

Lab 4

MỤC LỤC

3.1 Hàm băm cơ bản.....	2
1. Biên dịch và chạy chương trình trên	2
2. Hãy chỉ ra công thức toán của hàm băm trong đoạn code mẫu trên	2
3. Trong hàm khởi tạo bảng băm, chỉ ra tập U có bao nhiêu phần tử, tập các khoá k lưu trong bảng băm có bao nhiêu phần tử.....	2
4. Mô tả quy trình các bước chi tiết từ khi xây dựng bảng băm đến khi xuất ra kết quả tìm kiếm.	2
3.2 Hàm băm cho bài toán tra tên sinh viên.	4
1. Biên dịch đoạn chương trình trên.....	4
2. Chỉ ra những thay đổi của chương trình này so với chương trình cơ bản.	4
3. Nếu bỏ đoạn code sau trong hàm băm	5
4. Nếu thay $K[i].key$ trong đoạn code sau.....	5
5. Nếu thay giá trị $K[2].key$ chỗ dòng code sau.....	5
6. Viết lại chương trình trên, cho phép người dùng tự nhập số lượng giá trị k và thông tin của SV cần đưa vào bảng băm (ở đoạn code trên tác giả chỉ định cứng số lượng là 5).....	6
Bài tập nâng cao:	7
1. Cải tiến bài toán tra thông tin SV theo tên của SV (giá trị băm là tên SV, giá trị xuất ra là thông tin SV như email, số điện thoại, MSSV...). Sử dụng hàm băm hợp lý để băm chuỗi tên SV. Gợi ý: hàm băm là hàm chuyển một chuỗi sang một số theo mã ASCII.	7
2. Áp dụng phương pháp nối kết để giải quyết đụng độ cho bài toán tra tên SV.	9
3. Áp dụng phương pháp địa chỉ mở (dò tuyến tính) để giải quyết đụng độ cho bài toán tra tên SV.....	11
4. Áp dụng phương pháp địa chỉ mở (dò bậc 2) để giải quyết đụng độ cho bài toán tra tên SV.	14
5. Áp dụng phương pháp địa chỉ mở (băm kép) để giải quyết đụng độ cho bài toán tra tên SV.....	16

3.1 Hàm băm cơ bản

Bài toán: Cho một mảng các số nguyên U có M phần tử. U chính là bảng băm. Có k khoá cũng là các số nguyên, mỗi khoá được lưu vào bảng băm sử dụng công thức tính modulo ($H(k) = k \bmod M$). Sau đó nhập một giá trị khoá, in ra giá trị tương ứng.

Nhiệm vụ: giúp hiểu một cách cơ bản về bảng băm, hàm băm.

Cách thực hiện:

- Xây dựng hàm băm.
- Xây dựng hàm khởi tạo các giá trị cho bảng băm sử dụng hàm băm ở trên.
- Nhập khoá cần tìm, cũng dùng hàm băm ở trên.

1. Biên dịch và chạy chương trình trên

```
● @Glasspham on E:/Code/Github/DSA-GTVTHCM
# cd "e:\Code\Github\DSA-GTVTHCM\Lab_4" ; if ($?) { g++ *.cpp -o main } ; if ($?) { .\main }
Nhập khóa tìm kiếm: 4
Giá trị phần tử cần tìm kiếm: 4
○ @Glasspham on E:/Code/Github/DSA-GTVTHCM/Lab_4
#
```

2. Hãy chỉ ra công thức toán của hàm băm trong đoạn code mẫu trên

Công thức toán của hàm băm là: $H(k) = k \bmod M$

```
int Hash(int k, int M) {
    if (M == 0) return 0;
    return (k % M);
}
```

3. Trong hàm khởi tạo bảng băm, chỉ ra tập U có bao nhiêu phần tử, tập các khoá k lưu trong bảng băm có bao nhiêu phần tử.

Tập U (bảng băm)

- Trong chương trình, mảng U có kích thước $M=10M=10$, nghĩa là tập U chứa **10 phần tử**.
- Ban đầu, tất cả phần tử của U được khởi tạo bằng 0 ($U[i] = 0$).

Tập các khóa được lưu trong bảng băm

- Các khóa cần lưu vào bảng băm được khai báo trong mảng K:
- `int K[5] = {1,2,4,6,9};`

→ Có **5 khóa** được lưu vào bảng băm.

