TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

----------o0o----------

Bài tập lớn môn học

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Giảng viên hướng dẫn: **TS. Hoàng Văn Thông**

Sinh viên thực hiện: **Hoàng Mạnh Khiêm**

Lớp: **CNTT VA 1 – K63**

Hà Nội, tháng 11 năm 2023

MỤC LỤC

I.BÀI 1 – BÀI SỐ 29

1. Đề Bài

Tìm hiểu về mã hóa Huffman, hàng đợi ưu tiên và thuật toán xây dựng cây Huffman. Xây dựng lớp biểu diễn cây Huffman có các chức năng sau:

a. Cài đặt thuật toán xây dựng cây Huffman sử dụng hàng đợi ưu tiên.

b. Duyệt cây và gán các từ mã cho các ký tự.

c. Xây dựng ứng dụng nén chuỗi ký tự bằng thuật toán nén tĩnh và giải nén chuỗi ký tự đã được nén.

2. Phân tích bài toán

a. Yêu cầu

- Tìm hiểu mã hóa Huffman
- Tìm hiểu hàng đợi ưu tiên
- Thuật toán xây dựng cây Huffman

b. Tìm hiểu

b.1 Tìm hiểu mã hóa huffman

mã hóa Huffman là một thuật toán mã hóa dùng để nén dữ liệu. Nó dựa trên bảng tần suất xuất hiện các ký tự cần mã hóa để xây dựng một bộ mã nhị phân cho các ký tự đó sao cho dung lượng (số bit) sau khi mã hóa là nhỏ nhất.

b.2 Tìm hiểu hàng đợi ưu tiên

Priority Queue là một cấu trúc dữ liệu mà các phần tử được quản lý sẽ có "độ ưu tiên" khác nhau gắn với từng phần tử. Phần tử có thứ tự ưu tiên cao hơn trong Priority Queue sẽ được xếp lên trước và truy vấn trước.

Đơn giản hơn, Priority Queue là một cấu trúc cho phép nó tự động sắp xếp các phần tử của nó.

Một Priority Queue sẽ có các chức năng cơ bản sau:

- Thêm một phần tử vào tập quản lý
- Xóa bỏ phần tử có ưu tiên cao nhất
- Lấy ra phần tử có ưu tiên cao nhất

b.3 Thuật toán xây dựng Huffman

Ta sẽ lưu trữ cấu trúc của cây mã Huffman vào một mảng. Cây Huffman gồm n lá mỗi lá chứa chỉ số của chữ cái tương ứng. Mỗi lần ghép 2 cây thành một ta phải thêm một đỉnh, như vậy cây biểu diễn mã Huffman gồm 2.n-1 đỉnh. Ta ký hiệu cây này là Huff[1..2n-1]. Vì mỗi gốc mới bổ sung đều có trọng số nên ta mở rộng mảng W[1..n] các trọng số thành mảng W' [1..2n-1]. Gọi m là số đỉnh của cây sẽ xây dựng. lúc đầu ta có n lá, đỉnh bổ sung lần đầu sẽ là n+1, lần thứ 2 là n+2,... Khi lấy ra hai ký tự có tần số nhỏ nhất chẳng hạn ký tự thứ i làm con trái và ký tự thứ j làm con phải của đỉnh mới bổ sung có chỉ số m ta đặt Huff[i]=-m, Huff[j]=m.

c. Xác định các lớp, các thuộc tính, phương thức của các lớp

- Xây dựng cây Huffman sử dụng lớp Huffman được xây dựng và lưu trữ dưới dạng một file Header C++.

+ Các thuộc tính của cây Huffman bao gồm:
  Node: Cấu trúc node cho cây Huffman
  Char c: kí tự mà tại một node của cây đang chứa
  Int freq: Tần số của kí tự xuất hiện trong chuỗi cần giải mã, mã hóa
  Node *left, Node *right: Node trái, Node phải của cây
+ Các phương thức của cây Huffman bao gồm:
  Node *newNode(char c, int freq, Node *left, Node *right)
  void encode(Node *root, string str, map<char, string> &huffmanCode)
  void decode(Node *root, int &index, string str)
  void buildHuffmanTree_encode(string text)
  void buildHuffmanTree_decode(map<char, int> freq, string str)

d.Chức năng của các phương thức có trong cây Huffman

+ Node *newNode(char c, int freq, Node *left, Node *right): Tạo một node mới cho cây.
+ void encode(Node *root, string str, map<char, string> &huffmanCode): Hàm đệ quy để gán các các kí tự vào một node của cây, gán thêm giá trị "0" hoặc "1" cho node trái và node phải.
+ void decode(Node *root, int &index, string str): Duyệt đoạn code đã được mã hóa, đồng thời kiểm tra khi đến lá thì in ra được kí tự nằm ở nút lá cần được xác định.
+ void buildHuffmanTree_encode(string text): Xây dựng cây Huffman mã hóa code
+ void buildHuffmanTree_decode(map<char, int> freq, string str): Xây dựng cây Huffman giải mã code dựa trên tần suất xuất hiện và chuỗi đã mã hóa

3. Cài đặt các lớp và hàm main bằng C++

a. HuffmanTree.h

a.1 Khu vực lưu trữ

- Cây Huffman được xây dựng riêng tại một file Header C++ có tên là: **HuffmanTree.h**

a.2 Cài đặt phương thức:

a.2.1 Node *newNode(char c, int freq, Node *left, Node *right)

```
Node *newNode(char c, int freq, Node *left, Node *right) // tạo node mới
{
    Node *node = new Node();
    node->c = c;
    node->freq = freq;
    node->left = left;
    node->right = right;
    return node;
}
```

a.2.2 void encode(Node *root, string str, map<char, string> &huffmanCode)

```
void encode(Node *root, string str, map<char, string> &huffmanCode) // mã hóa
{
    if (root == nullptr)
        return;

    if (!root->left && !root->right)
    {
        huffmanCode[root->c] = str;
    }

    encode(root->left, str + "0", huffmanCode);
    encode(root->right, str + "1", huffmanCode);
}
```

a.2.3 void decode(Node *root, int &index, string str)

```
void decode(Node *root, int &index, string str) // giải mã
{
    if (root == nullptr)
    {
        return;
    }

    if (!root->left && !root->right)
    {
        cout << root->c;
        return;
    }

    index++;

    if (str[index] == '0')
        decode(root->left, index, str);
    else
        decode(root->right, index, str);
}
```

a.2.4 void buildHuffmanTree_encode(string text)

```
void buildHuffmanTree_encode(string text)
{
    map<char, int> freq;
    for (char c : text)
    {
        freq[c]++;
    }

    priority_queue<Node *, vector<Node *>, comp> pq;

    for (auto pair : freq)
    {
        pq.push(newNode(pair.first, pair.second, nullptr, nullptr));
    }

    while (pq.size() != 1)
    {
        Node *left = pq.top();
        pq.pop();
        Node *right = pq.top();
        pq.pop();

        int sum = left->freq + right->freq;
        pq.push(newNode('\0', sum, left, right));
    }

    Node *root = pq.top();

    map<char, string> huffmanCode;
    encode(root, "", huffmanCode);

    cout << "Huffman Codes are :\n";
    for (auto pair : huffmanCode)
    {
        cout << pair.first << " " << pair.second << '\n';
    }

    string str;
    for (char c : text)
    {
        str += huffmanCode[c];
    }
    cout << "\nEncoded string is :\n" << str << '\n';
}
```

a.2.5 void buildHuffmanTree_decode(map<char, int> freq, string str)

```
void buildHuffmanTree_decode(map<char, int> freq, string str)
{
    priority_queue<Node *, vector<Node *>, comp> pq;

    for (auto pair : freq)
    {
        pq.push(newNode(pair.first, pair.second, nullptr, nullptr));
    }

    while (pq.size() != 1)
    {
        Node *left = pq.top();
        pq.pop();
        Node *right = pq.top();
        pq.pop();

        int sum = left->freq + right->freq;
        pq.push(newNode('\0', sum, left, right));
    }
    cout << "Huffman Codes are :\n";
    for (auto pair : freq)
    {
        cout << pair.first << " " << pair.second << '\n';
    }
    Node *root = pq.top();
    int index = -1;
    cout << "\nDecoded string is: \n";
    while (index < (int)str.size() - 2)
    {
        decode(root, index, str);
    }
}
```

b. main.cpp

- Hàm main được xây dựng riêng trong một file tên main.cpp
- Sử dụng #include “HuffmanTree.h” để truy cập tới thư viện chứa cây Huffman
- Sử dụng `buildHuffmanTree_encode(text);` để chạy phương thức mã hóa của cây Huffman
- Sử dụng

```
map<char, int> text2 = { {'a',2 }, {'h', 3}, {'o', 1},
                         {'n', 2}, {'g', 1}, {'m', 2},
                         {'e', 1}, {'k', 1}, {'i', 1},
                         {' ', 2}};
buildHuffmanTree_decode(text2,"11100011010010110010111101001110011010111010101000011");
```

