**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**



**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**ĐỀ TÀI: CÀI ĐẶT HÀNG ĐỢI**

**Giáo viên hướng dẫn: Trần Quý Nam**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Stt** | **Mã sv** | **Họ và tên** | **Lớp** |
| 01 | 1871020622 | Nguyễn Văn Tuấn | CNTT18-02 |
|  |  |  |  |
|  |  |  |  |

**Hà Nội, năm 2024**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**



**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**ĐỀ TÀI: CÀI ĐẶT HÀNG ĐỢI**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| **1** | 1871020622 | Nguyễn Văn Tuấn | 08/07/2006 |  |  |
| **2** |  |  |  |  |  |
| **3** |  |  |  |  |  |

|  |  |
| --- | --- |
| **CÁN BỘ CHẤM THI 1**  **Trần Quý Nam** | **CÁN BỘ CHẤM THI 2** |

**CHƯƠNG 1 : CƠ BẢN VỀ HÀNG ĐỢI**

**1.1:cài đặt hàng số đợi số nguyên bằng mảng**

**a. Hàng đợi là gì?**

Hàng đợi (Queue) là một cấu trúc dữ liệu hoạt động theo nguyên tắc **FIFO** (First In, First Out) – nghĩa là phần tử nào vào trước thì sẽ được lấy ra trước.

Ví dụ:

* Xếp hàng mua vé xem phim: Người nào vào hàng trước sẽ được phục vụ trước.
* Hệ thống xử lý lệnh in: Lệnh nào được gửi trước thì sẽ in trước.

**b. Cài đặt Hàng đợi bằng Mảng**

**Cách hoạt động**

* Dùng một mảng cố định có kích thước MAX.
* Dùng hai biến front (đầu hàng đợi) và rear (cuối hàng đợi) để quản lý.
* Khi thêm (enqueue), phần tử sẽ được đưa vào cuối hàng đợi.
* Khi xóa (dequeue), phần tử ở đầu hàng đợi sẽ bị loại bỏ.

**c.Hàng đợi (Queue) sử dụng mảng có các thành phần:**

* front: Chỉ số phần tử đầu hàng đợi.
* rear: Chỉ số phần tử cuối hàng đợi.
* enqueue(value): Thêm phần tử vào hàng đợi.
* dequeue(): Xóa phần tử đầu tiên.
* display(): Duyệt và in hàng đợi.

Code Hàng đợi bằng Mảng

#include <stdio.h>

#include <stdlib.h>

#define MAX 5 // Kích thước tối đa của hàng đợi

typedef struct {

int data[MAX];

int front, rear;

} Queue;

// Khởi tạo hàng đợi

void initQueue(Queue \*q) {

q->front = -1;

q->rear = -1;

}

// Kiểm tra hàng đợi có rỗng không

int isEmpty(Queue \*q) {

return (q->front == -1);

}

// Kiểm tra hàng đợi có đầy không

int isFull(Queue \*q) {

return (q->rear == MAX - 1);

}

// Thêm phần tử vào hàng đợi

void enqueue(Queue \*q, int value) {

if (isFull(q)) {

printf("Hàng đợi đầy!\n");

return;

}

if (isEmpty(q)) {

q->front = 0;

}

q->data[++q->rear] = value;

printf("Đã thêm %d vào hàng đợi.\n", value);

}

// Xóa phần tử khỏi hàng đợi

void dequeue(Queue \*q) {

if (isEmpty(q)) {

printf("Hàng đợi rỗng!\n");

return;

}

printf("Đã xóa %d khỏi hàng đợi.\n", q->data[q->front]);

if (q->front == q->rear) {

initQueue(q); // Reset nếu hàng đợi trống

} else {

q->front++;

}

}

// Hiển thị hàng đợi

void display(Queue \*q) {

if (isEmpty(q)) {

printf("Hàng đợi rỗng!\n");

return;

}

printf("Hàng đợi: ");

for (int i = q->front; i <= q->rear; i++) {

printf("%d ", q->data[i]);

}

printf("\n");

