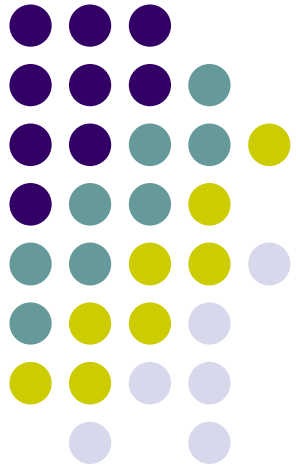
NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

PHẦN II – TIẾN TRÌNH PHẦN MỀM

Bộ môn Công nghệ phần mềm,
Khoa CNTT&TT, Đại học Cần Thơ

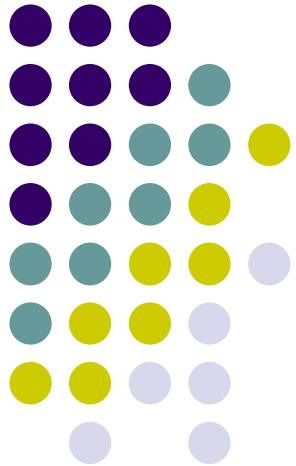


Nội dung

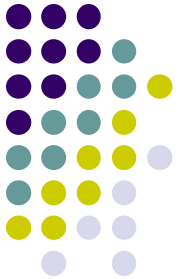
- Phân tích và Đặc tả
- Thiết kế
- Lập trình
- Kiểm thử
- Triển khai hệ thống và Bảo trì

TIẾN TRÌNH PHẦN MỀM

PHẦN II.3 – LẬP TRÌNH

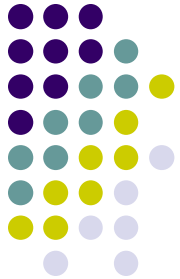


Nội dung



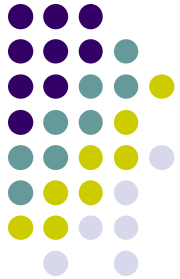
- Các chuẩn và thủ tục lập trình
- Chọn ngôn ngữ lập trình
- Nguyên tắc lập trình
- Tài liệu lập trình

Các chuẩn và thủ tục lập trình



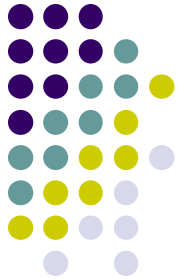
- Chuẩn và thủ tục giúp lập trình viên
  - Tổ chức các ý định và tránh các lỗi.
    - Các tài liệu theo chuẩn giúp ta quay lại công việc mà không mất dấu những gì đã làm.
    - Các tài liệu theo chuẩn giúp ta định vị các lỗi và tạo ra các thay đổi.
  - Dịch các thiết kế sang mã lệnh.

Các chuẩn và thủ tục lập trình



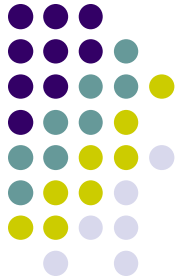
- Chuẩn và thủ tục giúp các thành viên khác
  - Giúp các thành viên khác hiểu được mã lệnh (do lập trình viên viết ra) làm gì và làm như thế nào nhằm thực hiện việc:
    - Tái sử dụng (lập trình viên khác).
    - Kiểm thử (kiểm thử viên).
    - Hiệu chỉnh hay hoàn thiện hệ thống (bảo trì viên).

Các chuẩn và thủ tục lập trình



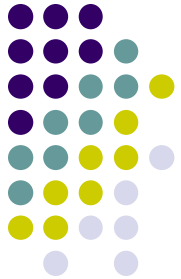
- Chuẩn và thủ tục giúp tạo ra sự tương ứng trực tiếp giữa các thành phần thiết kế và các thành phần cài đặt.
  - Các đặc trưng của chương trình nên giống như các đặc trưng của thiết kế: nối kết thấp, gắn kết cao và các giao diện rõ ràng =>
  - Các giải thuật, chức năng, giao diện và cấu trúc dữ liệu có thể được theo vết từ thiết kế sang chương trình và ngược lại một cách dễ dàng.

