THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Họ tên: Khương Thanh Bình

MSSV:3123411034

Bài Tập Ngăn Xếp Và Hàng Đợi

**BÀI TẬP THỰC HÀNH**

**Bài 1. Ngăn xếp số nguyên**

(a) Cài đặt CTDL StackInt và LinkedStackInt dùng để chứa các số nguyên, trong đó:

• Sử dụng mảng (StackInt) và danh sách liên kết (LinkedStackInt)

• Cài đặt các thao tác: InitStack, IsEmpty, IsFull, PopStack, PushStack, PeekStack, Clear

(b) Ứng dụng ngăn xếp để đảo số,

(c) Ứng dụng ngăn xếp để kiểm tra xâu đối xứng (mở rộng sang StackString),

(d) Ứng dụng ngăn xếp để đổi từ số thập phân sang nhị phân,

(e) Nhập một biểu thức từ bàn phím, hãy chuyển sang dạng hậu tố và tính giá trị biểu thức.

**- StackInt**

CODE

#include <iostream>

#include <stack>

#include <string>

using namespace std;

typedef struct {

int top; // dinh ngan xep

int count; // so phan tu

int size; // kich thuoc toi da

int\* arr; // chua du lieu

} Stack;

void Init(Stack& s, int size) {

s.top = -1;

s.count = 0;

s.size = size;

s.arr = new int[s.size];

}

void Delete(Stack& s) {

s.top = -1;

s.count = 0;

s.size = 0;

delete[] s.arr;

}

bool IsEmpty(Stack& s) {

return s.top == -1;

}

bool IsFull(Stack& s) {

return s.top == s.size - 1;

}

void Push(Stack& s, int x) {

if (IsFull(s)) {

cout << "Stack bi tran (stack overflow)" << endl;

return;

}

s.arr[++s.top] = x;

s.count++;

}

int Pop(Stack& s) {

if (IsEmpty(s)) {

cout << "Stack rong!" << endl;

return -1;

}

int x = s.arr[s.top--];

s.count--;

return x;

}

int PeekStack(Stack& s) {

if (IsEmpty(s)) return -1;

return s.arr[s.top];

}

void Clear(Stack& s) {

s.top = -1;

s.count = 0;

}

// (b) Dao so nguyen

int DaoSo(int so) {

bool isNegative = so < 0;

if (isNegative) so = -so; // Lam viec voi gia tri tuyet doi

Stack nganXep;

Init(nganXep, 10);

while (so > 0) {

Push(nganXep, so % 10);

so /= 10;

}

int soDao = 0, heSo = 1;

while (!IsEmpty(nganXep)) {

soDao += Pop(nganXep) \* heSo;

heSo \*= 10;

}

Delete(nganXep);

return isNegative ? -soDao : soDao; // Neu la so am, them lai dau "-"

}

// (c) Kiem tra xau doi xung

bool KiemTraDoiXung(string xau) {

Stack nganXep;

Init(nganXep, xau.length());

for (char c : xau) Push(nganXep, c);

for (char c : xau) {

if (c != Pop(nganXep)) {

Delete(nganXep);

return false;

}

}

Delete(nganXep);

return true;

}

// (d) Chuyen doi so thap phan sang nhi phan

string ThapPhanSangNhiPhan(int so) {

Stack nganXep;

Init(nganXep, 32);

while (so > 0) {

Push(nganXep, so % 2);

so /= 2;

}

string nhiPhan = "";

while (!IsEmpty(nganXep)) {

nhiPhan += to\_string(Pop(nganXep));

}

Delete(nganXep);

return nhiPhan;

}

int main() {

Stack s;

Init(s, 5);

cout << "Dao so -12345: " << DaoSo(-12345) << endl;

cout << "Dao so 67890: " << DaoSo(67890) << endl;

cout << "Dao so 0: " << DaoSo(0) << endl;

cout << "Kiem tra xau 'madam' co doi xung: ";

if (KiemTraDoiXung("madam")) cout << "Co" << endl;

else cout << "Khong" << endl;

cout << "Kiem tra xau 'hello' co doi xung: ";

if (KiemTraDoiXung("hello")) cout << "Co" << endl;

else cout << "Khong" << endl;

cout << "So 25 sang nhi phan: " << ThapPhanSangNhiPhan(25) << endl;

