# **HÀM BĂM**

## **Hàm băm cơ bản**

### **1. Biên dịch và chạy chương trình trên.**

- Hash để đại diện cho hàm băm tính chỉ số vị trí (theo công thức k mod M), còn mảng U là bảng băm quản lý các khóa. Chương trình khởi tạo bảng băm với năm khóa {1, 2, 4, 6, 9}, yêu cầu người dùng nhập khóa cần tìm, sau đó tính vị trí bằng hàm băm và in ra giá trị khóa nếu tìm thấy hoặc thông báo "Khong tim thay khoa trong bang bam" nếu không có.

A screenshot of a computer

AI-generated content may be incorrect.

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

H(k) = k mod 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 có 10 phần tử.

Tập các khóa k lưu trong bảng băm có 5 phần tử.

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

**Thứ 1: Khởi tạo bảng băm:**

Trong hàm main, biến M được gán bằng 10.

Bộ nhớ cho mảng U được cấp phát với 10 phần tử (U[0] đến U[9]).

Hàm InitHash(U, M) được gọi để khởi tạo bảng băm.

**Thứ 2:** **Khởi tạo trong hàm InitHash**

**Khởi tạo mảng U:**

Duyệt qua các phần tử của U từ 0 đến M–1 và gán giá trị 0

**Chèn các khóa vào bảng băm:**

Một mảng tạm K gồm 5 giá trị {1, 2, 4, 6, 9}.

Với mỗi khóa k trong mảng K:

Tính chỉ số lưu trữ bằng cách gọi pos = Hash(k, M), tức là tính pos = k % M.

Gán giá trị của khóa vào ô U[pos] của bảng băm.

**Thứ 3: Tìm kiếm khóa:**

Sau khi bảng băm đã được khởi tạo, chương trình yêu cầu user nhập vào một khóa để tìm kiếm.

User nhập một giá trị (ví dụ: x).

Hàm Hash được gọi với khóa x và M để tính toán vị trí pos = x % M trong bảng băm.

Chương trình kiểm tra:

* Nếu ô U[pos] chứa giá trị 0, kết luận rằng bảng băm không chứa khóa cần tìm và in ra thông báo "Khong tim thay khoa trong bang bam".
* Nếu ô U[pos] khác 0, nghĩa là tại vị trí đó đã có một khóa được lưu trữ, chương trình in ra kết quả với thông báo "Gia tri phan tu can tim kiem: ...".

**Thứ 4: Kết thúc chương trình**

Sau khi in kết quả tìm kiếm, chương trình kết thúc và trả về 0.

## **Hàm băm cho bài toán tra tên sinh viên**

### **1. Biên dịch đoạn chương trình trên.**

    Word để đại diện cho các phần tử trong bảng băm và InitHash để quản lý bảng băm. Chương trình khởi tạo bảng băm với năm khóa và tên tương ứng, sau đó yêu cầu người dùng nhập một khóa để tìm kiếm và in ra giá trị tương ứng nếu tìm thấy hoặc thông báo không tìm thấy.

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

    1. Dạng lưu trữ

Chương trình cơ bản sử dụng mảng int \*U để lưu trữ các khóa dưới dạng số nguyên

Còn chương trình này sử dụng cấu trúc Word để lưu trữ cả khóa (key) với giá trị (value), và sử dụng mảng Word \*U để quản lý bảng băm.

   2. Hàm InitHash

Chương trình cơ bản sử dụng hàm InitHash khởi tạo bảng băm bằng cách gán các khóa số nguyên trực tiếp vào mảng U.

*void InitHash(int \*&U, int M)*

*{*

*int K[5] = {1, 2, 4, 6, 9};*

*int i, pos;*

*for (i = 0; i < M; i++)*

*U[i] = 0;*

*for (i = 0; i < 5; i++)*

*{*

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

*U[pos] = K[i];*

*}*

*}*

Còn chương trình này sử dụng hàm InitHash khởi tạo bảng băm bằng cách gán các đối tượng Word (bao gồm cả khóa và giá trị) vào mảng U.

*void InitHash(Word \*&U, int M)*

*{*

*Word K[5];*

*K[0].key = 1;*

*strcpy(K[0].value, "Messi");*

*K[1].key = 3;*

*strcpy(K[1].value, "Ronaldo");*

*K[2].key = 5;*

*strcpy(K[2].value, "Rooney");*

*K[3].key = 7;*

*strcpy(K[3].value, "Drogba");*

*K[4].key = 9;*

*strcpy(K[4].value, "Xavi");*

*int i, pos;*

*for (i = 0; i < M; i++)*

*U[i].key = 0;*

*for (i = 0; i < 5; i++)*

*{*

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

*U[pos] = K[i];*

*}*

*}*

   3. Hàm main

Chương trình cơ bản thì hàm main chỉ xử lý và in ra các khóa số nguyên.