để giải mã Huffman

4. Phân tích thời gian chạy có trong phương thức của các lớp

a. HuffmanTree.h

*a.1 Phương thức Node *newNode(char c, int freq, Node *left, Node *right)*

- Có 4 phép gán giá trị cho node
- Có 1 phép cấp phát bộ nhớ để tạo một node mới
- Có 1 phép trả về node đã được tạo

=> Tổng số phép toán cơ sở: 6 phép toán

=> Độ phức tạp: O(6)

*a.2 Phương thức void encode(Node *root, string str, map<char, string> &huffmanCode)*

- Có 2 lệnh if tương đương với có 2 phép toán
- Phép gán giá trị: 1 phép toán
- Gọi đệ quy cho cây con trái và cây con phải: 2 phép toán

=> Tổng số phép toán cơ sở: 5 phép toán

=> Với cây Huffman có n lá thì chiều cao của cây khoảng log(n). Độ phức tạp: O(log (n))

a.3 Phương thức void decode(Node *root, int &index, string str)

- Có 2 lệnh if tương đương với có 2 phép toán

- Có một lệnh tăng giá trị index lên 1 đơn vị là 1 phép toán

- Một lệnh in ra màn hình: 1 phép toán

- Gọi đệ quy cho cây con trái và cây con phải: 2 phép toán

=> Tổng số phép toán cơ bản: 6 phép toán

- Với cây Huffman có chiều sâu là d, chuỗi đầu có độ dài n, trường hợp xấu nhất khi d = n. Khi đó ta có tổng phép toán cơ sở là: 2*n+6

=> Độ phức tạp: O(n)

a.4 Phương thức void buildHuffmanTree_encode(string text)

- Một lệnh khai báo map và gán tên biến cho cấu trúc dữ liệu map: 1 phép toán

- Duyệt chuỗi để tính tần suất xuất hiện của các kí tự trong chuỗi cần mã hóa: n phép toán

=> Độ phức tạp cho duyệt để lấy tần suất: O(n)

- Khai báo priority_queue: 1 phép toán

- Duyệt các tần suất khi đã lưu vào map và được đẩy vào priority_queue dưới dạng một node mới: 6*k*log(k) phép toán

=> Độ phức tạp cho duyệt để đẩy vào hàng đợi ưu tiên: O(k*log(k))

- Vòng lặp cài đặt các node trái và phải từ các giá trị đã được đẩy vào priority_queue : 5*k*log(k) phép toán.

=> Độ phức tạp cho duyệt hàng đợi và tạo các cây con trái phải cho từng giá trị: O(k*log(k))

- Khai báo một node cuối cùng còn lại trong priority_queue là node gốc: 1 phép toán.

- Khai báo map và gán tên biến cho map: 1 phép toán.

- Khai báo phương thức encode: 1 phép toán

=> Độ phức tạp cho phương thức encode tại đây: O(log(n))

- Vòng lặp và dòng lệnh được in ra các quy ước của mã huffman: n+1: phép toán

- Khai báo một chuỗi kí tự và duyệt vòng lặp in ra chuỗi đã mã hóa: n+2 phép toán

=> Độ phức tạp cho phần in ra giá trị mã hóa: O(n)

=> Độ phức tạp tổng hợp của phương thức này: O((n+k*log(k)+d+n+log(n))= O(n+klogk+d) với d là độ sâu của cây tại trường hợp xấu nhất

a.5 Phương thức void buildHuffmanTree_decode(map<char, int> freq, string str)

- Khai báo hàng đợi ưu tiên: 1 phép toán

- Duyệt tần suất và đưa vào hàng đợi ưu tiên dưới dạng 1 Node: k*log(k) với k là số lượng kí tự cần duyệt

- Vòng lặp xây dựng cây Huffman với hàng đợi ưu tiên: (5+log(k))*k/2 phép toán

- In ra dòng thông báo số tần suất và các tần suất: k+1 phép toán

- Gán giá trị nút gốc: 1 phép toán

- Gán giá trị index, đưa ra thông báo ngoài màn hình: 2 phép toán

- Vòng lặp giải mã chuỗi: d phép toán với d là độ sâu của chuỗi nhị phân

=> Tổng số phép toán cơ sở: (1+k*log(k) + (5+log(k))*k/2 + k + 1 + 3 +d)

=> Độ phức tạp: O(klog(k)+k+d)

b main.cpp

- Khai báo một chuỗi kí tự: 1 phép toán

- Gọi 2 hàm giải mã và mã hóa: 2 phép toán

- Khai báo map có các giá trị cho trước: 1+k phép toán

=> Tổng số phép toán cơ sở: k+4 phép toán

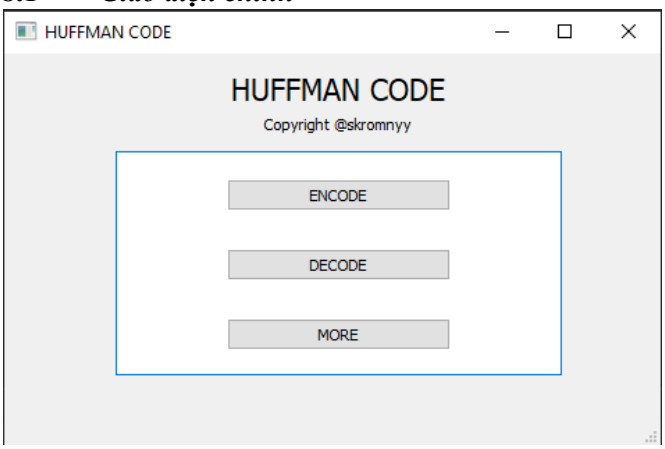
5. Ứng dụng của đề bài

a. Chạy trên console

- Thay đổi các giá trị trong hàm main và chạy code.

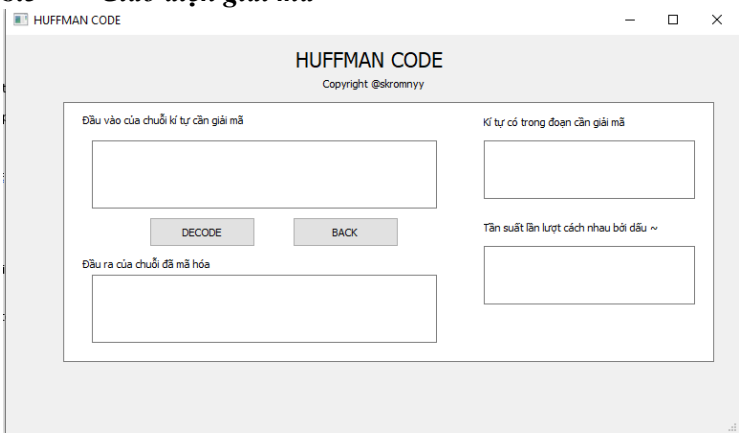
b. Phần mềm HUFFMAN code

b.1 Giao diện chính



b.2 Giao diện mã hóa

b.3 Giao diện giải mã



b.4 Phương thức hoạt động

b.4.1 Giao diện chính

- Có 2 lựa chọn: Giải mã và Mã hóa

b.4.2 Giao diện mã hóa

- Tại dòng input: Đầu vào của chuỗi kí tự cần mã hóa

- Nhập chuỗi và bấm encode: đáp án sẽ được hiển thị tại đầu ra của chuỗi mã hóa

- Ngoại lệ báo lỗi xảy ra khi người dùng không nhập chuỗi cần mã hóa

b.4.3 Giao diện giải mã

- Khu vực đầu vào chính đó chính là chuỗi bit đã mã hóa bao gồm số 0 và 1.