}

// Hàm main để kiểm tra

int main() {

Queue q;

initQueue(&q);

enqueue(&q, 10);

enqueue(&q, 20);

enqueue(&q, 30);

display(&q);

dequeue(&q);

display(&q);

return 0; }

**Chương 2: Mở rộng với Hàng đợi Chuỗi**

### **2.1. Cấu trúc và nguyên lý hoạt động**

* **Dùng mảng** để lưu trữ các chuỗi.
* **Hai biến front và rear** để theo dõi vị trí đầu và cuối của hàng đợi.
* **Kích thước cố định (MAX)** để tránh tràn bộ nhớ.
* **Các thao tác chính**:
  + **enqueue**: Thêm một chuỗi vào cuối hàng đợi.
  + **dequeue**: Xóa chuỗi đầu tiên trong hàng đợi.
  + **display**: In tất cả các chuỗi trong hàng đợi.

**2.2** Cài đặt Hàng đợi Chuỗi bằng Mảng

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#define MAX 5 // Kích thước tối đa của hàng đợi**

**#define STR\_SIZE 100 // Độ dài tối đa của mỗi chuỗi**

**typedef struct {**

**char data[MAX][STR\_SIZE]; // Mảng chứa chuỗi**

**int front, rear;**

**} Queue;**

**// Khởi tạo hàng đợi**

**void initQueue(Queue \*q) {**

**q->front = -1;**

**q->rear = -1;**

**}**

**// Kiểm tra hàng đợi có rỗng không**

**int isEmpty(Queue \*q) {**

**return (q->front == -1);**

**}**

**// Kiểm tra hàng đợi có đầy không**

**int isFull(Queue \*q) {**

**return (q->rear == MAX - 1);**

**}**

**// Thêm chuỗi vào hàng đợi**

**void enqueue(Queue \*q, char \*str) {**

**if (isFull(q)) {**

**printf("Hàng đợi đầy! Không thể thêm.\n");**

**return;**

**}**

**if (isEmpty(q)) {**

**q->front = 0;**

**}**

**q->rear++;**

**strcpy(q->data[q->rear], str);**

**printf("Đã thêm \"%s\" vào hàng đợi.\n", str);**

**}**

**// Xóa chuỗi khỏi hàng đợi**

**void dequeue(Queue \*q) {**

**if (isEmpty(q)) {**

**printf("Hàng đợi rỗng! Không thể xóa.\n");**

**return;**

**}**

**printf("Đã xóa \"%s\" khỏi hàng đợi.\n", q->data[q->front]);**

**if (q->front == q->rear) {**

**initQueue(q); // Reset nếu hàng đợi trống**

**} else {**

**q->front++;**

**}**

**}**

**// Hiển thị hàng đợi**

**void display(Queue \*q) {**

**if (isEmpty(q)) {**

**printf("Hàng đợi rỗng!\n");**

**return;**

**}**

**printf("Hàng đợi: ");**

**for (int i = q->front; i <= q->rear; i++) {**

**printf("\"%s\" ", q->data[i]);**

**}**

**printf("\n");**

**}**

**// Hàm main để kiểm tra**

**int main() {**

**Queue q;**

**initQueue(&q);**

**enqueue(&q, "Alice");**

**enqueue(&q, "Bob");**

**enqueue(&q, "Charlie");**

**display(&q);**

**dequeue(&q);**

**display(&q);**

**enqueue(&q, "David");**

**enqueue(&q, "Eve");**

**display(&q);**

**return 0;**

**}**

### **2.3. Giải thích Code**

* + 1. **Cấu trúc Queue chứa:**
  + data[MAX][STR\_SIZE]: Mảng lưu các chuỗi.
  + front: Vị trí phần tử đầu hàng đợi.
  + rear: Vị trí phần tử cuối hàng đợi.

**2.1.2 Các thao tác chính:**

* + **enqueue():** Kiểm tra đầy → Nếu chưa đầy, thêm chuỗi vào cuối.
  + **dequeue():** Kiểm tra rỗng → Nếu không rỗng, xóa phần tử đầu.
  + **display():** Duyệt từ front đến rear và in các chuỗi.

### **2.4. Kết quả Chạy Chương Trình**

css

Sao chépChỉnh sửa

Đã thêm "Alice" vào hàng đợi.

Đã thêm "Bob" vào hàng đợi.

Đã thêm "Charlie" vào hàng đợi.

Hàng đợi: "Alice" "Bob" "Charlie"

Đã xóa "Alice" khỏi hàng đợi.

