LECTURE 05

ARRAY

Big–O Coding

Website: www.bigocoding.com

Giới thiệu

BT1 – TÍNH TỔNG

- Viết chương trình đọc vào số nguyên dương n là số lượng bài post trên Facebook của một người dùng, và mảng a có n phần tử tương ứng với n bài post, mỗi phần tử là số lượng like có được của bài post đó.
- Tính tổng số lượng like của tất cả n bài post.
- VD:
  - Input:
    - 4
    - 3 3 2 1
  - Output: 9

BT1 – Gợi ý

- BT1 có thể giải sử dụng **kĩ thuật tính tổng** ở buổi 4, không cần sử dụng thêm kiến thức mới.

1. Đọc n, số lượng bài post
2. **ans = 0**
3. Sử dụng vòng lặp for i in range(0, n)
   a. Đọc x, số like của 1 bài post
   b. **Cộng dồn x vào ans**.
4. In kết quả

BT1 – Vấn đề

- Làm thế nào để **tách biệt** quá trình **đọc danh sách** phần tử và quá trình **cộng giá trị** các phần tử?
- Làm thế nào để **lưu trữ 1 mảng các phần tử** (trong trường hợp này là mảng số nguyên)?
- Khai báo n biến x1, x2, x3,..., xn?
  - Không khả thi, làm sao biết trước n bao nhiêu?
- Giải pháp: sử dụng kiểu dữ liệu mảng có trong mỗi ngôn ngữ lập trình.

Khai báo và sử dụng mảng

Khái niệm mảng 1 chiều

- Dùng để biểu diễn một dãy các đối tượng dữ liệu liên tiếp nhau.
- Chiều cao các bạn trong lớp (cm)

a

150	190	181	165

- Cân nặng của các bạn trong lớp (kg)

b

30.5	85.6	77	45.2

- Thông tin các sản phẩm trong giỏ hàng.

c

Táo 1.0 50,000	Cam 0.5 31,000	Mít 2.2 40,000	Nho 1.0 100,000

Tạo mảng số nguyên 3 phần tử (C++)

```cpp
int a[] = {10, 20, 30};
int n = 3;
```

```cpp
int a[100];
int n = 3;

a[0] = 10;
a[1] = 20;
a[2] = 30;
```

- Phải khai báo kiểu dữ liệu của mỗi phần tử. (int, double, Product…)
- Số phần tử trong mảng phải cụ thể, xác định lúc biên dịch, không dùng biến hoặc hằng số.
- Khai báo thêm biến n, số nguyên, số phần tử trong mảng a.
  - n ≤ 100

```cpp
int n;
cin >> n;
int a[n];
```

X

```cpp
int n = 3;
int a[n];
```

X

```cpp
const int n = 3;
int a[n];
```

X

Tạo mảng số nguyên 3 phần tử (Java)

```
int []a = {10, 20, 30};
```

```
int []a;
int n = 3;
a = new int[n];
a[0] = 10;
a[1] = 20;
a[2] = 30;
```

- Phải khai báo kiểu dữ liệu của mỗi phần tử. (int, double, Product…)
- Số phần tử trong mảng được xác định khi chạy chương trình.
- Không cần khai báo thêm biến n.
  - Thuộc tính a.length: số phần tử trong mảng.

Tạo mảng số nguyên 3 phần tử (Python)

```
a = [10, 20, 30]
```

```
a = []
a.append(10)
a.append(20)
a.append(30)
```

- Không cần khai báo kiểu dữ liệu.
- Số phần tử trong mảng được xác định khi chạy chương trình.
- Không cần khai báo thêm biến n.
  - Hàm len(a): số phần tử trong mảng.

Tạo mảng rỗng

- Mảng rỗng là mảng không có phần tử nào.

```
int a[100];
int n = 0;
```

```
a = new int[0];
System.out.print(a.length);
```

```
a = []
```

Xác định số phần tử của mảng

```cpp
int a[100] = {10, 20, 30};
int n = 3;
cout << n;
```

```java
int []a = {10, 20, 30};
System.out.print(a.length);
```

```python
a = [10, 20, 30]
print(len(a))
```

Truy xuất 1 phần tử trong mảng

- Sử dụng toán tử **[index]** để truy xuất 1 phần tử trong mảng.
- index: chỉ số các phần tử trong mảng bắt đầu từ 0.
- Như vậy, nếu mảng lst có n phần tử, thì chỉ số các phần tử trong mảng là 0, 1, 2, 3,..., n−2, n−1.