4. Mô tả quy trình các bước chi tiết từ khi xây dựng bảng băm đến khi xuất ra kết quả tìm kiếm.

Bước 1: Khởi tạo bảng băm

- Cấp phát động mảng U có $M = 10$ phần tử.

- Gán tất cả phần tử của U bằng 0 để đánh dấu ô trống.

Bước 2: Chèn khóa vào bảng băm

- Duyệt qua từng khóa trong $K = \{1, 2, 4, 6, 9\}$.
- Áp dụng hàm băm $h(k) = k \% M$ để xác định vị trí của khóa trong U.
- Gán giá trị $U[pos] = K[i]$ để lưu khóa vào bảng băm.

Khóa kk	Vị trí $h(k)=k \bmod 10$	Bảng băm sau khi chèn
1	1	$U[1] = 1$
2	2	$U[2] = 2$
4	4	$U[4] = 4$
6	6	$U[6] = 6$
9	9	$U[9] = 9$

Bước 3: Tìm kiếm khóa

- Nhập một khóa x từ bàn phím.
- Tính $pos = h(x) = x \% 10$ để xác định vị trí tìm kiếm.
- Nếu $U[pos] == 0$, thông báo "Không tìm thấy khóa".
- Ngược lại, in giá trị $U[pos]$ (khóa tìm thấy).

Ví dụ minh họa quá trình tìm kiếm

Trường hợp 1:

Người dùng nhập $x = 4$

- $pos = 4 \% 10 = 4$
- $U[4] = 4 \rightarrow$ Xuất: "Giá trị phần tử cần tìm kiếm: 4"

Trường hợp 2:

Người dùng nhập $x = 3$

- $pos = 3 \% 10 = 3$
- $U[3] = 0 \rightarrow$ Xuất: "Không tìm thấy khóa trong bảng băm"

3.2 Hàm băm cho bài toán tra tên sinh viên.

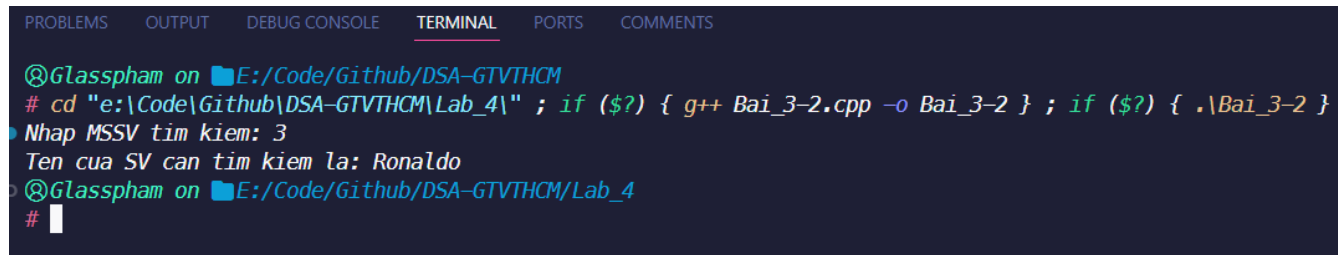
Sau khi đã nắm được những khái niệm về hàm băm, bảng băm và quy trình các bước chi tiết, sinh viên vận dụng vào bài toán tra tên SV theo MSSV.

Bài toán: Cho tập U các đối tượng sinh viên, thông tin mỗi SV bao gồm MSSV và tên SV đó. Giả sử ta có một tập k các SV trong Khoa CNTT, được lưu vào bảng băm U theo hàm băm modulo tương tự bài toán cơ bản. Hãy xây dựng ứng dụng cho phép tìm kiếm tên 1 SV theo MSSV được nhập từ bàn phím.

Cách thực hiện:

- Tạo cấu trúc Word biểu diễn cho thông tin của 1 SV, gồm hai trường: key (MSSV) và value (tên SV).
- Thực hiện hàm băm giống như bài toán cơ bản.
- Hàm khởi tạo bảng băm sẽ đưa danh sách các SV đang có vào bảng băm U.
- Khi tra cứu, băm MSSV của SV cần tra cứu, sau đó lấy thông tin của SV trong bảng băm U dựa theo giá trị vừa băm.