Hàng đợi: "Bob" "Charlie"

Đã thêm "David" vào hàng đợi.

Đã thêm "Eve" vào hàng đợi.

Hàng đợi: "Bob" "Charlie" "David" "Eve"

### **2.5. Đánh giá**

✅ **Ưu điểm:**

* Dễ hiểu, dễ triển khai.
* Truy cập nhanh vì dùng mảng (O(1)).

⚠️ **Nhược điểm:**

* Kích thước cố định (MAX = 5), nếu đầy không thể thêm.
* Khi xóa nhiều phần tử, khoảng trống ở đầu không thể tái sử dụng.

💡 **Giải pháp:** Nếu muốn tối ưu bộ nhớ, có thể **dùng danh sách liên kết** thay vì mảng.

Bạn muốn tiếp tục với hàng đợi chuỗi bằng **danh sách liên kết** không? 🚀

**2.6 Cài đặt hàng đợi chuỗi bằng danh sách liên kết.**

### ***2.6.1. Nguyên lý hoạt động***

* Sử dụng danh sách liên kết để lưu trữ chuỗi.
* Dùng hai con trỏ:
  + front: Trỏ đến phần tử đầu hàng đợi.
  + rear: Trỏ đến phần tử cuối hàng đợi.
* Khi thêm (enqueue): Tạo một **node mới** và đặt nó ở cuối.
* Khi xóa (dequeue): Xóa node ở đầu.

### ***2.6.2. Cài đặt Hàng đợi Chuỗi bằng Danh sách Liên kết***

c

Sao chépChỉnh sửa

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Cấu trúc một node trong danh sách liên kết

typedef struct Node {

char data[100]; // Chuỗi dữ liệu

struct Node \*next;

} Node;

// Cấu trúc hàng đợi

typedef struct {

Node \*front, \*rear;

} Queue;

// Hàm khởi tạo hàng đợi

void initQueue(Queue \*q) {

q->front = NULL;

q->rear = NULL;

}

// Kiểm tra hàng đợi có rỗng không

int isEmpty(Queue \*q) {

return (q->front == NULL);

}

// Hàm thêm một chuỗi vào hàng đợi (enqueue)

void enqueue(Queue \*q, char \*str) {

Node \*newNode = (Node \*)malloc(sizeof(Node));

if (!newNode) {

printf("Lỗi cấp phát bộ nhớ!\n");

return;

}

strcpy(newNode->data, str);

newNode->next = NULL;

if (isEmpty(q)) {

q->front = q->rear = newNode;

} else {

q->rear->next = newNode;

q->rear = newNode;

}

printf("Đã thêm \"%s\" vào hàng đợi.\n", str);

}

// Hàm xóa một chuỗi khỏi hàng đợi (dequeue)

void dequeue(Queue \*q) {

if (isEmpty(q)) {

printf("Hàng đợi rỗng! Không thể xóa.\n");

return;

}

Node \*temp = q->front;

printf("Đã xóa \"%s\" khỏi hàng đợi.\n", temp->data);

q->front = q->front->next;

if (q->front == NULL) {

q->rear = NULL;

}

free(temp);

}

// Hàm hiển thị hàng đợi

void display(Queue \*q) {

if (isEmpty(q)) {

printf("Hàng đợi rỗng!\n");

return;

}

Node \*temp = q->front;

printf("Hàng đợi: ");

while (temp) {

printf("\"%s\" ", temp->data);

temp = temp->next;

}

printf("\n");

}

// Hàm main để kiểm tra chương trình

int main() {

Queue q;

initQueue(&q);

enqueue(&q, "Alice");

enqueue(&q, "Bob");

enqueue(&q, "Charlie");

display(&q);

dequeue(&q);

display(&q);

enqueue(&q, "David");

enqueue(&q, "Eve");

display(&q);

return 0;

}

### ***2.6.3. Giải thích Code***

1. **Cấu trúc Node**:
   * data[100]: Lưu trữ chuỗi.
   * next: Con trỏ trỏ đến phần tử tiếp theo.
2. **Cấu trúc Queue**:
   * front: Con trỏ đến phần tử đầu hàng đợi.
   * rear: Con trỏ đến phần tử cuối hàng đợi.
3. **Các thao tác chính**:
   * **enqueue()**:
     + Cấp phát động một **node mới** chứa chuỗi.
     + Thêm vào cuối danh sách liên kết.
   * **dequeue()**:
     + Xóa phần tử đầu danh sách.
     + Nếu hàng đợi trống sau khi xóa, đặt rear = NULL.
   * **display()**:
     + Duyệt danh sách liên kết và in các phần tử.

