Bài 4

Logic lập trình
Mô-đun hoá chương trình

Nội dung

- Mô-đun hóa
- Mô-đun hóa một chương trình
- Khai báo và sử dụng biến trong mô-đun
- Mô-đun có tham số và giá trị trả về
- Cấu trúc chương trình mô-đun hóa
- Biểu đồ phân cấp chương trình

Mô-đun hóa

- **Mô-đun hóa** là quá trình chia nhỏ một chương trình lớn thành các mô-đun.
- Mô-đun hóa cung cấp sự trừu tượng.
  - Chú ý đến các đặc điểm quan trọng trong khi bỏ qua các chi tiết không thiết yếu.
  - Giúp người lập trình dễ dàng thấy được "bức tranh toàn cảnh" của chương trình.
- Mô-đun hóa cho phép nhiều lập trình viên cùng giải quyết một bài toán lập trình lớn.
  - Dễ dàng chia các nhiệm vụ cho nhiều lập trình viên hoặc nhiều nhóm cùng thực hiện.
- Mô-đun hóa giúp tái sử dụng và tăng độ tin cậy của mã nguồn.

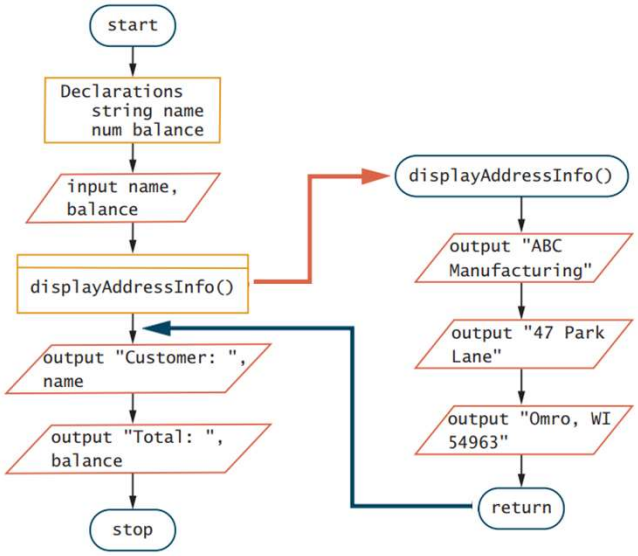
Mô-đun hóa một chương trình (1)

- **Chương trình chính** chứa các bước cơ bản hoặc logic chính của chương trình.
  - Chương trình chính truy cập các mô-đun chứa các chỉ thị chi tiết hơn.
- Một mô-đun khi được tạo bao gồm 3 phần sau:
  - **Tiêu đề mô-đun** có tên mô-đun hoặc thông tin khác như các tham số hình thức, kiểu trả về.
  - **Thân mô-đun** chứa các lệnh của mô-đun.
  - **Lệnh trả về của mô-đun (lệnh return)** đánh dấu kết thúc mô-đun và xác định vị trí điều khiển quay trở lại nơi gọi mô-đun.

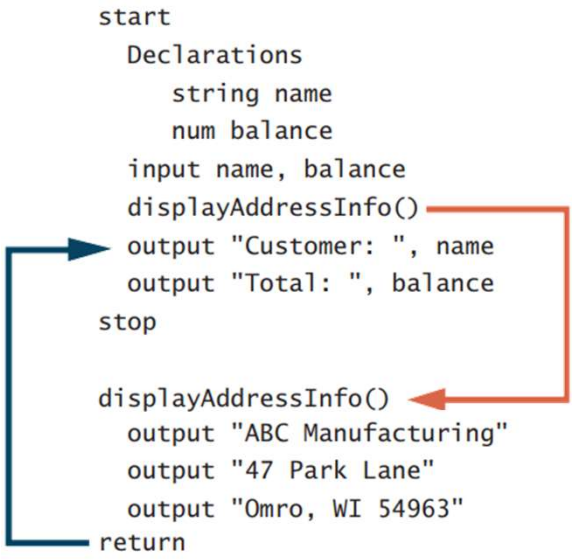
Mô-đun hóa một chương trình (2)

- Đặt tên cho mô-đun tương tự như đặt cho biến
  - Gồm các chữ cái, chữ số, dấu gạch nối _.
  - Dùng động từ hoặc cụm động từ.
  - Thêm cặp dấu ( ) vào cuối tên để phân biệt với tên biến
- Khi muốn sử dụng một mô-đun, thực hiện **lệnh gọi mô-đun** bằng cách sử dụng tên của mô-đun.
- Kí hiệu gọi mô-đun trong lưu đồ
  - Hình chữ nhật có thanh ngang ở trên, bên trong là lệnh gọi mô-đun.

