1) Trình bày mối quan hệ giữa cấu trúc giữ liệu và giải thuật, cho ví dụ minh hoạ.

2) Trình bày mối quan hệ giữa cấu trúc dữ liệu và các phép toán trên cấu trúc dữ liệu.

3) Trình bày sự khác nhau giữ cấu trúc dữ liệu và cấu trúc lưu trữ, cho ví dụ minh hoạ.

4) Trình bày những đặc điểm về cấu trúc dữ liệu trong các ngôn ngữ lập trình bậc cao, có liên hệ với ngôn ngữ C.

5) Trình bày nguyên tắc thiết kế Top-Down, cho ví dụ minh hoạ.

**1) Trình bày mối quan hệ giữa cấu trúc giữ liệu và giải thuật, cho ví dụ minh hoạ.**

Cấu trúc dữ liệu và giải thuật có mối quan hệ mật thiết.

Giải thuật là một hệ thống chặt chẽ và rõ ràng các qui tắc nhằm xác định 1 dãy các thao tác trên những đối tượng, sao cho sau 1 số bước hữu hạn thực hiện các thao tác đó ta thu được kết quả mong muốn.

Cấu trúc dữ liệu: là cách tổ chức, lưu trữ dữ liệu trong máy tính điện tử 1 cách có thứ tự, có hệ thống nhằm sử dụng dữ liệu 1 cách hiệu quả (trong giáo trình không có định nghĩa này)

Cấu trúc dữ liệu và giải thuật có mối liên hệ chặt chẽ với nhau, chúng luôn tồn tại song song đi kèm theo công thức: cấu trúc dữ liệu + giải thuật = chương trình.

Bản thân các phần tử của dữ liệu thường có mối quan hệ với nhau, ngoài ra nếu biết tổ chức chúng theo các cấu trúc dữ liệu thích hợp thì việc thực hiện các phép xử lý trên các dữ liệu sẽ càng thuận lợi hơn, đạt hiệu quả cao hơn. Với 1 cấu trúc dữ liệu đã chọn, ta sẽ có giải thuật xử lý tương ứng. Cấu trúc dữ liệu thay đổi thì giải thuật cũng thay đổi theo. Để có 1 ctrinh tốt, ta cần phải chọn được ctdl phù hợp và chọn được 1 gt đúng đắn.

Vậy, giữa cấu trúc dữ liệu và giải thuật có mối quan hệ mật thiết. Có thể coi chúng như hình với bóng, không thể nói tới cái này mà không nhắc tới cái kia.

Vd: Giả sử ta có 1 danh sách các trường đại học và cao đẳng trên cả nước mỗi trường có các thông tin sau: Tên trường, địa chỉ, sđt phòng đào tạo. Ta muốn viết một chương trình trên máy tính điện tử để khi cho biết “tên trường” máy sẽ hiện ra màn hình cho ta: “địa chỉ” và “số điện thoại phòng đào tạo” của trường đó.

Một cách đơn giản là cứ duyệt tuần tự các tên trường trong danh sách cho tới khi tìm thấy tên trường cần tìm, thì sẽ đối chiếu ra “địa chỉ” và “số điện thoại phòng đào tạo” của trường đó. Cách tìm tuần tự này rõ ràng chỉ chấp nhận được khi danh sách ngắn, còn danh sách dài thì rất mất thời gian.

Nếu ta biết tổ chức lại danh sách bằng cách sắp xếp theo thứ tự từ điển của tên trường, thì có thể áp dụng một giải thuật tìm kiếm khác tốt hơn, tương tự như ta vẫn thường làm khi tra từ điển. Cách tìm này nhanh hơn cách trên rất nhiều nhưng rõ ràng không thể áp dụng được với dữ liệu chưa được sắp xếp.

Nếu lại biết tổ chức thêm một bảng mục lục chỉ dẫn theo chữ cái đầu tiên của tên trường, thì khi tìm “địa chỉ” và “số điện thoại phòng đào tạo” của Học viện Kỹ thuật Mật mã, ta sẽ bỏ qua được các tên trường mà chữ cái đầu không phải là “H”.

**2) Trình bày mối quan hệ giữa cấu trúc dữ liệu và các phép toán trên cấu trúc dữ liệu.**

Đối với các bài toán phi số, đi đôi với các cấu trúc dữ liệu mới cũng xuất hiện các phép toán mới tác động trên các cấu trúc ấy. Thông thường có các phép toán như: phép tạo lập hoặc hủy bỏ một cấu trúc, phép truy nhập vào từng phần tử của cấu trúc, phép bổ sung hoặc loại bỏ một phần tử trên cấu trúc…

Các phép toán đó sẽ có những tác dụng khác nhau đối với từng cấu trúc. Có phép toán hữu hiệu đối với cấu trúc này nhưng lại tỏ ra không hữu hiệu trên các cấu trúc khác.