### ***2.6.4. Kết quả Chạy Chương Trình***

css

Sao chépChỉnh sửa

Đã thêm "Alice" vào hàng đợi.

Đã thêm "Bob" vào hàng đợi.

Đã thêm "Charlie" vào hàng đợi.

Hàng đợi: "Alice" "Bob" "Charlie"

Đã xóa "Alice" khỏi hàng đợi.

Hàng đợi: "Bob" "Charlie"

Đã thêm "David" vào hàng đợi.

Đã thêm "Eve" vào hàng đợi.

Hàng đợi: "Bob" "Charlie" "David" "Eve"

### ***2.6.5. Đánh giá***

✅ **Ưu điểm**:

* Không giới hạn kích thước (dùng bộ nhớ động).
* Không lãng phí ô nhớ như mảng.

⚠️ **Nhược điểm**:

* Cần cấp phát động và giải phóng bộ nhớ.
* Truy cập chậm hơn so với mảng (O(n)).

💡 **Bạn muốn cải tiến chương trình bằng cách nào?**

* Thêm chức năng **đếm số phần tử**?
* Tạo menu nhập từ bàn phím? 🚀

***2.7Các thao tác cơ bản: thêm, xóa, duyệt.***

### **a. Thêm (enqueue)**

* Tạo một **node mới** chứa chuỗi.
* Nếu hàng đợi **rỗng** → front và rear đều trỏ đến node mới.
* Nếu hàng đợi **không rỗng** → Gán next của rear vào node mới, cập nhật rear.

**Cài đặt:**

c

Sao chépChỉnh sửa

void enqueue(Queue \*q, char \*str) {

Node \*newNode = (Node \*)malloc(sizeof(Node));

if (!newNode) {

printf("Lỗi cấp phát bộ nhớ!\n");

return;

}

strcpy(newNode->data, str);

newNode->next = NULL;

if (isEmpty(q)) {

q->front = q->rear = newNode;

} else {

q->rear->next = newNode;

q->rear = newNode;

}

printf("Đã thêm \"%s\" vào hàng đợi.\n", str);

}

### b. **Xóa (dequeue)**

* Nếu hàng đợi **rỗng** → Không thể xóa.
* Nếu **chỉ có một phần tử** → Xóa phần tử và đặt front = rear = NULL.
* Nếu **có nhiều phần tử** → Di chuyển front sang phần tử kế tiếp, giải phóng bộ nhớ.

**Cài đặt:**

c

Sao chépChỉnh sửa

void dequeue(Queue \*q) {

if (isEmpty(q)) {

printf("Hàng đợi rỗng! Không thể xóa.\n");

return;

}

Node \*temp = q->front;

printf("Đã xóa \"%s\" khỏi hàng đợi.\n", temp->data);

q->front = q->front->next;

if (q->front == NULL) {

q->rear = NULL;

}

free(temp);

}

### **c. Duyệt (display)**

* Nếu hàng đợi **rỗng** → Thông báo.
* Duyệt từ front đến rear và in từng phần tử.

**Cài đặt:**

c

Sao chépChỉnh sửa

void display(Queue \*q) {

if (isEmpty(q)) {

printf("Hàng đợi rỗng!\n");

return;

}

Node \*temp = q->front;

printf("Hàng đợi: ");

while (temp) {

printf("\"%s\" ", temp->data);

temp = temp->next;

}

printf("\n");

}

### **d. Kết quả Chạy Chương Trình**

c

Sao chépChỉnh sửa

Queue q;

initQueue(&q);

enqueue(&q, "Alice");

enqueue(&q, "Bob");

enqueue(&q, "Charlie");

display(&q);

dequeue(&q);

display(&q);

enqueue(&q, "David");

enqueue(&q, "Eve");

display(&q);

🔹 **Output:**

css

Sao chépChỉnh sửa

Đã thêm "Alice" vào hàng đợi.

