Chương 1: TỔNG QUAN VỀ CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

1. Vai trò của Cấu trúc dữ liệu trong một dự án Tin học

Thực hiện một dự án Tin học là chuyển 1 bài toán thực tế thành 1 bài toán có thể xử lý trên máy tính.

Một bài toán thực tế nào cùng đều bao gồm:

1. Các đối tượng dữ liệu
2. Các yêu cầu xử lý trên các đối tượng đó

Vì vậy, để giải quyết và xây dựng một dự án Tin học bao giờ chúng ta cũng cần phải có 2 vấn đề:

1.1. Tổ chức biểu diễn các đối tượng thực tế

Các thành phần dữ liệu trong thực tế đa dạng, và có thể chứa đựng những quan hệ với nhau, do đó khi chúng ta lập trình để giải quyết các bài toàn đó, cần phải tổ chức, xây dựng các *CẤU TRÚC DỮ LIỆU* thích hợp sao cho:

- Phản ánh chính xác được các dữ liệu trong thực tế
- Có thể dễ dàng xử lý nó bằng máy tính

Công việc này được gọi là xây dựng ***Cấu trúc dữ liệu*** cho bài toán

1.2. Xây dựng các thao tác xử lý dữ liệu

Từ các yêu cầu xử lý bài toán thực tế, cần tìm ra các giải thuật tương ứng để xác định được trình tự các thao tác của máy tính phải thực hiện để cho ra kết quả mong muốn. Đây chính là bước ***Xây dựng giải thuật*** cho bài toán thực tế.

Tuy nhiên khi giải quyết một bài toán bằng máy tính, chúng ta thường có khuynh hướng chú trọng đến việc xây dựng giải thuật mà quên đi việc tổ chức dữ liệu. Giải thuật phản ánh các phép xử lý, trình tự xử lý dữ liệu.

Nếu xây dựng cấu trúc dữ liệu không hợp lý thì dù có giải thuật tốt đến mấy cũng khó giải quyết được bài toán, hoặc có thể giải quyết được nhưng tốn rất nhiều thời gian và dễ bị lỗi.

Như vậy trong một dự án Tin học, *giải thuật và cấu trúc dữ liệu* sẽ có 1 mối quan hệ chặt chẽ với nhau được mô tả qua công thức:

Cấu trúc dữ liệu + Giải thuật = Chương trình

Với một cấu trúc dữ liệu được chọn, sẽ có những giải thuật tương ứng. Khia cấu trúc dữ liệu thay đổi chúng ta cũng cần thay đổi giải thuật để giải quyết bài toán.

2. Các tiêu chuẩn đánh giá cấu trúc dữ liệu

Do tầm quan trọng của Cấu trúc dữ liệu nên cần phải có các tiêu chuẩn để xây dựng cấu trúc dữ liệu tối ưu. Một cấu trúc dữ liệu tốt phải thoả các yêu cầu sau:

Phản ánh đúng thực tế

Đây là tiêu chuẩn quan trọng nhất, quyết định tính đúng của bài toán.

Ví dụ: Chọn kiểu số nguyên **int** để biểu diễn tiền thưởng của nhân viên dựa trên *Trị giá hàng x 5,3%,* nếu vậy kết quả sẽ bị làm tròn ra số nguyên mà thực tế kết quả có thể có phần lẻ.

Tiết kiệm tài nguyên hệ thống

Cấu trúc dữ liệu chỉ nên sử dụng tài nguyên hệ thống (CPU và RAM) vừa đủ để đạt hiệu quả tốc độ xử lý tối ưu

Ví dụ: Cần khai báo kiểu dữ liệu cho cột **Tuổi** chúng ta chỉ cần sử dụng kiểu **Byte** là vừa đủ không cần sử dụng kiểu **Int**.

3. Kiểu dữ liệu

Máy tính thực sự chỉ có thể lưu trữ dữ liệu ở dạng nhị phân. Nếu muốn phản ánh các dữ liệu trong thực tế, chúng ta cần ánh xạ dữ liệu thực tế thành các kiểu dữ liệu.

3.1. Định nghĩa kiểu dữ liệu

Một kiểu dữ liệu **T** được xác định bởi một bộ **<V,O>** trong đó:
- **V** là tập hợp các giá trị mà 1 đối tượng kiểu **T** có thể lưu trữ
- **O** là tập hợp các thao tác xử lý có thể được thi hành trên đối tượng kiểu **T**

Ví dụ: Kiểu số nguyên **Integer** = <V,O> với
V = {-32768.. 32767}
O = {+, -, *, /, %}
Các thuộc tính của 1 kiểu dữ liệu bao gồm:

1. *Tên kiểu*
2. *Miền giá trị*
3. *Kích thước lưu trữ*
4. *Các phép toán xử lý trên kiểu*

3.2. Các kiểu dữ liệu cơ bản

Các kiểu dữ liệu cơ bản là các loại dữ liệu đơn giản, không có cấu trúc, như: số nguyên, số thực, các ký tự, các giá trị logic.

Mỗi ngôn ngữ lập trình thường dựng sẵn các kiểu này.

Thông thường các kiểu dữ liệu cơ bản bao gồm:

- *Kiểu có thứ tự rời rác: số nguyên, ký tự, logic, liệt kê…*
- *Kiểu không rời rạc: số thực*

3.3. Các kiểu dữ liệu có cấu trúc

Trong thực tế với các kiểu cơ bản như vậy chưa đủ phản ánh dữ liệu trong thực tế; Như vậy chúng ta cần phải xây dựng thêm các kiểu mới dựa trên những kiểu cơ bản. Những kiểu dữ liệu xây dựng như thế được gọi là các kiểu dữ liệu có cấu trúc.

Hầu hết các ngôn ngữ lập trình đều cung cấp sẵn một số kiểu có cấu trúc như:

- *Kiểu mảng*
- *Kiểu bản ghi*
- *Kiểu tập tin ..*

Đồng thời cung cấp thêm cho lập trình viên 1 cơ chế để tự định nghĩa các kiểu dữ liệu mới

4. Đánh giá độ phức tạp của giải thuật

Một bài toán thực tế đều có thể giải được bằng nhiều thuật toán khác nhau, nhưng chúng ta nên chọn 1 thuật toán tối ưu nhất để xử lý nó.

Khi nói đến hiệu quả 1 thuật toán người ta thường đề cập đến các vấn đề sau:

1. Hao tốn tài nguyên bộ nhớ (RAM)
2. Thời gian chiếm dụng CPU

Để đánh giá 1 thuật toán bằng thực nhiệm thường thực hiện qui trình như sau:

1. Cài đặt thuật toán bằng 1 ngôn ngữ lập trình
2. Đưa dữ liệu mẫu vào chương trình để Test
3. Thống kê các kết quả nhận được qua mỗi lần Test
4. Đánh giá thuật toán qua việc tổng hợp test

Tuy nhiên đó chỉ là 1 phương pháp gặp phải 1 số khó khăn:

- Do phải cài đặt trên 1 ngôn ngữ nên sẽ gặp phải hạn chế của ngôn ngữ lập trình này.
- Hiệu quả thuật toán sẽ bị ảnh hưởng bởi người cài đặt
- Việc tạo ra các bộ dữ liệu để Test tốn nhiều chi phí
- Các số liệu test nhận được phụ thuộc vào phần cứng của máy tính dùng để test

Vì vậy song song đánh giá thuật toán bằng phương pháp thực nghiệm, người ta còn dùng phương pháp đánh giá thuật toán theo hướng xấp xỉ tiệm cận qua các khái niệm toán học O lớn, o nhỏ