LECTURE 10

DATA ABSTRACTION

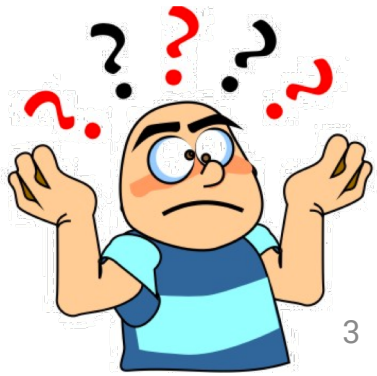
Big-O Coding

Website: www.bigocoding.com

Đặt vấn đề

BT1 – CỘNG 2 PHÂN SỐ

- Viết chương trình nhập vào 2 phân số. Cộng 2 phân số và in kết quả (sau khi rút gọn) ra màn hình.



BT1 – Gợi ý

1. Đọc phân số 1, lưu vào 2 biến num1 (tử số phân số 1), denom1 (mẫu số phân số 1).
2. Đọc phân số 2, lưu vào 2 biến num2, denom2.
3. Cộng 2 phân số:
   a. num3 = num1 * denom2 + num2 * denom1
   b. denom3 = denom1 * denom2
4. Rút gọn phân số: num3, denom3.
5. In kết quả: num3, denom3.

BT1 – Cài đặt (C++) – Xử lí chính

```cpp
int main() {
    int num1, denom1;
    int num2, denom2;
    int num3, denom3;

    cin >> num1 >> denom1;
    cin >> num2 >> denom2;

    sumFraction(num1, denom1, num2, denom2, num3, denom3);

    cout << num3 << " " << denom3;

    return 0;
}
```

BT1 – Cài đặt (C++) – Khai báo hàm

```
void sumFraction(int num1, int denom1, int num2, int denom2,
                 int &num3, int &denom3);

int gcd(int a, int b);
void reduceFraction(int &num, int &denom);
```

BT1 – Cài đặt (C++) – Cài đặt hàm

```
void sumFraction(int num1, int denom1, int num2, int denom2,
                 int &num3, int &denom3) {
    num3 = num1*denom2 + num2*denom1;
    denom3 = denom1 * denom2;
    reduceFraction(num3, denom3);
}
```

BT1 – Cài đặt (C++) – Cài đặt hàm

```
int gcd(int a, int b) {
    int r = 0;
    while(b != 0) {
        r = a % b;
        a = b;
        b = r;
    }
    return a;
}

void reduceFraction(int &num, int &denom) {
    if(num == 0){
        denom = 1;
        return;
    }
    int x = gcd(abs(num), abs(denom));
    num = num / x;
    denom = denom / x;
}
```

BT1 – Cài đặt (Python) – Xử lí chính

```
num1, denom1 = map(int, input().split())
num2, denom2 = map(int, input().split())

num3, denom3 = sumFraction(num1, denom1, num2, denom2)

print("{0} {1}".format(num3, denom3))
```

BT1 – Cài đặt (Python) – Cài đặt hàm

```
def gcd(a, b):
    while b != 0:
        r = a % b
        a = b
        b = r
    return a

def reduceFraction(num, denom):
    if num == 0:
        return 0, 1
    x = gcd(abs(num), abs(denom))
    num = num // x
    de = de // x
    return num, denom

def sumFraction(num1, denom1, num2, denom2):
     num3 = num1*denom2 + num2*denom1
     denom3 = denom1 * denom2
     num3, denom3 = reduceFraction(num3, denom3)
     return num3, denom3
```

BT1 – Cài đặt (Java)

- Trong Java, với những kiến thức hiện tại, không thể cài đặt hàm sumFraction() để trả về 2 giá trị num3, denom3.

Vấn đề

- Phải quản lí quá nhiều biến.
- Dễ nhầm lẫn.

```
sumFraction(num1, denom1, num2, denom2, num3, denom1);
```

```
num3, denom2 = sumFraction(num1, denom1, num2, denom1)
```

- Giải pháp: sử dụng struct/class để tạo ra kiểu dữ liệu mới: Fraction.

