúc nãy sau khi đã giả lập cập nhật dữ liệu.



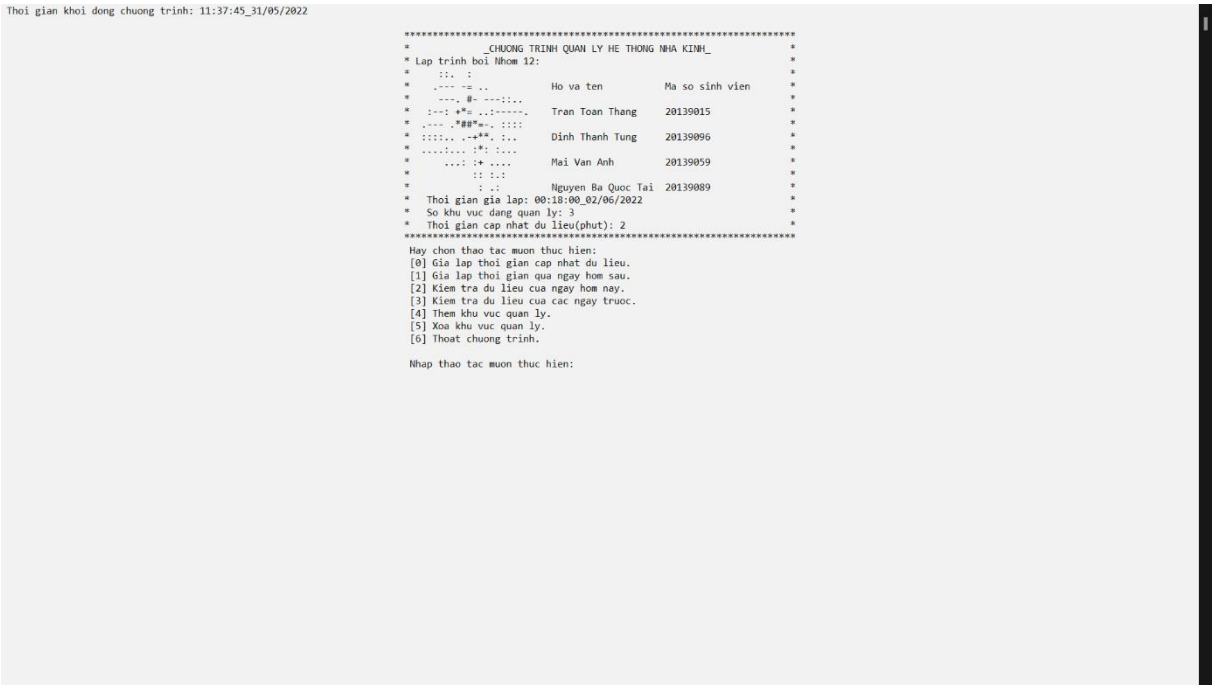
Hình 1.10 Hình ảnh giao diện khi muốn thực hiện thao tác thứ 4 của chương trình(thao tác kiểm tra dữ liệu của các ngày cũ)