- Khu vực đầu vào thứ 2 “Kí tự trong đoạn cần giải mã”: bao gồm các kí tự có trong ý nghĩa của dãy bit (không cần đúng thứ tự)

-Khu vực đầu vào thứ 3 “Nhập số lần xuất hiện cách nhau bởi -~- xác định được tần suất xuất hiện của các kí tự trong dãy bit mã hóa

- Bấm DECODE và đoạn được giải mã hiển thị tại đầu ra của chuỗi giải mã

- Ngoại lệ báo lỗi xảy ra khi người dùng:

    + Nhập thiếu một trong 3 thông tin đầu vào, số lượng tần suất không khớp

    + Không thể giải mã do một trong 3 đầu vào nhập sai

6. Đường dẫn tới source code phần mềm và source code báo cáo

- Đường dẫn tới source code báo cáo**: https://github.com/hoangmanhkhiem/Great-Exercises-On-Data-structures-and-Algorithms/tree/main/Problem_29**

- Đường dẫn tới source code phần mềm**: https://github.com/hoangmanhkhiem/Great-Exercises-On-Data-structures-and-Algorithms/tree/main/app/Project29**

7. Demo chạy code và chạy phần mềm

- Đường dẫn video demo mã hóa giải mã Huffman trên console:
https://drive.google.com/file/d/1IovnLTxYC3UL4GB0PT9f5kLjvFqXjwpH/view?usp=drive_link

- Đường dẫn video demo phần mềm mã hóa và giải mã Huffman:

https://drive.google.com/file/d/1uozcE1AKGUaSnv82vEQMqt5bWscai3Vg/view?usp=drive_link

II. BÀI 2 – BÀI SỐ 13

1. Đề bài

Một công ty điện thoại cần xây dựng một chương trình tính tiền điện thoại cho các khách hàng. Hiện tại, công ty có lưu trữ 2 tệp tin, một tệp về khách hàng, một tệp về các cuộc điện thoại đã gọi của khách hàng.

1. Tệp khách hàng có tên ***khachhang.txt***. Mỗi dòng của tệp này gồm có:

Tên khách hàng; số điện thoại

- Tên của khách hàng: tên là một xâu ký tự, độ dài không vượt quá 25 và kết thúc bởi dấu ";".
- Số điện thoại của khách hàng gồm 10 chữ số bắt đầu bằng chữ số 0

Ví dụ:

Nguyen Anh Tuấn; 0987654233
Le Nhat Anh; 0967456321
…

2. Tệp lưu trữ các cuộc điện thoại có tên ***cuocgoi.txt***. Mỗi dòng có những thông tin về một cuộc điện thoại như sau:

Số điện thoại; Số phút; Thời điểm bắt đầu gọi; Ngày gọi (dd/mm/yyyy); Vùng

(Vùng: nội hạt, lân cận, xa và rất xa, được viết tắt là: NH, LC, X, RX).
Ví dụ:

0987654233; 4; 8h23; 14/010/2018; NH
0987654233; 6; 15h10; 19/10/2018; LC
0967456321; 1; 23h05; 15/10/2018; RX
….

Hãy sử dụng cấu trúc dữ liệu thích hợp viết chương trình tính tiền điện thoại theo yêu cầu sau:
- Đọc thông tin trong tệp *khachhang.txt* và *cuocgoi.txt,* tính tiền cho từng khách hàng và ghi ra tệp *ketqua.txt*, mỗi dòng có các thông tin sau:

Tên KH; số đt; Số tiền, số cuộc gọi NH, số gọi LC, số cuộc X, số cuộc gọi RX.

Cách tính tiền cho mỗi cuộc gọi như sau:

*Tiền = Giá cơ bản * Số phút * Hệ số miền.*
Giá cơ bản là 1.100 đồng 1 phút.

Hệ số miền đối với nội hạt là 1, với lân cận là 2, với xa là 3, với rất xa là 4. Đối với các cuộc gọi bắt đầu từ 23h00 đến 5h00 các ngày trong tuần và ngày Thứ Bảy, Chủ nhật thì được giảm giá 30%.

Lưu ý: trong danh sách cuộc gọi mỗi khách có thể có nhiều cuộc gọi hoặc không có cuộc gọi nào.

2. Phân tích bài toán

a. Yêu cầu bài toán

Hãy sử dụng cấu trúc dữ liệu thích hợp viết chương trình tính tiền điện thoại theo yêu cầu sau:

- Đọc thông tin trong tệp khachhang.txt và cuocgoi.txt, tính tiền cho từng khách hàng và ghi

ra tệp ketqua.txt, mỗi dòng có các thông tin sau:

Tên KH; số đt; Số tiền, số cuộc gọi NH, số gọi LC, số cuộc X, số cuộc gọi RX.

b. Phân tích

- Tại vấn đề này, em sẽ xây dựng cấu trúc dữ liệu vector để lưu trữ các dữ liệu của người dùng và cuộc gọi.

- Vì có 2 tệp đầu vào nên em sẽ sử dụng 2 lớp để xây dựng nên cấu trúc của từng đối tượng và 2 lớp để quản lí 2 danh sách đối tượng đó.

- Xây dựng thêm 1 lớp để thống kê 2 đối tượng có chung giá trị SDT và 1 lớp để quản lí danh sách đối tượng đã được kết hợp

- Xây dựng 1 lớp để tính tiền điện thoại dựa trên những thông tin đã thống kê được.

- Vì đề bài yêu cầu phải xác định Thứ 7, Chủ nhật nên phải xây dựng thêm 1 lớp để chuyển đổi những ngày tháng về thứ theo đúng quy chuẩn hiện tại.

c. Xác định các lớp, các thuộc tính, phương thức của lớp, chức năng

c.1 Vector.hpp

c.1.1 Chức năng

- Vector trong C++ giống dynamic array (mảng động) nhưng có khả năng tự động thay đổi kích thước khi một phần tử được chèn hoặc xóa tùy thuộc vào nhu cầu của tác vụ được thực thi, với việc lưu trữ của chúng sẽ được vùng chứa tự động xử lý.

c.1.2 Thuộc tính

- int capacity;
- data* cur_user_array;
- int sizeIs;

c.1.3 Phương thức

- bool operator<(dynamicArray<data>& input);
- bool operator>(dynamicArray<data>& input);
- bool operator>=(dynamicArray<data>& input);
- bool operator<=(dynamicArray<data>& input);
- bool operator==(dynamicArray<data>& input);

Chức năng: Toán tử so sánh 2 vector

- dynamicArray<data> operator+(dynamicArray<data>& input1);

Chức năng: Khai báo vector với kiểu dữ liệu data

- data& operator[](int index);

Chức năng: Truy xuất và trả về tham chiếu đến phần tử tại chỉ mục index

- data& operator->();

Chức năng: Trả về con trỏ đến đối tượng hiện tại (giả sử data là một kiểu dữ liệu có thể truy cập thông qua con trỏ).

- dynamicArray();

Chức năng: Hàm khởi tạo mặc định, tạo một đối tượng dynamicArray mới.

- ~dynamicArray();

Hàm hủy, giải phóng bộ nhớ khi đối tượng dynamicArray bị hủy.

- bool is_empty();

Chức năng: Kiểm tra xem vector có trống không. Trả về true nếu trống, ngược lại trả về false.

- void auto_resize();

Chức năng: Thay đổi kích thước của vector khi nó đầy. Có thể thực hiện bằng cách tạo một vector mới với kích thước lớn hơn và sao chép dữ liệu từ vector cũ sang vector mới.

- void clear();

Chức năng: Xóa tất cả các phần tử trong vector.

- void push_back(data push_data_to_back_vector);

Chức năng: Thêm dữ liệu vào cuối vector.

- void remove_data(data lookFor);

Chức năng: xóa các phần tử có giá trị xác định khỏi vector

- void remove(int index);

Chức năng: Xóa phần tử có chỉ mục index khỏi vector

- int get_size();

Chức năng: Trả về kích thước hiện tại của vector

- void debugAry();

Chức năng: In thông tin debug về vector, chẳng hạn như các phần tử và kích thước, để kiểm tra và giải quyết lỗi.

c.2 customer_lib.h

c.2.1 Chức năng

- Lưu trữ và quản lí các thông tin của từng khách hàng

c.2.2 Thuộc tính

- string fullName;
- string numberPhone;

c.2.3 Phương thức

- Customer();

Chức năng: Hàm khởi tạo mặc định, tạo mới đối tượng customer mới

- Customer(string fullName, string numberPhone);

Chức năng: Hàm khởi tạo với tham số.