VD: Index hợp lệ

```cpp
int a[] = {10, 20, 30, 40};
int n = 4;

cout << a[0];
cout << a[3];

a[3] = 77;
```

```java
int []a = {10, 20, 30, 40};

System.out.print(a[0]);
System.out.print(a[3]);

a[3] = 77;
```

```python
a = [10, 20, 30, 40]

print(a[0])
print(a[3])

a[3] = 77
```

VD: Index out of range

```cpp
int a[] = {10, 20, 30, 40};
int n = 4;
cout << a[4];
cout << a[10];
```

```
// Giá trị rác
```

```java
int []a = {10, 20, 30, 40};
System.out.print(a[4]);
System.out.print(a[10]);
```

```
java.lang.ArrayIndexOutOfBoundsException
```

```python
a = [10, 20, 30, 40]
print(a[4])
print(a[10])
```

```
IndexError: list index out of range
```

VD: Index là số âm (valid)

Big-O Green

```cpp
int a[] = {10, 20, 30, 40};
int n = 4;
cout << a[-1];
cout << a[-2];
cout << a[-3];
cout << a[-4];
```

```
// Giá trị rác
```

```java
int []a = {10, 20, 30, 40};
System.out.print(a[-1]);
System.out.print(a[-2]);
System.out.print(a[-3]);
System.out.print(a[-4]);
```

```
java.lang.ArrayIndexOutOfBoundsException
```

```python
a = [10, 20, 30, 40]
print(a[-1])
print(a[-2])
print(a[-3])
print(a[-4])
```

```
40 30 20 10
```

VD4: Index là số âm (invalid)

```cpp
int a[100] = {10, 20, 30, 40};
int n = 4;
cout << a[-5];
cout << a[-10];
```

```
// Giá trị rác
```

```java
int []a = {10, 20, 30, 40};
System.out.print(a[-5]);
System.out.print(a[-10]);
```

```
java.lang.ArrayIndexOutOfBoundsException
```

```python
a = [10, 20, 30, 40]
print(a[-5])
print(a[-10])
```

```
IndexError: list index out of range
```

Duyệt qua các phần tử trong mảng (C1)

```cpp
int a[] = {10, 20, 30, 40};
int n = 4;

for(int i = 0; i < n; i++){
    cout << a[i] << endl;
}
```

```java
int []a = {10, 20, 30, 40};

for(int i = 0; i < a.length; i++){
    System.out.println(a[i]);
}
```

```python
a = [10, 20, 30, 40]

for i in range(len(a)):
    print(a[i])
```

Duyệt qua các phần tử trong mảng (C2)

```
// C++ không hỗ trợ cách duyệt này
```

```
int []a = {10, 20, 30, 40};

for(int x: a){
    System.out.println(x);
}
```

```
a = [10, 20, 30, 40]

for x in a:
    print(x)
```

Đọc mảng số nguyên, cùng 1 dòng, biết trước n

```
4
10 20 30 40
```

```
int a[100];
int n = 0;
cin >> n;
for(int i = 0; i < n; i++)
    cin >> a[i];
```

```
int []a;
int n = 0;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
a = new int[n];

for(int i = 0; i < n; i++)
    a[i] = sc.nextInt();
```

```
n = int(input())
a = list(map(int, input().split()))
```

Đọc mảng số nguyên, n dòng, biết trước n

```
4
10
20
30
40
```

```cpp
int a[100];
int n = 0;
cin >> n;
for(int i = 0; i < n; i++)
    cin >> a[i];
```

```java
int []a;
int n = 0;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
a = new int[n];

for(int i = 0; i < n; i++)
    a[i] = sc.nextInt();
```

```python
n = int(input())
a = []
for i in range(n):
    x = int(input())
    a.append(x)
```

Đọc mảng số nguyên, n dòng, không biết trước n (C++, Python)

```
10
20
30
40
0
```

```cpp
int a[100];
int n = 0;
int x = 0;
while(true){
    cin >> x;
    if(x == 0){
        break;
    }
    a[n] = x;
    n++;
}
```

```python
a = []
while(True):
    x = int(input())
    if(x == 0):
        break
    a.append(x)
```

Đọc mảng số nguyên, n dòng, không biết trước n (Java)

```
10
20
30
40
0
```

```java
int []a;

// Đọc vào mảng tạm b, để biết n
int []b = new int[100];
int n = 0;
int x = 0;
Scanner sc = new Scanner(System.in);
while(true){
    x = sc.nextInt();
    if(x == 0){
        break;
    }
    b[n] = x;
    n++;
}

// Copy từ mảng tạm b sang mảng a
a = new int[n];
for(int i = 0; i < n;i++){
    a[i] = b[i];
}
```

Các kĩ thuật xử lí trên mảng

BT1 – TÍNH TỔNG

- Viết chương trình đọc vào số nguyên dương n là số lượng bài post trên Facebook của một người dùng, và mảng a có n phần tử tương ứng với n bài post, mỗi phần tử là số lượng like có được của bài post đó.
- Tính tổng số lượng like của tất cả n bài post.