Đã thêm "Bob" vào hàng đợi.

Đã thêm "Charlie" vào hàng đợi.

Hàng đợi: "Alice" "Bob" "Charlie"

Đã xóa "Alice" khỏi hàng đợi.

Hàng đợi: "Bob" "Charlie"

Đã thêm "David" vào hàng đợi.

Đã thêm "Eve" vào hàng đợi.

Hàng đợi: "Bob" "Charlie" "David" "Eve"

💡 **Bạn muốn bổ sung chức năng nào khác?**

* **Kiểm tra kích thước hàng đợi?**
* **Xóa toàn bộ hàng đợi?** 🚀

4o

**CHƯƠNG 3 : ỨNG DỤNG HÀNG ĐỢI VỚI SINH VIÊN**

## **Ứng dụng Hàng đợi với Sinh viên**

Hàng đợi sinh viên không chỉ giúp quản lý dữ liệu theo nguyên tắc FIFO (First In, First Out) mà còn có nhiều ứng dụng thực tế trong giáo dục, quản lý và các hệ thống tự động. Dưới đây là một số ứng dụng phổ biến của hàng đợi trong bối cảnh sinh viên.

## ***3.1. Quản lý Đăng ký Môn học***

### **Mô tả**

* Khi sinh viên đăng ký môn học, họ sẽ được xếp vào hàng đợi dựa trên thời gian đăng ký.
* Người đăng ký trước được xử lý trước.
* Nếu môn học đã đủ số lượng sinh viên tối đa, những người đăng ký sau sẽ bị đưa vào danh sách chờ.

### **Cài đặt**

**Mỗi sinh viên bao gồm:**

* **Mã sinh viên**
* **Họ tên**
* **Môn học đăng ký**
* **Trạng thái (Đã đăng ký / Chờ đợi)**

c

Sao chépChỉnh sửa

typedef struct {

char studentID[15];

char name[50];

char course[50];

char status[20]; // "Đã đăng ký" hoặc "Chờ đợi"

struct Student \*next;

} Student;

**Cơ chế hoạt động:**

* Khi sinh viên đăng ký môn học, họ sẽ được thêm vào hàng đợi.
* Khi có chỗ trống, sinh viên đầu hàng đợi sẽ được chuyển từ trạng thái **"Chờ đợi" → "Đã đăng ký"**.

c

Sao chépChỉnh sửa

void processRegistration(Queue \*q, int maxStudents) {

int count = 0;

Student \*temp = q->front;

while (temp) {

if (count < maxStudents) {

strcpy(temp->status, "Đã đăng ký");

} else {

strcpy(temp->status, "Chờ đợi");

}

count++;

temp = temp->next;

}

}

## ***3.2. Hệ thống Xếp hàng tại Phòng Đào tạo***

### **Mô tả**

* Khi sinh viên đến phòng đào tạo để giải quyết thủ tục (ví dụ: xin bảng điểm, đăng ký học lại, đổi lớp...), họ sẽ lấy số thứ tự và xếp hàng.
* Hệ thống sẽ phục vụ sinh viên theo thứ tự đến trước.

### **Cài đặt**

**Mỗi sinh viên bao gồm:**

* **Mã số sinh viên**
* **Họ tên**
* **Lý do đến phòng đào tạo**
* **Số thứ tự**

c

Sao chépChỉnh sửa

typedef struct {

char studentID[15];

char name[50];

char request[100];

int queueNumber;

struct Student \*next;

} Student;

**Cơ chế hoạt động:**

* Khi một sinh viên đến, họ nhận được một số thứ tự và được đưa vào hàng đợi.
* Khi đến lượt, sinh viên đầu hàng đợi sẽ được phục vụ và xóa khỏi danh sách.

c

Sao chépChỉnh sửa

void serveStudent(Queue \*q) {

if (isEmpty(q)) {

printf("Không có sinh viên nào đang chờ!\n");

return;

}

Student \*temp = q->front;

printf("Phục vụ sinh viên: %s | Lý do: %s | Số thứ tự: %d\n",

temp->name, temp->request, temp->queueNumber);

dequeue(q);

}

## ***3.3. Hệ thống Gọi Sinh viên Phỏng vấn***

### **Mô tả**

* Khi có một đợt phỏng vấn việc làm hoặc xét học bổng, sinh viên sẽ đăng ký và được gọi theo thứ tự đăng ký.
* Hệ thống đảm bảo không có sinh viên nào bị bỏ sót.