1. Biên dịch đoạn chương trình trên.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
@Glasspham on E:/Code/Github/DSA-GTVTHCM
# cd "E:/Code/Github/DSA-GTVTHCM/Lab_4\" ; if ($?) { g++ Bai_3-2.cpp -o Bai_3-2 } ; if ($?) { .\Bai_3-2 }
Nhập MSSV tìm kiếm: 3
Tên của SV cần tìm kiếm là: Ronaldo
@Glasspham on E:/Code/Github/DSA-GTVTHCM/Lab_4
#
```

2. Chỉ ra những thay đổi của chương trình này so với chương trình cơ bản.

Câu 2: Những thay đổi so với chương trình cơ bản

So với chương trình đầu tiên, chương trình này có một số thay đổi quan trọng:

1. Cấu trúc dữ liệu:

- Thay vì dùng int để lưu giá trị trực tiếp trong bảng băm, chương trình này sử dụng struct Word chứa:

```
struct Word {
    int key;
    char value[128];
};
```

- key là số nguyên (MSSV), value là tên sinh viên (chuỗi ký tự).

2. Cách lưu trữ dữ liệu:

- Mảng K[5] chứa các cặp MSSV - Tên thay vì chỉ là một mảng số nguyên.

3. Cách kiểm tra tìm kiếm:

- Thay vì kiểm tra $U[pos] == 0$, chương trình kiểm tra $U[pos].key == 0$ để xác định xem vị trí trong bảng băm có dữ liệu hay không.

4. Kết quả tìm kiếm:

- Khi tìm thấy sinh viên, chương trình in ra value thay vì in số nguyên.

3. Nếu bỏ đoạn code sau trong hàm băm

`if (M == 0) return 0;`

Thì có được không? Giải thích lý do.

- Khi $M == 0$, phép $k \% M$ sẽ gây lỗi chia cho 0 (division by zero), dẫn đến lỗi chương trình.
- Việc kiểm tra `if (M == 0) return 0;` giúp ngăn chặn lỗi này bằng cách trả về giá trị mặc định 0 khi $M = 0$.

→ **Kết luận:** Không nên bỏ đoạn kiểm tra này.

4. Nếu thay `K[i].key` trong đoạn code sau

`pos = Hash(K[i].key, M);`

Thành

`pos = Hash(K[i], M);`

Thì chuyện gì xảy ra? Giải thích lý do tại sao.

- Hàm `Hash(int k, int M)` nhận vào một số nguyên k , nhưng `K[i]` là một struct kiểu `Word`, không phải số nguyên.
- Nếu gọi `Hash(K[i], M)`, chương trình sẽ báo lỗi biên dịch (error: cannot convert 'Word' to 'int').

→ **Kết luận:** Chương trình sẽ bị lỗi biên dịch vì truyền sai kiểu dữ liệu vào hàm băm.

5. Nếu thay giá trị `K[2].key` chỗ dòng code sau

`K[2].key = 5;`

Thành

`K[2].key = 13;`

Và lúc chạy, nhập MSSV là 3. Kết quả xuất ra là bao nhiêu? Đúng hay sai? Nếu sai, giải thích lý do tại sao?

Phân tích:

- Khi `K[2].key = 5`;, vị trí trong bảng băm sẽ là $5 \% 10 = 5$, lưu "Rooney".
- Nếu thay `K[2].key = 13`;, vị trí sẽ là $13 \% 10 = 3$, tức "Rooney" sẽ lưu tại `U[3]`.

Tình huống tìm MSSV 3:

- Khi nhập $x = 3$, chương trình tính $pos = 3 \% 10 = 3$.
- `U[3]` chứa "Ronaldo" (từ `K[1]` ban đầu).

- Kết quả xuất ra: "Ten của SV cần tìm kiếm là: Ronaldo", đúng.

→ **Kết luận:** Kết quả vẫn đúng vì U[3] vẫn chứa "Ronaldo".

6. Viết lại chương trình trên, cho phép người dùng tự nhập số lượng giá trị k và thông tin của SV cần đưa vào bảng băm (ở đoạn code trên tác giả chỉ định cứng số lượng là 5).

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct Word {
    int key;
    char value[128];
};

int Hash(int k, int M) {
    if (M == 0) return 0;
    return (k % M);
}

void InitHash(Word *&U, int M, Word *K, int n) {
    for (int i = 0; i < M; i++)
        U[i].key = 0; // Đánh dấu ô trống
    for (int i = 0; i < n; i++) {
        int pos = Hash(K[i].key, M);
        U[pos] = K[i]; // Lưu vào bảng băm
    }
}

int main() {
    int M = 10; // Kích thước bảng băm
    int n;      // Số lượng sinh viên nhập vào
    printf("Nhập số lượng sinh viên: ");
    scanf("%d", &n);

    Word *K = new Word[n]; // Mảng chứa dữ liệu nhập vào
    for (int i = 0; i < n; i++) {
```

```

    printf("Nhap MSSV thu %d: ", i + 1);

    scanf("%d", &K[i].key);

    printf("Nhap ten SV: ");

    scanf(" %[^\\n]", K[i].value);
}

Word *U = new Word[M]; // Bảng băm

InitHash(U, M, K, n);

// Tìm kiếm sinh viên

int x;

printf("Nhap MSSV tim kiem: ");

scanf("%d", &x);

int pos = Hash(x, M);

if (U[pos].key == 0)

    printf("Khong tim thay SV nao trong bang bam\\n");

else printf("Ten cua SV can tim kiem la: %s\\n", U[pos].value);

// Giải phóng bộ nhớ

delete[] K;

delete[] U;

return 0;

}