Chọn ngôn ngữ lập trình



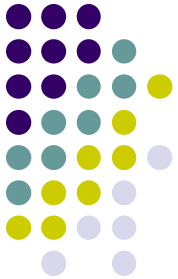
- Theo loại phần mềm
  - Phần mềm hệ thống: C, C++
  - Phần mềm thời gian thực: C, C++, Assembly
  - Phần mềm nhúng: C++, Java
  - Phần mềm nghiệp vụ (HTTT):
    - CSDL: Oracle, MySQL, SQL Server
    - NNLT: VB, VC++
  - Phần mềm trí tuệ nhân tạo: Prolog, Lisp
  - Phần mềm (dịch vụ) Web: PHP, Java Script
- Theo đặc trưng của ngôn ngữ
- Theo năng lực và kinh nghiệm của nhóm phát triển phần mềm
- Theo yêu cầu của khách hàng

Nguyên tắc lập trình



- Dù bất cứ ngôn ngữ lập trình nào được sử dụng, mỗi thành phần chương trình đều liên quan tới ít nhất 3 khía cạnh chính:
  - Cấu trúc điều khiển.
  - Giải thuật.
  - Cấu trúc dữ liệu.

Cấu trúc điều khiển



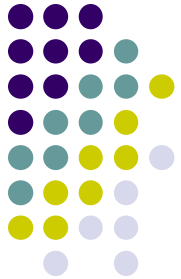
- Tạo ra mã lệnh dễ đọc

  Ví dụ: Cho đoạn chương trình

```
            benefit = minimum;
            if (age < 75) goto A;
            benefit = maximum;
            goto C;
            if (age < 65) goto B;
            if (age < 55) goto C;
      A:    if (age < 65) goto B;
            benefit = benefit * 1.5 + bonus;
            goto C;
      B:    if (age < 55) goto C;
            benefit = benefit * 1.5;
      C:    next statement
```

  Điều khiển của đoạn chương trình này bỏ qua một số câu lệnh => Rất khó đọc.

Cấu trúc điều khiển



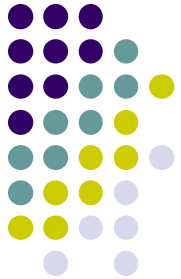
- Ví dụ: Đoạn chương trình

```
            benefit = minimum;
            if (age < 75) goto A;
            benefit = maximum;
            goto C;
            if (age < 65) goto B;
            if (age < 55) goto C;
      A:    if (age < 65) goto B;
            benefit = benefit * 1.5 + bonus;
            goto C;
      B:    if (age < 55) goto C;
            benefit = benefit * 1.5;
      C:    next statement
```

Được viết lại:

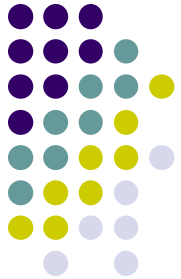
```
if (age < 55) benefit = minimum;
else if (age < 65) benefit = minimum *1.5;
else if (age < 75) benefit = minimum * 1.5 + bonus;
else benefit = maximum;
```

Cấu trúc điều khiển



- Xây dựng chương trình từ các khối mô đun nhằm làm cho hệ thống dễ hiểu, dễ kiểm thử và dễ bảo trì.
- Viết mã lệnh có tính tổng quát nhưng không làm ảnh hưởng đến sự thực hiện và sự hiểu biết.
- Sử dụng các tên tham số và các chú thích để biểu thị sự kết hợp giữa các thành phần.
- Tạo sự phụ thuộc giữa các thành phần một cách rõ ràng.