Vì vậy khi chọn một cấu trúc dữ liệu ta phải nghĩ ngay tới các phép toán tác động trên cấu trúc ấy và ngược lại, nói tới phép toán thì lại phải chú ý tới phép đó được tác động trên cấu trúc dữ liệu nào. Cho nên người ta thường quan niệm: nói tới cấu trúc dữ liệu là bao hàm luôn cả phép toán tác động đến cấu trúc ấy.

**3) Trình bày sự khác nhau giữ cấu trúc dữ liệu và cấu trúc lưu trữ, cho ví dụ minh hoạ.**

Các cách biểu diễn một cấu trúc dữ liệu trong bộ nhớ máy tính điện tử được gọi là cấu trúc lưu trữ (storage structures). Đó chính là cách cài đặt cấu trúc ấy trên máy tính điện tử, và trên cơ sở cấu trúc lưu trữ này mà thực hiện các phép xử lý. Ta cần phân biệt giữa cấu trúc dữ liệu và cấu trúc lưu trữ tương ứng. Có thể có nhiều cấu trúc lưu trữ khác nhau cho cùng một cấu trúc dữ liệu, cũng như có thể có những cấu trúc dữ liệu khác nhau mà được thể hiện trong bộ nhớ bởi cùng một kiểu cấu trúc lưu trữ.

Ví dụ: Cấu trúc lưu trữ kế tiếp (mảng) và cấu trúc lưu trữ móc nối đều có thể được dùng để cài đặt cấu trúc dữ liệu ngăn xếp (STACK). Mặt khác, các cấu trúc dữ liệu như: danh sách, ngăn xếp và cây đều có thể cài đặt trên máy thông qua cấu trúc lưu trữ móc nối.

**4) Trình bày những đặc điểm về cấu trúc dữ liệu trong các ngôn ngữ lập trình bậc cao, có liên hệ với ngôn ngữ C.**

Một cấu trúc dữ liệu là một dữ liệu phức hợp, gồm nhiều thành phần dữ liệu, mỗi thành phần hoặc là dữ liệu cơ sở (số nguyên, số thực, ký tự,… ) hoặc là một cấu trúc dữ liệu đã được xây dựng. Các thành phần dữ liệu tạo nên một cấu trúc dữ liệu được liên kết với nhau theo một cách nào đó.

Trong các ngôn ngữ lập trình bậc cao, các dữ liệu được phân thành các kiểu dữ liệu. Kiểu dữ liệu của một biến được xác định bởi một tập các giá trị mà biến đó có thể nhận và các phép toán có thể thực hiện trên các giá trị đó. Mỗi ngôn ngữ lập trình cung cấp cho chúng ta một số kiểu dữ liệu cơ bản (basic data types). Trong các ngôn ngữ lập trình khác nhau, các kiểu dữ liệu cơ bản có thể khác nhau. Các ngôn ngữ lập trình như: Pascal, C / C + + … có các kiểu dữ liệu cơ bản rất phong phú.

Các kiểu dữ liệu được tạo thành từ nhiều kiểu dữ liệu khác (các kiểu này có thể là kiểu cơ bản hoặc kiểu dữ liệu đã được xây dựng) được gọi là kiểu dữ liệu có cấu trúc. Các dữ liệu thuộc kiểu dữ liệu có cấu trúc được gọi là các cấu trúc dữ liệu. Ví dụ, các mảng, các cấu trúc, các danh sách móc nối … trong ngôn ngữ C, là các cấu trúc dữ liệu.

Từ các kiểu cơ bản, bằng cách sử dụng các qui tắc cú pháp để kiến tạo các kiểu dữ liệu, người lập trình có thể xây dựng nên các kiểu dữ liệu mới thích hợp cho từng vấn đề. Các kiểu dữ liệu mà người lập trình xây dựng nên được gọi là các kiểu dữ liệu được xác định bởi người sử dụng (user-defined data types).

Trong ngôn ngữ lập trình C phương pháp để liên kết dữ liệu :

+) Liên kết dữ liệu cùng kiểu tạo thành mảng dữ liệu.

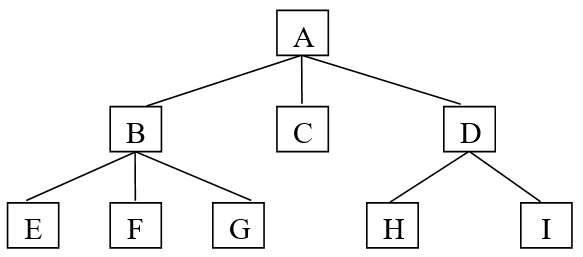
+) Liên kết các dữ liệu thành mảng cấu trúc trong C.

+) Sử dụng con trỏ để liên kết dữ liệu.