### **Cài đặt**

**Mỗi sinh viên bao gồm:**

* **Mã sinh viên**
* **Họ tên**
* **Vị trí trong danh sách phỏng vấn**

c

Sao chépChỉnh sửa

typedef struct {

char studentID[15];

char name[50];

int interviewPosition;

struct Student \*next;

} Student;

**Cơ chế hoạt động:**

* Khi sinh viên đến đăng ký, họ sẽ được thêm vào hàng đợi.
* Khi đến lượt, sinh viên đầu hàng đợi sẽ được gọi vào phỏng vấn.

c

Sao chépChỉnh sửa

void callForInterview(Queue \*q) {

if (isEmpty(q)) {

printf("Không có sinh viên nào đang chờ phỏng vấn!\n");

return;

}

Student \*temp = q->front;

printf("Gọi sinh viên vào phỏng vấn: %s | Vị trí: %d\n",

temp->name, temp->interviewPosition);

dequeue(q);

}

## ***3.4. Xếp hàng Nhận Học bổng***

### **Mô tả**

* Sinh viên đăng ký nhận học bổng sẽ được xét duyệt theo thứ tự.
* Khi có suất học bổng, sinh viên đầu tiên trong hàng đợi sẽ được xét trước.

### **Cài đặt**

**Mỗi sinh viên bao gồm:**

* **Mã số sinh viên**
* **Họ tên**
* **Điểm trung bình**
* **Trạng thái (Đã nhận học bổng / Đang xét duyệt)**

c

Sao chépChỉnh sửa

typedef struct {

char studentID[15];

char name[50];

float gpa;

char status[20]; // "Đã nhận học bổng" hoặc "Đang xét duyệt"

struct Student \*next;

} Student;

**Cơ chế hoạt động:**

* Nếu sinh viên có GPA cao và có suất học bổng, họ sẽ được xét duyệt trước.
* Nếu suất học bổng hết, những sinh viên còn lại trong hàng đợi sẽ tiếp tục chờ đợt sau.

c

Sao chépChỉnh sửa

void processScholarship(Queue \*q, int availableScholarships) {

int count = 0;

Student \*temp = q->front;

while (temp && count < availableScholarships) {

strcpy(temp->status, "Đã nhận học bổng");

printf("Sinh viên %s đã nhận học bổng.\n", temp->name);

count++;

temp = temp->next;

}

while (temp) {

strcpy(temp->status, "Đang xét duyệt");

temp = temp->next;

}

}

*3.5 Cài đặt hàng đợi sinh viên bằng danh sách liên kết.*

## ***3.5.1. Định nghĩa cấu trúc Sinh viên và Hàng đợi***

c

Sao chépChỉnh sửa

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Định nghĩa cấu trúc Sinh viên

typedef struct Student {

char studentID[15]; // Mã số sinh viên

char name[50]; // Họ tên sinh viên

int birthYear; // Năm sinh

float gpa; // Điểm trung bình

struct Student \*next;

} Student;

// Định nghĩa cấu trúc Hàng đợi

typedef struct {

Student \*front;

Student \*rear;

} Queue;

## ***3.5.2. Hàm Khởi tạo Hàng đợi***

c

Sao chépChỉnh sửa

// Khởi tạo hàng đợi rỗng

void initializeQueue(Queue \*q) {

q->front = q->rear = NULL;

}

## ***3.5.3. Hàm Thêm Sinh viên vào Hàng đợi (Enqueue)***

c

Sao chépChỉnh sửa

// Thêm sinh viên vào cuối hàng đợi

void enqueue(Queue \*q, char studentID[], char name[], int birthYear, float gpa) {

Student \*newStudent = (Student \*)malloc(sizeof(Student));

if (!newStudent) {

printf("Lỗi cấp phát bộ nhớ!\n");

return;

}

strcpy(newStudent->studentID, studentID);

strcpy(newStudent->name, name);

newStudent->birthYear = birthYear;

newStudent->gpa = gpa;

newStudent->next = NULL;

if (q->rear == NULL) {

q->front = q->rear = newStudent;