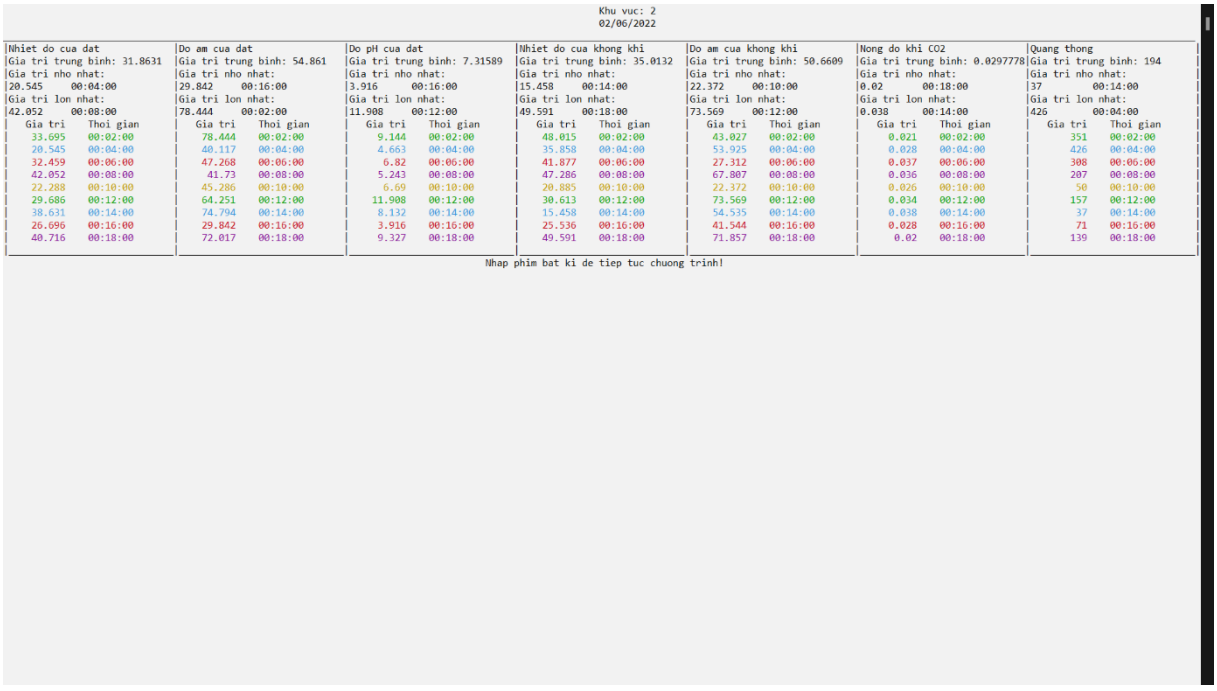
Hình 1.11 Hình ảnh giao diện chương trình chính sau khi thực hiện thao tác thứ 5 và nhập số lượng khu vực muốn thêm là 2, số lượng khu vực đang quản lý được tăng lên thành 4



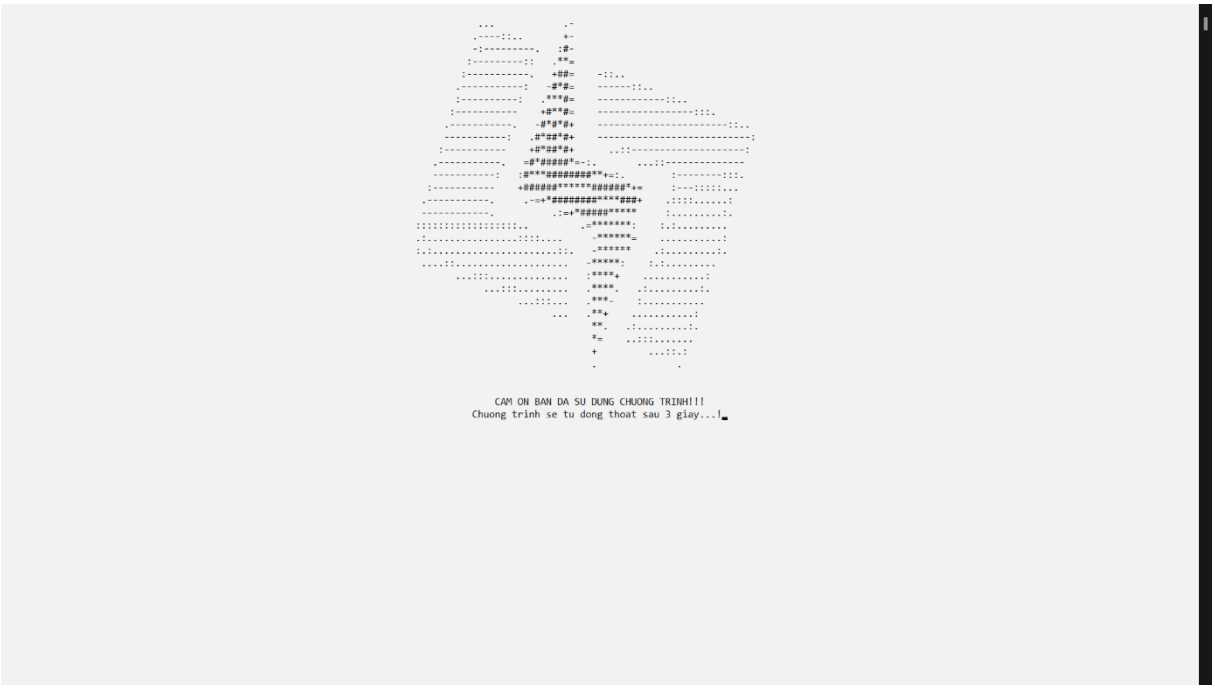
Hình 1.12 Hình ảnh giao diện lựa chọn khu vực để xóa sau khi thực hiện thao tác thứ 6 trong giao diện chính.