- string get_Name();

Chức năng: Trả về tên đầy đủ của khách hàng.

- string get_numberPhone();

Chức năng: Trả về số điện thoại của khách hàng.

c.3 infoCall_lib.h

c.3.1 Chức năng

- Quản lí và lưu trữ thông tin một cuộc gọi

c.3.2 Thuộc tính

- string numberPhone;
- int minutes;
- string start_time;
- int start_hour;
- string callDate;
- string location;

c.3.3 Phương thức

- infoCall();

Chức năng: Hàm khởi tạo mặc định.

- infoCall(string numberPhone, int minutes, string start_time, int start_hour, string callDate, string location);

Chức năng: Hàm khởi tạo với tham số.

- string get_numberPhone();

Chức năng: Trả về số điện thoại của cuộc gọi.

- int get_minutes();

Chức năng: Trả về thời lượng của cuộc gọi (phút).

- string get_start_time();

Chức năng: Trả về thời gian bắt đầu cuộc gọi dưới dạng chuỗi.

- int get_start_hour();

Chức năng: Trả về giờ bắt đầu cuộc gọi.

- string get_callDate();

Chức năng: Trả về ngày cuộc gọi diễn ra.

- string get_location();

Chức năng: Trả về địa điểm của cuộc gọi.

c.4 Statistic_lib.h

c.4.1 Chức năng

- Lưu trữ và quản lý thông tin thống kê về cuộc gọi của một người dùng cụ thể, bao gồm tên đầy đủ, số điện thoại, cước phí cuộc gọi, và số lượng cuộc gọi cho từng loại (NH, LC, X, RX).

c.4.2 Thuộc tính

- string fullName;
- string numberPhone;
- int callFee;
- int callNumber_NH;
- int callNumber_LC;
- int callNumber_X;
- int callNumber_RX;

c.4.3 Phương thức

- Statistic();

Chức năng: Hàm khởi tạo mặc định.

- Statistic(string fullName, string numberPhone, int callFee, int callNumber_NH, int callNumber_LC, int callNumber_X, int callNumber_RX);

Chức năng: Hàm khởi tạo với tham số.

- string get_fullName();

Chức năng: Trả về tên đầy đủ của người dùng.

- string get_numberPhone();

Chức năng: Trả về số điện thoại của người dùng.

- int get_callFee();

Chức năng: Trả về cước phí cuộc gọi.

- int get_callNumber_NH();

Chức năng: Trả về số lượng cuộc gọi loại NH

- int get_callNumber_LC();

Chức năng: Trả về số lượng cuộc gọi loại LC

- int get_callNumber_X();

Chức năng: Trả về số lượng cuộc gọi loại X

- int get_callNumber_RX();

Chức năng: Trả về số lượng cuộc gọi loại RX

c.5 list_customer_lib.h

c.5.1 Chức năng

- thực hiện các hoạt động như đọc dữ liệu từ tệp, xuất dữ liệu, quản lí danh sách các khách hàng

c.5.2 Thuộc tính

- dynamicArray<Customer> listCustomer;

c.5.3 Phương thức

- void inputFile();

Chức năng: Đọc dữ liệu từ tệp vào danh sách khách hàng.

- dynamicArray<Customer> get_list_Customer();

Chức năng: Trả về danh sách khách hàng.

c.6 list_called_lib.h

c.6.1 Chức năng

- thực hiện các hoạt động như đọc dữ liệu từ tệp, xuất dữ liệu, quản lí danh sách các cuộc gọi

c.6.2 Thuộc tính

- dynamicArray<infoCall> callList;

c.6.3 Phương thức

- List_Called();

Chức năng: Hàm xóa danh sách cuộc gọi

- void inputFile();

Chức năng: Đọc dữ liệu từ tệp vào danh sách cuộc gọi.

- dynamicArray<infoCall> get_List_Called();

Chức năng: Trả về danh sách cuộc gọi.

c.7 list_statistic_lib.h

c.7.1 Chức năng

- Thực hiện các hoạt động thống kê tổng các cuộc gọi của các khách hàng từ 2 lớp danh sách khách hàng và danh sách cuộc gọi

c.7.2 Thuộc tính

dynamicArray<Statistic> statisticList;

c.7.3 Phương thức

- List_Statistic();

Chức năng: Hàm xóa danh sách thống kê

- void add(Statistic Listt);

Chức năng: Thêm một đối tượng Statistic vào danh sách thống kê.

- void outPut();

Chức năng: Xuất dữ liệu thống kê.

c.8 systemStatistic_lib.h

c.8.1 Chức năng

- Tính toán tổng số tiền của các khách hàng thông qua danh sách khách hàng, cuộc gọi và thống kê

c.8.2 Thuộc tính

- Customer_List _listCustomer;
- List_Called _listCalled;
- List_Statistic _listStatistic;

c.8.3 Phương thức

- systemStatistics();

Chức năng: Hàm khởi tạo mặc định để đọc các dữ liệu từ 2 tệp khách hàng, cuộc gọi qua đó thống kê lại

- void tinhTien();

Chức năng: Tính cước phí cho các cuộc gọi.

- void xuat();

Chức năng: Xuất thông tin về file kết quả

c.9 converttime.h

c.9.1 Chức năng

- Chuyển đổi thời gian về thứ mấy

c.9.2 Thuộc tính

- int total_day;
- int day;
- int month;
- int year;
- string date;
- string celandar;

c.9.3 Phương thức

- converttime();

Chức năng: Hàm khởi tạo mặc định.

- converttime(int day, int month, int year);

Chức năng: Hàm khởi tạo với tham số ngày, tháng, năm.

- converttime(string date);

Chức năng: Hàm khởi tạo với tham số ngày dưới dạng chuỗi.

- string convert_date_to_celandar(int total_day);

Chức năng: Chuyển đổi tổng số ngày thành thứ mấy.

- int get_day();

Chức năng: Trả về giá trị ngày.

- int get_month();

Chức năng: Trả về giá trị tháng.

- int get_year();

Chức năng: Trả về giá trị năm.

- int get_total_day();

Chức năng: Trả về tổng số ngày.

- string get_date();

Chức năng: Trả về ngày dưới dạng chuỗi.

- int Caculate_Day(int day,int month,int year)

Chức năng: tính tổng số ngày từ ngày cần xác định đến ngày mặc định

3. Cài đặt các lớp và hàm main bằng C++

a. Vector.hpp

a.1 Khu vực lưu trữ

- Trong file **Vector.hpp**

a.2 Cài đặt phương thức

a.2.1 dynamicArray();

```
template<typename data>
dynamicArray<data>::dynamicArray()
{
    capacity = 1; //default 1?
    cur_user_array = new data[capacity]; //this value has to modify on the heap
    sizeIs = 0;
}
```

a.2.2 ~dynamicArray();

```
template<typename data>
dynamicArray<data>::~dynamicArray()
{
    delete[] cur_user_array;
}
```

a.2.3 bool is_empty();

```
template<typename data>
bool dynamicArray<data>::is_empty()
{
    if (sizeIs == 0)
        return true;
    return false;
}
```

a.2.4 void auto_resize();

```
template<typename data>
void dynamicArray<data>::auto_resize()
{
    if (capacity == 0)
        capacity++;
    else if (capacity == sizeIs)
    {
        capacity *= 2;
        //cout << "size has been doubled" << endl;
        //cout << "capacity: " << capacity << "|sizeIs: " << sizeIs << endl;

        data* temp_arr = new data[capacity];
        for (int i = 0; i < capacity / 2; i++)
        {
            temp_arr[i] = cur_user_array[i];
        }
        delete[] cur_user_array;
        cur_user_array = temp_arr;
    }
}
```

a.2.5 void clear();

```
template<typename data>
void dynamicArray<data>::clear()
{
    sizeIs = 0;
}
```

