Bài thực hành 1: Con trỏ và cấp phát động

Trong bài này các bạn sẽ làm quen với việc sử dụng con trỏ và cấp phát động.

Phần 1. Thực hành về con trỏ

Bài tập 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.

In [ ]:

```
# 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 #
    *****************/

    return 0;
}
```

Bài tập 2

Viết chương trình in ra địa chỉ của 5 phần tử đầu tiên trong mảng được định nghĩa sau đây:
int a[7]= {13, -355, 235, 47, 67, 943, 1222};

Lưu ý:

Để in địa chỉ con trỏ các bạn sử dụng ký tự định dạng **%p**

Để lấy địa chỉ của một biến ta có thể dùng phép toán **&**

In [ ]:

```
#include <stdio.h>
int main(){
    int a[7]= {13, -355, 235, 47, 67, 943, 1222};
    printf("address of first five elements in memory.\n");
    for (int i=0; i<5;i++)  printf("\ta[%d] ",i);
    printf("\n");

    /*****************
    # YOUR CODE HERE #
    *****************/

    return 0;
}
```

Bài tập 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.

In [ ]:

```
#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 #
    *****************/

    printf("Once again, here are the values of x, y, and z:\n");
    printf("%d %d %d\n", x, y, z);
    return 0;
}
```

Phần 2. Con trỏ và mảng

Bài tập 4

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

In [ ]:

```
int counteven(int* arr, int size){
    int count = 0;

    /*****************
    # YOUR CODE HERE #
    *****************/

    return count;
}
```

Bài tập 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.

In [ ]:

```
double* maximum(double* a, int size){
    double *max;
    max = a;
    if (a==NULL) return NULL;

    /*****************
    # YOUR CODE HERE #
    *****************/

    return max;
}
```

Bài tập 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].

In [ ]:

```
void reversearray(int arr[], int size){
    int l = 0, r = size - 1, tmp;

    /*****************
    # YOUR CODE HERE #
    *****************/
}
```

```
void ptr_reversearray(int *arr, int size){
    int l = 0, r = size - 1, tmp;

    /*****************
    # YOUR CODE HERE #
    *****************/
}
```

Phần 3. Cấp phát động

Bài tập 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.

In [ ]:

```
#include <stdio.h>

int *a;
int n, tmp;

int main(){
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    //#Allocate memory

    /*****************
    # 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

    /*****************
    # 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 tập 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(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`.

In [ ]:

```
#include <stdio.h>

void allocate_mem(int ***mt, int m, int n){
    //#Allocate memory for the matrix

    /*****************
    # YOUR CODE HERE #
    *****************/
}


void input(int **mt, int m, int n){
    //#Input elements of the matrix

    /*****************
    # YOUR CODE HERE #
    *****************/
}

void output(int **mt, int m, int n){
    //# Print all elements of the matrix

    /*****************
    # YOUR CODE HERE #
    *****************/

}

int process(int **mt, int m, int n){
    int tong = 0;
    //# Calculate the sum of all even elements in the matrix

    /*****************
    # YOUR CODE HERE #
    *****************/

    return tong;
}

void free_mem(int **mt, int m, int n){
    //# Free memory

    /*****************
    # YOUR CODE HERE #
    *****************/
}

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", process(mt, m, n));
```

```
    free_mem(mt, m, n);
    return 0;
}
```

Phần 4. Bài tập về nhà

Bài tập 9

Viết chương trình in ra tất cả các dãy con của một dãy cho trước. Ví dụ dãy 1 3 4 2 có các dãy con sau:

1
1 3
1 3 4
1 3 4 2
3
3 4
3 4 2
4
4 2
2

Bài tập 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.