} else {

q->rear->next = newStudent;

q->rear = newStudent;

}

printf("Đã thêm sinh viên: %s - %s\n", studentID, name);

}

## ***3.5.4. Hàm Xóa Sinh viên khỏi Hàng đợi (Dequeue)***

c

Sao chépChỉnh sửa

// Xóa sinh viên đầu tiên khỏi hàng đợi

void dequeue(Queue \*q) {

if (q->front == NULL) {

printf("Hàng đợi rỗng! Không có sinh viên nào để xóa.\n");

return;

}

Student \*temp = q->front;

q->front = q->front->next;

if (q->front == NULL) {

q->rear = NULL; // Nếu hàng đợi trống sau khi xóa

}

printf("Đã xóa sinh viên: %s - %s\n", temp->studentID, temp->name);

free(temp);

}

## ***3.5.5. Hàm Duyệt Hàng đợi (Hiển thị danh sách sinh viên)***

c

Sao chépChỉnh sửa

// Hiển thị danh sách sinh viên trong hàng đợi

void displayQueue(Queue \*q) {

if (q->front == NULL) {

printf("Hàng đợi rỗng!\n");

return;

}

printf("Danh sách sinh viên trong hàng đợi:\n");

Student \*temp = q->front;

while (temp != NULL) {

printf("MSSV: %s | Họ tên: %s | Năm sinh: %d | GPA: %.2f\n",

temp->studentID, temp->name, temp->birthYear, temp->gpa);

temp = temp->next;

}

}

## ***3.5.6. Chương trình Chính***

c

Sao chépChỉnh sửa

int main() {

Queue studentQueue;

initializeQueue(&studentQueue);

// Thêm sinh viên vào hàng đợi

enqueue(&studentQueue, "SV001", "Nguyen Van A", 2003, 3.5);

enqueue(&studentQueue, "SV002", "Tran Thi B", 2002, 3.8);

enqueue(&studentQueue, "SV003", "Le Van C", 2001, 3.2);

// Hiển thị danh sách sinh viên

displayQueue(&studentQueue);

// Xóa sinh viên khỏi hàng đợi

dequeue(&studentQueue);

displayQueue(&studentQueue);

return 0;

}

## ***3.5.7. Kết quả chạy chương trình***

yaml

Sao chépChỉnh sửa

Đã thêm sinh viên: SV001 - Nguyen Van A

Đã thêm sinh viên: SV002 - Tran Thi B

Đã thêm sinh viên: SV003 - Le Van C

Danh sách sinh viên trong hàng đợi:

MSSV: SV001 | Họ tên: Nguyen Van A | Năm sinh: 2003 | GPA: 3.50

MSSV: SV002 | Họ tên: Tran Thi B | Năm sinh: 2002 | GPA: 3.80

MSSV: SV003 | Họ tên: Le Van C | Năm sinh: 2001 | GPA: 3.20

Đã xóa sinh viên: SV001 - Nguyen Van A

Danh sách sinh viên trong hàng đợi:

MSSV: SV002 | Họ tên: Tran Thi B | Năm sinh: 2002 | GPA: 3.80

MSSV: SV003 | Họ tên: Le Van C | Năm sinh: 2001 | GPA: 3.20

## **3.5.8. Giải thích Chương trình**

* **Cấu trúc Student** chứa thông tin của sinh viên.
* **Cấu trúc Queue** có hai con trỏ front và rear để quản lý hàng đợi.
* **initializeQueue(q)**: Khởi tạo hàng đợi rỗng.
* **enqueue(q, studentID, name, birthYear, gpa)**: Thêm sinh viên vào cuối hàng đợi.
* **dequeue(q)**: Xóa sinh viên đầu hàng đợi.
* **displayQueue(q)**: Hiển thị danh sách sinh viên trong hàng đợi.
* **Trong main()**, chương trình thực hiện thêm, xóa, và hiển thị danh sách sinh viên.

## **3.5.9. Mở rộng Chương trình**

Có thể bổ sung các chức năng nâng cao như: ✅ **Sắp xếp sinh viên theo điểm GPA trong hàng đợi.**  
**✅ Tìm kiếm sinh viên theo mã số sinh viên.  
✅ Lưu danh sách sinh viên vào file và đọc lại từ file.**