Data abstraction

Ví dụ

Không áp dụng Data Abstraction	Áp dụng Data Abstraction
- Dùng **2 biến** int num1; int denom1; biểu diễn 1 phân số	- Tạo struct / class Fraction - Dùng **1 biến** Fraction p1; biểu diễn 1 phân số
- Dùng **2 biến** double x; double y; biểu diễn 1 điểm trong hệ tọa độ Oxy	- Tạo struct / class Point2D - Dùng **1 biến** Point2D p1; biểu diễn 1 điểm
- Dùng **3 biến** int id; char name[51]; float gpa; biểu diễn 1 sinh viên	- Tạo struct / class Student - Dùng **1 biến** Student st1; biểu diễn 1 sinh viên

Sử dụng struct / class

BT1 – Sử dụng struct Fraction (C++)

```
struct Fraction{
    int num;
    int denom;
};

Fraction sumFraction(Fraction p1, Fraction p2);
void reduceFraction(Fraction &p);
int gcd(int a, int b);
```

- Dùng từ khóa struct để tạo ra 1 kiểu dữ liệu mới
- Bên trong khai báo các thuộc tính
- Thuộc tính là các biến, mô tả thông tin của struct

- Khai báo các hàm liên quan đến Fraction ở ngoài struct
- Thay vì có 2 biến num, denom, ta sử dụng 1 biến duy nhất

BT1 – Sử dụng struct Fraction (C++)

```
Fraction sumFraction(Fraction p1, Fraction p2) {
    Fraction p3;
    p3.num = p1.num * p2.denom + p1.denom * p2.num;
    p3.denom = p1.denom * p2.denom;
    reduceFraction(p3);
    return p3;
}
```

- Dùng toán tử dấu chấm, để truy xuất đến các thuộc tính
- Gọi hàm truyền tham số: như trước
- return: như trước

BT1 – Sử dụng struct Fraction (C++)

```
void reduceFraction(Fraction &p) {
    if(p.num == 0) {
        p.denom = 1;
        return;
    }
    int x = gcd(abs(p.num), abs(p.denom));
    p.num = p.num / x;
    p.denom = p.denom / x;
}
```

- Dùng toán tử dấu chấm, để truy xuất đến các thuộc tính

BT1 – Sử dụng struct Fraction (C++)

```
int gcd(int a, int b) {
    int r = 0;
    while(b != 0) {
        r = a % b;
        a = b;
        b = r;
    }
    return a;
}
```

BT1 – Xử lí chính (C++)

```
int main() {
    Fraction p1;
    Fraction p2;
    Fraction p3;

    cin >> p1.num >> p1.den
    cin >> p2.num >> p2.den

    p3 = sumFraction(p1, p2);

    cout << p3.num << " " << p3.denom;

    return 0;
}
```

- Thay vì có 6 biến int, ta sử dụng 3 biến Fraction

- Dùng toán tử dấu chấm, để truy xuất đến các thuộc tính

- Gọi hàm, truyền tham số p1, p2, nhận kết quả trả về p3, như các biến thông thường trước đây

BT1 – Sử dụng class Fraction (Java)

```
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    // Các hàm
}
```

- Sử dụng từ khóa class để tạo kiểu dữ liệu mới
- Trong class, khai báo các thuộc tính
- Các thuộc tính là các biến, mô tả thông tin của class

- private: che dấu, không cho phép class khác truy cập
- Thông thường, các thuộc tính là private
- public: công khai, cho phép class khác truy cập

- Các hàm cũng nằm trong class

BT1 – Sử dụng class Fraction (Java)

```
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    // Các hàm
    public Fraction sumFraction(Fraction p2){
        Fraction p3 = new Fraction();
        p3.num = num * p2.denom + denom * p2.num;
        p3.denom = denom * p2.denom;
        p3.reduceFraction();
        return p3;
    }
}
```

- Các hàm cũng nằm trong class
- Các hàm thường public để class khác gọi sử dụng được
- Số lượng tham số của hàm giảm 1. Vì lúc sử dụng sẽ viết thành p1.sumFraction(p2);

- Dùng new để tạo phân số mới

BT1 – Sử dụng class Fraction (Java)

```
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    // Các hàm
    public void reduceFraction(){
        if(num == 0){
            denom = 1;
            return;
        }

        int x = gcd(Math.abs(num), Math.abs(denom));
        num = num / x;
        denom = denom / x;
    }
}
```

- Hàm reduceFraction không có tham số, vì khi sử dụng sẽ là p.reduceFraction();

