I. Toán tử so sánh.

- Xem ví dụ sau:

  Ví dụ 1:

```
printf("%d\n", 2 > 1);
printf("%d", 2 < 1);
```

  Ví dụ 2:

```
if(1) printf("1\n");
if(0) printf("0\n");
if(2) printf("2\n");
if(-1) printf("-1\n");
```

➔ Số nguyên 0 tương đương với điều kiện sai, các số nguyên còn lại tương với điều kiện đúng.

Bài 1: Cho đoạn code sau. Hỏi màn hình sẽ hiển thị những gì sau khi chạy?

```
#include <stdio.h>
#include <string.h>

int main(){
    char s[40];
    strcpy(s, "BBBBBB");
    for(int i = 0; i < strlen(s); i++){
        if(i%2)
            s[i]++;
        else
            s[i]--;
    }
    puts(s);
}
```

➔ ACACAC

Bài 2: Cho đoạn code sau. Hỏi màn hình sẽ hiển thị những gì sau khi chạy?

```
#include <stdio.h>
#include <string.h>

int main(){
    char s1[40], s2[40];
    strcpy(s1, "BBBBBB");
    strcpy(s2, s1);
    for(int i = 0; i < strlen(s1); i++){
        if(i%2)
            s1[i]++;
        else
            s1[i]--;
    }

    for(int i = 0; i < strlen(s2); i++){
        if(i%3)
            s2[i] += 3;
        else
            s2[i] += 2;
    }

    strcat(s1, s2);
    puts(s1);
}
```

➔ **ACACACDEEDEE**

II. Phép gán.

- Xem ví dụ sau:

Ví dụ 1:

```
char a = -130;  //111...111 0111 1110
printf("%d\n", a);

int c = 260;    //00...00 0001 0000 0100
char b = c;
printf("%d\n", b);
```

➔ **Khi gán một số nguyên lớn hơn miền giá trị của biến số nguyên cho một biến số nguyên, thì biến số nguyên chỉ lấy số lượng bit đúng bằng số lượng bit cần dùng để lưu trữ 1 biến theo kiểu số nguyên đó, từ vị trí của bit LSB sáng trái.**

Ví dụ 2:

```
#include <stdio.h>
#include <string.h>

int main(){
    char a = 140.2345;
    printf("%d\n", a);

    char b = -150.123456;
    printf("%d\n", b);
}
```

➔ **Khi gán một số thập phân lớn hơn miền giá trị của biến số nguyên cho một biến số nguyên, thì biến sẽ lưu số nguyên lớn nhất có thể nếu số thập phân dương và lưu số nguyên nhỏ nhất có thể nếu số thập phân âm.**

Bài 1: Cho đoạn code sau. Hỏi màn hình sẽ hiển thị những gì sau khi chạy? Biết số 260 có mã nhịu phân là 0001 0000 0100.

```
char a = 260;
char b = 260.0;
printf("%d %d", a, b);
```

➔ **4 127**

Bài 2: Cho đoạn code sau. Hỏi màn hình sẽ hiển thị những gì sau khi chạy? Biết số 260 có mã nhị phân là 0001 0000 0100.

```
char a = 260.0;
int b = 260;

char *aPtr = &a;
char *bPtr = (char*)&b;

*aPtr -= 10;
*bPtr += 1;

printf("%d %d", *aPtr, *bPtr);
```

➔ **117 5**

III. Hàm đệ quy.

- Khái niệm: là hàm tự gọi chính nó.

  Ví dụ 1: Hàm nhận một tham số n, tính tổng từ 0 đến n.

```c
#include <stdio.h>

int sum(int n){
    if(n == 0){
        return 0;
    } else{
        return n + sum(n-1);
    }
}

int main(){
    printf("%d", sum(5));
}
```

Ví dụ 2: Hàm nhận vào một tham số n, tính tổng các số chẵn từ 0 đến n.

```c
#include <stdio.h>

int sumEven(int n){
    if(n == 0){
        return 0;
    } else{
        if(n % 2 == 0){
            return n + sumEven(n-1);
        } else{
            return sumEven(n-1);
        }
    }
}

int main(){
    printf("%d", sumEven(6));
}
```

Bài 1: Tạo hàm đệ quy nhận vào một tham số n, tính n!.

```c
int factorial(int n){
    if(n == 0){
        return 1;
    } else{
        return n * factorial(n-1);
    }
}
```

Bài 2: Tạo hàm đệ quy nhận vào 2 tham số n và x, tính n mũ x.

```c
int power(int n, int x){
    if(x == 0){
        return 1;
    } else{
        return n * power(n, x - 1);
    }
}
```