Delete(s);

return 0;

}

**- LinkedStackInt**

CODE

#include <iostream>

#include <string>

using namespace std;

struct Node {

int data;

Node\* next;

};

struct LinkedStack {

Node\* top;

int count;

};

void Init(LinkedStack& s) {

s.top = nullptr;

s.count = 0;

}

void Delete(LinkedStack& s) {

while (s.top) {

Node\* temp = s.top;

s.top = s.top->next;

delete temp;

}

s.count = 0;

}

bool IsEmpty(LinkedStack& s) {

return s.top == nullptr;

}

void Push(LinkedStack& s, int x) {

Node\* newNode = new Node{ x, s.top };

s.top = newNode;

s.count++;

}

int Pop(LinkedStack& s) {

if (IsEmpty(s)) {

cout << "Stack rong!" << endl;

return -1;

}

int x = s.top->data;

Node\* temp = s.top;

s.top = s.top->next;

delete temp;

s.count--;

return x;

}

int PeekStack(LinkedStack& s) {

if (IsEmpty(s)) return -1;

return s.top->data;

}

void Clear(LinkedStack& s) {

Delete(s);

}

// (b) Dao so nguyen

int DaoSo(int so) {

bool isNegative = so < 0;

if (isNegative) so = -so;

LinkedStack nganXep;

Init(nganXep);

while (so > 0) {

Push(nganXep, so % 10);

so /= 10;

}

int soDao = 0, heSo = 1;

while (!IsEmpty(nganXep)) {

soDao += Pop(nganXep) \* heSo;

heSo \*= 10;

}

Delete(nganXep);

return isNegative ? -soDao : soDao;

}

// (c) Kiem tra xau doi xung

bool KiemTraDoiXung(string xau) {

LinkedStack nganXep;

Init(nganXep);

for (char c : xau) Push(nganXep, c);

for (char c : xau) {

if (c != Pop(nganXep)) {

Delete(nganXep);

return false;

}

}

Delete(nganXep);

return true;

}

// (d) Chuyen doi so thap phan sang nhi phan

string ThapPhanSangNhiPhan(int so) {

LinkedStack nganXep;

Init(nganXep);

while (so > 0) {

Push(nganXep, so % 2);

so /= 2;

}

string nhiPhan = "";

while (!IsEmpty(nganXep)) {

nhiPhan += to\_string(Pop(nganXep));

}

Delete(nganXep);

return nhiPhan;

}

int main() {

LinkedStack s;

Init(s);

cout << "Dao so -12345: " << DaoSo(-12345) << endl;

cout << "Dao so 67890: " << DaoSo(67890) << endl;

cout << "Dao so 0: " << DaoSo(0) << endl;

cout << "Kiem tra xau 'madam' co doi xung: ";

if (KiemTraDoiXung("madam")) cout << "Co" << endl;

else cout << "Khong" << endl;

cout << "Kiem tra xau 'hello' co doi xung: ";

if (KiemTraDoiXung("hello")) cout << "Co" << endl;

else cout << "Khong" << endl;

cout << "So 25 sang nhi phan: " << ThapPhanSangNhiPhan(25) << endl;

Delete(s);

return 0;

}

Run code

-> A black screen with white text

AI-generated content may be incorrect.