```

Bài tập nâng cao:

1. Cải tiến bài toán tra thông tin SV theo tên của SV (giá trị băm là tên SV, giá trị xuất ra là thông tin SV như email, số điện thoại, MSSV...). Sử dụng hàm băm hợp lý để băm chuỗi tên SV. Gợi ý: hàm băm là hàm chuyển một chuỗi sang một số theo mã ASCII.

```

#include <iostream>

#include <unordered_map>

#include <string>

using namespace std;

// Cấu trúc lưu trữ thông tin sinh viên

struct Student {

```

```

string email;

string phone;

int studentID;

};

// Bảng băm dùng `unordered_map` để lưu sinh viên theo tên
unordered_map<string, Student> studentTable;

// Hàm thêm sinh viên vào bảng băm
void insertStudent(string name, int id, string email, string phone) {
    studentTable[name] = {email, phone, id};
}

// Hàm tìm kiếm sinh viên theo tên
void searchStudent(string name) {
    auto it = studentTable.find(name);
    if (it != studentTable.end()) {
        cout << "Thông tin SV:\n";
        cout << "Ten: " << name << "\n";
        cout << "MSSV: " << it->second.studentID << "\n";
        cout << "Email: " << it->second.email << "\n";
        cout << "SDT: " << it->second.phone << "\n";
    } else cout << "Không tìm thấy sinh viên.\n";
}

int main() {
    cout << "Nhập số lượng sinh viên: ";

    int n; cin >> n;

    cin.ignore();

    for (int i = 0; i < n; i++) {
        string name, email, phone;

        int id;

        cout << "Nhập tên SV: ";

        getline(cin, name);
    }
}

```



```

    cout << "Nhap MSSV: ";

    cin >> id;

    cin.ignore();

    cout << "Nhap email: ";

    getline(cin, email);

    cout << "Nhap so dien thoai: ";

    getline(cin, phone);

    insertStudent(name, id, email, phone);

}

string searchName;

cout << "\nNhap ten SV can tim: ";

getline(cin, searchName);

searchStudent(searchName);

return 0;

}

```

2. Áp dụng phương pháp nối kết để giải quyết dụng độ cho bài toán tra tên SV.

```

#include <iostream>

#include <list>

#include <vector>

using namespace std;

struct Student {

    string name;

    string email;

    string phone;

    int studentID;

};

const int MAXN = 10;

vector<list<Student>> hashTable(MAXN);

// Hàm băm chuỗi theo mã ASCII

int hashFunction(string name) {

```

```

int value = 0;

for (char c : name) value += c;

return value % MAXN;
}

// Hàm thêm sinh viên vào bảng băm
void insertStudent(string name, int id, string email, string phone) {
    int index = hashFunction(name);

    hashTable[index].push_back({name, email, phone, id});
}

// Hàm tìm kiếm sinh viên theo tên
void searchStudent(string name) {
    int index = hashFunction(name);

    for (Student s : hashTable[index]) {
        if (s.name == name) { // Kiểm tra chính xác tên
            cout << "Thông tin SV:\n";

            cout << "Ten: " << s.name << "\n";

            cout << "MSSV: " << s.studentID << "\n";

            cout << "Email: " << s.email << "\n";

            cout << "SDT: " << s.phone << "\n";

            return;
        }
    }

    cout << "Khong tim thay sinh vien.\n";
}

int main() {
    cout << "Nhap so luong sinh vien: ";

    int n; cin >> n;

    cin.ignore();

    for (int i = 0; i < n; i++) {
        string name, email, phone;

