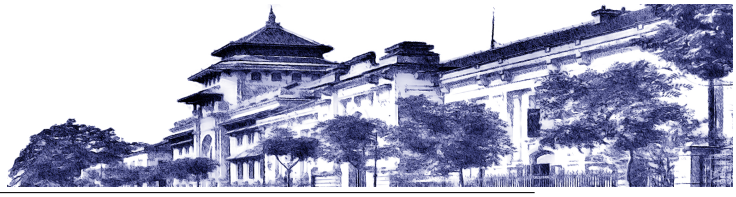
ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
Khoa Vật lý, Bộ môn Tin học Vật lý
P. 308D-T1, 334 Nguyễn Trãi, Thanh Xuân, Hà Nội



Phần 5

Nội dung thực hành

- Con trỏ: *khai báo con trỏ, thao tác trên vùng nhớ của biến khác thông qua con trỏ*
- Cấp phát bộ nhớ động: *cấp phát/giải phóng bộ nhớ bằng các hàm, quản lý bộ nhớ được cấp phát bằng con trỏ*
- Sử dụng con trỏ và các hàm cấp phát bộ nhớ động cho mảng: *cấp phát bộ nhớ, xuất/nhập và thao tác với các phần tử của mảng, giải phóng bộ nhớ được cấp*

Ví dụ

1. Thay đổi dữ liệu của các biến bằng con trỏ

```c
#include <stdio.h>
main() {
    int x = 1789, y = 1776; // khoi tao gia tri ban dau cho x va y
    int *VN;                // Khai bao con tro VN
    printf("Gia tri ban dau cua x la: %d\n", x);
    printf("Gia tri ban dau cua y la: %d\n", y);

    VN = &x;                // Tro VN vao vung nho cua bien x;
    *VN = 1954;             // Thay doi gia tri cua bien x thong qua con tro VN
                            // ->Thay gia tri 1954 vao vung nho ma VN tro den
    printf("\nGia tri cua x sau khi bi con tro VN doi la: %d\n", x);
    printf("Du lieu tai vung nho ma VN tro den la: %d\n", *VN);

    VN = &y;                // Tro VN vao vung nho cua bien y;
    *VN = 1975;             // Thay doi gia tri cua bien y thong qua con tro VN
                            // ->Thay gia tri 1975 vao vung nho ma VN tro den
    printf("\nGia tri cua y sau khi bi con tro VN doi la: %d\n", y);
    printf("Du lieu tai vung nho ma VN tro den la: %d\n", *VN);
}
```

2. Tính trung bình cộng (TBC) của n số

```c
#include <stdio.h>
#include <stdlib.h>
main() {
    int i, n, sum = 0;
    int *a; // Khai bao con tro a

    printf("Nhap vao gia tri n: ");
    scanf("%d", &n);
    // Cap phat n*sizeof(int) byte cho a
    a = (int*)malloc(n * sizeof(int));

    if(a == NULL) exit(1);

    for(i = 0; i < n; i++) {
        printf("\n a[%d] = ", i);
        scanf("%d", (a+i)); // Nhap a[i]
    }

    for (i = 0; i < n; i++) {
        sum = sum + *(a+i); // Tinh tong
    }

    printf("TBC: %f \n", (float)sum/n);
    free(a); // Giai phong vung nho
}
```

3. Tìm phần tử lớn nhất trong ma trận có m hàng, n cột

```c
#include <stdio.h>
#include <stdlib.h>
main() {
    int m, n, max, i, j, *b;

    printf("Nhap vao gia tri m, n: ");
    scanf("%d%d", &m, &n);
    // Cap phat bo nho cho mxn phan tu
    b = (int*)calloc(m * n, sizeof(int));

    if(b == NULL)  exit(1);

    for(i = 0; i < m; i++)
        for(j = 0; j < n; j++){
            printf("\na[%d, %d]= ", i, j);
            scanf("%d", (b+i*n+j));// Nhap b[i][j]
        }
    max = *b; // *b la gia tri cua b[0][0]
    for(i = 0; i < m; i++)
        for(j = 0; j < n; j++)
            if( *(b+i*n+j) > max)
                max = *(b+i*n+j);

    printf("Gia tri lon nhat la: %d\n", max);
    free(b); // Giai phong vung nho
}
```

Bài tập

Lưu ý:
- Các bài tập dưới đây bắt buộc phải sử dụng kiến thức về *con trỏ/địa chỉ/cấp phát bộ nhớ động*!
- Trong các bài tập 1 và 2, địa chỉ của các biến thay đổi sau mỗi lần chạy lại chương trình.