**Bài 2. Hàng đợi số nguyên**

(a) Cài đặt CTDL QueueInt và LinkedQueueInt dùng để chứa các số nguyên, trong đó:

• Sử dụng mảng (StackInt) và danh sách liên kết (LinkedStackInt)

• Cài đặt các thao tác: InitStack, IsEmpty, IsFull, PopStack, PushStack, PeekStack, Clear

(b) Ứng dụng hàng đợi để làm bài toán xếp lịch cặp múa nam/nữ (như trong bài giảng),

(c) Ứng dụng hàng đợi để cài thuật toán RadixSort

- **QueueInt**

CODE

#include <iostream>

#include <vector>

using namespace std;

// Định nghĩa QueueInt (hàng đợi dùng mảng)

struct QueueInt {

int front, rear, count, size;

string\* arr;

// Hàm khởi tạo hàng đợi

void Init(int sz = 100) {

front = rear = -1;

count = 0;

size = sz;

arr = new string[size];

}

// Kiểm tra hàng đợi rỗng

bool IsEmpty() { return count == 0; }

// Kiểm tra hàng đợi đầy

bool IsFull() { return count == size; }

// Thêm phần tử vào hàng đợi

void Enqueue(string value) {

if (IsFull()) return;

if (front == -1) front = 0;

rear = (rear + 1) % size;

arr[rear] = value;

count++;

}

// Lấy phần tử ra khỏi hàng đợi

string Dequeue() {

if (IsEmpty()) return "";

string value = arr[front];

front = (front + 1) % size;

count--;

if (count == 0) front = rear = -1;

return value;

}

// Hiển thị hàng đợi

void Show(string label) {

cout << label << ": ";

for (int i = front, j = 0; j < count; j++, i = (i + 1) % size)

cout << arr[i] << " ";

cout << endl;

}

};

// Xếp lịch cặp múa nam/nữ

void FormLines(QueueInt& male, QueueInt& female) {

string dancers[] = {

"F Trang", "M Truc", "M Thien", "M Bao", "F Thu",

"M Tien", "F Thuy", "M Nghia", "F Thao", "M Phuoc",

"M Hung", "F Vy"

};

int n = 12;

for (int i = 0; i < n; i++) {

if (dancers[i][0] == 'M')

male.Enqueue(dancers[i].substr(2)); // Bỏ "M "

else

female.Enqueue(dancers[i].substr(2)); // Bỏ "F "

}

}

// Ghép cặp vũ công

void StartDancing(QueueInt& male, QueueInt& female) {

cout << "Cac cap dien vien:\n";

int pairs = 4; // Số cặp có thể nhảy đồng thời

for (int i = 0; i < pairs; i++) {

if (!male.IsEmpty() && !female.IsEmpty())

cout << male.Dequeue() << " - " << female.Dequeue() << endl;

else

break;

}

}

// Hiển thị người tiếp theo

void HeadOfLine(QueueInt& male, QueueInt& female) {

if (!male.IsEmpty() && !female.IsEmpty())

cout << "Cap dien vien ke tiep: " << male.arr[male.front] << " - " << female.arr[female.front] << endl;

else if (!male.IsEmpty())

cout << "Dien vien nam ke tiep: " << male.arr[male.front] << endl;

else if (!female.IsEmpty())

cout << "Dien vien nu ke tiep: " << female.arr[female.front] << endl;

}

// Định nghĩa QueueInt cho Radix Sort (hàng đợi số nguyên)

struct RadixQueue {

int front, rear, count, size;

int\* arr;

// Khởi tạo hàng đợi

void Init(int sz = 100) {

front = rear = -1;

count = 0;

size = sz;

arr = new int[size];

}

bool IsEmpty() { return count == 0; }

void Enqueue(int value) {

if (count == size) return;

rear = (rear + 1) % size;

arr[rear] = value;

if (front == -1) front = 0;

count++;

}

int Dequeue() {

if (IsEmpty()) return -1;

int value = arr[front];

front = (front + 1) % size;

count--;

return value;

}

};

// Thuật toán Radix Sort

void RadixSort(vector<int>& arr) {

RadixQueue buckets[10]; // 10 hàng đợi (0-9)

for (int i = 0; i < 10; i++) buckets[i].Init(arr.size());

int maxValue = \*max\_element(arr.begin(), arr.end());

int exp = 1;

while (maxValue / exp > 0) {

for (int num : arr) {

int bucketIndex = (num / exp) % 10;

buckets[bucketIndex].Enqueue(num);

