# Bài 3: Trừu tượng hóa dữ liệu

Giảng viên: Hoàng Thị Điệp

Khoa Công nghệ Thông tin – Đại học Công Nghệ

# Nội dung chính

- Biểu diễn dữ liệu trong các ngôn ngữ lập trình
- Sự trừu tượng hóa dữ liệu
- Kiểu dữ liệu trừu tượng
  - Đặc tả
  - Cài đặt

# Biểu diễn như thế nào?

- Tuổi của một người.
- Điểm của một môn học tín chỉ.
- Một phân số. Một dãy phân số.
- Một điểm ảnh (pixel) của ảnh RGB biết cường độ mỗi màu nằm trong [0; 255]. Một ảnh RGB.
- Một điểm, một đoạn thẳng, một tam giác trong hệ tọa độ 2 chiều.
- Một đa thức bậc n.
- Giá trị của n! với n nhỏ. Giá trị của n! với n lớn.

# Dữ liệu

- Dữ liệu là những thông tin mà máy tính có thể xử lý: số nguyên, số thực, xâu kí tự, và các dữ liệu phức tạp được tạo từ nhiều thành phần
- Trong bộ nhớ máy tính, dữ liệu được biểu diễn dưới dạng nhị phân (dãy 0, 1)
- Trong các ngôn ngữ lập trình bậc cao (C++, Java..), dữ liệu được biểu diễn dưới dạng trừu tượng, xuất phát từ biểu diễn toán học và dễ hiểu cho con người:
  - int age
  - double weight



diepht@vnu

# Kiểu dữ liệu cơ bản

Kiểu dữ liệu được xác định bởi:

1. Phạm vi giá trị
2. Các phép toán

Ví dụ trong C++

| kiểu | phạm vi | phép toán thường dùng |
|---|---|---|
| bool | true/false | &&, \|\|, ! |
| char | -128 -> 127 | >, <, == |
| short int | -32,768 -> 32,767 | >, <, ==, +, -, *, /, % |
| float | +/- 3.4e +/- 38 | >, <, ==, +, -, *, / |
| double | +/- 1.7e +/- 308 | >, <, ==, +, -, *, / |



diepht@vnu

# Kiểu dữ liệu có cấu trúc

Ngôn ngữ lập trình cung cấp cho ta những luật để xây dựng kiểu dữ liệu mới $T$ từ những kiểu dữ liệu đã biết $t_1, t_2, \ldots, t_n$.

Ví dụ trong C++:

```
struct T {
    t1  x1;
    t2  x2;
    ……
    tn  xn;
};
```



diepht@vnu

# Kiểu dữ liệu có cấu trúc

- Xây dựng cấu trúc dữ liệu để biểu diễn dữ liệu của 1 điểm trên mặt phẳng

```
struct  Point {
    double      x;
    double      y;
};
```

- Xây dựng cấu trúc dữ liệu để  biểu diễn dữ liệu của 1 đoạn thẳng trên mặt phẳng

```
struct  Line {
        Point    start;
        Point    end;
};
```



diepht@vnu

# Phạm vi và các phép toán trên kiểu dữ liệu có cấu trúc

Xét kiểu dữ liệu mới *T* được tạo từ những kiểu dữ liệu đã biết $t_1, t_2, \ldots, t_n$,

Ví dụ:

```
struct  Complex{
        double          real;
        double          image;
};
```

Phạm vi: Xác định bởi phạm vi của các kiểu dữ liệu thành phần

- real: là số thực nằm trong phạm vi kiểu 'double'
- image: là số thực nằm trong phạm vi kiểu 'double'



diepht@vnu

# Phạm vi và các phép toán trên kiểu dữ liệu có cấu trúc

## Phép toán: Do người dùng định nghĩa

Ví dụ:

```
struct  Complex{
        double   real;
        double              image;
};

Complex  createComplex (double real, double image) {
   Complex  c;
   c.real = real;
   c.image = image;
   return c;
}
```



diepht@vnu

# Phạm vi và các phép toán trên kiểu dữ liệu có cấu trúc

```
Complex add (Complex c1, Complex c2) {
    Complex c12;
    c12.real = c1.real + c2.real;
    c12.image = c1.image + c2.image;
    return c12;
}

Complex multiply (Complex c1, Complex c2) {
    Complex c12;
    c12.real = (c1.real * c2.real) – (c1.image * c2.image);
    c12.image = (c1.real * c2.image) + (c1.image * c2.real);
    return c12;
}
```



diepht@vnu

# Trừu tượng hóa dữ liệu (abstraction)

1. Đặc tả đối tượng dữ liệu (các thành phần dữ liệu của đối tượng)

   Ví dụ: đối tượng số phức (Complex)
   - real
   - image

2. Đặc tả các phép toán trên đối tượng dữ liệu (operations)

   Ví dụ: đối tượng số phức (Complex)
   - createComplex (real, image)
   - getReal (complexNumber)
   - getImage (complexNumber)
   - add (complexNumber1, complexNumber2)
   - multiply (complexNumber2, complexNumber2)
   - print (complexNumber)



diepht@vnu

# Trừu tượng hóa dữ liệu

## Trừu tượng hóa đối tượng sinh viên (Student)

1. Đặc tả đối tượng dữ liệu

   name, age, sex,  address

2. Đặc tả các phép toán trên đối tượng dữ liệu

   createStudent (name, age, sex, address)

   compare (student1, student2)

   getName (student)

   getAge (student)

   getSex (student)

   getAdd (student)



diepht@vnu

# Trừu tượng hóa dữ liệu

## Trừu tượng hóa đối tượng lớp học (StudentClass)

1. Đặc tả đối tượng dữ liệu

   className, numberStudent, studentArr, address

2. Đặc tả các phép toán trên đối tượng dữ liệu

   addStudent (studentClass, student)

