Nội dung

[Bài thực hành số 01 – Tuần 8 2](#_Toc180858114)

[BÀI TẬP TRÊN LAB 2](#_Toc180858115)

[**Bài 1.1.** Viết một chương trình C nhập vào 3 số nguyên. Thiết lập một con trỏ để lần lượt trỏ tới từng số nguyên và hiển thị kết quả giá trị tham chiếu ngược của con trỏ. 2](#_Toc180858116)

[**Bài 1.3.**Viết chương trình yêu cầu nhập giá trị cho 3 biến số nguyên x, y, z kiểu int. Sau đó sử dụng duy nhất một con trỏ để cộng giá trị của mỗi biến thêm 100. 4](#_Toc180858117)

[**Bài 1.4.**Viết hàm countEven(int\*, int) nhận một mảng số nguyên và kích thước của mảng, trả về số lượng số chẵn trong mảng??? 6](#_Toc180858118)

[**Bài 1.5.**Viết hàm trả về con trỏ trỏ tới giá trị lớn nhất của một mảng các số double. Nếu mảng rỗng hãy trả về NULL. 8](#_Toc180858119)

[**Bài 1.6.** Viết hàm đảo ngược một mảng các số nguyên theo hai cách: dùng chỉ số và dùng con trỏ. 10](#_Toc180858120)

[**Bài 1.7.**Viết chương trình nhập vào một mảng các số nguyên với số lượng các phần tử nhập từ bàn phím. Sau đó sắp xếp mảng theo thứ tự tăng dần. Hiển thị danh sách mảng trước và sau khi sắp xếp. 12](#_Toc180858121)

[**Bài 1.8.**Viết chương trình nhập vào một ma trận 2 chiều kích thước m\*n với m và n nhập từ bàn phím. Sau đó đưa ra tổng các phần tử chẵn của ma trận đó. 14](#_Toc180858122)

[BÀI TẬP VỀ NHÀ 18](#_Toc180858123)

[**Bài 1.9.** Viết chương trình in ra tất cả các dãy con của một dãy cho trước. 18](#_Toc180858124)

[**Bài 1.10.** Viết chương trình nhập vào 2 ma trận vuông cùng kích thước n\*n, trong đó n nhập từ bàn phím. Sau đó tính tổng và tích của hai ma trận đó và đưa kết quả ra màn hình. 21](#_Toc180858125)

# Bài thực hành số 01 – Tuần 8

## BÀI TẬP TRÊN LAB

### **Bài 1.1.** Viết một chương trình C nhập vào 3 số nguyên. Thiết lập một con trỏ để lần lượt trỏ tới từng số nguyên và hiển thị kết quả giá trị tham chiếu ngược của con trỏ.

Lưu ý: Phép toán & trả về địa chỉ của biến.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, số, ảnh chụp màn hình

Mô tả được tạo tự động

# include <stdio.h>

int main(){

int x, y, z;

int\* ptr;

printf("Enter three integers: ");

scanf("%d %d %d", &x, &y, &z);

printf("\nThe three integers are:\n");

ptr = &x;

printf("x = %d\n", \*ptr);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# YOUR CODE HERE #

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ptr = &y;

printf("y = %d\n", \*ptr);

ptr = &z;

printf("z = %d\n", \*ptr);

return 0;

}

### **Bài 1.3.**Viết chương trình yêu cầu nhập giá trị cho 3 biến số nguyên x, y, z kiểu int. Sau đó sử dụng duy nhất một con trỏ để cộng giá trị của mỗi biến thêm 100.

Ảnh có chứa văn bản, phần mềm, số, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, số, phần mềm

Mô tả được tạo tự động

#include <stdio.h>

int main()

{

int x, y, z;

int \*ptr;

scanf("%d %d %d", &x, &y, &z);

printf("Here are the values of x, y, and z:\n");

printf("%d %d %d\n", x, y, z);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# YOUR CODE HERE #

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int tmp = 100;

ptr = &tmp;

x += \*ptr;

y += \*ptr;

z += \*ptr;

printf("Once again, here are the values of x, y, and z:\n");

printf("%d %d %d\n", x, y, z);

return 0;

}

### **Bài 1.4.**Viết hàm countEven(int\*, int) nhận một mảng số nguyên và kích thước của mảng, trả về số lượng số chẵn trong mảng???