```

```

    int id;

    cout << "Nhap ten SV: ";

    getline(cin, name);

    cout << "Nhap MSSV: ";

    cin >> id;

    cin.ignore();

    cout << "Nhap email: ";

    getline(cin, email);

    cout << "Nhap so dien thoai: ";

    getline(cin, phone);

    insertStudent(name, id, email, phone);

}

string searchName;

cout << "\nNhap ten SV can tim: ";

getline(cin, searchName);

searchStudent(searchName);

return 0;

}

```

3. Áp dụng phương pháp địa chỉ mở (dò tuyến tính) để giải quyết độ cho bài toán tra tên SV.

```

#include <iostream>

#include <vector>

using namespace std;

struct Student {

    string name;

    string email;

    string phone;

    int studentID;

    bool isOccupied = false;

};

const int MAXN = 10;

```

```

vector<Student> hashTable(MAXN);
// Hàm băm chuỗi theo mã ASCII
int hashFunction(string name) {
    int value = 0;
    for (char c : name)
        value += c;
    return value % MAXN;
}
// Hàm chèn sinh viên vào bảng băm (Dò tuyến tính)
void insertStudent(string name, int id, string email, string phone) {
    int index = hashFunction(name);
    int originalIndex = index;
    while (hashTable[index].isOccupied) {
        index = (index + 1) % MAXN;
        if (index == originalIndex) {
            cout << "Bang bam day! Khong the chen them.\n";
            return;
        }
    }
    hashTable[index] = {name, email, phone, id, true};
}
// Hàm tìm kiếm sinh viên theo tên (Dò tuyến tính)
void searchStudent(string name) {
    int index = hashFunction(name);
    int originalIndex = index;
    while (hashTable[index].isOccupied) {
        if (hashTable[index].name == name) {
            cout << "Thong tin SV:\n";
            cout << "Ten: " << hashTable[index].name << "\n";
            cout << "MSSV: " << hashTable[index].studentID << "\n";
        }
        index = (index + 1) % MAXN;
    }
}

```

```

        cout << "Email: " << hashTable[index].email << "\n";

        cout << "SDT: " << hashTable[index].phone << "\n";

        return;
    }

    index = (index + 1) % MAXN;

    if (index == originalIndex) break;
}

cout << "Khong tim thay sinh vien.\n";
}

int main() {
    cout << "Nhap so luong sinh vien: ";

    int n; cin >> n;

    cin.ignore();

    for (int i = 0; i < n; i++) {
        string name, email, phone;

        int id;

        cout << "Nhap ten SV: ";

        getline(cin, name);

        cout << "Nhap MSSV: ";

        cin >> id;

        cin.ignore();

        cout << "Nhap email: ";

        getline(cin, email);

        cout << "Nhap so dien thoai: ";

        getline(cin, phone);

        insertStudent(name, id, email, phone);
    }

    string searchName;

    cout << "\nNhap ten SV can tim: ";

    getline(cin, searchName);