a.2.6 void push_back(data push_data_to_back_vector);

```
template<typename data>
void dynamicArray<data>::push_back(data push_data_to_back_vector/*user_input*/)
{
    sizeIs++;
    auto_resize();
    //cout << "push_back called" << endl;
    cur_user_array[sizeIs - 1] = push_data_to_back_vector;
}
```

a.2.7 void remove_data(data lookFor);

```
template<typename data>
void dynamicArray<data>::remove_data(data lookFor)
{
    int sizeIsnew = sizeIs;
    for (int i = 0; i < sizeIs; i++)
    {
        if (cur_user_array[i] == lookFor)
        {
            for (int j = i; j < sizeIs; j++)
            {
                cur_user_array[j] = cur_user_array[j + 1];
            }
            sizeIs--;
        }
    }
    if (sizeIsnew == sizeIs) cout << "No data found" << endl;
}
```

a.2.8 void remove(int index);

```
template<typename data>
void dynamicArray<data>::remove(int index)
{
    if (!is_empty())
    {
        for (int i = index; i < sizeIs; i++)
        {
            cur_user_array[i] = cur_user_array[i + 1];
        }
        sizeIs--;
    }
}
```

a.2.9 int get_size();

```
template<typename data>
int dynamicArray<data>::get_size() //size of the array currently
{
    return sizeIs;
}
```

a.2.10 void debugAry();

```
template<typename data>
void dynamicArray<data>::debugAry()
{
    //cout << endl;
    cout << "[";
    for (int i = 0; i < sizeIs; i++)
    {
        cout << "Element " << i << ": ";
        cout << cur_user_array[i] << endl;//", ";
    }
    //cout << cur_user_array[sizeIs];
    cout << "]";
    cout << endl;
}
```

a.2.11 bool operator<(dynamicArray<data>& input);

```
template<typename data>
bool dynamicArray<data>::operator<(dynamicArray<data>& input)
{
    if (sizeIs != input.sizeIs)
        return (sizeIs < input.sizeIs);

    if (sizeIs == input.sizeIs)
    {
        for (int i = 0; i < sizeIs; i++)
        {
            if (cur_user_array[i]/*or you can use "this[i]"*/ < input[i])
                return true;
            else if (cur_user_array[i] > input[i])
                return false;
        }
        return false;
    }
}
```

a.2.12 bool operator>(dynamicArray<data>& input);

```
template<typename data>
bool dynamicArray<data>::operator>(dynamicArray<data>& input)
{
    if (sizeIs != input.sizeIs)
        return(sizeIs > input.sizeIs);

    if (sizeIs == input.sizeIs)
    {
        for (int i = 0; i < sizeIs; i++)
        {
            if (cur_user_array[i] > input[i])
                return true;
            else if (cur_user_array[i] > input[i])
                return false;
        }
        return false;
    }
}
```

a.2.13 bool operator>=(dynamicArray<data>& input);

```
template<typename data>
bool dynamicArray<data>::operator>=(dynamicArray<data>& input)
{
    if (sizeIs != input.sizeIs)
        return(sizeIs > input.sizeIs);

    if (sizeIs == input.sizeIs)
    {
        for (int i = 0; i < sizeIs; i++)
        {
            if (cur_user_array[i] > input[i])
                return true;
            else if (cur_user_array[i] > input[i])
                return false;
        }
        return false;
    }
}
```

a.2.14 bool operator<=(dynamicArray<data>& input);

```
template<typename data>
bool dynamicArray<data>::operator<=(dynamicArray<data>& input)
{
    if (sizeIs != input.sizeIs)
        return (sizeIs < input.sizeIs);

    if (sizeIs == input.sizeIs)
    {
        for (int i = 0; i < sizeIs; i++)
        {
            if (cur_user_array[i]/*or you can use "this[i]"*/ < input[i])
                return true;
            else if (cur_user_array[i] > input[i])
                return false;
        }
        return false;
    }
}
```

a.2.15 bool operator==(dynamicArray<data>& input);

```
template<typename data>
bool dynamicArray<data>::operator==(dynamicArray<data>& input)
{
    if (/*cur_user_array*/this->sizeIs != input.sizeIs)
        return false;

    for (int i = 0; i < sizeIs; i++)
    {
        if (cur_user_array[i] != input[i])
            return false;
    }

    return true;
}
```

a.2.16 dynamicArray<data> operator+(dynamicArray<data>& input1);

```
template<typename data>
dynamicArray<data> dynamicArray<data>::operator+(dynamicArray<data>& input1)
{
    dynamicArray<data> temp;
    for (int i = 0; i < sizeIs; i++)
    {
        temp.push_back(cur_user_array[i] + input1[i]);
    }
    return temp;
}
```

a.2.17 data& operator[](int index);

```
template<typename data>
data& dynamicArray<data>::operator[](int index)
{
    return cur_user_array[index];
}
```

a.2.18 data& operator->();

```
template<typename data>
data& dynamicArray<data>::operator->()
{
    return cur_user_array;
}
```

b. customer_lib.h

b.1 Khu vực lưu trữ

- Lưu trong file có tên customer_lib.h và phân thực thi trong customer_lib.cpp

b.2 Cài đặt phương thức

b.2.1 Customer();

```
Customer::Customer() {
    fullName = "";
    numberPhone = "";
}
```

b.2.2 Customer(string fullName, string numberPhone);

```
Customer::Customer(string fullName, string numberPhone) {
    this->fullName = fullName;
    this->numberPhone = numberPhone;
}
```

b.2.3 string get_Name();

```
string Customer::get_Name() {
    return fullName;
}
```

b.2.4 string get_numberPhone();

```
string Customer::get_numberPhone() {
    return numberPhone;
}
```

c. infoCall_lib.h

c.1 Khu vực lưu trữ

- Các phương thức và các câu lệnh thực thi được viết ở trong infoCall_lib.h và infoCall_lib.cpp

c.2 Cài đặt phương thức

c.2.1 infoCall();

```
infoCall::infoCall() {
    numberPhone = "";
    minutes = 0;
    start_time = "";
    callDate = "";
    location = "";
}
```

c.2.2 infoCall(string numberPhone, int minutes, string start_time,int start_hour, string callDate, string location);

```
infoCall::infoCall(string numberPhone, int minutes, string start_time,int start_hour, string callDate, string location) {
    this->numberPhone = numberPhone;
    this->minutes = minutes;
    this->start_time = start_time;
    this->start_hour = start_hour;
    this->callDate = callDate;
    this->location = location;
}
```

c.2.3 string get_numberPhone();

```
string infoCall::get_numberPhone() {
    return numberPhone;
}
```

c.2.4 int get_minutes();

```
int infoCall::get_minutes() {
    return minutes;
}
```

c.2.5 string get_start_time();

```
string infoCall::get_start_time() {
    return start_time;
}
```

c.2.6 int get_start_hour();

```
int infoCall::get_start_hour() {
    return start_hour;
}
```

c.2.7 string get_callDate();

```
string infoCall::get_callDate() {
    return callDate;
}
```

c.2.8 string get_location();

```
string infoCall::get_location() {
    return location;
}
```

d. Statistic_lib.h

d.1 Khu vực lưu trữ

- Được lưu trữ trong 2 file Statistic_lib.h và Statistic_lib.cpp

d.2 Cài đặt phương thức

d.2.1 Statistic();

```
Statistic::Statistic() {
    fullName = "";
    numberPhone = "";
    callFee = 0;
    callNumber_NH = 0;
    callNumber_LC = 0;
    callNumber_X = 0;
    callNumber_RX = 0;
}
```

d.2.2 Statistic(string fullName, string numberPhone, int callFee, int callNumber_NH, int callNumber_LC, int callNumber_X, int callNumber_RX);

```
Statistic::Statistic(string fullName, string numberPhone, int callFee, int callNumber_NH, int callNumber_LC, int callNumber_X, int callNumber_RX) {
    this->fullName = fullName;
    this->numberPhone = numberPhone;
    this->callFee = callFee;
    this->callNumber_NH = callNumber_NH;
    this->callNumber_LC = callNumber_LC;
    this->callNumber_X = callNumber_X;
    this->callNumber_RX = callNumber_RX;
}
```

d.2.3 string get_fullName();

```
string Statistic::get_fullName() {
    return fullName;
}
```

d.2.4 string get_numberPhone();

```
 string Statistic::get_numberPhone() {
     return numberPhone;
 }
```