Hình 1.13 Hình ảnh sau khi lựa chọn khu vực để xóa, số lượng khu vực giảm từ 4 thành 3



Hình 1.14 Hình ảnh giao diện in hàng loạt các dữ liệu được lựa chọn



Hình 1.15 Hình ảnh giao diện thoát chương trình, chương trình tự động thoát sau 3 giây

2.Nhóm đánh giá chương trình

Chương trình hoạt động khá ổn định, thực hiện được các yêu cầu ban đầu đặt ra, có thể tùy chỉnh lượng dữ liệu đầu vào, có thể lưu dữ liệu với số lượng khu vực có thể nói là không giới hạn. Chương trình có giao diện trực quan, có các lệnh và hướng dẫn lựa chọn thao tác.

Tuy nhiên vẫn còn một số điểm chưa tốt cần được khắc phục: chưa có chức năng xuất dữ liệu ra file output, cần phát triển ứng dụng này hoạt động tốt trên các máy tính nhúng, vi điều khiển, cần phải tối ưu được thuật toán và bộ nhớ để có thể mở rộng với những dữ liệu lớn và thời gian thực.

Các vấn đề được giáo viên hướng dẫn góp ý:

- Cần có chức năng in hàng loạt các dữ liệu mong muốn trên cùng một màn hình(đã thực hiện được)
- Cần có chức năng xuất dữ liệu theo yêu cầu ra một file bên ngoài(chưa thực hiện được)
- Thêm chức năng cảnh báo và gợi ý hành động cần thực hiện cho các dữ liệu có giá trị nằm ngoài phạm vi an toàn.(đã thực hiện được)

PHẦN 5: KẾT LUẬN

Qua dự án báo cáo lần này, nhóm đã nghiên cứu thành công và nắm rõ các lý thuyết về cấu trúc dữ liệu, lý thuyết về hướng đối tượng, nâng cao kĩ năng giải quyết vấn đề, xử lý các lỗi phát sinh trong quá trình phát triển dự án, cải thiện kĩ năng làm việc nhóm.

Chương trình này sẽ là một nền tảng cho nhóm có thêm kinh nghiệm để phát triển các dự án liên quan đến quản lý dữ liệu hoặc dự án liên quan đến phát triển tự động hóa nông nghiệp.

Hướng phát triển cho dự án:

- Phát triển các thuật toán xử lý dữ liệu tốt hơn và nhanh hơn.
- Mở rộng thêm các loại dữ liệu môi trường khác.
- Có thể chủ động định nghĩa hoặc đặt tên cho các khu vực và các loại dữ liệu môi trường.
- Tối ưu mã chương trình cho việc chỉnh sửa và nâng cấp.

TÀI LIỆU THAM KHẢO

-Hàm getFileContents() dùng để đọc dữ liệu của file chứa bản vẽ logo bằng bảng mã ASCII: How to print ascii art in c++? - C++ Forum (cplusplus.com)

-Lệnh "::SendMessage(::GetConsoleWindow(),WM_SYSKEYDOWN,

VK_RETURN,0x20000000);" trong hàm LogoStartup dùng để đặt cửa sổ console thành toàn màn hình ngay khi khởi động: visual c++ - Running a C++ Console Program in full screen - Stack Overflow

-Lệnh "system("Color F0")" đầu hàm main của chương trình dùng để đặt màn hình console thành nền trắng chữ đen: How to print Colored text in C++ - GeeksforGeeks

- Hàm compareNode và lệnh sort trong header Sector.hpp dùng để so sánh và sắp xếp các nút giá trị theo thứ tự thời gian: std::sort() in C++ STL - GeeksforGeeks

- Dòng "HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE)" và lệnh SetConsoleTextAttribute trong các header Sector.hpp, Binary_Tree.hpp dùng để thay đổi màu sắc của các giá trị khi được in ra: colors - Colorizing text in the console with C++ - Stack Overflow

-Hàm Random để tạo ra giá trị ngẫu nhiên thay thế cho các thiết bị ngoại vi: c++ - Random float number generation - Stack Overflow