**Chương 1: Cấu trúc dữ liệu và giải thuật.**

**(Data Structure and Algorithms)**

**Câu 1: Trong khoa học máy tính, cấu trúc dữ liệu được hiểu như thế nào? Cho ví dụ.**

Cấu trúc dữ liệu (Data Structure) là cách lưu trữ, tổ chức dữ liệu có thứ tự, có hệ thống để dữ liệu có thể được sử dụng một cách hiệu quả.

Đặc điểm của một cấu trúc dữ liệu:

- Chính xác: Sự triển khai của Cấu trúc dữ liệu nên triển khai Interface của nó một cách chính xác.

- Độ phức tạp về thời gian (Time Complexity): Thời gian chạy hoặc thời gian thực thi của các phép tính của cấu trúc dữ liệu phải là nhỏ nhất có thể.

- Độ phức tạp về bộ nhớ (Space Complexity): Sự sử dụng bộ nhớ của mỗi phép tính của cấu trúc dữ liệu nên là nhỏ nhất có thể.

Ví dụ:

- Về tìm kiếm dữ liệu: Giả sử có 1 triệu hàng hóa được lưu giữ vào trong kho hàng hóa. Và giả sử có một ứng dụng cần để tìm kiếm một hàng hóa. Thì mỗi khi thực hiện tìm kiếm, ứng dụng này sẽ phải tìm kiếm 1 hàng hóa trong 1 triệu hàng hóa. Khi dữ liệu tăng lên thì việc tìm kiếm sẽ càng trở lên chậm và tốn kém hơn.

Tốc độ bộ vi xử lý: Mặc dù bộ vi xử lý có tốc độ rất cao, tuy nhiên nó cũng có giới hạn và khi lượng dữ liệu lên tới hàng tỉ bản ghi thì tốc độ xử lý cũng sẽ không còn được nhanh nữa.

Đa yêu cầu: Khi hàng nghìn người dùng cùng thực hiện một phép tính tìm kiếm trên một Web Server thì cho dù Web Server đó có nhanh đến mấy thì việc phải xử lý hàng nghìn phép tính cùng một lúc là thực sự rất khó.

=> Để xử lý các vấn đề trên, cấu trúc dữ liệu là một giải pháp tốt nhất. Dữ liệu có thể được tổ chức trong cấu trúc dữ liệu theo một cách để khi thực hiện tìm kiếm một phần tử nào đó thì dữ liệu yêu cầu sẽ được tìm thấy ngay lập tức.

**Câu 2: Trong khoa học máy tính, giải thuật được hiểu như thế nào? Cho ví dụ.**

Giải thuật (hay còn gọi là thuật toán - tiếng Anh là Algorithms) là một tập hợp hữu hạn các chỉ thị để được thực thi theo một thứ tự nào đó để thu được kết quả mong muốn. Nói chung thì giải thuật là độc lập với các ngôn ngữ lập trình, tức là một giải thuật có thể được triển khai trong nhiều ngôn ngữ lập trình khác nhau.

Đặc điểm của giải thuật:

- Tính xác định: Giải thuật phải rõ ràng và không mơ hồ. Mỗi một giai đoạn (hay mỗi bước) chỉ nên mang một mục đích nhất định.

- Dữ liệu đầu vào xác định: Một giải thuật nên có 0 hoặc nhiều hơn dữ liệu đầu vào đã xác định.

- Kết quả đầu ra: Một giải thuật nên có một hoặc nhiều dữ liệu đầu ra đã xác định, và nên kết nối với kiểu kết quả mong muốn.

-Tính dừng: Các giải thuật phải kết thúc sau một số hữu hạn các bước.

- Tính hiệu quả: Một giải thuật nên là có thể thi hành được với các nguồn có sẵn, tức là có khả năng giải quyết hiệu quả vấn đề trong điều kiện thời gian và tài nguyên cho phép.

- Tính phổ biến: Một giải thuật có tính phổ biến nếu giải thuật này có thể giải quyết được một lớp các vấn đề tương tự.

- Độc lập: Một giải thuật nên có các chỉ thị độc lập với bất kỳ phần code lập trình nào.

Ví dụ viết giải thuật

Bài toán: Thiết kế một giải thuật để cộng hai số và hiển thị kết quả.