d.2.5 int get_callFee();

```
int Statistic::get_callFee() {
    return callFee;
}
```

d.2.6 int get_callNumber_NH();

```
int Statistic::get_callNumber_NH() {
    return callNumber_NH;
}
```

d.2.7 int get_callNumber_LC();

```
int Statistic::get_callNumber_LC() {
    return callNumber_LC;
}
```

d.2.8 int get_callNumber_X();

```
int Statistic::get_callNumber_X() {
    return callNumber_X;
}
```

d.2.9 int get_callNumber_RX();

```
int Statistic::get_callNumber_RX() {
    return callNumber_RX;
}
```

e. list_customer_lib.h

e.1 Khu vực lưu trữ

- Lưu trong 2 file list_customer_lib.h và list_customer_lib.cpp

e.2 Cài đặt phương thức

e.2.1 void inputFile();

```
void Customer_List::inputFile() {
    ifstream fileIn;
    fileIn.open("../khachhang.txt", ios::in);
    if (fileIn.is_open()) {
        string line;
        while (!fileIn.eof()) {
            getline(fileIn, line);
            if (line == "") {
                break;
            }
            stringstream ss(line);
            string fName;
            string nPhone;
            getline(ss, fName, ';');
            getline(ss, nPhone, ';');
            Customer _customer(fName, nPhone);
            listCustomer.push_back(_customer);
        }
    }
    fileIn.close();
}
```

e.2.2 dynamicArray<Customer> get_list_Customer();

```
dynamicArray<Customer> Customer_List::get_list_Customer() {
    return listCustomer;
}
```

f. list_called_lib.h

f.1 Khu vực lưu trữ

- Lưu trữ trong 2 file list_called_lib.h và list_called_lib.cpp

f.2 Cài đặt phương thức

f.2.1 List_Called();

```
List_Called::List_Called() {
    callList.clear();
}
```

f.2.2 void inputFile();

```
void List_Called::inputFile() {
    ifstream fileIn;
    fileIn.open("../cuocgoi.txt", ios::in);
    if (fileIn.is_open()) {
        string line;
        while (!fileIn.eof()) {
            getline(fileIn, line);
            if (line == "") {
                break;
            }
            stringstream ss(line);
            string nPhone;
            string minutes;
            string timeStart;
            int hourStart;
            string callDate;
            string location;
            getline(ss, nPhone, ';');
            getline(ss, minutes, ';');
            getline(ss, timeStart, ';');
            getline(ss, callDate, ';');
            getline(ss, location, ';');
            hourStart = stoi(timeStart.substr(0,2));
            infoCall iCall(nPhone, stoi(minutes), timeStart,hourStart, callDate, location);
            callList.push_back(iCall);
        }
    }
    fileIn.close();
}
```

f.2.3 dynamicArray<infoCall> get_List_Called();

```
dynamicArray<infoCall> List_Called::get_List_Called() {
    return callList;
}
```

g. list_statistic_lib.h

g.1 Khu vực lưu trữ

- Lưu trữ trong 2 tệp list_statistic_lib.h và list_statistic_lib.cpp

g.2 Cài đặt phương thức

g.2.1 List_Statistic();

```
List_Statistic::List_Statistic() {
    statisticList.clear();
}
```

g.2.2 void add(Statistic Listt);

```
void List_Statistic::add(Statistic Listt) {
    statisticList.push_back(Listt);
}
```

g.2.3 void outPut();

```
void List_Statistic::outPut() {
    ofstream fileOut;
    fileOut.open("../ketqua.txt", ios::out);
    if (fileOut.is_open()) {
        fileOut << std::left
                << "|" << std::setw(30) << "Full Name"
                << "|" << std::setw(15) << "NumberPhone"
                << "|" << std::setw(15) << "Total_FEE"
                << "|" << std::setw(10) << "number_NH"
                << "|" << std::setw(10) << "number_LC"
                << "|" << std::setw(10) << "number_X"
                << "|" << std::setw(10) << "number_RX"
                << "|\n";
        fileOut << "|------------------------------|---------------|---------------|----------|----------|----------|----------|\n";
        for (int i = 0; i < statisticList.get_size(); i++) {
            fileOut << "|" << std::setw(30) << std::left << statisticList[i].get_fullName();
            fileOut << "|" << std::setw(15) << std::left << statisticList[i].get_numberPhone().substr(1, 10);
            fileOut << "|" << std::setw(15) << std::left << statisticList[i].get_callFee();
            fileOut << "|" << std::setw(10) << std::left << statisticList[i].get_callNumber_NH();
            fileOut << "|" << std::setw(10) << std::left << statisticList[i].get_callNumber_LC();
            fileOut << "|" << std::setw(10) << std::left << statisticList[i].get_callNumber_X();
            fileOut << "|" << std::setw(10) << std::left << statisticList[i].get_callNumber_RX();
            fileOut << "|\n";

            fileOut << "|------------------------------|---------------|---------------|----------|----------|----------|----------|\n";
        }
    }
    fileOut.close();
```

h. systemStatistic_lib.h

h.1 Khu vực lưu trữ

- Lưu trữ trong 2 tệp systemStatistic_lib.h và systemStatistic_lib.cpp

h.2 Cài đặt phương thức

h.2.1 systemStatistics();

```
systemStatistics::systemStatistics() {
    _listCustomer.inputFile();
    _listCalled.inputFile();
}
```

h.2.2 void tinhTien();

```
void systemStatistics::tinhTien() {
    dynamicArray<Customer> listCustomerinFile = this->_listCustomer.get_list_Customer();
    dynamicArray<infoCall> listCalledinFile = this->_listCalled.get_List_Called();
    for (int i = 0; i < listCustomerinFile.get_size(); i++) {
        int total_FEE = 0;
        int _numberCall_NH = 0;
        int _numberCall_LC = 0;
        int _numberCall_X = 0;
        int _numberCall_RX = 0;
        int factor_location = 1;
        for (int j = 0; j < listCalledinFile.get_size(); j++){
            string num1 = listCustomerinFile[i].get_numberPhone();
            num1.erase(0, 1); // erase the backspace in num1
            string num2 = listCalledinFile[j].get_numberPhone();
            if (checking_stt(num1, num2)) {
                string location = listCalledinFile[j].get_location();
                location.erase(0, 1); // erase the backspace in location
                if (checking_stt(location, "NH")) {
                    _numberCall_NH++;
                    factor_location = 1;
                }
                else if (checking_stt(location, "LC")){
                    _numberCall_LC++;
                    factor_location = 2;
                }
                else if (checking_stt(location, "X")){
                    _numberCall_X++;
                    factor_location = 3;
                }
                else if (checking_stt(location, "RX")){
                    _numberCall_RX++;
                    factor_location = 4;
                }
                int minute = listCalledinFile[j].get_minutes();
                if ((listCalledinFile[j].get_start_hour() >= 23 && listCalledinFile[j].get_start_hour() <= 5)
                    || listCalledinFile[j].get_callDate() == "Saturday" ||
                    listCalledinFile[j].get_callDate() == "Sunday") {
                    total_FEE += 770 * minute * factor_location;
                }
                else total_FEE += 1100 * minute * factor_location;
            }
        }
        Statistic ketQua(listCustomerinFile[i].get_Name(), listCustomerinFile[i].get_numberPhone(), total_FEE,
                         _numberCall_NH, _numberCall_LC, _numberCall_X, _numberCall_RX);
        _listStatistic.add(ketQua);
    }
}
```

h.2.3 void xuat();

```
void systemStatistics::xuat() {
    cout << "Complete!";
    _listStatistic.outPut();
}
```

i. converttime.h

i.1 converttime();

```
converttime::converttime() {
    day = 0;
    month = 0;
    year = 0;
    date = "01/01/0001";
}
```

i.2 converttime(int day, int month, int year);

```
converttime::converttime(int day, int month, int year) {
    this->day = day;
    this->month = month;
    this->year = year;
    date = "(" + to_string(day) + "/" + to_string(month) + "/" + to_string(year) + ")";
    total_day = Caculate_Day(day,month,year);
}
```

i.3 converttime(string date);

```
converttime::converttime(string date) {
    this->date = date;
    stringstream ss(date);
    string _day, _month, _year;
    getline(ss, _day, '/');
    getline(ss, _month, '/');
    getline(ss, _year, '/');
    day = stoi(_day);
    month = stoi(_month);
    year = stoi(_year);
    total_day = Caculate_Day(day,month,year);
}
```

i.4 string convert_date_to_celandar(int total_day);

```
string converttime::convert_date_to_celandar(int total_day){
    int day = total_day%7;
    string celandars[7] = {"Monday","Tuesday","Wednesday","Thursday"
                           ,"Friday","Saturday","Sunday"}; // 0 1 2 3 4 5 6
    return celandars[day];
}
```

i.5 int get_day();

```
int converttime::get_day() {
    return day;
}
```

i.6 int get_month();

```
int converttime::get_month() {
    return month;
}
```

i.7 int get_year();

```
int converttime::get_year() {
    return year;
}
```

i.8 int get_total_day();

```
int converttime::get_total_day() {
    return total_day;
}
```

i.9 string get_date();

```
string converttime::get_date() {
    return date;
}
```

k. main.cpp

```
#include "libcs/systemStatistic_lib.h"

#include <iostream>

using namespace std;

int main(){
   systemStatistics tinhTienDienThoai;
   tinhTienDienThoai.tinhTien();
   tinhTienDienThoai.xuat();
}
```