}

int index = 0;

for (int i = 0; i < 10; i++) {

while (!buckets[i].IsEmpty())

arr[index++] = buckets[i].Dequeue();

}

exp \*= 10;

}

}

int main() {

// \*\*Bài toán xếp lịch cặp múa nam/nữ\*\*

cout << "\n=== Bai toan xep lich cap mua nam/nu ===\n";

QueueInt males, females;

males.Init(10);

females.Init(10);

FormLines(males, females);

males.Show("Danh sach dien vien nam");

females.Show("Danh sach dien vien nu");

StartDancing(males, females);

if (!males.IsEmpty() || !females.IsEmpty()) {

HeadOfLine(males, females);

StartDancing(males, females);

}

// \*\*Bài toán Radix Sort\*\*

cout << "\n=== Bai toan Radix Sort ===\n";

vector<int> arr = { 170, 45, 75, 90, 802, 24, 2, 66 };

cout << "Mang truoc khi sap xep: ";

for (int num : arr) cout << num << " ";

cout << endl;

RadixSort(arr);

cout << "Mang sau khi sap xep: ";

for (int num : arr) cout << num << " ";

cout << endl;

return 0;

}

- **LinkedQueueInt**

CODE

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

// Cấu trúc Node cho danh sách liên kết

struct Node {

string data;

Node\* next;

};

// Cấu trúc LinkedQueueInt (hàng đợi sử dụng danh sách liên kết)

struct LinkedQueueInt {

Node\* front;

Node\* rear;

int count;

// Khởi tạo hàng đợi

void Init() {

front = rear = nullptr;

count = 0;

}

// Kiểm tra hàng đợi rỗng

bool IsEmpty() {

return count == 0;

}

// Thêm phần tử vào hàng đợi

void Enqueue(string value) {

Node\* newNode = new Node{ value, nullptr };

if (IsEmpty()) {

front = rear = newNode;

}

else {

rear->next = newNode;

rear = newNode;

}

count++;

}

// Lấy phần tử ra khỏi hàng đợi

string Dequeue() {

if (IsEmpty()) return "";

Node\* temp = front;

string value = front->data;

front = front->next;

if (front == nullptr) rear = nullptr;

delete temp;

count--;

return value;

}

// Hiển thị hàng đợi

void Show(string label) {

cout << label << ": ";

Node\* temp = front;

while (temp) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

};

// Xếp lịch cặp múa nam/nữ

void FormLines(LinkedQueueInt& male, LinkedQueueInt& female) {

string dancers[] = {

"F Trang", "M Truc", "M Thien", "M Bao", "F Thu",

"M Tien", "F Thuy", "M Nghia", "F Thao", "M Phuoc",

"M Hung", "F Vy"

};

int n = 12;

for (int i = 0; i < n; i++) {

if (dancers[i][0] == 'M')

male.Enqueue(dancers[i].substr(2)); // Bỏ "M "

else

female.Enqueue(dancers[i].substr(2)); // Bỏ "F "

}

}

// Ghép cặp vũ công

void StartDancing(LinkedQueueInt& male, LinkedQueueInt& female) {

cout << "Cac cap dien vien:\n";

int pairs = 4; // Số cặp có thể nhảy đồng thời

for (int i = 0; i < pairs; i++) {

if (!male.IsEmpty() && !female.IsEmpty())

cout << male.Dequeue() << " - " << female.Dequeue() << endl;

else

break;

}

}

// Hiển thị người tiếp theo

void HeadOfLine(LinkedQueueInt& male, LinkedQueueInt& female) {

if (!male.IsEmpty() && !female.IsEmpty())

cout << "Cap dien vien ke tiep: " << male.front->data << " - " << female.front->data << endl;

else if (!male.IsEmpty())

cout << "Dien vien nam ke tiep: " << male.front->data << endl;

else if (!female.IsEmpty())

cout << "Dien vien nu ke tiep: " << female.front->data << endl;

