TIN ĐẠI CƯƠNG

BÀI 1: MỞ ĐẦU

Nội dung chính

1. Giới thiệu môn học
2. Viết chương trình cho máy tính
3. Làm quen với ngôn ngữ C/C++
4. Các cấu trúc điều khiển
5. Thuật toán
6. Một số khái niệm liên quan
7. Câu hỏi và bài tập

Phần 1

Giới thiệu môn học

Giáo trình & Giờ học

- Thời lượng: 3 tín chỉ (2 lý thuyết, 1 thực hành)
- Giáo trình chính
  - "*Introduction to Engineering Programming: Solving Problems with Algorithms*" (James Paul Holloway)
  - Đã có bản dịch tiếng Việt
- Công cụ trên máy tính: Dev-C++ 5.11
  - Hoặc những công cụ tương đương
- Giờ lý thuyết: lý thuyết + chữa bài tập
- Giờ thực hành: viết chương trình trên máy

Nội dung giảng dạy

- Khái niệm cơ bản
- Các lệnh cơ bản
- Câu lệnh lặp
- Câu lệnh lựa chọn
- Chuỗi (string)
- Mảng (vector)
- Bài tập tổng hợp

Mục tiêu của môn học

- Hiểu biết cơ bản về ngôn ngữ lập trình C/C++
- Biết cách triển khai (lập trình) một số thuật toán trên máy tính
- Biết cách viết, dịch, sửa lỗi và chạy một chương trình viết bằng C++
- Biết giải được một số bài toán đơn giản bằng lập trình C++
- Biết ứng dụng kiến thức lập trình vào những công việc sau này

Tại sao phải học môn này?

- Hiểu biết hơn về máy tính
- Làm quen với máy tính theo cách của giới làm kỹ thuật
- Hiểu cách thức giải quyết một vấn đề bằng máy tính
- Nâng cao tư duy logic và tư duy thuật toán
- Lấy kiến thức
- Lấy bằng đại học

Thi & Tính điểm

- Tính điểm:
  - Điểm bài tập (20%)
  - Điểm kiểm tra giữa kì (20%)
  - Điểm kiểm tra cuối kì (60%, thi viết)
- Chú trọng vào viết chương trình, không thi lý thuyết, được tham khảo tài liệu khi thi
- Giảng viên:
  - Họ tên: Trương Xuân Nam, khoa CNTT
  - Email: truongxuannam@gmail.com

Một vài chú ý khác

- Cần xem trước giáo trình và xem lại bài cũ trước khi lên lớp
- Phải làm hết bài tập (được giao trên lớp và trong giờ thực hành)
- Yêu cầu hỗ trợ của giáo viên khi cần thiết
- Mọi thông tin cần thiết về môn học được đưa lên http://txnam.net mục BÀI GIẢNG
- Cách học hợp lý môn này: không nên ghi chép nhiều trong giờ lý thuyết

Phần 2

Viết chương trình cho máy tính

Máy tính chỉ hiểu con số

- Mọi thông tin đều có thể chuyển về dạng số:
  - Các số → giữ nguyên → số
  - Âm thanh → số hóa (tần số) → số
  - Hình ảnh → số hóa (ma trận điểm) → số
  - ...

➔ Máy tính xử lý các thông tin ở dạng số

➔ Mọi thông tin trong máy tính đều được lưu ở dạng số, cụ thể là số ở dạng nhị phân

➔ Ra lệnh cho máy tính phải viết lệnh ở dạng số

Các lệnh máy là các dãy số

- Máy tính chỉ hiểu một số lệnh cơ bản:
  - Thao tác bộ nhớ: đọc/ghi số
  - Tính toán: cộng 2 số, trừ 2 số,...
  - So sánh: so sánh 2 số với nhau
  - ...
- Chương trình máy tính = dãy các lệnh máy để chỉ thị từng bước làm việc nhỏ
- Kích thước một chương trình máy tính
  - Loại nhỏ ~ vài chục nghìn lệnh máy
  - Loại vừa ~ vài trăm nghìn lệnh máy
  - Loại lớn ~ vài triệu lệnh máy

