I. Tạo mảng 1 chiều.

- Cú pháp:

  <Kiểu dữ liệu> <Tên mảng>[<Kích thước của mảng>];

  <Kiểu dữ liệu> <Tên mảng>[<Kích thước của mảng>] = {<Các gía trị>};

  Ví dụ:

```
int a[10];
int b[5] = {1, 2, 3, 4, 5};
char c[5] = {'H', 'C', 'M', 'U', 'T'};
char d[6] = "HCMUT";
double e[] = {1.2, 2.2, 3.3};  //mang co kich thuoc la 3
```

- **Lưu ý 1:** Nếu các phần tử trong mảng không được khởi tạo giá trị ngay từ đầu thì các phần tử trong mảng sẽ lưu giá trị 0.
- **Lưu ý 2:** Nếu không khai báo kích thước cho mảng thì chương trình sẽ tự tạo mảng với kích thước bằng số lượng giá trị được khởi tạo.
- Truy xuất đến các phần tử của mảng 1 chiều bằng cú pháp:

  <Tên mảng>[<chỉ số của phần tử cần truy xuất>]

- Chỉ số của các phần tử trong mảng sẽ được đánh số từ 0 trở lên.

  Ví dụ:

```
#include <stdio.h>

int main(){
    int a[10] = {1, 2, 3, 4, 5};
    a[0] = 0;
    printf("%d %d %d", a[0], a[4], a[9]);

    return 0;
}
```

Bài tập 1: Tạo một mảng có kích thước là 10 và cho nhập các giá trị của mảng. Hãy tính tổng giá trị các phần tử của mảng và in ra màn hình.

```
int a[10];

for(int i = 0; i < 10; i++){
    scanf("%d", &a[i]);
}

int sum = 0;
for(int i = 0; i < 10; i++){
    sum = sum + a[i]; // sum += a[i]
}

printf("%d", sum);
```

Bài tập 2: Tạo một mảng có kích thước là 10 và cho nhập các giá trị của mảng. Hãy tính tổng giá trị của phần tử đầu và phần tử cuối.

```
int a[10];
```

```
for(int i = 0; i < 10; i++){
    scanf("%d", &a[i]);
}

printf("%d", a[0] + a[9]);
```

Bài tập 3: Nhập vào số nguyên n là kích thước của mảng và các giá trị của mảng. In ra màn hình lần lượt là chỉ số và giá trị của phần tử có giá trị nhỏ nhất trong mảng.

```
int n, a[100]; // 0 < n <= 100

do{
    printf("Moi nhap n: ");
    scanf("%d", &n);
} while(n <= 0 || n > 100); // || <=> or

for(int i = 0; i < n; i++){
    scanf("%d", &a[i]);
}

int p, min = 32767; // [-32768; 32767]
for(int i = 0; i < n; i++){
    if(a[i] < min){
        min = a[i];
        p = i;
    }
}
printf("%d: %d", p, min);
```

Bài tập 4: Nhập vào số nguyên n là kích thước của mảng và các giá trị của mảng. In ra các số lẻ trong mảng.

```
int n, a[100]; // 0 < n <= 100

do{
    printf("Moi nhap n: ");
    scanf("%d", &n);
} while(n <= 0 || n > 100); // || <=> or

for(int i = 0; i < n; i++){
    scanf("%d", &a[i]);
}

for(int i = 0; i < n; i++){
    if(a[i] % 2 == 1){
        printf("%d ", a[i]);
    }
}
```

Bài tập 5: Nhập vào số nguyên n là kích thước của mảng và các giá trị của mảng, nhập một số nguyên k. In ra màng hình số lượng các phần tử trong mảng có giá trị bằng k.

```
int n, a[100]; // 0 < n <= 100

do{
    printf("Moi nhap n: ");
    scanf("%d", &n);
} while(n <= 0 || n > 100); // || <=> or

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

```
    scanf("%d", &a[i]);
}

int k;
printf("Moi nhap k: ");
scanf("%d", &k);

int count = 0;
for(int i = 0; i < n; i++){
    if(a[i] == k){
        count = count + 1; // count++ // count += 1
    }
}
```

Bài tập 6: Nhập vào số nguyên n là kích thước của mảng và các giá trị của mảng. In ra màn hình tổng giá trị các phần tử là số dương lẻ.