}

// Cấu trúc LinkedQueueInt cho Radix Sort (hàng đợi số nguyên)

struct RadixQueue {

struct IntNode {

int data;

IntNode\* next;

};

IntNode\* front;

IntNode\* rear;

int count;

void Init() {

front = rear = nullptr;

count = 0;

}

bool IsEmpty() { return count == 0; }

void Enqueue(int value) {

IntNode\* newNode = new IntNode{ value, nullptr };

if (IsEmpty()) {

front = rear = newNode;

}

else {

rear->next = newNode;

rear = newNode;

}

count++;

}

int Dequeue() {

if (IsEmpty()) return -1;

IntNode\* temp = front;

int value = front->data;

front = front->next;

if (front == nullptr) rear = nullptr;

delete temp;

count--;

return value;

}

};

// Thuật toán Radix Sort sử dụng hàng đợi liên kết

void RadixSort(vector<int>& arr) {

RadixQueue buckets[10]; // 10 hàng đợi (0-9)

for (int i = 0; i < 10; i++) buckets[i].Init();

int maxValue = \*max\_element(arr.begin(), arr.end());

int exp = 1;

while (maxValue / exp > 0) {

for (int num : arr) {

int bucketIndex = (num / exp) % 10;

buckets[bucketIndex].Enqueue(num);

}

int index = 0;

for (int i = 0; i < 10; i++) {

while (!buckets[i].IsEmpty())

arr[index++] = buckets[i].Dequeue();

}

exp \*= 10;

}

}

int main() {

// \*\*Bài toán xếp lịch cặp múa nam/nữ\*\*

cout << "\n=== Bai toan xep lich cap mua nam/nu ===\n";

LinkedQueueInt males, females;

males.Init();

females.Init();

FormLines(males, females);

males.Show("Danh sach dien vien nam");

females.Show("Danh sach dien vien nu");

StartDancing(males, females);

if (!males.IsEmpty() || !females.IsEmpty()) {

HeadOfLine(males, females);

StartDancing(males, females);

}

// \*\*Bài toán Radix Sort\*\*

cout << "\n=== Bai toan Radix Sort ===\n";

vector<int> arr = { 170, 45, 75, 90, 802, 24, 2, 66 };

cout << "Mang truoc khi sap xep: ";

for (int num : arr) cout << num << " ";

cout << endl;

RadixSort(arr);

cout << "Mang sau khi sap xep: ";

for (int num : arr) cout << num << " ";

cout << endl;

return 0;

}

Run code

-> A computer screen with white text

AI-generated content may be incorrect.

**BÀI TẬP MỞ RỘNG**

**Bài 1. Dùng stack khử đệ qui**

(a) Viết chương trình đệ qui / khử đệ qui (dùng stack) cho bài toán tính số Fibonancy, đảo ngược số

(b) Viết chương trình đệ qui / khử đệ qui (dùng stack) cho bài toán đảo ngược số

(c) HanoiTower là một tháp có nhiều tầng, tầng nhỏ nằm trên tầng lớn. Sau đó viết phương thức di chuyển tháp này

từ vị trí 1 đến vị trí 3 thông qua vị trí trung gian 2; mỗi lần chỉ được di chuyển tầng trên cùng của tháp và tại mỗi

vị trí 1, 2, và 3 đều là tháp. Hãy viết chương trình đệ qui / khử đệ qui cho bài toán tháp Hà Nội.