BT1 – Sử dụng class Fraction (Java)

```
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    // Các hàm
    public int gcd(int a, int b){
        int r = 0;
        while(b != 0){
            r = a % b;
            a = b;
            b = r;
        }
        return a;
    }
}
```

BT1 – Sử dụng class Fraction (Java)

```
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    // Các hàm
    @Override
    public String toString() {
        String s = String.format("%d %d", num, de);
        return s;
    }
}
```

- Trong mỗi class đều có 1 hàm toString(), chuyển từ đối tượng thành chuỗi, từ đó in ra màn hình
- Nhớ `@Override`

BT1 – Sử dụng class Fraction (Java)

```
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    // Các hàm
    public Fraction(){
        num = 0;
        denom = 1;
    }
}
```

- Ngoài ra, trong class còn có 1 loại hàm đặc biệt, gọi là hàm dựng, constructor
- Hàm dựng dùng khi tạo phân số
- Ở đây ta có hàm dựng mặc định, hay hàm dựng không tham số
- Sử dụng: `Fraction p = new Fraction();`

BT1 – Sử dụng class Fraction (Java)

```
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    // Các hàm
    public Fraction(int a, int b){
        num = a;
        denom = b;
    }
}
```

- Đây cũng là constructor, nhưng có 2 tham số
- Sử dụng: `Fraction p = new Fraction(3, 4);`

BT1 – Xử lí chính (Java)

Big-O Green

```
public class L10P01 {
    public static void main(String []args){
        Fraction p1;
        Fraction p2;
        Fraction p3;

        int x, y;

        Scanner sc = new Scanner(System.in);
        x = sc.nextInt();
        y = sc.nextInt();
        p1 = new Fraction(x, y);

        x = sc.nextInt();
        y = sc.nextInt();
        p2 = new Fraction(x, y);

        p3 = p1.sumFraction(p2);

        System.out.println(p3.toString());
    }
}
```

- Thay vì dùng 6 biến int, ta dùng 3 biến Fraction
- Sử dụng hàm dựng 2 tham số
- Sử dụng hàm dựng 2 tham số
- Gọi hàm, truyền tham số, nhận kết quả
- Sử dụng hàm toString()

BT1 – Sử dụng class Fraction (Python)

```
class Fraction:
    def __init__(self, a = 0, b = 1):
        self.num = a
        self.denom = b
```

- Sử dụng từ khóa class để tạo kiểu dữ liệu mới
- Python không có khai báo thuộc tính
- Thuộc tính được xác định thông qua từ khóa self.tên_thuộc_tính

- Hàm __init__() chính là hàm dựng, constructor trong Python
- Dùng để tạo đối tượng phân số mới

- Có thể viết hàm trong class (như Java). Khi đó tham số đầu tiên của hàm là self
- self đại diện cho đối tượng gọi hàm

- Khác với Java (nghiêm ngặt), Python (dễ dãi) qui định các thuộc tính, hàm đều là public

BT1 – Sử dụng class Fraction (Python)

```
class Fraction:
    def sumFraction(self, p2):
        p3 = Fraction()
        p3.num = self.num * p2.denom + self.denom * p2.num
        p3.denom = self.denom * p2.denom
        p3.reduceFraction()
        return p3
```

- Có thể viết hàm trong class (như Java). Khi đó tham số đầu tiên của hàm là self
- Dòng code p3 = Fraction() sẽ gọi hàm __init__() và tạo phân số 0/1

- Khi viết p3 = p1.sumFraction(p2) thì self chính là p1

BT1 – Sử dụng class Fraction (Python)

```
class Fraction:
    def reduceFraction(self):
        if(self.num == 0):
            self.denom = 
            return
        x = self.gcd(abs(self.num), abs(self.denom))
        self.num = self.num // x
        self.denom = self.denom // x
```

- Khi viết p3.reduceFraction() thì self là p3

BT1 – Sử dụng class Fraction (Python)

```
class Fraction:
    def gcd(self, a, b):
        r = 0
        while(b != 0):
            r = a % b
            a = b
            b = r
        return a
```

BT1 – Sử dụng class Fraction (Python)

```
class Fraction:
    def __str__(self):
        s = "{0} {1}".format(self.num, self.denom)
        return s
```

- Hàm __str__() trong Python, tương tự hàm toString() trong Java, dùng để chuyển đối tượng thành phân số

BT1 – Xử lí chính (Python)

```
x, y = map(int, input().split())
p1 = Fraction(x, y)

x, y = map(int, input().split())
p2 = Fraction(x, y)

p3 = p1.sumFraction(p2)

print(p3)
```

- Chỗ này sẽ gọi hàm dựng __init__()

- Chỗ này sẽ gọi hàm dựng __init__()

1 hoặc 2?