```
int n, a[100]; // 0 < n <= 100

do{
    printf("Moi nhap n: ");
    scanf("%d", &n);
} while(n <= 0 || n > 100); // || <=> or

for(int i = 0; i < n; i++){
    scanf("%d", &a[i]);
}

int sum = 0;

for(int i = 0; i < n; i++){
    if(a[i] > 0 && a[i] % 2 == 1){
        sum = sum + a[i]; // sum += a[i]
    }
}

printf("%d", sum);
```

Bài tập 7: Nhập vào số nguyên n là kích thước của mảng và các giá trị của mảng. In ra màng hình các phần tử có giá trị thuộc đoạn [0,10].

```
int n, a[100]; // 0 < n <= 100

do{
    printf("Moi nhap n: ");
    scanf("%d", &n);
} while(n <= 0 || n > 100); // || <=> or

for(int i = 0; i < n; i++){
    scanf("%d", &a[i]);
}

for(int i = 0; i < n; i++){
    if(a[i] >= 0 && a[i] <= 10){
        printf("%d ", a[i]);
    }
}
```

II. Mảng 2 chiều.

Ví dụ:

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

```c
int main(){
    int a[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9},
    };

    a[0][0] = 0;

    printf("%d %d %d", a[0][0], a[1][0], a[2][1]);

    return 0;
}
```

- **Lưu ý:** Khi tạo hàm với tham số là mảng thì chiều đầu tiên của mảng không cần xác định cụ thể, từ chiều thứ 2 trở đi thì cần xác định cụ thể.

Ví dụ: void sumArray(int a[][4], int n) {}

void sortArray(int a[], int n) {}

III. Chuỗi ký tự.

- Chuỗi ký tự là mảng ký tự được kết thúc bằng một ký tự NULL (‘\0’).
- **Ví dụ:** char s[] = “Hello”;

char c[] = {‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’};

- Nhập một chuỗi: **scanf(“%s”, <Tên mảng ký tự>);**
- Tuy nhiên, hàm scanf() không thể nhận được chuỗi chứa dấu cách, dấu tab và ‘\n’. Để nhập chuỗi chứa các ký tự trên: **gets(<Tên mảng>);**
- Để xuất một chuỗi, có 2 cách: **printf(“%s”, <Tên mảng ký tự>);**

puts(<Tên mảng>);

- Tuy nhiên, có một số trường hợp chương trình không cho nhập chuỗi, để khắc phục cần nhập lệnh sau phía trước lên gets(): **fflush(stdin);**

Ví dụ:

```c
#include <stdio.h>

int main(){
    int n;
    char s[10];

    scanf("%d", &n);
    fflush(stdin);
    gets(s);

    return 0;
}
```

Các hàm trong thư viện string.h:

- Hàm strcpy(), cú pháp: **strcpy(<Chuỗi 1>, <Chuỗi 2>);**

➔ Gán **<Chuỗi 2>** vào **<Chuỗi 1>**.

- Hàm strcat(), cú pháp: **strcat(<Chuỗi 1>, <Chuỗi 2>);**

  ➔ Nối **<Chuỗi 2>** vào đuôi của **<Chuỗi 1>**.

- Hàm strlen(), cú pháp: **strlen(<Chuỗi>);**

  ➔ Trả về kích thước của **<Chuỗi>**.

- Hàm strcmp(), cú pháp: **strcmp(<Chuỗi 1>, <Chuỗi 2>);**

  ➔ Nếu **<Chuỗi 1>** giống **<Chuỗi 2>** thì trả về gián trị 0, ngược lại sẽ trả về giá trị khác 0.

- Link tham khảo thư viện string.h: https://nguyenvanhieu.vn/cac-ham-trong-thu-vien-string-h/

Một số hàm khác có liên quan:

- Hàm **getchar()**: cho phép nhập vào 1 ký tự, các ký tự được nhập sẽ được hiển thị ra màn hình, đợi nhấn Enter rồi sẽ trả về ký tự được nhập và kết thúc hàm. Nếu nhập nhiều hơn 1 ký tự thì chỉ nhân ký tự đầu tiên.

Ví dụ:

```c
#include <stdio.h>

int main(){
    char c;
    c = getchar();
    printf("%c", c);
    return 0;
}
```

- Hàm **getche()**: cho phép nhập vào 1 ký tự, ký tự được nhập sẽ được hiển thị ra màn hình, hàm không đợi nhấn phím Enter mà sẽ ngay lập tức trả về ký tự vừa nhập và kết thúc.

Ví dụ:

```c
#include <stdio.h>

int main(){
    char c;
    c = getche();
    printf("%c", c);
    return 0;
}
```

- Hàm **getch()**:cho phép nhập vào 1 ký tự, ký tự được nhập sẽ được hiển thị ra màn hình, hàm không đợi nhấn phím Enter mà sẽ ngay lập tức trả về ký tự vừa nhập và kết thúc.

Ví dụ:

```c
#include <stdio.h>

int main(){
    char c;
    c = getch();
```

```
    printf("%c", c);
    return 0;
}
```

Ví dụ: Viết chương trình cho phép nhập các ký tự từ bàn phím và in ra mã ASCII của các ký tự đó, chương trình sẽ lặp lại cho đến khi nhấn phím ESC (mã ASCII của phím ESC là 27).

```
#include <stdio.h>

int main(){
    char c;
    do{
        c = getch();
        printf("%d\n", c);
    } while(c != 27);
}
```

Bài tập 1: Nhập vào một chuỗi s và một sô nguyên k. In ra màn hình ký tự thứ k trong chuỗi.

```
char s[100];
int k;

gets(s);

// 0 < k <= strlen(s)
do{
    printf("Moi nhap k: ");
    scanf("%d", &k);
} while(k <= 0 || k > strlen(s));

printf("%c", s[k-1]);
```

Bài tập 2: Nhập vào một chuỗi s và một ký tự c. In ra màn hình số lần xuất hiện của ký tự c trong chuỗi.

```
char s[100];
char c;

gets(s);
printf("Moi nhap c: ");
scanf("%c", &c);

int count = 0;
for(int i = 0; i < strlen(s); i++){
    if(s[i] == c){
        count++; // count += 1 // coutn = count + 1
    }
}

printf("%d", count);
```

Bài tập 3: Nhập vào một chuỗi s và một ký tự c. In ra màn hình vị trí xuất hiện cuối cùng của ký tự c trong chuỗi s, vị trí bắt đầu tính từ số 1, nếu không có thì in ra 0.

```
char s[100];
char c;

gets(s);
printf("Moi nhap c: ");
scanf("%c", &c);
```

```
int a = 0;
for(int i = 0; i < strlen(s); i++){
    if(s[i] == c){
        a = i + 1;
    }
}
printf("%d", a);
```

Bài tập 4: Nhập vào một chuỗi s và một ký tự c. In ra màn hình vị trí xuất hiện đầu tiên của ký tự c trong chuỗi s, vị trí bắt đầu tính từ số 1, nếu không có thì in ra 0.

```
char s[100];
char c;

gets(s);
printf("Moi nhap c: ");
scanf("%c", &c);

int a = 0;
for(int i = 0; i < strlen(s); i++){
    if(s[i] == c){
        a = i + 1;
        break;
    }
}
printf("%d", a);
```

Bài tập 5: Nhập vào một chuỗi s. Chuyển đổi tất cả các ký tự ‘e’ thành ‘3’ rồi in ra chuỗi đã chuyển đổi.

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

int main(){
    char s[100];
    gets(s);

    for(int i = 0; i < strlen(s); i++){
        if(s[i] == 'e'){
            s[i] = '3';
        }
    }

    puts(s);

    return 0;
}
```

IV. Hàm.

- Cú pháp 1: hàm chỉ thực hiện các lệnh mà không trả về giá trị.

  void <Tên hàm>(<Kiểu dữ liệu 1> <Tham số 1>, <Kiểu dữ liệu 2> <Tham số 2>){

  <Các câu lệnh trong hàm>;

  }

- Cú pháp 2: hàm có trả về một giá trị.

  <Kiểu dữ liệu> <Tên hàm>(<Kiểu dữ liệu 1> <Tham số 1>, <Kiểu dữ liệu 2> <Tham số 2>){

```
    <Các câu lệnh trong hàm>;
    return <Giá trị muốn trả về>;
}
```

- Kiểu dữ liệu của hàm: là kiểu dữ liệu của giá trị mà hàm trả về.
- Tham số là các biến được sử dụng để truyền dữ liệu vào hàm khi gọi hàm, để các câu lệnh trong hàm sử dụng; khi gọi hàm phải cung cấp đủ tham số mà hàm yêu cầu, nếu không chương trình sẽ báo lỗi.
- Lệnh **return**, có 2 cách dùng: `return;`