Kĩ thuật tính tổng / tính tích

BT1 – Gợi ý

1. Đọc mảng a.
2. Khởi tạo biến tổng: **ans = 0**.
3. Sử dụng vòng lặp for, duyệt qua các phần tử trong mảng a.
   - **ans += a[i]**.
4. In kết quả.

BT1 – Minh họa – Tính tổng like

15	19	8	7	20	6

ans = 0
i = 0

i = 0 < n = 6: True
ans = ans + a[i] = 0 + 15 = 15
i = i + 1 = 0 + 1 = 1

i = 1 < n = 6: True
ans = ans + a[i] = 15 + 19 = 34
i = i + 1 = 1 + 1 = 2

i = 2 < n = 6: True
ans = ans + a[i] = 34 + 8 = 42
i = i + 1 = 2 + 1 = 3

i = 3 < n = 6: True
ans = ans + a[i] = 42 + 7 = 49
i = i + 1 = 3 + 1 = 4

i = 4 < n = 6: True
ans = ans + a[i] = 49 + 20 = 69
i = i + 1 = 4 + 1 =5

i = 5 < n = 6: True
ans = ans + a[i] = 69 + 6 = 75
i = i + 1 = 5 + 1 = 6

i = 6 < n = 6: False

BT2 – TÌM POST NHIỀU LIKE NHẤT

- Viết chương trình đọc vào số nguyên dương n là số lượng bài post trên Facebook, và mảng a có n phần tử tương ứng với n bài post, mỗi phần tử là số lượt like có được của bài post đó.
- Bạn hãy tìm bài post có nhiều lượt like nhất.

Kĩ thuật đặt lính canh

BT2 – Gợi ý

1. Đọc mảng a.
2. Khởi tạo lính canh là phần tử đầu tiên: **ans = a[0]**.
3. Sử dụng vòng lặp for, duyệt qua các phần tử trong mảng a.
   - Nếu phần tử a[i] > lính canh ans, cập nhật **ans = a[i]**.
4. In kết quả.

BT2 – Minh họa – Tìm post nhiều like nhất

15	19	1	7	20	6

ans = 15
i = 0

i = 0 < n = 6: True
a[i] = 15 > ans = 15: False
i = i + 1 = 0 + 1 = 1

i = 1 < n = 6: True
a[i] = 19 > ans = 15: True, ans = 19
i = i + 1 = 1 + 1 = 2

i = 2 < n = 6: True
a[i] = 1 > ans = 19: False
i = i + 1 = 2 + 1 = 3

i = 3 < n = 6: True
a[i] = 7 > ans = 19: False
i = i + 1 = 3 + 1 = 4

i = 4 < n = 6: True
a[i] = 20 > ans = 19: True, ans = 20
i = i + 1 = 4 + 1 = 5

i = 5 < n = 6: True
a[i] = 6 > ans = 20: False
i = i + 1 = 5 + 1 = 6

i = 6 < n = 6: False

BT3 – LIỆT KÊ CHIA HẾT CHO 2

- Liệt kê các số chia hết cho 2 trong mảng một chiều các số nguyên.

Kĩ thuật liệt kê

BT3 – Gợi ý

1. Đọc mảng a.

2. Sử dụng vòng lặp for, duyệt qua các phần tử a[i] trong mảng a.
   - **Nếu a[i] là số chẵn, in a[i].**

BT3 – Minh họa – Liệt kê số chẵn

15	19	8	7	20	6

i = 0

i = 0 < n = 6: True
a[i] = 15 % 2 == 0: False
i = i + 1 = 0 + 1 = 1

i = 1 < n = 6: True
a[i] = 19 % 2 == 0: False
i = i + 1 = 1 + 1 = 2

i = 2 < n = 6: True
a[i] = 8 % 2 == 0: True, In 8
i = i + 1 = 2 + 1 = 3

i = 3 < n = 6: True
a[i] = 7 % 2 == 0: False
i = i + 1 = 3 + 1 = 4

i = 4 < n = 6: True
a[i] = 20 % 2 == 0: True, In 20
i = i + 1 = 4 + 1 = 5

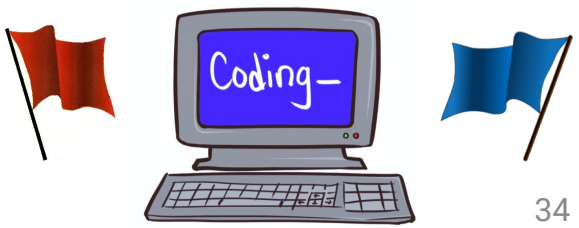
i = 5 < n = 6: True
a[i] = 6 % 2 == 0: True, In 6
i = i + 1 = 5 + 1 = 6