Bài 3: Cho đoạn code sau. Hỏi màn hình sẽ hiển thị những gì sau khi chạy và hàm thực hiện chức năng gì?

```c
#include <stdio.h>

void func(int n){
    if(n == 0){
        printf("0");
    } else{
        func(n/2);
        printf("%d", n%2);
    }
}

int main(){
    func(7);
}
```

➔ **0111**

Bài 4: Cho đoạn code sau. Hỏi màn hình sẽ hiển thị những gì sau khi chạy và hàm thực hiện chức năng gì?

```c
#include <stdio.h>

void func(int n){
    if(n != 0){
        func(n/8);
        printf("%d", n%8);
    } else{
        printf("0");
    }
}

int main(){
    func(9);
}
```

➔ **011**

Bài 5: Cho đoạn code sau. Hỏi màn hình sẽ hiển thị những gì sau khi chạy và hàm thực hiện chức năng gì?

```c
#include <stdio.h>

void func(int a, int b){
    if(a != 0){
        func(a/b, b);
        printf("%d", a%b);
    } else{
        printf("0");
    }
}

int main(){
    func(50, 7);
}
```

➔ **0101**

IV. Bài tập mảng:

Link bài tập: https://vnoi.info/wiki/algo/basic/two-pointers.md

Bài 1: Nhập vào 2 mảng sô nguyên a, b (kích thước không quá 50). Tạo hàm sắp xếp mảng theo thứ tự tăng dần, dùng hàm vừa tạo để sắp xếp 2 mảng a, b. Tạo hàm có chức năng tạo một mảng c gồm các phần tử trong 2 mảng a, b được sắp xếp theo thứ tự từ nhỏ đến lớn.

Bài 2: Nhập vào mảng a (kích thước không quá 50). Sắp xếp mảng a tăng dần. Nhập vào một số nguyên x, in ra màn hình một cặp số khác nhau bất kỳ thuộc mảng a sao cho tổng 2 số đó bằng x.

Bài 3: Nhập vào mảng a (kích thước không quá 50). Sắp xếp mảng a tăng dần. Tìm độ dài mảng con dài nhất trong dãy sao cho tổng các phần tử trong mảng con đó không quá 20.

Đáp án bài 1, 2, 3 tương ứng với hàm solve1(), solve2(), solve3():

```c
#include <stdio.h>

/* run this program using the console pauser or add your own getch, system("pause") or in
put loop */

//void getInput(int *a, int *n){
void getInput(int a[], int *n){
    do{
        printf("Moi nhap kich thuoc mang: ");
        scanf("%d", n);
    } while(*n <= 0 || *n >= 50);

    printf("Moi nhap cac phan tu:\n");
    for(int i = 0; i < *n; i++){
        scanf("%d", &a[i]);
//      scanf("%d", a+i);
    }
}

//void printData(int *a, int *n){
void printData(int a[], int n){
    printf("\n");
    for(int i = 0; i < n; i++){
        printf("%d ", a[i]);
//      printf("%d ", *(a+i));
    }
}

void sort(int a[], int n){
    for(int i = 0; i < n - 1; i++){
        for(int j = i + 1; j < n; j++){
            if(a[i] > a[j]){
                int k = a[i];
                a[i] = a[j];
                a[j] = k;
            }
        }
    }
}

void solve1(int c[], int a[], int b[], int n, int m){
```

```
    int i = 0, j = 0, k = 0;
    while(i < n || j < m){
        if((a[i] <= b[j] && i <= n) || j == m + 1){
            c[k] = a[i];
            i++;
            k++;
        } else{
            c[k] = b[j];
            j++;
            k++;
        }
    }
}

void solve2(int a[], int n){
    int x;

    printf("\nMoi nhap x: ");
    scanf("%d", &x);

    int i = 0, j = n-1;
    while(i < j){
        if(a[i] + a[j] == x){
            printf("%d - %d", a[i], a[j]);
            return;
        }
        if(a[i] + a[j] < x){
            i++;
        } else{
            j--;
        }
    }
    printf("No solution.");
}

void solve3(int a[], int n){
    int ans = 0, sum = 0;
    for(int l = 0, r = 0; r < n; r++){
        sum += a[r];
        while(sum > 20){
            sum -= a[l];
            l++;
        }
        ans = (ans > (r - l + 1)) ? ans : (r - l + 1);
    }
    printf("\nAnswer = %d\n", ans);
}

int main() {
    int a[50], b[50], n, m, c[100];

    getInput(a, &n);
    printData(a, n);
//  sort(a, n);
//  printData(a, n);

//  getInput(b, &m);
```

```
//  printData(b, m);
//  sort(b, m);
//  printData(b, m);

//  solve1(c, a, b, n, m);
//  printData(c, n + m);
//  solve2(a, n);
//  solve3(a, n);

    return 0;
}
```