```

```
    searchStudent(searchName);  
  
    return 0;  
}
```

4. Áp dụng phương pháp địa chỉ mở (dò bậc 2) để giải quyết đụng độ cho bài toán tra tên SV.

```
#include <iostream>  
#include <vector>  
using namespace std;  
struct Student {  
    string name;  
    string email;  
    string phone;  
    int studentID;  
    bool isOccupied = false;  
};  
const int MAXN = 10;  
vector<Student> hashTable(MAXN);  
// Hàm băm chuỗi theo mã ASCII  
int hashFunction(string name) {  
    int value = 0;  
    for (char c : name) value += c;  
    return value % MAXN;  
}  
// Hàm chèn sinh viên vào bảng băm (Dò bậc 2)  
void insertStudent(string name, int id, string email, string phone) {  
    int index = hashFunction(name);  
    int i = 1; // Biến kiểm soát bước nhảy bậc 2  
    while (hashTable[index].isOccupied) {  
        index = (index + i * i) % MAXN;  
        i++;  
        if (i == MAXN) {
```

```

        cout << "Bang bam day! Khong the chen them.\n";

        return;

    }

}

hashTable[index] = {name, email, phone, id, true};
}

// Hàm tìm kiếm sinh viên theo tên (Dò bậc 2)
void searchStudent(string name) {
    int index = hashFunction(name);
    int i = 1;
    while (hashTable[index].isOccupied) {
        if (hashTable[index].name == name) {
            cout << "Thong tin SV:\n";
            cout << "Ten: " << hashTable[index].name << "\n";
            cout << "MSSV: " << hashTable[index].studentID << "\n";
            cout << "Email: " << hashTable[index].email << "\n";
            cout << "SDT: " << hashTable[index].phone << "\n";
            return;
        }
        index = (index + i * i) % MAXN;
        i++;
        if (i == MAXN) break;
    }
    cout << "Khong tim thay sinh vien.\n";
}

int main() {
    cout << "Nhap so luong sinh vien: ";
    int n; cin >> n;
    cin.ignore();
    for (int i = 0; i < n; i++) {

```

```

    string name, email, phone;

    int id;

    cout << "Nhap ten SV: ";

    getline(cin, name);

    cout << "Nhap MSSV: ";

    cin >> id;

    cin.ignore();

    cout << "Nhap email: ";

    getline(cin, email);

    cout << "Nhap so dien thoai: ";

    getline(cin, phone);

    insertStudent(name, id, email, phone);

}

string searchName;

cout << "\nNhap ten SV can tim: ";

getline(cin, searchName);

searchStudent(searchName);

return 0;

}

```

5. Áp dụng phương pháp địa chỉ mở (băm kép) để giải quyết đụng độ cho bài toán tra tên SV.

```

#include <iostream>

#include <vector>

using namespace std;

struct Student {

    string name;

    string email;

    string phone;

    int studentID;

    bool isOccupied = false;

};

```



```
const int MAXN = 11;

vector<Student> hashTable(MAXN);

// Hàm băm chính (chuyển tên thành chỉ số trong bảng băm)
int hash1(string name) {
    int value = 0;
    for (char c : name) value += c;
    return value % MAXN;
}

// Hàm băm phụ để tạo bước nhảy khi đụng độ
int hash2(string name) {
    int value = 0;
    for (char c : name)
        value = (value * 31 + c) % MAXN;
    return (value % (MAXN - 1)) + 1;
}

// Hàm chèn sinh viên vào bảng băm (Băm kép)
void insertStudent(string name, int id, string email, string phone) {
    int index = hash1(name);
    int step = hash2(name);
    int i = 0;
    while (hashTable[index].isOccupied) {
        index = (index + i * step) % MAXN;
        i++;
        if (i == MAXN) {
            cout << "Bang bam day! Khong the chen them.\n";
            return;
        }
    }
    hashTable[index] = {name, email, phone, id, true};
}
```

// Hàm tìm kiếm sinh viên theo tên (Băm kép)

```
void searchStudent(string name) {
    int index = hash1(name);
    int step = hash2(name);
    int i = 0;
    while (hashTable[index].isOccupied) {
        if (hashTable[index].name == name) {
            cout << "Thông tin SV:\n";
            cout << "Ten: " << hashTable[index].name << "\n";
            cout << "MSSV: " << hashTable[index].studentID << "\n";
            cout << "Email: " << hashTable[index].email << "\n";
            cout << "SDT: " << hashTable[index].phone << "\n";
            return;
        }
        index = (index + i * step) % MAXN;
        i++;
        if (i == MAXN) break;
    }
    cout << "Không tìm thấy sinh viên.\n";
}

int main() {
    cout << "Nhập số lượng sinh viên: ";
    int n; cin >> n;
    cin.ignore();
    for (int i = 0; i < n; i++) {
        string name, email, phone;
        int id;
        cout << "Nhập tên SV: ";
        getline(cin, name);
        cout << "Nhập MSSV: ";
```

```
    cin >> id;

    cin.ignore();

    cout << "Nhap email: ";

    getline(cin, email);

    cout << "Nhap so dien thoai: ";

    getline(cin, phone);

    insertStudent(name, id, email, phone);

}

string searchName;

cout << "\nNhap ten SV can tim: ";

getline(cin, searchName);

searchStudent(searchName);

return 0;

}
```