Chương trình này thì xử lý cả khóa và giá trị tương ứng (tên sinh viên), và in ra tên sinh viên nếu tìm thấy khóa.

### **3. Nếu bỏ đoạn code sau trong hàm băm.**

**if (M == 0)**

**return 0;**

**Thì có được không? Giải thích lý do.**

    Không nên bỏ đoạn code trên. Vì đoạn if (M == 0) return 0; nhằm ngăn chặn việc thực hiện phép chia hoặc phép lấy dư với M = 0.

### **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.**

    Thì sẽ bị lỗi biên dịch trình biên dịch C++ sẽ không thể tự động chuyển đổi loại Word sang int. Do đó, sẽ phát sinh lỗi biên dịch tương tự như sau: error: cannot convert 'Word' to 'int' for argument '1' to 'int Hash(int, int)'. Vì không có phép chuyển đổi ngầm định**:** Trong C++, không có phép chuyển đổi ngầm định nào từ cấu trúc Word sang kiểu số nguyên int. Do đó, trình biên dịch không thể nhận dạng cách chuyển đổi này và sẽ báo lỗi.

### **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?**

    Kết quả xuất ra là Rooney.

Sai. Vì K[2].key = 5; Thành K[2].key = 13 là sẽ ghi đè lên Ronaldo vị trí 3 trong bảng băm bị ghi đè bởi khóa 13 và Ronaldo sẽ không tồn tại ở bảng băm.

### **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>

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, int N)

{

    Word \*K = **new** Word[N];

    for(int i = 0; i < N; i++)

    {

        printf("Nhap MSSV cho sinh vien %d: ", i+1);

        scanf("%d", &K[i].key);

        printf("Nhap ten cho sinh vien %d: ", i+1);

        scanf(" %127[^\n]", K[i].value);

    }

    for(int i = 0; i < M; i++)

        U[i].key = 0;

    for(int i = 0; i < N; i++)

    {

        int pos = Hash(K[i].key, M);

        U[pos] = K[i];

    }

    delete[] K;

}

int main(int argc, char \*argv[])

{

    int M = 10; // Kích thước bảng băm

    Word \*U = **new** Word[M];

    int N;

    printf("Nhap so luong sinh vien can them vao bang bam: ");

    scanf("%d", &N);

    // Khởi tạo bảng băm với N sinh viên

    InitHash(U, M, N);

    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

    {

        // Kiểm tra xem khóa có trùng với MSSV tìm kiếm hay không

        if (U[pos].key == x)

        {

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

        }

        else

        {

            // Nếu không trùng, thông báo không tìm thấy

            printf("Khong tim thay SV voi MSSV %d trong bang bam\n", x);

        }

    }

    // Giải phóng bộ nhớ đã cấp phát cho bảng băm

    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 <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Student {

    char name[128];

    char email[128];

    char phone[20];

    int MSSV;

};

int Hash(const char\* name, int M) {

    int hashValue = 0;

    for(int i = 0; name[i] != '\0'; i++) {

        hashValue += (int)name[i];

    }

    if (M == 0)

        return 0;

    return (hashValue % M);

}

void InitHash(Student \*&U, int M, int N) {

    for(int i = 0; i < M; i++) {

        U[i].MSSV = 0;

    }

    for(int i = 0; i < N; i++) {

        printf("Nhap thong tin cho sinh vien %d:\n", i+1);

        printf("Nhap ten: ");

        scanf(" %127[^\n]", U[i].name);

        printf("Nhap email: ");

        scanf(" %127[^\n]", U[i].email);

        printf("Nhap so dien thoai: ");

        scanf(" %19[^\n]", U[i].phone);

        printf("Nhap MSSV: ");

        scanf("%d", &U[i].MSSV);

        int pos = Hash(U[i].name, M);

        int originalPos = pos;

        int attempt = 0;

        while(U[pos].MSSV != 0 && strcmp(U[pos].name, U[i].name) != 0) {

            attempt++;

            pos = (Hash(U[i].name, M) + attempt) % M;

            if(pos == originalPos) {

                printf("Bang bam da day, khong the them sinh vien %d\n", i+1);

                break;

            }

        }

        if(U[pos].MSSV == 0 || strcmp(U[pos].name, U[i].name) == 0) {

            strcpy(U[pos].name, U[i].name);

            strcpy(U[pos].email, U[i].email);

            strcpy(U[pos].phone, U[i].phone);