   findStudent (studentClass, student)

   deleteStudent (studentClass, student)

   getClassName (studentClass)

   getNumberStudent (studentClass)

   getStudentArr (studentClass)

   getClassAddress (studentClass)



diepht@vnu

# Giải một bài toán tin học

- Đặc tả vấn đề
- Thiết kế cấu trúc dữ liệu
- Thiết kế giải thuật
- Cài đặt (C++, Java…)
- Thử nghiệm và sửa lỗi
- Tối ưu chương trình



diepht@vnu

# Ví dụ

- Bài toán: Giả sử chúng ta cần viết chương trình lập lịch thi. Vấn đề như sau. Mỗi người dự thi đăng kí thi một số môn trong số các môn tổ chức thi. Chúng ta cần xếp lịch thi, mỗi ngày thi một số môn trong cùng một thời gian, sao cho mỗi người dự thi có thể thi tất cả các môn họ đã đăng kí.
- Đặc tả bằng danh sách?
- Đặc tả bằng đồ thị?

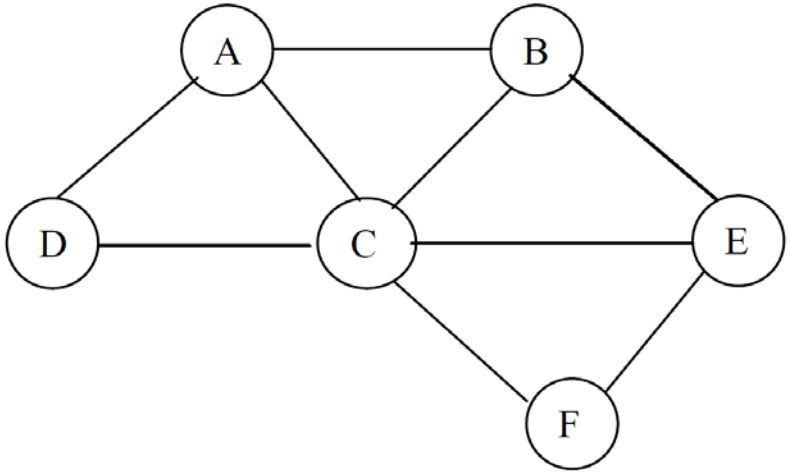
# Ví dụ: Đặc tả bằng danh sách

- Input:
  - 1 danh sách người dự thi
  - mỗi người dự thi biểu diễn bằng 1 danh sách môn thi người đó đăng kí
- Output:
  - 1 danh sách ngày thi
  - mỗi ngày thi là 1 danh sách các môn thi
    - không có môn thi nào xuất hiện trong 2 ngày
    - không có 2 môn thi nào trong 1 ngày cùng xuất hiện trong danh sách của 1 người dự thi nào đó

# Ví dụ: Đặc tả bằng đồ thị

- Input:
  - 1 đồ thị
    - đỉnh biểu diễn môn thi
    - giữa 2 môn thi có cạnh nối nếu chúng được đăng kí bởi cùng 1 người dự thi nào đó
- Output:
  - 1 danh sách ngày thi
  - mỗi ngày thi là 1 danh sách các đỉnh không kề nhau

# Ví dụ: Đặc tả bằng đồ thị




- Ví dụ:

  S1 = (A, B, C)  S2 = (A, C, D)
  S3 = (C, E, F)  S4 = (A, C)
  S5 = (C, B, E)  S6 = (C, F)

- Thuật toán
  - i = 0, $L_i$=() // ds môn thi ngày đầu
  - Lặp:
    - Chọn đỉnh bất kì chưa đánh dấu, đưa vào $L_i$
    - Đánh dấu nó và các đỉnh kề nó
    - Nếu tất cả đều đã đánh dấu:
      - Loại các đỉnh trong $L_i$ khỏi đồ thị.
        - Nếu đồ thị rỗng: Kết thúc!
        - Ngược lại: i++, $L_i$=(), xóa đánh dấu // Xây dựng ds cho ngày tiếp
- Kết quả:



diepht@vnu

# Lập trình hướng đối tượng

## Object oriented programming (OOP)

- Lập trình hướng đối tượng giúp chúng ta cài đặt các mô tả trừu tượng (đối tượng dữ liệu và các phép toán) thành  các đoạn mã chương trình
- Chương trình được thiết kế thành từng đoạn nhỏ, mỗi đoạn mô tả về một đối tượng (thuộc tính dữ liệu, các phép toán trên dữ liệu)
- Hai thuộc tính quan trọng: đóng gói (encapsulation) và thừa kế (inheritance)



diepht@vnu

# OOP: Tính đóng gói (encapsulation)

**Class:** Cài đặt một lớp đối tượng dữ liệu trừu tượng. Việc cài đặt bao gồm cài đặt các thành phần dữ liệu và các phép toán trên dữ liệu

Ví dụ:

```
class Complex {
        double real;
        double image;
public:
        void create (double r, double i) {
            real = r;  image = i;
        }
        double getReal () {
            return real;
        }
        ……….
        void   print {
            cout <<  real << " +i " << image << " \ n" ;
        }
};
```

- Liên kết chặt chẽ giữa dữ liệu và phép toán
- Che giấu dữ liệu
- Dễ dàng tìm lỗi
- Các đối tượng liên kết với nhau thông qua các phép toán



diepht@vnu

# OOP: Tính đóng gói (encapsulation)

**Object:** Biểu diễn cho một đối tượng cụ thể của một lớp

```
Complex    c1;
Complex    c2;
```

# C++

- Lập trình tổng quát (Generic programming)
- Con trỏ và cấp phát động
- Lớp



diepht@vnu