CODE

#include <iostream>

#include <stack>

using namespace std;

// (a) Fibonacci bang de quy

int fibonacciDeQuy(int n) {

if (n <= 1) return n;

return fibonacciDeQuy(n - 1) + fibonacciDeQuy(n - 2);

}

// Fibonacci khong de quy (dung stack)

int fibonacciStack(int n) {

if (n <= 1) return n;

stack<int> s;

s.push(n);

int a = 0, b = 1;

while (!s.empty()) {

int num = s.top(); s.pop();

if (num == 1) return b;

if (num == 0) return a;

s.push(num - 1);

s.push(num - 2);

}

return 0;

}

// (b) Dao nguoc so bang de quy

int daoNguocSoDeQuy(int n, int rev = 0) {

if (n == 0) return rev;

return daoNguocSoDeQuy(n / 10, rev \* 10 + n % 10);

}

// Dao nguoc so khong de quy (dung stack)

int daoNguocSoStack(int n) {

stack<int> s;

while (n > 0) {

s.push(n % 10);

n /= 10;

}

int reversed = 0, factor = 1;

while (!s.empty()) {

reversed += s.top() \* factor;

s.pop();

factor \*= 10;

}

return reversed;

}

// (c) Thap Ha Noi bang de quy

void hanoiDeQuy(int n, char from, char to, char aux) {

if (n == 1) {

cout << "Move disk 1 from " << from << " to " << to << endl;

return;

}

hanoiDeQuy(n - 1, from, aux, to);

cout << "Move disk " << n << " from " << from << " to " << to << endl;

hanoiDeQuy(n - 1, aux, to, from);

}

int main() {

int n = 5;

cout << "Fibonacci (de quy) cua " << n << ": " << fibonacciDeQuy(n) << endl;

cout << "Fibonacci (stack) cua " << n << ": " << fibonacciStack(n) << endl;

int num = 1234;

cout << "Dao nguoc (de quy) cua " << num << ": " << daoNguocSoDeQuy(num) << endl;

cout << "Dao nguoc (stack) cua " << num << ": " << daoNguocSoStack(num) << endl;

int disks = 3;

cout << "Thap Ha Noi (de quy) voi " << disks << " dia:" << endl;

hanoiDeQuy(disks, 'A', 'C', 'B');

return 0;

}

Run code

-> A screen shot of a computer

AI-generated content may be incorrect.

**Bài 2. Mê Cung (MECUNG.\*)**

Cho mê cung kích thước N dòng và M cột. Ô (i,j) có giá trị 0 (không có chướng ngại), 1 (có chướng ngại). Một người

xuất phát từ ô (x0,y0) và di chuyển qua trái, qua phải, trên và dưới qua các ô không có chướng ngại. Hỏi người đó có đi

tới được ô (x1,y1)? Nếu có, chỉ đường đi từ (x0,y0) đến (x1,y1). (Áp dụng BFS – hàng đợi và DFS – ngăn xếp).

Dữ liệu:

• Dòng đầu chứa N M x0 y0 x1 y1, ứng với số dòng, số cột mê cung, điểm xuất phát, điểm cần đi tới.

• N dòng tiếp, mỗi dòng có M số chứa 0 (không có chướng ngại) hoặc 1 (có chướng ngại).

Kết quả:

• Dòng đầu tiên ghi số 0 hoặc K với 0 ứng với không có lộ trình, K ứng với có lộ trình và đi qua K ô.

• Nếu có lộ trình, K dòng, mỗi dòng ghi u v, trong đó (u,v) là 1 ô trong lộ trình từ (x0,y0) đến (x1,y1).

CODE

#include <iostream>

#include <vector>

using namespace std;

struct Node {

int x, y;

Node\* next;

};

struct Stack {

Node\* top = nullptr;

void push(int x, int y) {

Node\* newNode = new Node{ x, y, top };

top = newNode;

}

void pop(int& x, int& y) {

if (!top) {

x = -1; y = -1;

return;

}

Node\* temp = top;

x = temp->x;

y = temp->y;

top = top->next;

delete temp;

}

bool empty() { return top == nullptr; }

};

struct Queue {

Node\* front = nullptr, \* rear = nullptr;

void push(int x, int y) {

Node\* newNode = new Node{ x, y, nullptr };

if (!rear) front = rear = newNode;

else {

rear->next = newNode;

rear = newNode;

}

}

void pop(int& x, int& y) {

if (!front) {

x = -1; y = -1;

return;