Bước 1: Bắt đầu

Bước 2: Lấy giá trị của a & b

Bước 3: c ← a + b

Bước 4: Hiển thị c

Bước 5: Kết thúc.

**Câu 3: Tại sao nói cấu trúc dữ liệu và giải thuật có quan hệ mật thiết với nhau? Liệt kê một ví dụ nói về cách thiết kế cấu trúc dữ liệu sẽ ảnh hưởng đến giải thuật, giải thích tại sao?**

Mối liên hệ giữa cấu trúc dữ liệu và giải thuật:

Thực hiện một đề án tin học là chuyển bài toán thực tế thành bài toán có thể giải quyết trên máy tính. Một bài toán thực tế bất kỳ đều bao gồm các đối tượng dữ liệu và các yêu cầu xử lý trên những đối tượng đó. Vì thế, để xây dựng một mô hình tin học phản ánh được bài toán thực tế cần chú trọng đến hai vấn đề :

- Tổ chức biểu diễn các đối tượng thực tế :

Các thành phần dữ liệu thực tế đa dạng, phong phú và thường chứa đựng những quan hệ nào đó với nhau, do đó trong mô hình tin học của bài toán, cần phải tổ chức, xây dựng các cấu trúc thích hợp nhất sao cho vừa có thể phản ánh chính xác các dữ liệu thực tế này, vừa có thể dễ dàng dùng máy tính để xử lý. Công việc này được gọi là xây dựng cấu trúc dữ liệu cho bài toán.

- Xây dựng các thao tác xử lý dữ liệu:

Từ những yêu cầu xử lý thực tế, cần tìm ra các giải thuật tương ứng để xác định trình tự các thao tác máy tính phải thi hành để cho ra kết quả mong muốn, đây là bước xây dựng giải thuật cho bài toán.

Tuy nhiên khi giải quyết một bài toán trên máy tính, chúng ta thường có khuynh hướng chỉ chú trọng đến việc xây dựng giải thuật mà quên đi tầm quan trọng của việc tổ chức dữ liệu trong bài toán. Giải thuật phản ánh các phép xử lý, còn đối tượng xử lý của giải thuật lại là dữ liệu, chính dữ liệu chứa đựng các thông tin cần thiết để thực hiện giải thuật. Để xác định được giải thuật phù hợp cần phải biết nó tác động đến loại dữ liệu nào (ví dụ để làm nhuyễn các hạt đậu, người ta dùng cách xay chứ không băm bằng dao, vì đậu sẽ văng ra ngoài) và khi chọn lựa cấu trúc dữ liệu cũng cần phải hiểu rõ những thao tác nào sẽ tác động đến nó (ví dụ để biểu diễn các điểm số của sinh viên người ta dùng số thực thay vì chuỗi ký tự vì còn phải thực hiện thao tác tính trung bình từ những điểm số đó). Như vậy trong một đề án tin học, giải thuật và cấu trúc dữ liệu có mối quan hệ chặt chẽ với nhau, được thể hiện qua công thức:

**Cấu trúc dữ liệu + Giải thuật = Chương trình**

Với một cấu trúc dữ liệu đã chọn, sẽ có những giải thuật tương ứng, phù hợp. Khi cấu trúc dữ liệu thay đổi thường giải thuật cũng phải thay đổi theo để tránh việc xử lý gượng ép, thiếu tự nhiên trên một cấu trúc không phù hợp. Hơn nữa, một cấu trúc dữ liệu tốt sẽ giúp giải thuật xử lý trên đó có thể phát huy tác dụng tốt hơn, vừa đáp ứng nhanh vừa tiết kiệm vật tư, giải thuật cũng dễ hiễu và đơn giản hơn.

Ví dụ nói về cách thiết kế cấu trúc dữ liệu sẽ ảnh hưởng đến giải thuật, giải thích

Một chương trình quản lý điểm thi của sinh viên cần lưu trữ các điểm số của 3 sinh viên. Do mỗi sinh viên có 4 điểm số ứng với 4 môn học khác nhau nên dữ liệu có dạng bảng như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sinh viên | Môn 1 | Môn 2 | Môn3 | Môn4 |
| SV 1 | 7 | 9 | 5 | 2 |
| SV 2 | 5 | 0 | 9 | 4 |
| SV 3 | 6 | 3 | 7 | 4 |

Chỉ xét thao tác xử lý là xuất điểm số các môn của từng sinh viên.