1. struct/class Triangle có 3 thuộc tính: Point vertexA, vertexB, vertexC.

2. struct/class Triangle có 3 thuộc tính: double edgeAB, edgeBC, edgeCA.

What's wrong? (C++)

```
Fraction sumFraction(Fraction p1, Fraction p2) {
    Fraction p3;
    num3 = num1 * p2.denom + p1.denom * p2.num;
    denom3 = p1.denom * p2.denom;
    reduceFraction(p3);
    return p3;
}
```

What's wrong? (Java)

```
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    // Các hàm
    public Fraction sumFraction(Fraction p2){
        Fraction p3;
        p3.num = p1.num * p2.denom + p1.denom * p2.num;
        p3.denom = p1.denom * p2.denom;
        p3.reduceFraction();
        return p3;
    }
}
```

What's wrong? (Java)

```
public class L10P01 {
    public static void main(String []args){
        Fraction p1;
        Fraction p2;
        Fraction p3;

        int x, y;

        Scanner sc = new Scanner(System.in);
        x = sc.nextInt();
        y = sc.nextInt();
        p1 = Fraction(x, y);

        x = sc.nextInt();
        y = sc.nextInt();
        p2 = Fraction(x, y);

        p3 = sumFraction(p1, p2);

        System.out.println(p3.toString());
    }
}
```

What's wrong? (Python)

```
class Fraction:
    def sumFraction(p2):
        p3 = Fraction()
        p3.num = self.num * p2.denom + self.denom * p2.num
        p3.denom = self.denom * p2.denom
        p3.reduceFraction()
        return p3
```

Lưu ý

1. get/set giá trị của thuộc tính ở bên ngoài struct (C++)

- Trong C++, thuộc tính trong struct là public, nên bên ngoài struct có thể get/set giá trị thuộc tính thoải mái.

```
int main() {
    Fraction p;
    p.num = 3;
    p.denom = 4;
    cout << p.num << " " << p.denom;
    return 0;
}
```

1. get/set giá trị của thuộc tính ở bên ngoài class(Java)

- Trong Java, thông thường, các thuộc tính là private, nên ở ngoài class, ta không thể get/set trực tiếp thuộc tính.
- Cần cài đặt và sử dụng các hàm getters, setters.

```java
public static void main(String []args){
    Fraction p = new Fraction();
    p.num = 3; // error
    p.denom = 4; // error
    // error
    System.out.printf("%d %d",
        p.num, p.denom);

    p.setNum(3); // ok
    p.setDenom(4); // ok
    // ok
    System.out.printf("%d %d",
        p.getNum(), p.getDenom());
}
```

```java
class Fraction{
    // Các thuộc tính
    private int num;
    private int denom;

    public int getNum() {
        return num;
    }

    public int getDenom() {
        return denom;
    }

    public void setNum(int a)
{
        num = a;
    }

    public void setDenom(int
b) {
        denom = b;
    }

    public Fraction(){
        num = 0;
        denom = 1;
    }
```

1. get/set giá trị của thuộc tính ở bên ngoài class(Python)

- Python không qui định chặt chẽ thuộc tính hay hàm nào là private, nên bên ngoài class có thể truy xuất trực tiếp vào thuộc tính.

```
p = Fraction()
p.num = 3
p.denom = 4
print("{0} {1}".format(p.num, p.denom))
```

2. Gán bằng 2 struct (C++)

- Có thể gán bằng 2 biến thuộc kiểu dữ liệu mới tạo.
- Sau khi gán bằng, thay đổi giá trị trên biến này, ko ảnh hưởng đến biến kia.

```
Fraction a;
a.num = 3;
a.denom = 4;

Fraction b = a;

cout << a.num << " " <<
a.denom;
b.num = 3000;
cout << a.num << " " <<
a.denom;
```

```
3 4

3 4
```

2. Gán bằng 2 class (Java)

- Sau khi gán bằng, thay đổi giá trị trên biến này, sẽ ảnh hưởng đến biến kia.

```
Fraction a = new Fraction(3, 4);
Fraction b = a;

System.out.println(a.toString());
b.setNum(3000);
System.out.println(a.toString());
```

```
3 4

3000 4
```