i = 6 < n = 6: False

BT4 – KIỂM TRA LƯỢT LIKE CỦA POST

- Viết chương trình đọc vào số nguyên dương n với n là số lượng bài post trên Facebook, và mảng a có n phần tử tương ứng với n bài post, mỗi phần tử là số lượt like có được của bài post đó.
- Kiểm tra xem có phải tất cả các bài post đều có người like hay không (có thể có bài post có 0 like)?

Kĩ thuật đặt cờ hiệu



BT4 – Gợi ý

1. Đọc mảng a.
2. Khởi tạo cờ hiệu: **allPositive = True**.
3. Sử dụng vòng lặp for, duyệt qua các phần tử a[i] trong mảng a.
   - Nếu a[i] == 0, cập nhật cờ hiệu **allPositive = False**.
4. Dựa vào biến allPositive, in kết quả.

BT4 – Minh họa

15	19	0	7	20	6

allPositive = True
i = 0

i = 0 < n = 6: True
a[i] = 15 == 0: False
i = i + 1 = 0 + 1 = 1

i = 1 < n = 6: True
a[i] = 19 == 0: False
i = i + 1 = 1 + 1 = 2

i = 2 < n = 6: True
a[i] = 0 == 0: True, allPostive = False
i = i + 1 = 2 + 1 = 3

i = 3 < n = 6: True
a[i] = 7.8 == 0: False
i = i + 1 = 4 + 1 = 4

i = 4 < n = 6: True
a[i] = 20 == 0: False
i = i + 1 = 5 + 1 = 5

i = 5 < n = 6: True
a[i] = 6 == 0: False
i = i + 1 = 5 + 1 = 6

i = 6 < n = 6: False

BT5 – ĐẾM SỐ NGUYÊN TỐ

- Đếm xem mảng có bao nhiêu số nguyên tố.

Kĩ thuật đếm

BT5 – Gợi ý

1. Đọc mảng a.
2. Khởi tạo biến đếm: **count = 0**.
3. Sử dụng vòng lặp for, duyệt qua các phần tử x trong mảng a.
   - **Nếu isPrime(x), tăng biến count lên 1**.
4. In kết quả.

BT5 – Minh họa – Đếm số nguyên tố

15	19	8	7	20	6

count = 0
i = 0

i = 0 < n = 6: True
isPrime(15): False
i = i + 1 = 0 + 1 = 1

i = 1 < n = 6: True
isPrime(19): True, count = 0 + 1 = 1
i = i + 1 = 1 + 1 = 2

i = 2 < n = 6: True
isPrime(8): False
i = i + 1 = 2 + 1 = 3

i = 3 < n = 6: True
isPrime(7): True, count = 1 + 1 = 2
i = i + 1 = 3 + 1 = 4

i = 4 < n = 6: True
isPrime(20): False
i = i + 1 = 4 + 1 = 5

i = 5 < n = 6: True
isPrime(6): False
i = i + 1 = 5 + 1 = 6

i = 6 < n = 6: False

Một số lưu ý trên mảng

Copy dữ liệu giữa 2 mảng (C++)

- Nếu gán b = a, source code bị lỗi compiled-time error.
- Phải dùng vòng lặp, copy từ giá trị a[i] qua b[i]

```
int a[100] = {10, 20, 30};
int n = 3;

int b[100];
int m = n;
for(int i = 0; i < m; i++){
    b[i] = a[i];
}
```

Copy dữ liệu giữa 2 mảng (Java)

- Nếu gán b = a, thay đổi giá trị mảng a gây ảnh hưởng giá trị mảng b và ngược lại.
- Tạo mảng b có kích thước bằng mảng a, sử dụng vòng lặp gán b[i] = a[i].

```
int []a = {10, 20, 30};

int []b;
System.out.println(a[0]);
b = a;
b[0] = 100;
System.out.println(a[0]);
```

X

```
int []a = {10, 20, 30};

int []b;
b = new int[a.length];
for(int i = 0; i < a.length; i++)
    b[i] = a[i];
```