**5) Trình bày nguyên tắc thiết kế Top-Down, cho ví dụ minh hoạ.**

Trước những bài toán và vấn đề ngày một đa dạng và phức tạp và các giải thuật và chương trình để giải chúng có quy mô ngày càng lớn, ta nhận thấy rằng mọi việc sẽ đơn giản hơn nếu như chia được bài toán thành nhiều bài toán nhỏ hơn. Điều đó cũng có nghĩa là nếu coi bài toán của ta như một mô-đun chính thì cần chia nó thành các mô-đun con, và dĩ nhiên, với tinh thần như thế, đến lượt nó, mỗi môn-đun con này lại được chia tiếp cho tới những mô-đun ứng với các phần việc cơ bản mà ta đã biết cách giải quyết.

Như vậy, việc tổ chức lời giải của bài toán sẽ được thể hiện theo một cấu trúc phân cấp có dạng như sau:



Cách giải quyết bài toán theo tinh thần như vậy được gọi là chiến thuật “chia để trị” (divide and conquer). Để thể hiện chiến thuật đó, người ta dùng cách thiết kế “ đỉnh\_xuống” (top-down design).

Đó là cách phân tích tổng quát toàn bộ vấn đề, xuất phát từ dữ kiện và các mục tiêu đặt ra để đề cập đến những công việc chủ yếu trước, rồi sau đó mới đi dần vào giải quyết các phần việc cụ thể một cách chi tiết hơn, cũng vì vậy mà người ta còn gọi cách thiết kế này là cách thiết kế từ khái quát đến chi tiết.

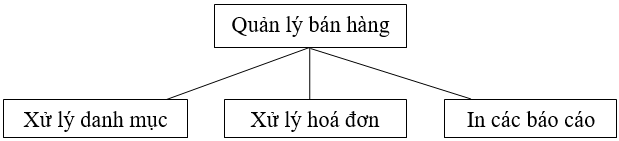
Ví dụ: Viết chương trình quản lý bán hàng chạy trên máy tính, với các yêu cầu là: Hàng ngày phải nhập các hóa đơn bán hàng, hóa đơn nhập hàng, tìm kiếm các hóa đơn đã nhập để xem hoặc sửa lại. in các hóa đơn cho khách hàng; tính doanh thu, lợi nhuận trong khoảng thời gian bất kì; tính tổng hợp kho, tính doanh số của từng mặt hàng, từng khách hàng.

→ Đối diện với nhiều yêu cầu, ta không thể có ngay giải thuật để xử lý. Lúc này, ta nên chia bài toán thành ba nhiệm vụ chính cần giải quyết là:

1) Xử lý các danh mục để quản lý và theo dõi các thông tin về hàng hoá và khách hàng.

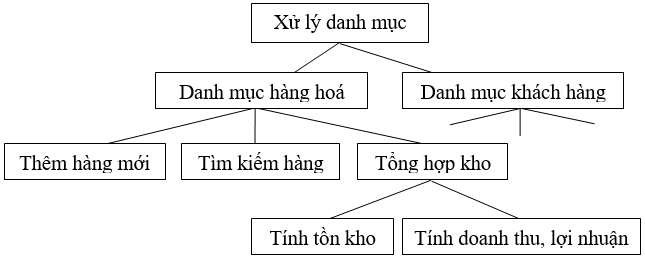
2) Xử lý dữ liệu về các hoá đơn bán hàng, hoá đơn nhập hàng.

3) In các báo cáo về doanh thu, lợi nhuận.



Từ ba nhiệm vụ chính, ta cần chia tiếp thành các nhiệm vụ nhỏ hơn.

“Xử lý danh mục” được chia làm hai là “danh mục hàng hóa” và “ danh mục khách hàng”. Trong “Danh mục hàng hoá” lại có thể chia thành các nhiệm vụ nhỏ hơn như thêm hàng mới, tìm kiếm hàng, tổng hợp kho… Các nhiệm vụ con này có thể chia thành những nhiệm vụ nhỏ hơn nữa.



Trong thực tế, việc phân tích bài toán thành các bài toán con như thế không phải là việc dễ dàng. Chính vì vậy mà có những bài toán, nhiệm vụ phân tích và thiết kế giải thuật giải bài toán còn mất nhiều thời gian và công sức hơn cả nhiệm vụ lập trình.

Cách thiết kế giải thuật theo kiểu top-down như trên giúp cho việc giải quyết bài toán được định hướng rõ ràng, tránh sa đà ngay vào các chi tiết phụ. Nó cũng là các nền tảng cho việc lập trình có cấu trúc.

Thông thường, đối với các bài toán lớn, việc giải quyết nó phải do nhiều người cùng làm . Chính phương pháp mô đun hóa sẽ cho phép tách bài toán ra thành các phần độc lập, tạo điều kiện cho các nhóm giải quyết phần việc của mình mà không ảnh hưởng gì đến nhóm khác. Với chương trình được xây dựng trên cơ sở của các giải thuật được thiết kế theo cách này, thì việc tìm hiểu cũng như sửa chữa, chỉnh lí sẽ đơn giản hơn.