4. PHÂN TÍCH THỜI GIAN CHẠY CÓ TRONG PHƯƠNG THỨC CỦA CÁC LỚP

a. vector.hpp

a.1 bool operator<(dynamicArray<data>& input);

- Có 2 câu lệnh điều kiện: 2 phép toán
- Vòng lặp: n phép toán

- Câu lệnh điều kiện trong vòng lặp có n lần so sánh điều kiện: n phép toán
- Trường trường hợp có 1 phép toán trả về: 1 phép toán

=> Tổng : 2n +3 phép toán cơ sở

=> Độ phức tạp: O(n)

a.2 bool operator>(dynamicArray<data>& input);

- Tương tự a.1 có độ phức tạp O(n) và có 2n+2 phép toán cơ sở

a.3 bool operator>=(dynamicArray<data>& input);

- Tương tự a.1 có độ phức tạp O(n) và có 2n+2 phép toán cơ sở

a.4 bool operator<=(dynamicArray<data>& input);

- Tương tự a.1 có độ phức tạp O(n) và có 2n+2 phép toán cơ sở

a.5 bool operator==(dynamicArray<data>& input);

- Tương tự a.1 có độ phức tạp O(n) và có 2n+2 phép toán cơ sở

a.6 dynamicArray<data> operator+(dynamicArray<data>& input1);

- Khai báo một mảng: 1 phép toán
- Vòng lặp n lần: n phép toán
- Câu lệnh trả về: 1 phép toán

=> Tổng: n+2 phép toán cơ sở

=> Độ phức tạp: O(n)

a.7 data& operator[](int index);

- Một câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

a.8 data& operator->();

- Một câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

a.9 dynamicArray();

- 2 câu lệnh gán giá trị và một câu lệnh cấp phát động: 3 phép toán

=> Độ phức tạp O(3)

a.10 ~dynamicArray();

- Xóa, giải phóng bộ nhớ: n phép toán

=> Độ phức tạp: O(n)

a.11 bool is_empty();

- Một câu lệnh điều kiện và một câu lệnh trả về: 2 phép toán

=> Độ phức tạp: O(2)

a.12 void auto_resize();

- Một câu lệnh điều kiện: 1 phép toán
- Phép tăng giá trị lên 1 đơn vị: 1 phép toán
- Phép nhân 2: 1 phép toán
- Cấp phát động: n phép toán
- Vòng lặp có n/2 phép toán
- Lệnh gán giá trị: n/2 phép toán
- Giải phóng bộ nhớ: n phép toán
- Lệnh gán cuối cùng: 1 phép toán

=> Tổng: 3n+4 phép toán cơ sở

=> Độ phức tạp: O(n)

a.13 void clear();

- Một câu lệnh gán độ dài mảng bằng 0

=> Độ phức tạp: O(1)

a.14 void push_back(data push_data_to_back_vector);

- Câu lệnh tang giá trị lên 1 đơn vị: 1 phép toán
- Gọi hàm auto_resize();: O(n)
- Gán dữ liệu mới vào vector: 1 phép toán

=> Độ phức tạp: O(n)

a.15 void remove_data(data lookFor);

- Gán giá trị cho 1 biến sizeIsnew: 1 phép toán
- Vòng lặp: n+2 phép toán
- Câu lệnh điều kiện: 1 phép toán
- Vòng lặp con: n+2-i phép toán
- Giảm giá trị 1 đơn vị: n phép toán
- Câu lệnh điều kiện cuối cùng: 1 phép toán

=> Tổng: 3n+7-i phép toán

=> Độ phức tạp: O(n)

a.16 void remove(int index);

- Một câu lệnh điều kiện : 1 phép toán
- Gọi hàm is_empty(): O(2)
- Vòng lặp: n-d+2 phép toán

- n-d+2 lần gán giá trị
- 1 lệnh giá biến size đi 1 đơn vị

=> Tổng: 2n-2d+8 phép toán

=> Độ phức tạp: O(n)

a.17 int get_size();

- Trả về giá trị size: 1 phép toán

=> Độ phức tạp: O(1)

a.18 void debugAry();

- Vòng lặp: n+2 phép toán
- Có 3 + n lần in ra màn hình

=> Tổng: 2n+5 phép toán

=> Độ phức tạp: O(n)

b. customer_lib.h

b.1 Customer();

- 2 câu lệnh gán giá trị: 2 phép toán

=> Độ phức tạp: O(2)

b.2 Customer(string fullName, string numberPhone);

- 2 câu lệnh gán giá trị: 2 phép toán

=> Độ phức tạp: O(2)

b.3 string get_Name();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

b.4 string get_numberPhone();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

c. infoCall_lib.h

c.1 infoCall();

- 5 lệnh gán giá trị:

=> Độ phức tạp: O(5)

c.2 infoCall(string numberPhone, int minutes, string start_time,int start_hour, string callDate, string location)

- 5 lệnh gán giá trị:

=> Độ phức tạp: O(5)

c.3 string get_numberPhone();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

c.4 int get_minutes();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

c.5 string get_start_time();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

c.6 int get_start_hour();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

c.7 string get_callDate();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

c.8 string get_location();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

d. Statistic_lib.h

d.1 Statistic();

- 7 câu lệnh gán giá trị: 7 phép toán

=> Độ phức tạp: O(7)

d.2 Statistic(string fullName, string numberPhone, int callFee, int callNumber_NH, int callNumber_LC, int callNumber_X, int callNumber_RX)

- 7 câu lệnh gán giá trị: 7 phép toán

=> Độ phức tạp: O(7)

d.3 string get_fullName();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

d.4 string get_numberPhone();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

d.5 int get_callFee();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

d.6 int get_callNumber_NH();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

d.7 int get_callNumber_LC();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

d.8 int get_callNumber_X();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

d.9 int get_callNumber_RX();

- 1 câu lệnh trả về: 1 phép toán

=> Độ phức tạp: O(1)

e. list_customer_lib.h

e.1 void inputFile();

- Câu lệnh khai báo con trỏ file: 1 phép toán
- Câu lệnh truy cập một địa chỉ file tĩnh: 1 phép toán
- câu lệnh điều kiện : 1 phép toán
- khai báo 1 biến string: 1 phép toán
- vòng lặp : n phép toán với n là số dòng ở trong tệp đọc được.
- câu lệnh đọc từng dòng: n phép toán
- câu lệnh điều kiện dừng vòng lặp: n phép toán
- câu lệnh khai báo chia chuỗi thành các chuỗi con : n phép toán
- khai báo 2 biến string: 2*n phép toán
- 2 câu lệnh đọc hết dòng : 2*n phép toán
- Khởi tạo lớp Customer: n phép toán
- Đẩy vào vector : n phép toán
- Đóng file: 1 phép toán

=> Tổng : 10*n+5 phép toán cơ sở

=> Độ phức tạp O(n)

e.2 dynamicArray<Customer> get_list_Customer();

- 1 Câu lệnh trả về: 1 phép toán

=> Đô phức tạp O(1)

f. list_called_lib.h

f.1 List_Called();

- 1 câu lệnh xóa dữ liệu vector : 1 phép toán

f.2 void inputFile();