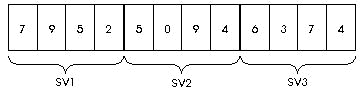
Giả sử có các phương án tổ chức lưu trữ sau:

**Phương án 1 : Sử dụng mảng một chiều** *(Cấu trúc dữ liệu 1)*

Có tất cả 3(SV)\*4(Môn) = 12 điểm số cần lưu trữ, do đó khai báo mảng result như sau :

int result [ 12 ] = {7, 9, 5, 2, 5, 0, 9, 4, 6, 3, 7, 4};

khi đó trong mảng result các phần tử sẽ được lưu trữ như sau:



Và truy xuất điểm số môn j của sinh viên i - là phần tử tại (dòng i, cột j) trong bảng - phải sử dụng một công thức xác định chỉ số tương ứng trong mảng result:

result[((i-1)\*số cột) + j] &&bảngđiểm(dòng i, cột j)

Ngược lại, với một phần tử bất kỳ trong mảng, muốn biết đó là điểm số của sinh viên nào, môn gì, phải dùng công thức xác định sau

result[ i ] && bảngđiểm (dòng((i / số cột) +1), cột (i % số cột) )

Với phương án này, thao tác xử lý được cài đặt như sau: *(Giải thuật 1****)***

void XuatDiem() //Xuất điểm số của tất cả sinh viên

{

const int so\_mon = 4;int sv,mon;for (int i=0; i<12; i+)

{

sv = i/so\_mon; mon = i % so\_mon; printf("Điểm môn %d của sv %d là: %d", mon, sv,result[i]);

}

}

**Phương án 2 : Sử dụng mảng 2 chiều** *(Cấu trúc dữ liệu 2)*

Khai báo mảng 2 chiều result có kích thước 3 dòng\* 4 cột như sau :

int result[3][4] ={{ 7, 9, 5, 2},{ 5, 0, 9, 4},{ 6, 3, 7, 4 }};

khi đó trong mảng result các phần tử sẽ được lưu trữ như sau :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cột 0 | Cột 1 | Cột 2 | Cột 3 |
| Dòng 0 | result[0][0] =7 | result[0][1] =9 | result[0][2] =5 | result[0][3] =2 |
| Dòng 1 | result[1][0] =5 | result[1][1] =0 | result[1][2] =9 | result[1][3] =4 |
| Dòng 2 | result[2][0] =6 | result[2][1] =3 | result[2][2] =7 | result[2][3] =4 |

Và truy xuất điểm số môn j của sinh viên i - là phần tử tại (dòng i, cột j) trong bảng - cũng chính là phần tử nằm ở vị trí (dòng i, cột j) trong mảng bảngđiểm(dòng i,cột j) Þ result[ i] [j]

Với phương án này, thao tác xử lý được cài đặt như sau: *(Giải thuật 2)*

void XuatDiem() //Xuất điểm số của tất cả sinh viên

{

int so\_mon = 4, so\_sv =3;for ( int i=0; i<so\_sv; i+) for ( int j=0; i<so\_mon; j+) printf("Điểm môn %d của sv %d là: %d", j, i,result[i][j]);

}

NHẬN XÉT

Có thể thấy rõ phương án 2 cung cấp một cấu trúc lưu trữ phù hợp với dữ liệu thực tế hơn phương án 1, và do vậy giải thuật xử lý trên cấu trúc dữ liệu của phương án 2 cũng đơn giản, tự nhiên hơn.

**Câu 4: Đếm số phép so sánh trong giải thuật ở ví dụ 1.12**

**Ví dụ 1.12**: Phân tích độ phức tạp của giải thuật Insertion Sort

void InsertionSort(int a[], int n)

{

int i, j, x;

for(i=1; i<n; i++) (i chạy từ 1 đến n-1; i ++ tương đương i=i+1)

{

x=a[i]; j=i; (x có n-1 giá trị; i có n-1 giá trị)

while (j>0 && a[j-1]>x) (so sánh j và x: n-1 lần

{

a[j]=a[j-1];

j--;

}

a[j]=x;

}

}

Kết luận: Số phép so sánh trong giải thuật InsertionSort là (n-1)