2. Gán bằng 2 class (Python)

- Sau khi gán bằng, thay đổi giá trị trên biến này, sẽ ảnh hưởng đến biến kia.

```
a = Fraction(3, 4)
b = a

print(a)
b.num = 3000
print(a)
```

```
3 4

3000 4
```

3. deep-copy 2 struct (C++)

- Sử dụng toán tử gán bằng.

```
Fraction a;
a.num = 3;
a.denom = 4;

Fraction b = a;

cout << a.num << " " <<
a.denom;
b.num = 3000;
cout << a.num << " " <<
a.denom;
```

```
3 4

3 4
```

3. deep-copy 2 class (Java)

- Cài đặt và sử dụng hàm clone().

```
class Fraction{
    public Fraction clone(){
        Fraction ans = new Fraction();
        ans.num = num;
        ans.denom = denom;
        return ans;
    }
}
```

```
Fraction a = new Fraction(3, 4);
Fraction b = a.clone();

System.out.println(a.toString());
b.setNum(3000);
System.out.println(a.toString());
```

3. deep-copy 2 class (Python)

- Cài đặt và sử dụng hàm clone().

```
class Fraction:
    def clone(self):
        ans = Fraction()
        ans.num = self.num
        ans.denom = self.denom
        return ans
```

```
a = Fraction(3, 4)
b = a.clone()

print(a)
b.num = 3000
print(a)
```

4. Sử dụng struct này trong struct kia (C++)

```cpp
struct Point2D {
    double x;
    double y;
};

struct Triangle {
    Point2D A;
    Point2D B;
    Point2D C;
};
```

4. Sử dụng class này trong class kia (Java)

```
class Point2D{
    private double x;
    private double y;
}

class Triangle{
    private Point2D A;
    private Point2D B;
    private Point2D C;
}
```

4. Sử dụng class này trong class kia (Python)

```python
class Point2D:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

class Triangle:
    def __init__(self, A, B, C):
        self.A = A
        self.B = B
        self.C = C
```

5. Mảng các đối tượng (C++)

```
Fraction a[100];
int n;
cin >> n;
for(int i = 0; i < n; i++){
    cin >> a[i].num >> a[i].denom;
}
```

5. Mảng các đối tượng (Java)

```
Fraction []a;
int n;

Scanner sc = new Scanner(System.in);
n = sc.nextInt();
a = new Fraction[n];

int x, y;
for(int i = 0; i < n; i++){
    x = sc.nextInt();
    y = sc.nextInt();
    a[i] = new Fraction(x, y);
}
```

5. Mảng các đối tượng (Python)

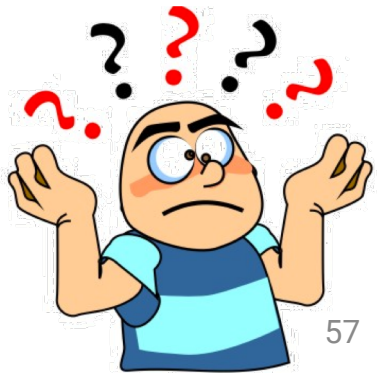
```
a = []
n = int(input())
for i in range(n):
    x, y = map(int, input().split())
    p = Fraction(x, y)
    a.append(p)
```

Bài tập và gợi ý

Khai báo ít nhất 1 struct/class cho mỗi bài tập

BT1 – CỘNG 2 PHÂN SỐ

- Viết chương trình nhập vào 2 phân số. Cộng 2 phân số và in kết quả (sau khi rút gọn) ra màn hình.



BT1 – Gợi ý – struct/class

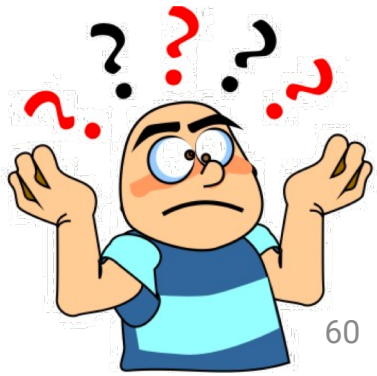
- struct/class: Fraction
- Thuộc tính:
  - num: tử số
  - denom: mẫu số
- Hàm xử lí:
  - Hàm sumFraction(): cộng 2 phân số
  - Hàm reduceFraction(): rút gọn phân số
  - Fraction() / __init__() và toString() / __str__() (Java / Python)

BT1 – Gợi ý – Xử lí chính

1. Đọc tử, mẫu vào phân số p1.
2. Đọc tử, mẫu vào phân số p2.
3. Gọi hàm sumFraction(), nhận kết quả vào phân số p3.
4. In kết quả.