- Câu lệnh khai báo con trỏ file: 1 phép toán
- Câu lệnh truy cập một địa chỉ file tĩnh: 1 phép toán
- câu lệnh điều kiện : 1 phép toán
- khai báo 1 biến string: 1 phép toán
- vòng lặp : n phép toán với n là số dòng ở trong tệp đọc được.
- câu lệnh đọc từng dòng: n phép toán
- câu lệnh điều kiện dừng vòng lặp: n phép toán
- câu lệnh khai báo chia chuỗi thành các chuỗi con : n phép toán
- khai báo 5 biến string: 5*n phép toán
- Khai báo 1 biến số nguyên: n phép toán
- 5 câu lệnh đọc hết dòng : 5*n phép toán
- Câu lệnh chuyển chuỗi về số nguyên : 1 phép toán
- Khởi tạo lớp Customer: n phép toán
- Đẩy vào vector : n phép toán
- Đóng file: 1 phép toán

=> Tổng : 17*n + 6 phép toán cơ sở

=> Độ phức tạp O(n)

f.3 dynamicArray<infoCall> get_List_Called();

- 1 Câu lệnh trả về: 1 phép toán

=> Đô phức tạp O(1)

g. list_statistic_lib.h

g.1 List_Statistic();

- 1 câu lệnh xóa dữ liệu vector : 1 phép toán

g.2 void add(Statistic Listt);

- 1 câu lệnh thêm vào vector: 1 phép toán

g.3 void outPut();

- Câu lệnh khai báo con trỏ file: 1 phép toán
- Câu lệnh truy cập một địa chỉ file tĩnh: 1 phép toán
- câu lệnh điều kiện : 1 phép toán
- câu lệnh in đầu bảng: 2 phép toán
- Vòng lặp : n+3 phép toán
- Mỗi lần in ra cần 4*(n+3) phép toán
- Lần in xuống dòng và hàng ngăn cách : 2*(n+3) phép toán
- Đóng file : 1 phép toán

=> Tổng : 6+(n+3)*6 phép toán cơ sở

=> Độ phức tạp: O(n)

h. systemStatistic_lib.h

h.1 systemStatistics();

- 2 câu lệnh truy cập 2 hàm: 1 phép toán

h.2 void tinhTien()

- 2 lệnh gán lấy danh sách từ 2 file đầu vào: 2 phép toán
- Vòng lặp đầu: n+3 phép toán với N là kích thước của listCustomerinFile
- Vòng lặp 2: m + 3 phép toán với M là kích thước của listCalledinFile.
- Số phép toán trong vòng for đầu là : 6 phép toán
- Gọi k là tổng số phép toán trong vòng for thứ 2
- Có 2 lệnh tạo một đối tượng và đẩy đối tượng vào vector: 2*n phép toán

=> Tổng: 2+ (n+3) + (m+3) + 8*n + (k*m)*n phép toán cơ sở

=> Độ phức tạp: O(m*n*k)

h.3 void xuat();

- 1 câu lệnh in ra dòng thành công: 1 phép toán
- 1 câu lệnh khai báo 1 hàm output: 1 phép toán

=> Độ phức tạp: O(2)

i. converttime.h

i.1 converttime();

- 4 lệnh gán: 4 phép toán

=> Độ phức tạp: O(4)

i.2 converttime(int day, int month, int year);

- 4 lệnh gán: 4 phép toán

- 1 lệnh gọi hàm tính tổng số ngày: 1 phép toán

=> Tổng : 5 phép toán

i.3 converttime(string date);

- 9 lệnh gán: 9 phép toán
- 1 lệnh gọi hàm tính tổng số ngày: 1 phép toán

=> Tổng : 10 phép toán

i.4 string convert_date_to_celandar(int total_day);

- 1 lệnh khai báo biến string
- 1 lệnh khai báo mảng
- 1 lệnh trả về giá trị của mảng: 2 phép toán

=> Tổng 4 phép toán

i.5 int get_day();

- 1 câu lệnh trả về : 1 phép toán

=> Độ phức tạp: O(1)

i.6 int get_month();

- 1 câu lệnh trả về : 1 phép toán

=> Độ phức tạp: O(1)

i.7 int get_year();

- 1 câu lệnh trả về : 1 phép toán

=> Độ phức tạp: O(1)

i.8 int get_total_day();

- 1 câu lệnh trả về : 1 phép toán

=> Độ phức tạp: O(1)

i.9 string get_date();

- 1 câu lệnh trả về : 1 phép toán

=> Độ phức tạp: O(1)

k. main.cpp

- 1 câu lệnh khởi tạo 1 biến systemStatistics
- 2 câu lệnh gọi hàm

5. Ứng dụng của đề bài

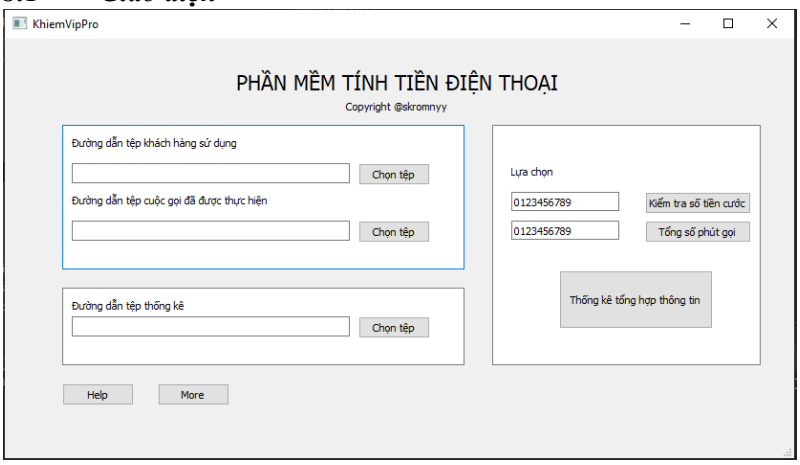
a. Chạy trên console

Thay đổi giá trị trong 2 file input: khanhhang.txt và cuocgoi.txt

Chạy thành công khi console hiện: complete và dữ liệu được đưa vào file ketqua.txt

b. Phần mềm tính tiền điện thoại

b.1 Giao diện



b.2 Phương thức hoạt động

- Tại giao diện của phần mềm
- Có 3 phần chọn đường dẫn tới tệp dữ liệu cần nhập và cần xuất
- Nhấn chọn tệp để mở lên cửa sổ file brower để chọn tệp đầu vào
- Tại phần chọn tệp cho đường dẫn thống kê : chọn tệp để đưa vào tệp cần xuất hoặc nếu không file sẽ xuất theo đường dẫn mặc định.
- Tại table lựa chọn có 3 option chính:

+ Xuất toàn bộ file

+ Kiểm tra số tiền cước của một số điện thoại: Nếu sdt đó có trong CSDL thì sẽ xuất ra họ tên, sdt, số tiền, nếu không sẽ in ra thông báo: sdt không tồn tại trong CSDL

+ Kiểm tra tổng số phút gọi của một số điện thoại: Nếu sdt đó có trong CSDL thì sẽ xuất ra họ tên, sdt, số phút gọi, nếu không sẽ in ra thông báo: sdt không tồn tại trong CSDL

- Ngoại lệ xảy ra nếu chưa nhập đủ 2 đường dẫn đầu vào, lựa chọn kiểm tra số tiền hoặc số phút nếu không có đầu vào.

6. Đường dẫn tới source code phần mềm và source code báo cáo

- Đường dẫn tới source code báo cáo: **https://github.com/hoangmanhkhiem/Great-Exercises-On-Data-structures-and-Algorithms/tree/main/Problem_13**

- Đường dẫn tới source code phần mềm: **https://github.com/hoangmanhkhiem/Great-Exercises-On-Data-structures-and-Algorithms/tree/main/app/Project13**

7. Demo chạy code và chạy phần mềm

- Đường dẫn video demo tính tiền điện thoại trên console:
https://drive.google.com/file/d/1WPFsT4vJtgDsOzl2s_gnJwx2TquYl9nM/view?usp=drive_link

- Đường dẫn video demo phần mềm tính tiền điện thoại:

https://drive.google.com/file/d/1kHv7V6BjDEozaecYJHuLjlgxbwT3P9Eu/view?usp=drive_link