Thực hiện một chương trình

- **Bước 1**: người dùng ra lệnh cho máy tính thực hiện một chương trình
- **Bước 2**: máy tính đọc file chương trình trên đĩa và nạp chương trình vào bộ nhớ
- **Bước 3**: hệ thống có một số thao tác chuẩn bị để chương trình sẵn sàng chạy
- **Bước 4**: máy tính đọc từng lệnh trong bộ nhớ và thực hiện từng lệnh một

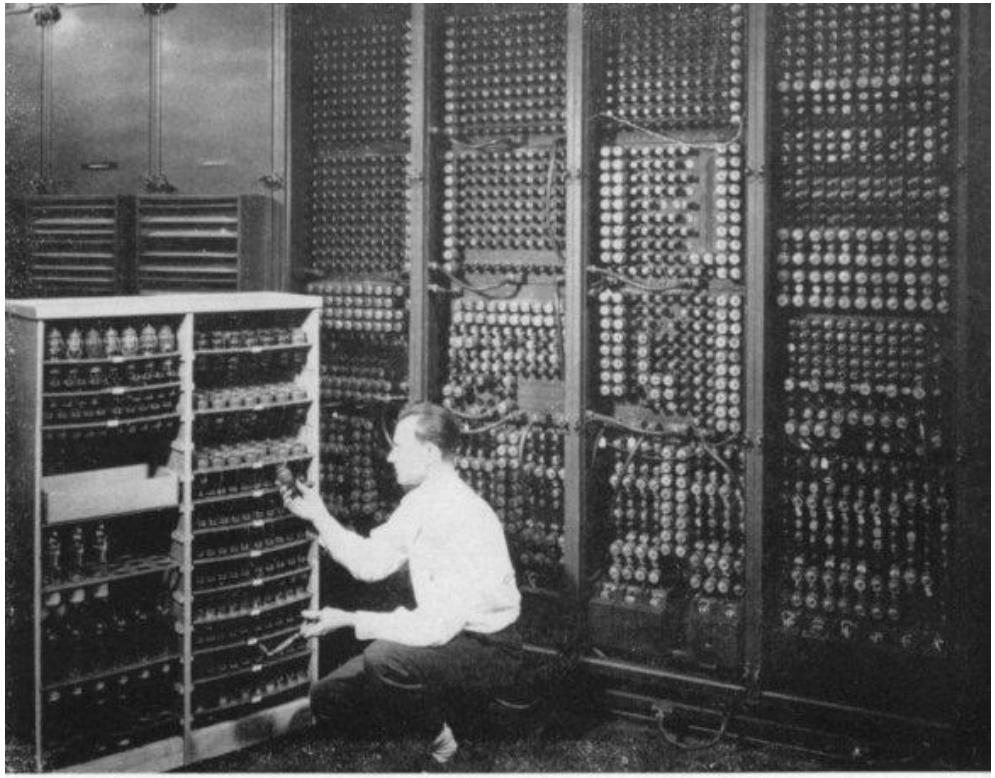
Máy tính thực hiện từng lệnh một

- Chương trình máy tính được ghi trên đĩa ở dạng file chương trình (.COM, .EXE, .DLL,...)
- Máy tính đọc lệnh máy trong bộ nhớ và thực hiện từng lệnh một

00011000 00010000	Nạp số 16 vào ô nhớ số 8
00011001 00001111	Nạp số 15 vào ô nhớ số 9
00101010 10001001	Cộng hai số ở ô nhớ số 8 và ô nhớ số 9 sau đó ghi kết quả vào ô nhớ số 10

Viết chương trình ~ viết dãy số?

- Thời kì đầu: viết trực tiếp lệnh máy (dãy số)
  - Bất lợi: khó hiểu, dễ nhầm lẫn, viết lâu,...



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

Viết chương trình ~ viết dãy số?

- Hợp ngữ: sử dụng các kí hiệu đơn giản bằng tiếng Anh, gần gũi với lệnh máy
  - Bất lợi: người lập trình phải biết rõ về từng lệnh máy, viết dài, dễ nhầm lẫn

Viết chương trình ~ viết dãy số?