Trình bày mô-đun bằng lưu đồ



Trình bày mô-đun bằng mã giả

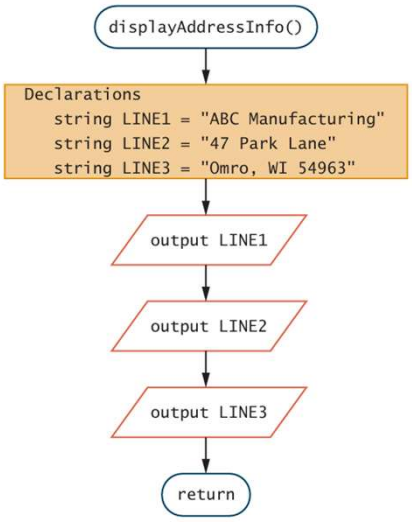


```
start
  Declarations
     string name
     num balance
  input name, balance
  displayAddressInfo()
  output "Customer: ", name
  output "Total: ", balance
stop

displayAddressInfo()
  output "ABC Manufacturing"
  output "47 Park Lane"
  output "Omro, WI 54963"
return
```


Khai báo và dùng biến trong mô-đun (1)

- Trong các mô-đun có thể đặt lệnh
  - Cho một hoạt động bất kì (đầu vào, xử lý, đầu ra).
  - Khai báo các biến và hằng.
- Các biến và hằng được khai báo trong mô-đun chỉ có thể được sử dụng trong mô-đun, chúng có tính **cục bộ**.
- Các biến và hằng được khai báo ở cấp độ chương trình (thường là bên ngoài tất cả các mô-đun) có thể được sử dụng trong các mô-đun của chương trình, chúng có tính **toàn cục**.

Khai báo và dùng biến trong mô-đun (2)

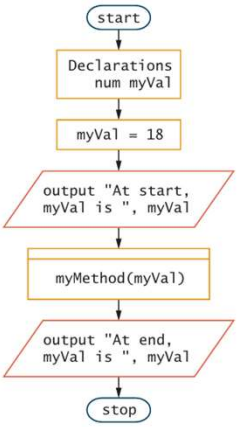


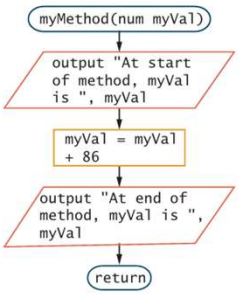

```
start
  Declarations
     string name
     num balance
  input name, balance
  displayAddressInfo()
  output "Customer: ", name
  output "Total: ", balance
stop

displayAddressInfo()
  Declarations
     string LINE1 = "ABC Manufacturing"
     string LINE2 = "47 Park Lane"
     string LINE3 = "Omro, WI 54963"
  output LINE1
  output LINE2
  output LINE3
return
```


Mô-đun có tham số và giá trị trả về (1)

- Nếu mô-đun cần các giá trị từ bên ngoài để thực hiện công việc:
  - Ta phải khai báo các **tham số** trong phần tiêu đề của mô-đun để nhận các giá trị.
  - Khi gọi, các giá trị cần gửi cho mô-đun được đặt vào vị trí tương ứng với tham số trong lời gọi mô-đun. Các giá trị này còn được gọi là **đối số**.
- Các tham số cũng được xem như các biến của mô-đun, chúng cũng có tính cục bộ.
- Nếu mô-đun cần trả về một giá trị khi kết thúc công việc, dùng lệnh return với giá trị cần trả về.