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

int counteven(int\* arr, int size){

int count = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# YOUR CODE HERE #

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (int i = 0; i < size; ++i) {if (arr[i]%2==0) {++count;}}

return count;

}

### **Bài 1.5.**Viết hàm trả về con trỏ trỏ tới giá trị lớn nhất của một mảng các số double. Nếu mảng rỗng hãy trả về NULL.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, hàng

Mô tả được tạo tự động

Ảnh có chứa văn bản, phần mềm, số, ảnh chụp màn hình

Mô tả được tạo tự động

double\* maximum(double\* a, int size) {

if (a == NULL || size <= 0) return NULL;

double \*max = a;

for (int i = 1; i < size; ++i) {

if (a[i] > \*max) {

max = &a[i];

}

}

return max;

}

### **Bài 1.6.** Viết hàm đảo ngược một mảng các số nguyên theo hai cách: dùng chỉ số và dùng con trỏ.

Ví dụ mảng đầu vào là [9, -1, 4, 5, 7] thì kết quả là [7, 5, 4, -1, 9].

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

void reversearray(int arr[], int size){

int l = 0, r = size - 1, tmp;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# YOUR CODE HERE #

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while (l < r) {

tmp = arr[l];

arr[l] = arr[r];

arr[r] = tmp;

l++;

r--;

}

}

void ptr\_reversearray(int \*arr, int size){

int l = 0, r = size - 1, tmp;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# YOUR CODE HERE #

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while (l < r) {

### **Bài 1.7.**Viết chương trình nhập vào một mảng các số nguyên với số lượng các phần tử nhập từ bàn phím. Sau đó sắp xếp mảng theo thứ tự tăng dần. Hiển thị danh sách mảng trước và sau khi sắp xếp.

Yêu cầu chỉ sử dụng con trỏ để truy cập mảng, không truy cập theo index mảng.

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

#include <stdio.h>

#include <stdlib.h>

int \*a;

int n, tmp;

int main(){

    printf("Enter the number of elements: ");

    scanf("%d", &n);

    //#Allocate memory

    a = (int\*)malloc(sizeof(int));

    if (a == NULL) {

        printf("Cấp phát bộ nhớ thất bại!\n");

        return 1;

    }

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    for(int i = 0; i < n; i++)

        scanf("%d", a + i);

    printf("The input array is: \n");

    for(int i = 0; i < n; i++)

        printf("%d ", \*(a + i));

    printf("\n");

    //#Sort array

    for (int i = 0; i < n - 1; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (\*(a + j) > \*(a + j + 1)) {

                tmp = \*(a + j);

                \*(a + j) = \*(a + j + 1);

                \*(a + j + 1) = tmp;

            }

        }

    }

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    printf("The sorted array is: \n");

    for(int i = 0; i < n; i++)

        printf("%d ", \*(a + i));

    printf("\n");

    delete [] a;

    return 0;

}

### **Bài 1.8.**Viết chương trình nhập vào một ma trận 2 chiều kích thước m\*n với m và n nhập từ bàn phím. Sau đó đưa ra tổng các phần tử chẵn của ma trận đó.

**Lưu ý:** Khi viết hàm cấp phát bộ nhớ cho một ma trận hai chiều biểu diễn bởi con trỏ int \*\*mt, nếu ta truyền con trỏ theo kiểu địa chỉ void allocate\_mem(int \*\*mt, int m, int n) sẽ dẫn tới việc cấp phát bộ nhớ cho một bản sao của con trỏ \*\*mt. Do đó, sau khi gọi hàm thì con trỏ \*\*mt gốc vẫn không được cấp phát bộ nhớ. Để cấp phát thành công cần truyền con trỏ theo dạng địa chỉ, ví dụ sử dụng con trỏ cấp 3 dạng int \*\*\*mt.