- Ngôn ngữ lập trình bậc cao: các lệnh ở dạng gần gũi với ngôn ngữ tự nhiên, trình biên dịch chuyển một lệnh này thành các lệnh máy
  - Ngôn ngữ bậc cao đơn giản: BASIC, FORTRAN,...
  - Ngôn ngữ lập trình thủ tục: ALGOL, PASCAL, C,...
  - Ngôn ngữ lập trình hướng đối tượng: SmallTalk, C++, Object Pascal, Java, C#,...
- Các ngôn ngữ lập trình đặc biệt (dùng cho những mục đích riêng): Prolog, SQL,...

Phần 3

Làm quen với ngôn ngữ C/C++

Các bước xây dựng chương trình

- Một chương trình máy tính được xây dựng để giải quyết một bài toán cụ thể nào đó
- Việc xây dựng một chương trình máy tính luôn tuân theo các bước sau:
  - Bước 1: xác định (mô tả) bài toán cần giải quyết
  - Bước 2: xây dựng lời giải (thuật toán)
  - Bước 3: chuyển lời giải bài toán thành chương trình viết bằng một ngôn ngữ lập trình nào đó
  - Bước 4: dịch chương trình thành dạng mã máy để máy tính có thể thực hiện được

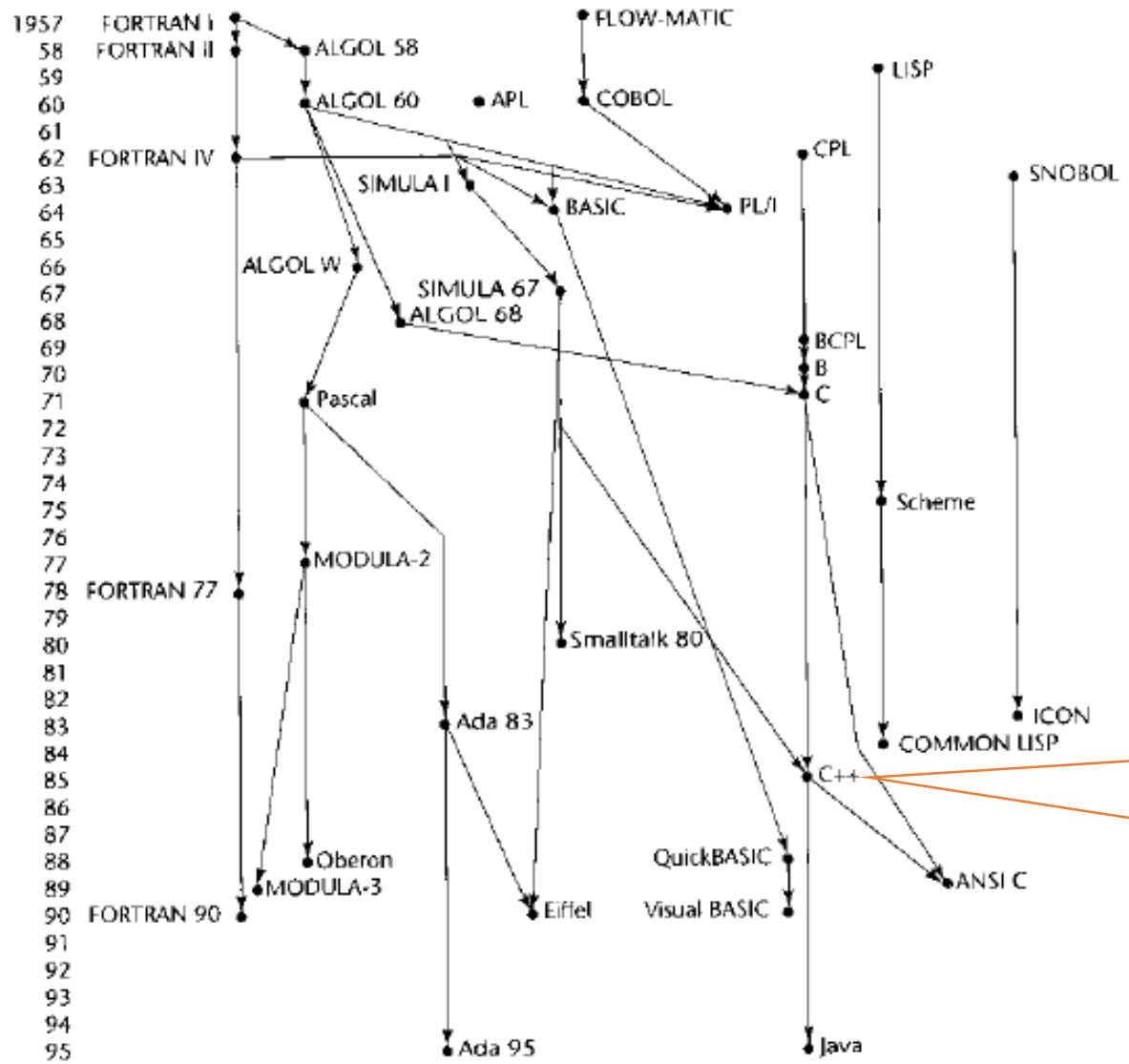
Các bước xây dựng chương trình

- Bước 1 - xác định (mô tả) bài toán cần giải quyết:
  - Ví dụ: bài toán tính A^2
  - Xác định bài toán: người dùng cho số A, máy tính cần tính A^2 dựa trên số A đã biết
- Bước 2 - xây dựng lời giải (thuật toán):
  - Có nhiều cách mô tả thuật toán (bằng lời hoặc bằng sơ đồ khối)
  - Ví dụ (mô tả bằng lời): nhập A từ bàn phím, sau đó tính giá trị A x A và in kết quả ra màn hình

Các bước xây dựng chương trình

- Bước 3 - chuyển lời giải bài toán thành chương trình viết bằng một ngôn ngữ lập trình nào đó:
  - Chọn ngôn ngữ lập trình thích hợp với bài toán
  - Viết chương trình theo thuật toán đã định
- Bước 4 - dịch chương trình thành dạng mã máy để máy tính có thể thực hiện được:
  - Sử dụng trình biên dịch của ngôn ngữ đã chọn và dịch chương trình sang dạng mã máy
  - Nếu xảy ra lỗi, tìm và sửa lỗi trong chương trình sau đó dịch lại đến khi không còn lỗi nữa

Ngôn ngữ lập trình C/C++



Ngôn ngữ lập trình C/C++

Giới thiệu ngôn ngữ C/C++

- Công cụ Dev-C++
- Hướng dẫn cơ bản
  - Bắt đầu vào chương trình
  - Viết mã
  - Dịch
  - Chạy
  - Sửa lỗi
- Một vài ví dụ đơn giản