Copy dữ liệu giữa 2 mảng (Python)

- Tương tự Java, nếu gán b = a, thay đổi giá trị mảng a gây ảnh hưởng giá trị mảng b và ngược lại.
- Phải sử dụng toán tử slicing để copy mảng a qua mảng b.

```
a = [10, 20, 30]

print(a[0])
b = a
b[0] = 100
print(a[0])
```

```
a = [10, 20, 30]

print(a[0])
b = a[:]
b[0] = 100
print(a[0])
```

Slicing (Python)

```
lst = [10, 20, 30, 40, 50, 60]

# start = 1, stop = 3
a = lst[1:3]

# start = 0, stop = 4
b = lst[:4]

# start = 3
c = lst[3:]

# all
a = lst[:]
```

- Python hỗ trợ toán tử slicing, cho phép lấy nhanh một dãy các phần tử liên tiếp trong mảng.
- Sau khi slicing, thay đổi giá trị trên mảng mới (a, b, c, d) không ảnh hưởng đến mảng cũ. (lst)

Slicing (C++)

```
int lst[] = {10, 20, 30, 40, 50, 60};

int start = 1;
int stop = 3;
int na = stop - start;
int a[100];
for(int i = 0; i < na; i++)
    a[i] = lst[start + i];
```

- C++ không hỗ trợ toán tử slicing.
- Phải dùng vòng lặp, copy từng giá trị từ mảng cũ (lst) qua mảng mới (a).

Slicing (Java)

```
int []lst = {10, 20, 30, 40, 50, 60};

int start = 1;
int stop = 3;
int na = stop - start;
int []a = new int[na];
for(int i = 0; i < na; i++)
    a[i] = lst[start + i];
```

- Java cũng không hỗ trợ toán tử slicing.
- Sử dụng toán tử new để tạo mảng mới (a).
- Dùng vòng lặp, copy từng giá trị từ mảng cũ (lst) qua mảng mới (a).

Thêm phần tử vào vị trí cuối trong mảng (Python)

```
a = [10, 20, 30]
# 10, 20, 30

a.append(40)
# 10, 20, 30, 40

a.append(50)
# 10, 20, 30, 40, 50
```

- Sử dụng hàm append()

Thêm phần tử vào vị trí cuối trong mảng (C++)

```
int a[100] = {10, 20, 30};
int n = 3;
// 10, 20, 30

a[n] = 40;
n++;
// 10, 20, 30, 40

a[n] = 50;
n++;
// 10, 20, 30, 40, 50
```

- Nếu n < số phần tử tối đa, gán a[n] = x và tăng n.

Thêm phần tử vào vị trí cuối trong mảng (Java)

```
int []a = {10, 20, 30};

int []b;
b = new int[a.length + 1];
for(int i = 0; i <a.length; i++)
    b[i] = a[i];
b[b.length-1] = 40;

a = b;
```

- Tạo mảng tạm b ➔ copy giá trị từ a qua b ➔ thêm x vào cuối b ➔ gán a = b.

Sử dụng mảng làm tham số của hàm

- Trong C++, Java và Python, khi truyền mảng a là tham số của hàm, ta **có thể thay đổi** giá trị các phần tử a[i] trong mảng.
- Sau khi kết thúc hàm, các giá trị ở mảng tương ứng bên ngoài cũng thay đổi theo.

VD: Sử dụng mảng làm tham số của hàm (C++)

```
void inc(int a[], int n){
    for(int i = 0; i < n; i++){
        a[i] *= 2;
    }
}

int main(){
    int a[] = {10, 20, 30};
    int n = 3;
    for(int i = 0; i < n; i++){
        cout << a[i] << " ";
    }

    inc(a, n);

    for(int i = 0; i < n; i++){
        cout << a[i] << " ";
    }
    return 0;
}
```

Sử dụng mảng làm tham số của hàm (Java)

```java
static void inc(int []a){
    for(int i = 0; i < a.length; i++){
        a[i] *= 2;
    }
}

public static void main(String[] args) {
    int []a = {10, 20, 30};
    for(int i = 0; i < a.length; i++){
        System.out.printf("%d ", a[i]);
    }

    inc(a);

    for(int i = 0; i < a.length; i++){
        System.out.printf("%d ", a[i]);
    }
}
```

Sử dụng mảng làm tham số của hàm (Python)

```
def inc(l):
    for i in range(len(l)):
        l[i] *= 2

lst = [10, 20, 30]

for x in lst:
    print(x, end=" ")

inc(lst)

for x in lst:
    print(x, end=" ")
```

Hỏi đáp