            U[pos].MSSV = U[i].MSSV;

        }

    }

}

int main(int argc, char \*argv[]) {

    int M = 10;

    Student \*U = **new** Student[M];

    int N;

    printf("Nhap so luong sinh vien can them vao bang bam: ");

    scanf("%d", &N);

    InitHash(U, M, N);

    char searchName[128];

    printf("Nhap ten sinh vien can tim kiem: ");

    scanf(" %127[^\n]", searchName);

    int pos = Hash(searchName, M);

    int originalPos = pos;

    int attempt = 0;

    bool found = false;

    while(U[pos].MSSV != 0) {

        if(strcmp(U[pos].name, searchName) == 0) {

            found = true;

            break;

        }

        attempt++;

        pos = (Hash(searchName, M) + attempt) % M;

        if(pos == originalPos) {

            break;

        }

    }

    if(found) {

        printf("Thong tin sinh vien:\n");

        printf("Ten: %s\n", U[pos].name);

        printf("Email: %s\n", U[pos].email);

        printf("So dien thoai: %s\n", U[pos].phone);

        printf("MSSV: %d\n", U[pos].MSSV);

    }

    else {

        printf("Khong tim thay sinh vien voi ten \"%s\" trong bang bam\n", searchName);

    }

    delete[] U;

    return 0;

}

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

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Student {

    char name[128];

    char email[128];

    char phone[20];

    int MSSV;

};

struct Node {

    Student data;

    Node\* next;

};

int Hash(const char\* name, int M) {

    int hashValue = 0;

    for(int i = 0; name[i] != '\0'; i++) {

        hashValue += (int)name[i];

    }

    if (M == 0)

        return 0;

    return (hashValue % M);

}

void InitHash(Node\*\*& U, int M, int N) {

    for(int i = 0; i < M; i++) {

        U[i] = NULL;

    }

    for(int i = 0; i < N; i++) {

        Student s;

        printf("Nhap thong tin cho sinh vien %d:\n", i+1);

        printf("Nhap ten: ");

        scanf(" %127[^\n]", s.name);

        printf("Nhap email: ");

        scanf(" %127[^\n]", s.email);

        printf("Nhap so dien thoai: ");

        scanf(" %19[^\n]", s.phone);

        printf("Nhap MSSV: ");

        scanf("%d", &s.MSSV);

        int pos = Hash(s.name, M);

        Node\* newNode = **new** Node;

        newNode->data = s;

        newNode->next = NULL;

        if(U[pos] == NULL) {

            U[pos] = newNode;

        } else {

            newNode->next = U[pos];

            U[pos] = newNode;

        }

    }

}

bool FindStudent(Node\*\* U, int M, const char\* searchName, Student& result) {

    int pos = Hash(searchName, M);

    Node\* current = U[pos];

    while(current != NULL) {

        if(strcmp(current->data.name, searchName) == 0) {

            result = current->data;

            return true;

        }

        current = current->next;

    }

    return false;

}

void FreeHashTable(Node\*\*& U, int M) {

    for(int i = 0; i < M; i++) {

        Node\* current = U[i];

        while(current != NULL) {

            Node\* temp = current;

            current = current->next;

            delete temp;

        }

        U[i] = NULL;

    }

    delete[] U;

}

int main(int argc, char \*argv[]) {

    int M = 10;

    Node\*\* U = **new** Node\*[M];

    int N;

    printf("Nhap so luong sinh vien can them vao bang bam: ");

    scanf("%d", &N);

    InitHash(U, M, N);

    char searchName[128];

    printf("Nhap ten sinh vien can tim kiem: ");

    scanf(" %127[^\n]", searchName);

    Student foundStudent;

    bool found = FindStudent(U, M, searchName, foundStudent);

    if(found) {

        printf("Thong tin sinh vien\n");

        printf("Ten: %s\n", foundStudent.name);

        printf("Email: %s\n", foundStudent.email);

        printf("So dien thoai: %s\n", foundStudent.phone);

        printf("MSSV: %d\n", foundStudent.MSSV);

    }

    else {

        printf("Khong tim thay sinh vien voi ten \"%s\" trong bang bam\n", searchName);

    }

    FreeHashTable(U, M);

    return 0;

}

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

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Student {

    char name[128];

    char email[128];

    char phone[20];

    int MSSV;

};

int Hash(const char\* name, int M) {

    int hashValue = 0;

    for(int i = 0; name[i] != '\0'; i++) {

        hashValue += (int)name[i];

    }

    if (M == 0)

        return 0;

    return (hashValue % M);