1. Cho số n. Hãy cấp phát bộ nhớ động cho mảng a có n phần tử là các số có kiểu *int*, sau đó nhập vào các giá trị của mảng a rồi in ra giá trị và địa chỉ của các phần tử của mảng.

   Input
   dòng 1 là số n
   dòng 2 là n số tương ứng với giá trị các phần tử của mảng a
   Output
   dòng 1 là n số tương ứng với giá trị các phần tử của mảng a
   dòng 2 là n số tương ứng với địa chỉ các phần tử của mảng a

Input	Output
3 4 2 7	4 2 7 82656 82660 82664

   Yêu cầu: Thay đổi kiểu dữ liệu của các phần tử từ *int* thành *short*, chạy lại chương trình và nhận xét kết quả thu được!

2. Cho một ma trận có m hàng n cột. Hãy cấp phát bộ nhớ động cho mảng a chứa các phần tử kiểu *int* của ma trận đó. Nhập vào các giá trị của mảng a rồi in ra giá trị và địa chỉ của các phần tử của mảng (theo hàng và cột).

   Input
   dòng 1 là 2 số m và n
   m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận
   Output
   m dòng, mỗi dòng chứa n số là giá trị các phần tử ma trận
   m dòng tiếp, mỗi dòng chứa n số là địa chỉ các phần tử ma trận

Input	Output
2 3 4 2 7 3 5 1	4 2 7 3 5 1 93824 93828 93832 93836 93840 93844

   Yêu cầu: Nhận xét về sự sắp xếp các phần tử của mảng hai chiều trong bộ nhớ máy tính!

3. Cho một mảng có n phần tử. Xóa một phần tử ở một vị trí được chọn trong mảng.

   Input
   dòng 1 là số nguyên n
   dòng 2 là n số tương ứng với giá trị các phần tử của mảng
   dòng 3 là 1 số chỉ vị trí phần tử bị xóa
   Output
   $n-1$ số là các phần tử của mảng sau khi đã xóa bớt

Input	Output
7 5 3 6 2 7 1 4 5	5 3 6 2 1 4

   Gợi ý: Sử dụng hàm *realloc* của thư viện *stdlib*.

   Mở rộng bài toán: Tương tự, thực hiện thao tác thêm một phần tử vào mảng.

4. Cho một mảng có n phần tử. Hãy cấp phát thêm bộ nhớ cho mảng đó: từ n ô nhớ cho n phần tử tăng lên thành $2n$ ô nhớ cho $2n$ phần tử. Sau đó, mỗi phần tử của mảng ban đầu sẽ có thêm một bản sao có vị trí ngay sát phần tử đó trong mảng. In ra mảng đó sau khi thực hiện các thao tác trên.

   Input
   dòng 1 là số n
   dòng 2 là n số tương ứng với giá trị các phần tử của mảng
   Output
   $2n$ số là các phần tử của mảng sau khi đã thực hiện thao tác

Input	Output
3 4 2 7	4 4 2 2 7 7

   Gợi ý: Sử dụng hàm *realloc* của thư viện *stdlib*.

5. Cho một mảng có n phần tử. Sắp xếp lại các phần tử của mảng đó theo thứ tự tăng dần.

   Input
   dòng 1 là số nguyên n
   dòng 2 là n số nguyên
   Output
   n số là các phần tử của mảng sau khi đã sắp xếp

Input	Output
7 5 3 6 2 7 1 4	1 2 3 4 5 6 7

6. *(Algebra)*
Cho một ma trận có $m \times n$ phần tử. Tính tổng các phần tử trên đường chéo chính của ma trận đó.

Input
dòng 1 là 2 số m và n
m dòng tiếp, mỗi dòng chứa n số là các phần tử của ma trận
Output
1 số duy nhất là tổng các phần tử trên đường chéo chính

Input	Output
3 4 2 3 5 0 8 7 3 9 1 4 2 6	11

7. *(Algebra)*
Cho một ma trận có $m \times n$ phần tử. Hãy chuyển vị ma trận đó.

Input
dòng 1 là 2 số m và n
m dòng tiếp, mỗi dòng chứa n số là các phần tử của ma trận
Output
n dòng, mỗi dòng chứa m số là các phần tử của ma trận sau khi chuyển vị

Input	Output
3 4 2 3 5 0 8 7 3 9 1 4 2 6	2 8 1 3 7 4 5 3 2 0 9 6

8. *(Algebra)*
Cho ma trận A có m hàng và n cột, và ma trận B có p hàng q cột. Xét xem hai ma trận A và B có thực hiện được phép nhân ma trận (đại số tuyến tính) không, nếu có thì thực hiện phép toán và in ra kết quả, nếu không thì in ra dòng chữ "INVALID".