}

Node\* temp = front;

x = temp->x;

y = temp->y;

front = front->next;

if (!front) rear = nullptr;

delete temp;

}

bool empty() { return front == nullptr; }

};

void inMeCung(const vector<vector<int>>& maze) {

for (const auto& row : maze) {

for (int cell : row) {

cout << (cell ? "# " : ". ");

}

cout << endl;

}

cout << endl;

}

bool bfsMeCung(vector<vector<int>>& maze, int N, int M, int x0, int y0, int x1, int y1) {

Queue q;

vector<vector<bool>> visited(N, vector<bool>(M, false));

vector<vector<int>> parentX(N, vector<int>(M, -1));

vector<vector<int>> parentY(N, vector<int>(M, -1));

q.push(x0, y0);

visited[x0][y0] = true;

int dx[] = { 1, -1, 0, 0 }, dy[] = { 0, 0, 1, -1 };

while (!q.empty()) {

int x, y;

q.pop(x, y);

if (x == x1 && y == y1) {

vector<int> pathX, pathY;

while (x != -1 && y != -1) {

pathX.push\_back(x);

pathY.push\_back(y);

int px = parentX[x][y];

int py = parentY[x][y];

x = px;

y = py;

}

cout << "Duong di BFS:\n";

for (int i = pathX.size() - 1; i >= 0; i--) {

cout << pathX[i] << " " << pathY[i] << endl;

}

return true;

}

for (int i = 0; i < 4; i++) {

int nx = x + dx[i], ny = y + dy[i];

if (nx >= 0 && ny >= 0 && nx < N && ny < M && maze[nx][ny] == 0 && !visited[nx][ny]) {

visited[nx][ny] = true;

parentX[nx][ny] = x;

parentY[nx][ny] = y;

q.push(nx, ny);

}

}

}

return false;

}

bool dfsMeCung(vector<vector<int>>& maze, int N, int M, int x0, int y0, int x1, int y1) {

Stack s;

vector<vector<bool>> visited(N, vector<bool>(M, false));

vector<vector<int>> parentX(N, vector<int>(M, -1));

vector<vector<int>> parentY(N, vector<int>(M, -1));

s.push(x0, y0);

visited[x0][y0] = true;

int dx[] = { 1, -1, 0, 0 }, dy[] = { 0, 0, 1, -1 };

while (!s.empty()) {

int x, y;

s.pop(x, y);

if (x == x1 && y == y1) {

vector<int> pathX, pathY;

while (x != -1 && y != -1) {

pathX.push\_back(x);

pathY.push\_back(y);

int px = parentX[x][y];

int py = parentY[x][y];

x = px;

y = py;

}

cout << "Duong di DFS:\n";

for (int i = pathX.size() - 1; i >= 0; i--) {

cout << pathX[i] << " " << pathY[i] << endl;

}

return true;

}

for (int i = 0; i < 4; i++) {

int nx = x + dx[i], ny = y + dy[i];

if (nx >= 0 && ny >= 0 && nx < N && ny < M && maze[nx][ny] == 0 && !visited[nx][ny]) {

visited[nx][ny] = true;

parentX[nx][ny] = x;

parentY[nx][ny] = y;

s.push(nx, ny);

}

}

}

return false;

}

int main() {

vector<vector<int>> maze = {

{0, 1, 0, 0},

{0, 0, 0, 1},

{1, 0, 1, 0},

{0, 0, 0, 0}

};

int N = 4, M = 4, x0 = 0, y0 = 0, x1 = 3, y1 = 3;

cout << "Me cung ban dau:\n";

inMeCung(maze);

if (!bfsMeCung(maze, N, M, x0, y0, x1, y1)) {

cout << "Khong co duong di BFS" << endl;

}

if (!dfsMeCung(maze, N, M, x0, y0, x1, y1)) {

cout << "Khong co duong di DFS" << endl;

}

return 0;

}

Run code

-> A screenshot of a computer

AI-generated content may be incorrect.