}

void InitHash(Student \*U, int M, int N) {

    for(int i = 0; i < M; i++) {

        U[i].MSSV = 0;

    }

    for(int i = 0; i < N; i++) {

        Student s;

        printf("Nhap thong tin cho sinh vien %d:\n", i+1);

        printf("Nhap ten: ");

        scanf(" %127[^\n]", s.name);

        printf("Nhap email: ");

        scanf(" %127[^\n]", s.email);

        printf("Nhap so dien thoai: ");

        scanf(" %19[^\n]", s.phone);

        printf("Nhap MSSV: ");

        scanf("%d", &s.MSSV);

        int pos = Hash(s.name, M);

        int originalPos = pos;

        int attempt = 0;

        while(U[pos].MSSV != 0 && strcmp(U[pos].name, s.name) != 0) {

            attempt++;

            pos = (originalPos + attempt) % M;

            if(pos == originalPos) {

                printf("Bang bam da day, khong the them sinh vien %d\n", i+1);

                break;

            }

        }

        if(U[pos].MSSV == 0 || strcmp(U[pos].name, s.name) == 0) {

            strcpy(U[pos].name, s.name);

            strcpy(U[pos].email, s.email);

            strcpy(U[pos].phone, s.phone);

            U[pos].MSSV = s.MSSV;

        }

    }

}

bool FindStudent(Student \*U, int M, const char\* searchName, Student& result) {

    int pos = Hash(searchName, M);

    int originalPos = pos;

    int attempt = 0;

    while(U[pos].MSSV != 0) {

        if(strcmp(U[pos].name, searchName) == 0) {

            result = U[pos];

            return true;

        }

        attempt++;

        pos = (originalPos + attempt) % M;

        if(pos == originalPos) {

            break;

        }

    }

    return false;

}

int main(int argc, char \*argv[]) {

    int M = 10;

    Student \*U = **new** Student[M];

    int N;

    printf("Nhap so luong sinh vien can them vao bang bam: ");

    scanf("%d", &N);

    InitHash(U, M, N);

    char searchName[128];

    printf("Nhap ten sinh vien can tim kiem: ");

    scanf(" %127[^\n]", searchName);

    Student foundStudent;

    bool found = FindStudent(U, M, searchName, foundStudent);

    if(found) {

        printf("Thong tin sinh vien:\n");

        printf("Ten: %s\n", foundStudent.name);

        printf("Email: %s\n", foundStudent.email);

        printf("So dien thoai: %s\n", foundStudent.phone);

        printf("MSSV: %d\n", foundStudent.MSSV);

    }

    else {

        printf("Khong tim thay sinh vien voi ten \"%s\" trong bang bam\n", searchName);

    }

    delete[] U;

    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 <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Student {

    char name[128];

    char email[128];

    char phone[20];

    int MSSV;

};

int Hash(const char\* name, int M) {

    int hashValue = 0;

    for(int i = 0; name[i] != '\0'; i++) {

        hashValue += (int)name[i];

    }

    if (M == 0)

        return 0;

    return (hashValue % M);

}

void InitHash(Student \*U, int M, int N) {

    for(int i = 0; i < M; i++) {

        U[i].MSSV = 0;

    }

    for(int i = 0; i < N; i++) {

        Student s;

        printf("Nhap thong tin cho sinh vien %d:\n", i+1);

        printf("Nhap ten: ");

        scanf(" %127[^\n]", s.name);

        printf("Nhap email: ");

        scanf(" %127[^\n]", s.email);

        printf("Nhap so dien thoai: ");

        scanf(" %19[^\n]", s.phone);

        printf("Nhap MSSV: ");

        scanf("%d", &s.MSSV);

        int pos = Hash(s.name, M);

        int originalPos = pos;

        int attempt = 1;

        while(U[pos].MSSV != 0 && strcmp(U[pos].name, s.name) != 0) {

            pos = (originalPos + attempt \* attempt) % M;

            if(pos == originalPos) {

                printf("Bang bam da day, khong the them sinh vien %d\n", i+1);

                break;

            }

            attempt++;

        }

        if(U[pos].MSSV == 0 || strcmp(U[pos].name, s.name) == 0) {

            strcpy(U[pos].name, s.name);

            strcpy(U[pos].email, s.email);

            strcpy(U[pos].phone, s.phone);

            U[pos].MSSV = s.MSSV;

        }

    }

}

bool FindStudent(Student \*U, int M, const char\* searchName, Student& result) {

    int pos = Hash(searchName, M);

    int originalPos = pos;

    int attempt = 1;

    while(U[pos].MSSV != 0) {

        if(strcmp(U[pos].name, searchName) == 0) {

            result = U[pos];

            return true;