Input
dòng 1 là 2 số m và n
m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A
dòng tiếp là 2 số p và q
p dòng tiếp, mỗi dòng chứa q số là các phần tử ma trận B
Output
đáp án của bài toán

Input	Output
1 2 2 1 2 3 1 2 3 4 5 6	6 9 12

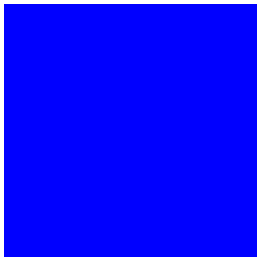
9. Cho một bảng số có m hàng, hàng thứ i có n_i phần tử (các số n_i không nhất thiết bằng nhau). Xóa một hàng được chọn trong bảng rồi in ra các giá trị bảng đó.

Input
dòng 1 là số m
dòng 2 là m số n_i ứng với số phần tử hàng thứ i
m dòng tiếp, dòng thứ i chứa n_i số là các phần tử của bảng
dòng cuối là 1 số chỉ vị trí dòng bị xóa
Output
$m - 1$ dòng, dòng thứ i chứa n_i số là các phần tử của bảng sau khi đã xóa bớt

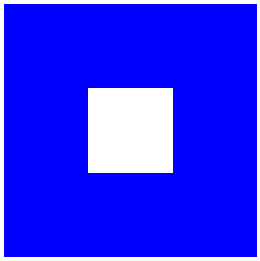
Input	Output
3 3 4 2 2 3 5 8 7 3 9 1 4 2	2 3 5 1 4

☞ Gợi ý: Nên sử dụng con trỏ "đa cấp" (con trỏ trỏ vào con trỏ) để lưu giữ dữ liệu. Như vậy, thao tác xóa một hàng của bảng sẽ đơn giản tương tự như bài xóa một phần tử của mảng trước đó.

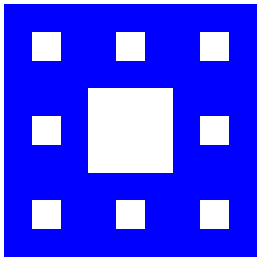
10. Thảm Sierpiński (*Sierpiński carpet*) là công trình khoa học về hình học tự đồng dạng của nhà toán học Wacław Sierpiński. Thảm này được tạo ra như sau: Đầu tiên cho một hình vuông (cấp 0), ta sẽ nhân hình vuông này thành 9 phần bằng nhau (3×3), rồi xóa phần chính giữa thì được hình vuông mới (cấp 1). Tiếp theo, ta lại nhân hình vuông cấp 1 đó thành 9 phần bằng nhau, rồi xóa đi phần chính giữa của chúng để được hình vuông mới (cấp 2). Lặp lại việc này vô hạn lần (cấp n), ta thu được tấm thảm Sierpiński.
Giả sử ký tự '*' mô tả hình vuông cấp 0. Cho n là số cấp (độ phức tạp) của tấm thảm. In ra tấm thảm Sierpiński cấp n.



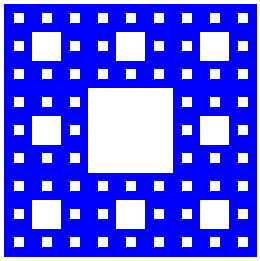
Cấp 0



Cấp 1



Cấp 2



Cấp 3

Input
1 số n duy nhất là cấp của tấm thảm
Output
các dòng mô tả tấm thảm cấp n

Input	Output
3	``` ************************** * ** ** ** ** ** ** ** ** * ************************** *** ****** ****** *** * * * ** * * ** * * * *** ****** ****** *** ************************** * ** ** ** ** ** ** ** ** * ************************** ********* ********* * ** ** * * ** ** * ********* ********* *** *** *** *** * * * * * * * * *** *** *** *** ********* ********* * ** ** * * ** ** * ********* ********* ************************** * ** ** ** ** ** ** ** ** * ************************** *** ****** ****** *** * * * ** * * ** * * * *** ****** ****** *** ************************** * ** ** ** ** ** ** ** ** * ************************** ```

☞ Mở rộng bài toán: Tìm ra quy luật và vẽ lại tam giác Sierpiński (*Sierpiński triangle*).