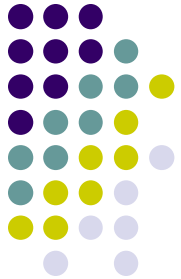
Giải thuật



- Mục tiêu chung: tăng hiệu quả thực hiện (tốc độ).
- Tính hiệu quả trong sự thực hiện có thể có các chi phí ẩn.
  - Tốn thời gian để viết mã lệnh thực thi nhanh hơn.
  - Tốn thời gian để kiểm thử mã lệnh.
  - Tốn thời gian để người dùng hiểu mã lệnh.
  - Tốn thời gian để hiệu chỉnh mã lệnh, nếu cần.

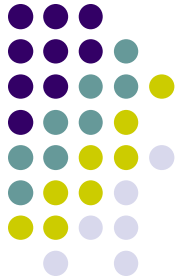
  => Cân bằng thời gian thực thi với chất lượng thiết kế, chuẩn và các yêu cầu của khách hàng.

Cấu trúc dữ liệu



- Ta nên định dạng và lưu trữ dữ liệu sao cho việc quản lý và thực hiện trên dữ liệu là không phức tạp.
- Các cấu trúc dữ liệu: danh sách, ngăn xếp, cây, v.v.
- Một số kỹ thuật sử dụng cấu trúc dữ liệu để tổ chức chương trình
  - Làm cho chương trình đơn giản.
  - Sử dụng cấu trúc dữ liệu để xác định cấu trúc chương trình.

Cấu trúc dữ liệu



- Làm cho chương trình đơn giản

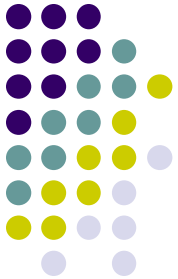
  Ví dụ: xác định thuế thu nhập

  - Với $10,000 thu nhập đầu tiên, thuế là 10%
  - Với $10,000 thu nhập tiếp theo (trên $10,000), thuế là 12%
  - Với $10,000 thu nhập tiếp theo (trên $20,000), thuế là 15%
  - Với $10,000 thu nhập tiếp theo(trên $30,000), thuế là 18%
  - Với bất cứ thu nhập nào trên $40,000, thuế là 20%

Cấu trúc dữ liệu

```
tax = 0.
if (taxable_income == 0) goto EXIT;
if (taxable_income > 10000) tax = tax + 1000;
else {
      tax = tax + .10*taxable_income;
      goto EXIT;                                          }
if (taxable_income > 20000) tax = tax + 1200;
else {
      tax = tax + .12*(taxable_income-10000):
      goto EXIT;                                          }
if (taxable_income > 30000) tax = tax + 1500;
else {
      tax = tax + .15*(taxable_income-20000);
      goto EXIT;                                          }
if (taxable_income < 40000){
      tax = tax + .18*(taxable_income-30000);
      goto EXIT;                                          }
else
      tax = tax + 1800. + .20*(taxable_income-40000);
EXIT:
```

Cấu trúc dữ liệu



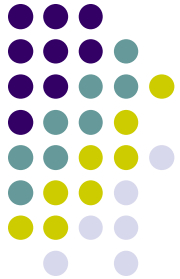
- Xác định bảng thuế

Bracket	Base	Percent
0	0	10
10,000	1000	12
20,000	2200	15
30,000	3700	18
40,000	55000	20

- Giải thuật được đơn giản hóa

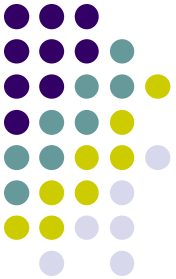
```
for (int i=2; level=1; i <= 5; i++)
        if (taxable_income > bracket[i])
                level = level + 1;
 tax = base[level]+percent[level]*(taxable_income-bracket[level]);
```