Phần 4

Các cấu trúc điều khiển

Các cấu trúc điều khiển

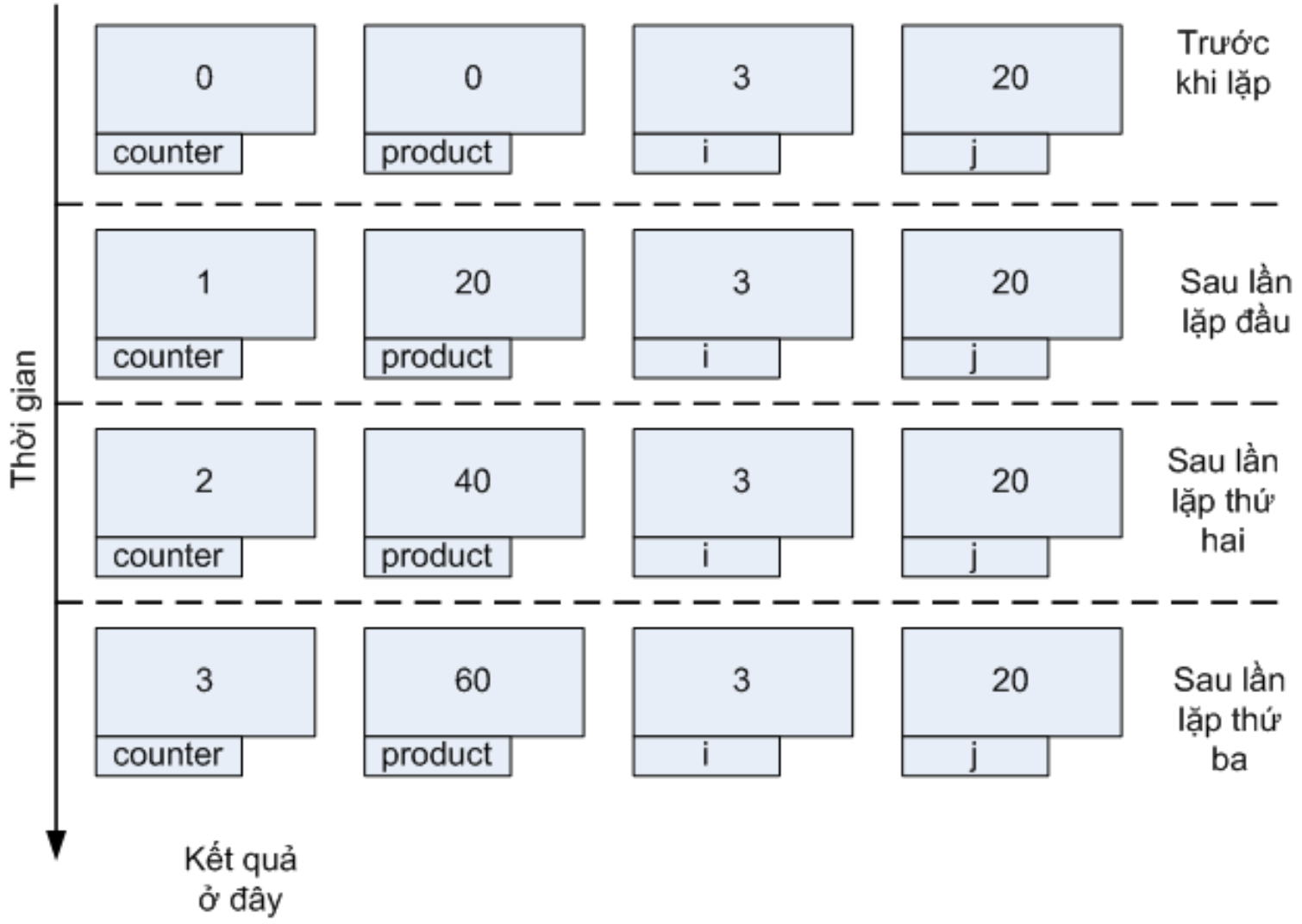
- Có 3 cấu trúc điều khiển cơ bản
  - Tuần tự: Thực hiện tuyến tính từng việc một
  - Lặp: Thực hiện lặp lại một hoặc nhiều việc cho đến khi điều kiện nhất định được thỏa mãn
  - Lựa chọn (rẽ nhánh): Chọn thực hiện một hoặc nhiều việc dựa trên một điều kiện nhất định
- Các cấu trúc này tương tự như nhiều hành vi trong cuộc sống
- Đọc trước các chương 2, 3 và 4 của giáo trình để tìm hiểu về các điều khiển này

Ví dụ minh họa

(*phần 1.1.2 của giáo trình*) Nhân 2 số tự nhiên i và j trên máy tính không có phép nhân

1. Đặt biến **product = 0**
2. Đặt biến **counter = 0**
3. Lặp lại bước 4 và 5 chừng nào **counter < i**:
4. Đặt product bằng chính nó cộng với j
5. Tăng biến counter lên 1
6. Trả về kết quả là giá trị của product

Ví dụ minh họa



Ví dụ mở rộng

Nhân 2 số nguyên i và j trên máy tính không có phép nhân (bỏ điều kiện i và j không âm)

if cả i và j đều không âm **then**

- sử dụng thuật toán nhân không âm i và j và lưu kết quả trong product

else if cả i và j đều âm **then**

- đổi dấu cả i và j, lúc này chúng đều không âm
- sử dụng thuật toán nhân không âm i và j và lưu kết quả trong product

else

Ví dụ mở rộng

- **if i<0 then**
  - đổi dấu của i
- **else**
  - đổi dấu của j
- **end if**
- sử dụng thuật toán nhân không âm i và j và lưu kết quả trong product
- đảo dấu của product

end if

return product

Phần 5

Thuật toán

Định nghĩa và đặc trưng

- Định nghĩa: các bước cần tiến hành để giải quyết một công việc cụ thể nào đó
- Thuật toán phổ biến trong cuộc sống, có trước máy tính và có nhiều dạng khác nhau
- Đặc trưng (*xem phần 1.1 của giáo trình*):
  - Tính hữu hạn
  - Tính máy móc
  - Tính dừng
  - Mở rộng: Tính đúng
  - Mở rộng: Tính tổng quát