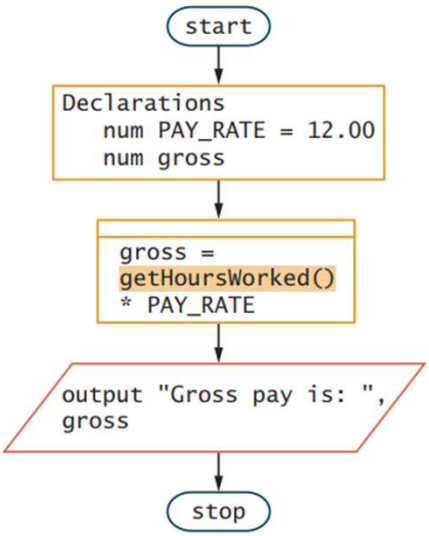
Mô-đun có tham số và giá trị trả về (2)

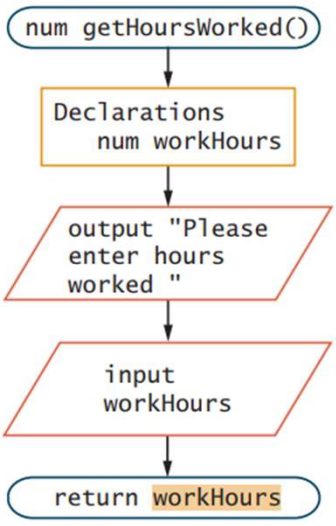





```
start
   Declarations
      num myVal
   myVal = 18
   output "At start, myVal is ", myVal
   myMethod(myVal)
   output "At end, myVal is ", myVal
stop

myMethod(num myVal)
   output "At start of method, myVal is ", myVal
   myVal = myVal + 86
   output "At end of method, myVal is ", myVal
return
```


Mô-đun có tham số và giá trị trả về (3)

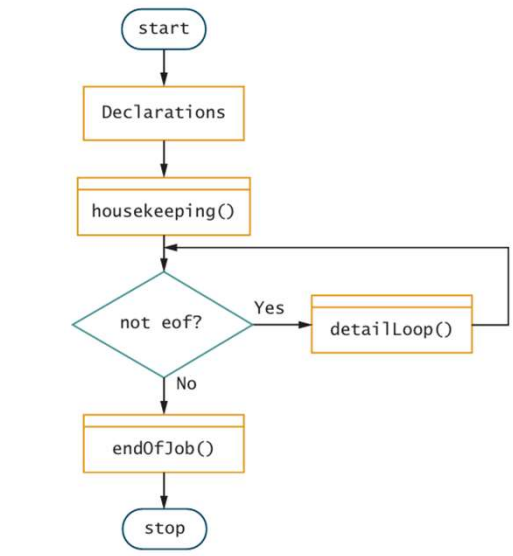




Cấu trúc chương trình mô-đun hóa (1)

- Logic chính của hầu hết mọi chương trình máy tính được mô-đun hóa có thể tuân theo cấu trúc chung như sau:
  - Khai báo cho các biến và hằng toàn cục.
  - **Công tác chuẩn bị (housekeeping tasks)** bao gồm các bước phải thực hiện khi bắt đầu một chương trình.
  - Vòng lặp cho **công tác cốt lõi (detail loop tasks)** để thực hiện lặp lại các bước chi tiết cho công việc chính của chương trình.
  - **Công tác kết thúc (end-of-job tasks)** gồm các bước phải thực hiện ở cuối chương trình.

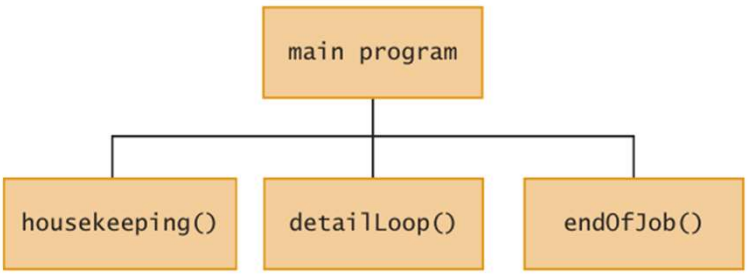
Cấu trúc chương trình mô-đun hóa (2)




```
start
  Declarations
  housekeeping()
  while not eof
     detailLoop()
  endwhile
  endOfJob()
stop
```

Biểu đồ phân cấp chương trình (1)

- Biểu diễn bức tranh tổng thể về mối liên hệ giữa các mô-đun.
- Cho ta biết mô-đun nào tồn tại trong một chương trình và mô-đun nào gọi các mô-đun khác.



Biểu đồ phân cấp chương trình (2)

- Cho biết một mô-đun cụ thể có thể được gọi từ nhiều vị trí trong chương trình.
- Công cụ lập kế hoạch, công cụ tài liệu.