Ảnh có chứa văn bản, phần mềm, số, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

#include <stdio.h>

#include <stdlib.h>

void allocate\_mem(int \*\*\*mt, int m, int n) {

// Cấp phát bộ nhớ cho ma trận

\*mt = (int \*\*)malloc(m \* sizeof(int \*));

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

(\*mt)[i] = (int \*)malloc(n \* sizeof(int));

}

}

void input(int \*\*mt, int m, int n) {

// Nhập các phần tử của ma trận

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

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

printf("mt[%d][%d] = ", i, j);

scanf("%d", &mt[i][j]);

}

}

}

void output(int \*\*mt, int m, int n) {

// In tất cả các phần tử của ma trận

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

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

printf("%d ", mt[i][j]);

}

printf("\n");

}

}

int process(int \*\*mt, int m, int n) {

int tong = 0;

// Tính tổng các phần tử chẵn trong ma trận

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

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

if (mt[i][j] % 2 == 0) {

tong += mt[i][j];

}

}

}

return tong;

}

void free\_mem(int \*\*mt, int m) {

// Giải phóng bộ nhớ

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

free(mt[i]);

}

free(mt);

}

int main() {

int m, n, \*\*mt;

printf("Enter m, n = ");

scanf("%d%d", &m, &n);

allocate\_mem(&mt, m, n);

input(mt, m, n);

output(mt, m, n);

printf("The sum of all even elements is %d\n", process(mt, m, n));

free\_mem(mt, m);

return 0;

}

# 

## BÀI TẬP VỀ NHÀ

### **Bài 1.9.** Viết chương trình in ra tất cả các dãy con của một dãy cho trước.

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Phông chữ

Mô tả được tạo tự động

#include <stdio.h>

void generateSubsequences(int *arr*[], int *n*) {

    for (int i = 0; i < *n*; i++) {

        for (int j = i; j < *n*; j++) {

            for (int k = i; k <= j; k++) {

                printf("%d ", *arr*[k]);

            }

            printf("\n");

        }

    }

}

int main() {

    int n;

    scanf("%d", &n);

    int arr[n];

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    generateSubsequences(arr, n);

    return 0;

}

**Case 1:**

**Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động**

**Case 2:**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động**

### **Bài 1.10.** Viết chương trình nhập vào 2 ma trận vuông cùng kích thước n\*n, trong đó n nhập từ bàn phím. Sau đó tính tổng và tích của hai ma trận đó và đưa kết quả ra màn hình.

Yêu cầu sử dụng cấp phát động để cấp phát bộ nhớ cho các ma trận.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động

#include <stdio.h>

#include <stdlib.h>

void inputMatrix(int\*\* matrix, int n) {

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

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

scanf("%d", &matrix[i][j]);

}

}

}

void printMatrix(int\*\* matrix, int n) {

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

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

printf("%d ", matrix[i][j]);

}

printf("\n");

}

}

int\*\* addMatrices(int\*\* A, int\*\* B, int n) {

int\*\* result = (int\*\*)malloc(n \* sizeof(int\*));

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

result[i] = (int\*)malloc(n \* sizeof(int));

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

result[i][j] = A[i][j] + B[i][j];

}

}

return result;

}

int\*\* multiplyMatrices(int\*\* A, int\*\* B, int n) {

int\*\* result = (int\*\*)malloc(n \* sizeof(int\*));

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

result[i] = (int\*)malloc(n \* sizeof(int));

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

result[i][j] = 0;

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

result[i][j] += A[i][k] \* B[k][j];

}

}

}

return result;

}

void deleteMatrix(int\*\* matrix, int n) {

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

free(matrix[i]);

}

free(matrix);

}

int main() {

int n;

scanf("%d", &n);

int\*\* A = (int\*\*)malloc(n \* sizeof(int\*));

int\*\* B = (int\*\*)malloc(n \* sizeof(int\*));

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

A[i] = (int\*)malloc(n \* sizeof(int));

B[i] = (int\*)malloc(n \* sizeof(int));

}

inputMatrix(A, n);

inputMatrix(B, n);

int\*\* sum = addMatrices(A, B, n);

int\*\* product = multiplyMatrices(A, B, n);

printMatrix(sum, n);

printMatrix(product, n);

deleteMatrix(A, n);

deleteMatrix(B, n);

deleteMatrix(sum, n);

deleteMatrix(product, n);

return 0;

}

**Case 1:**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

**Case 2:**

**Ảnh có chứa ảnh chụp màn hình, văn bản, Phông chữ

Mô tả được tạo tự động**