BT2 – TÌM ĐIỂM XA M NHẤT

- Trong danh sách n điểm Oxy, hãy tìm điểm xa nhất M(x, y) nhất.



BT2 – Gợi ý – struct/class

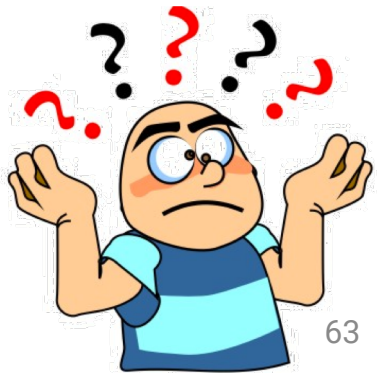
- struct/class: Point2D
- Thuộc tính:
  - x: hoành độ
  - y: tung độ
- Hàm xử lí:
  - Hàm distance(): tính khoảng cách 2 điểm
  - Point2D() / __init__() và toString() / __str__() (Java / Python)

BT2 – Gợi ý – Xử lí chính

1. Đọc tọa độ x, y, lưu vào điểm M.
2. Đọc số lượng điểm, lưu vào n
3. Sử dụng vòng lặp, đọc vào tọa độ các điểm, lưu vào mảng a.
4. Đặt lính canh: ans = a[0].
5. Sử dụng vòng lặp, duyệt qua mảng a.
    a. Nếu distance(a[i], M) > distance(ans, M)
        i. Cập nhật ans.
6. In kết quả, ans.

BT3 – ĐẾM SỐ SINH VIÊN XUẤT SẮC

- Đếm số lượng sinh viên có điểm trung bình ≥ 9.



BT3 – Gợi ý – struct/class

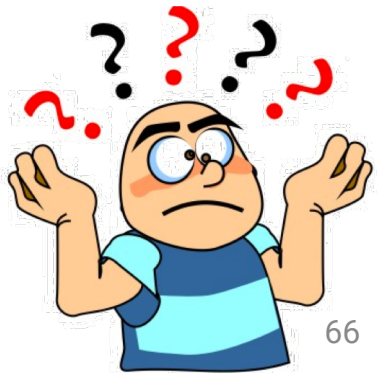
- struct/class: Student
- Thuộc tính:
  - fullname: họ tên
  - literature: điểm văn
  - math: điểm toán
- Hàm xử lí:
  - Hàm gpa(): tính điểm trung bình
  - Student() / __init__() và toString() / __str__() (Java / Python)

BT3 – Xử lí chính

1. Đọc số lượng sinh viên, lưu vào n.
2. Sử dụng vòng lặp, đọc điểm văn, điểm toán, họ tên của từng sinh viên, lưu vào mảng a.
3. Khởi tạo biến đếm: count = 0.
4. Sử dụng vòng lặp, duyệt qua mảng a.
    a. Nếu điểm trung bình của sinh viên a[i] >= 9.0, tăng biến đếm.
5. In kết quả.

BT4 – TÍNH TỔNG DIỆN TÍCH TAM GIÁC

- Tính tổng diện tích các tam giác trong mảng.



BT4 – Gợi ý – struct/class

- struct/class: Triangle
- Thuộc tính:
  - A, B, C: 3 đỉnh
- Hàm xử lí:
  - Hàm area(): tính diện tích của 1 tam giác
  - Triangle() / __init__() và toString() / __str__() (Java / Python)

BT4 – Gợi ý – Xử lí chính

1. Đọc số lượng tam giác, lưu vào n.
2. Sử dụng vòng lặp, đọc tọa độ 3 đỉnh của từng tam giác, lưu vào mảng a.
3. Khởi tạo tổng: sum = 0.
4. Sử dụng vòng lặp, duyệt qua mảng a.
    a. sum = sum + diện tích tam giác a[i].
5. In kết quả.

Hỏi đáp