Cấu trúc dữ liệu



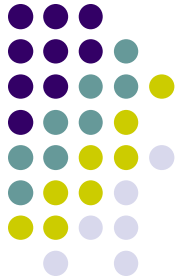
- Sử dụng cấu trúc dữ liệu để xác định cấu trúc chương trình
  - Cấu trúc dữ liệu chi phối tổ chức chương trình.
  - Cấu trúc dữ liệu có thể tác động đến việc chọn ngôn ngữ.
    - Nếu cấu trúc dữ liệu đệ quy, chọn ngôn ngữ cho phép lập trình các thủ tục đệ quy.
    - Nếu cấu trúc dữ liệu là danh sách, chọn ngôn ngữ Lisp.
    - …

Nguyên tắc chung



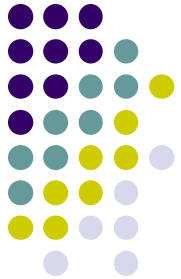
- Cục bộ hóa input và output
  - Các phần đọc input và sinh ra output là những phần có khả năng phải thay đổi nhiều nhất khi phần cứng và phần mềm được sửa đổi.
  - Chúng được tách riêng khỏi phần còn lại của mã lệnh nhằm giúp ta dễ hiểu và dễ thay đổi hệ thống hơn.
- Tái sử dụng
  - Tái sử dụng của nhà sản xuất: tạo ra các thành phần mà chúng được tái sử dụng trong các ứng dụng nối tiếp.
  - Tái sử dụng của người tiêu thụ: tái sử dụng các thành phần mà ban đầu chúng được phát triển cho các dự án khác.

Nguyên tắc chung



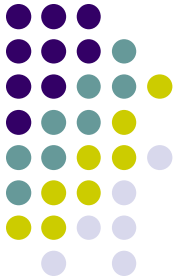
- Một số lưu ý khi lập trình
  - Bắt ngoại lệ.
  - Quên cấp phát và thu hồi bộ nhớ.
  - Vấn đề làm tròn số.
  - Hạn chế gọi nhiều lần các hàm có qui mô nhỏ.
  - Truyền tham số.
  - Tránh dùng mảng nhiều chiều.
  - Sử dụng các phép toán nhanh.

Tài liệu chương trình



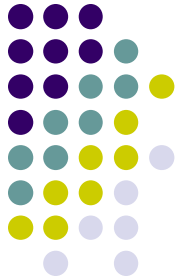
- Tài liệu chương trình: một tập các mô tả giải thích với người đọc về các chương trình làm gì và chúng làm như thế nào.
- Các loại tài liệu chương trình
  - Tài liệu nội là mô tả được viết trực tiếp trong mã lệnh và được sử dụng bởi những người đọc mã lệnh.
  - Tài liệu ngoại gồm những tài liệu khác và được sử dụng bởi những người không trực tiếp đọc mã lệnh.

Tài liệu chương trình

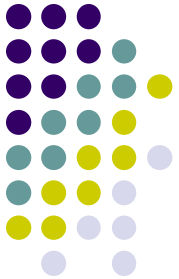


- Tài liệu nội
  - Khối chú thích ở phần đầu của từng thành phần: ai, cái gì, tại sao, khi nào, như thế nào và ở đâu.
  - Các chú thích khác giúp ta hiểu chương trình rõ hơn: thêm thông tin mới, không phải diễn đạt lại những gì đã rõ ràng.
  - Nhãn cho câu lệnh và tên biến phải có nghĩa.
  - Định dạng để gia tăng sự hiểu biết: thụt lề, khoảng trắng.

Tài liệu chương trình



- Tài liệu ngoại
  - Mô tả vấn đề: mô tả các lựa chọn được xem xét để giải quyết vấn đề và tại sao một giải pháp cụ thể được chọn.
  - Mô tả giải thuật: giải thích từng giải thuật được sử dụng trong thành phần (công thức, ranh giới, các điều kiện đặc biệt, bắt lỗi, v.v).
  - Mô tả dữ liệu: từ điển dữ liệu.



HẾT PHẦN II.3