        }

        pos = (originalPos + attempt \* attempt) % M;

        if(pos == originalPos) {

            break;

        }

        attempt++;

    }

    return false;

}

int main(int argc, char \*argv[]) {

    int M = 10;

    Student \*U = **new** Student[M];

    int N;

    printf("Nhap so luong sinh vien can them vao bang bam: ");

    scanf("%d", &N);

    InitHash(U, M, N);

    char searchName[128];

    printf("Nhap ten sinh vien can tim kiem: ");

    scanf(" %127[^\n]", searchName);

    Student foundStudent;

    bool found = FindStudent(U, M, searchName, foundStudent);

    if(found) {

        printf("Thong tin sinh vien:\n");

        printf("Ten: %s\n", foundStudent.name);

        printf("Email: %s\n", foundStudent.email);

        printf("So dien thoai: %s\n", foundStudent.phone);

        printf("MSSV: %d\n", foundStudent.MSSV);

    }

    else {

        printf("Khong tim thay sinh vien voi ten \"%s\" trong bang bam\n", searchName);

    }

    delete[] U;

    return 0;

}

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

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Student {

    char name[128];

    char email[128];

    char phone[20];

    int MSSV;

};

int Hash1(const char\* name, int M) {

    int hashValue = 0;

    for(int i = 0; name[i] != '\0'; i++) {

        hashValue += (int)name[i];

    }

    if (M == 0)

        return 0;

    return (hashValue % M);

}

int Hash2(const char\* name, int M) {

    int hashValue = 0;

    for(int i = 0; name[i] != '\0'; i++) {

        hashValue += (int)name[i] \* (i + 1);

    }

    return (1 + (hashValue % (M - 1)));

}

void InitHash(Student \*U, int M, int N) {

    for(int i = 0; i < M; i++) {

        U[i].MSSV = 0;

    }

    for(int i = 0; i < N; i++) {

        Student s;

        printf("Nhap thong tin cho sinh vien %d:\n", i+1);

        printf("Nhap ten: ");

        scanf(" %127[^\n]", s.name);

        printf("Nhap email: ");

        scanf(" %127[^\n]", s.email);

        printf("Nhap so dien thoai: ");

        scanf(" %19[^\n]", s.phone);

        printf("Nhap MSSV: ");

        scanf("%d", &s.MSSV);

        int h1 = Hash1(s.name, M);

        int h2 = Hash2(s.name, M);

        int pos = h1;

        int attempt = 0;

        while(U[pos].MSSV != 0 && strcmp(U[pos].name, s.name) != 0) {

            attempt++;

            pos = (h1 + attempt \* h2) % M;

            if(pos == h1) {

                printf("Bang bam da day, khong the them sinh vien %d\n", i+1);

                break;

            }

        }

        if(U[pos].MSSV == 0 || strcmp(U[pos].name, s.name) == 0) {

            strcpy(U[pos].name, s.name);

            strcpy(U[pos].email, s.email);

            strcpy(U[pos].phone, s.phone);

            U[pos].MSSV = s.MSSV;

        }

    }

}

bool FindStudent(Student \*U, int M, const char\* searchName, Student& result) {

    int h1 = Hash1(searchName, M);

    int h2 = Hash2(searchName, M);

    int pos = h1;

    int attempt = 0;

    while(U[pos].MSSV != 0) {

        if(strcmp(U[pos].name, searchName) == 0) {

            result = U[pos];

            return true;

        }

        attempt++;

        pos = (h1 + attempt \* h2) % M;

        if(pos == h1) {

            break;

        }

    }

    return false;

}

int main(int argc, char \*argv[]) {

    int M = 10;

    Student \*U = **new** Student[M];

    int N;

    printf("Nhap so luong sinh vien can them vao bang bam: ");

    scanf("%d", &N);

    InitHash(U, M, N);

    char searchName[128];

    printf("Nhap ten sinh vien can tim kiem: ");

    scanf(" %127[^\n]", searchName);

    Student foundStudent;

    bool found = FindStudent(U, M, searchName, foundStudent);

    if(found) {

        printf("Thong tin sinh vien:\n");

        printf("Ten: %s\n", foundStudent.name);

        printf("Email: %s\n", foundStudent.email);

        printf("So dien thoai: %s\n", foundStudent.phone);

        printf("MSSV: %d\n", foundStudent.MSSV);

    }

    else {

        printf("Khong tim thay sinh vien voi ten \"%s\" trong bang bam\n", searchName);

    }

    delete[] U;

    return 0;

}