  `return <Giá trị muốn trả về>;`
- Khi gặp lệnh return, chương trình sẽ ngay lập tức thoát khỏi hàm đang chạy, nếu lệnh return đi kèm với **<Giá trị muốn trả về>** thì chương trình sẽ trả về giá trị đó rồi mới thoát khỏi hàm.
- Trước khi sử dụng hàm thì phải tạo hàm.
- Cách gọi hàm: **<Tên hàm>(<Danh sách tham số>);**

Ví dụ 1: Tạo hàm gồm 2 tham số nguyên a, b (a < b) và in ra màn hình các số nguyên từ a đến b.

```c
#include <stdio.h>

void print(int a, int b){
    for(int i = a; i <= b; i++){
        printf("%d ", i);
    }
}

int main(){
    int a, b;
    scanf("%d%d", &a, &b);
    print(a, b);

    return 0;
}
```

Ví dụ 2: Tạo hàm gồm 2 tham số nguyên a, b (a < b) và tính tổng các số nguyên từ a đến b.

```c
#include <stdio.h>

int sum(int a, int b){
    int s = 0;
    for(int i = a; i <= b; i++){
        s += i;
    }
    return s;
}

int main(){
    int a, b;
    scanf("%d%d", &a, &b);
    int s = sum(a, b);
```

```
    printf("\n%d", s);
    printf("\n%d", sum(a, b));

    return 0;
}
```

Bài tập 1: Cho đoạn chương trình bên dưới. Điền vào chỗ … đoạn chương trình phù hợp để chương trình in ra màn hình các số chia hết cho cả 2 và 3 trong đoạn [0,30].

```
#include <stdio.h>
. . .
int main(){
    show();
    return 0;
}
```

Giải:

```
#include <stdio.h>

void show(){
    for(int i = 0; i <= 30; i++){
        if(i % 6 == 0){
            printf("%d ", i);
        }
    }
}

int main(){
    show();
    return 0;
}
```

Bài tập 2: Cho đoạn chương trình bên dưới. Điền vào chỗ … đoạn chương trình phù hợp để chương trình thực hiện: Khi nhập vào 1 chuỗi ký tự name không chứa dấu cách, sẽ in ra màn hình chuỗi “Hello [name] !!!”.

```
#include <stdio.h>

. . .

int main(){
    char name[100];
    scanf("%s", name);
    show(name);

    return 0;
}
```

Giải:

```
#include <stdio.h>

void show(char name[]){
    printf("Hello %s!!!", name);
}

int main(){
    char name[100];
    scanf("%s", name);
    show(name);

    return 0;
```

```
}
```

Bài tập 3: Cho đoạn chương trình bên dưới. Điền vào chỗ … đoạn chương trình phù hợp để chương trình thực hiện: Khi nhập vào 2 chuỗi ký tự name1 và name2 không chứa dấu cách, sẽ in ra màn hình chuỗi "Hello [name1] !!!\nHi [name2] !!!".

```
#include <stdio.h>

. . .

int main(){
    char name1[100], name2[100];
    scanf("%s%s", name1, name2);
    show(name1, name2);

    return 0;
}
```

Giải:

```
#include <stdio.h>

void show(char name1[], char name2[]){
    printf("Hello %s!!!\nHi %s!!!", name1, name2);
}

int main(){
    char name1[100], name2[100];
    scanf("%s%s", name1, name2);
    show(name1, name2);

    return 0;
}
```

Bài tập 4: Viết hàm nhận vào một số nguyên, kiểm tra xem số đó có phải là số nguyên tố hay không. Nếu là số nguyên tố thì trả về giá trị 1, nếu không phải thì trả về giá trị 0.

Giải:

```
int isPrime(int x){
    if(x < 2){
        return 0;
    }
    for(int i = 2; i < x; i++){
        if(x % i == 0){
            return 0;
        }
    }
    return 1;
}
```

Bài tập 5: Cho đoạn chương trình bên dưới. Điền vào chỗ … đoạn chương trình phù hợp để chương trình thực hiện: Khi nhập vào số nguyên n (0 < n < 100) là chiều dài của một mảng số nguyên và các giá trị của mảng số nguyên đó, chương trình sẽ in ra màn hình dãy số đã được sắp xếp. Nếu nhập n sai thì yêu cầu nhập lại.

```
#include <stdio.h>

void sortArray(. . .){
```

```
    . . .
}

int main(){
    int n, a[100];

    do{
        printf("Moi nhap n: ");
        scanf("%d", &n);
    } while(n <= 0 || n >= 100);

    for(int i = 0; i<n; i++){
        scanf("%d", &a[i]);
    }

    sortArray(a, n);

    return 0;
}
```

Giải:

```
#include <stdio.h>

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

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

int main(){
    int n, a[100];

    do{
        printf("Moi nhap n: ");
        scanf("%d", &n);
    } while(n <= 0 || n >= 100);

    for(int i = 0; i<n; i++){
        scanf("%d", &a[i]);
    }

    sortArray(a, n);

    return 0;
}
```

Bài tập 6: Cho đoạn chương trình bên dưới. Điền vào chỗ … đoạn chương trình phù hợp để chương trình thực hiện: Khi nhập vào số nguyên n (0 < n < 100) là chiều dài của một mảng số nguyên và các giá trị của mảng số nguyên đó, chương trình sẽ tính tổng

các phần tử là số nguyên tố trong mảng rồi in ra màn hình. Nếu nhập n sai thì yêu cầu nhập lại.

```
#include <stdio.h>

int n, a[100];

void getInput(){
    . . .
}

int isPrime(int x){
    . . .
}

int sum(){
    . . .
}

int main(){
    getInput();
    printf("%d", sum());

    return 0;
}
```

Giải:

```
#include <stdio.h>

int n, a[100];

void getInput(){
    do{
        printf("Moi nhap n: ");
        scanf("%d", &n);
    } while(n <= 0 || n >= 100);

    for(int i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
}

int isPrime(int x){
    if(x < 2){
        return 0;
    }
    for(int i = 2; i < x; i++){
        if(x % i == 0){
            return 0;
        }
    }
    return 1;
}

int sum(){
    int sum = 0;
    for(int i = 0; i < n; i++){
        if(isPrime(a[i]) == 1){
            sum += a[i];
        }
    }
    return sum;
```

```
}


int main(){
    getInput();
    printf("%d", sum());

    return 0;
}
```

Bài tập 7: Cho đoạn chương trình bên dưới. Điền vào chỗ … đoạn chương trình phù hợp để chương trình thực hiện: Khi nhập vào số dòng m, số cột n (0 < m, n < 20) và các giá trị của mảng số nguyên đó, chương trình sẽ tính trung bình các phần tử trong mảng. Nếu nhập m, n sai thì yêu cầu nhập lại.

Giải:

```
#include <stdio.h>

int m, n, a[20][20];

void getInput(){
    do{
        printf("Moi nhap m: ");
        scanf("%d", &m);
    } while(m <= 0 || m >= 20);

    do{
        printf("Moi nhap n: ");
        scanf("%d", &n);
    } while(n <= 0 || n >= 20);

    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            scanf("%d", &a[i][j]);
        }
    }
}

double calcAvr(){
    double avr = 0.0;
    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            avr += a[i][j];
        }
    }
    return avr/(m*n);
}

int main(){
    getInput();
    printf("%lf", calcAvr());

    return 0;
}
```

Bài tập 8: Cho đoạn chương trình bên dưới. Điền vào chỗ … đoạn chương trình phù hợp để chương trình thực hiện: Khi nhập vào số dòng m, số cột n (0 < m, n < 20) và các giá trị của mảng số nguyên đó, chương trình sẽ tính trung bình các phần tử là số nguyên tố trong mảng. Nếu nhập m, n sai thì yêu cầu nhập lại.

Giải:

```c
#include <stdio.h>

int m, n, a[20][20];

void getInput(){
    do{
        printf("Moi nhap m: ");
        scanf("%d", &m);
    } while(m <= 0 || m >= 20);

    do{
        printf("Moi nhap n: ");
        scanf("%d", &n);
    } while(n <= 0 || n >= 20);

    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            scanf("%d", &a[i][j]);
        }
    }
}

int isPrime(int x){
    if(x < 2){
        return 0;
    }
    for(int i = 2; i < x; i++){
        if(x % i == 0){
            return 0;
        }
    }
    return 1;
}

double calcAvrPrime(){
    double avr = 0.0;
    int count  = 0;
    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            if(isPrime(a[i][j]) == 1){
                avr += a[i][j];
                count++;
            }
        }
    }
//  Cách 1:
    if(count == 0){
        return 0;
    } else{
        return avr/count;
    }

//  Cách 2: Dùng toán tử 3 ngôi (xem trong file buổi 1).
//  return (count == 0) ? 0 : avr/count;
}
```

```c
int main(){
    getInput();
    printf("%lf", calcAvrPrime());

    return 0;
}
```