Ví dụ đơn giản

Ví dụ: Tính bình phương của số m

Bước 1: Nhập giá trị cho m

Bước 2: Tính giá trị m × m và đưa vào s

Bước 3: Trả về giá trị s cho chương trình gọi

Ta thấy:

- Thuật toán có 3 bước
- Viết rõ ràng, không thể hiểu sai
- Làm theo thuật toán ta có thể tính được kết quả mà không cần hiểu khái niệm "bình phương" là gì

Ví dụ phức tạp hơn

Giải phương trình $ax^2 + bx + c = 0$ (với $a \neq 0$)

- Bước 1: Nhập các giá trị a, b, c
- Bước 2: Nếu a = 0 thì thông báo lỗi và dừng
- Bước 3: Tính $d = b^2 - 4 \times a \times c$
- Bước 4: Nếu d < 0 thì thông báo vô nghiệm và dừng
- Bước 5: Nếu d > 0 thì thực hiện Bước 7
- Bước 6: Thông báo có nghiệm x = -b/2/a và dừng
- Bước 7: Thông báo có hai nghiệm
  - $x_1 = (-b + \sqrt{d})/2/a$
  - $x_2 = (-b - \sqrt{d})/2/a$

Ví dụ vui

Cho con sư tử vào tủ lạnh

- Bước 1: Mở cửa tủ lạnh
- Bước 2: Cho con sư tử vào
- Bước 3: Đóng cửa tủ lạnh

Câu hỏi: *hãy nêu một vài thuật toán tương tự trong cuộc sống (thuật toán nhưng không dính dáng gì tới máy tính)*

Phần 6

Một số khái niệm liên quan

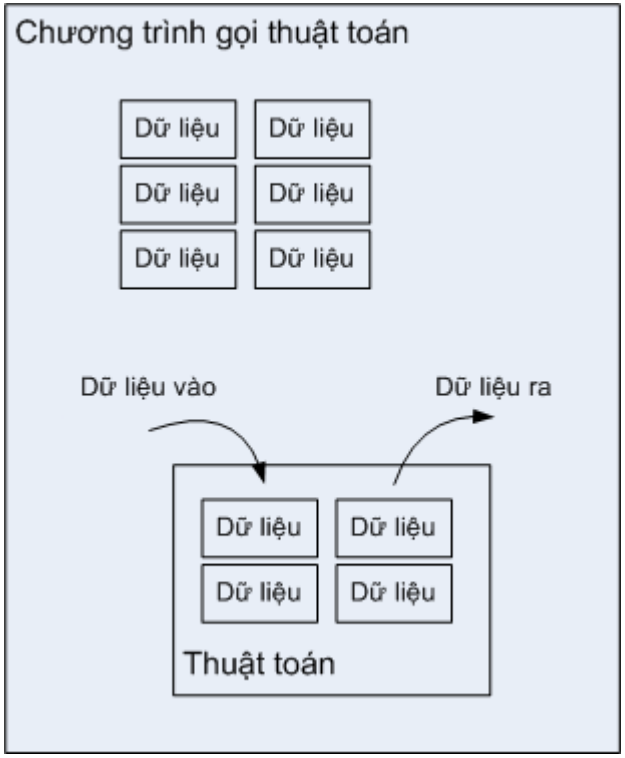
Một số khái niệm liên quan

- Khái niệm "môi trường hoạt động":
  - Cung cấp dữ liệu (số liệu) để thuật toán hoạt động (dữ liệu vào)
  - Là nơi nhận kết quả của thuật toán (dữ liệu ra)
- Mở rộng: cung cấp ngữ cảnh để thuật toán hoạt động tốt

(*những khái niệm này được đề cập trong phần 1.1.1 của giáo trình*)

Một số khái niệm liên quan

- Như vậy thêm một đặc trưng mới cho thuật toán: giao diện (interface)
  - Dữ liệu đầu vào để thực hiện thuật toán
  - Dữ liệu đầu ra để thuật toán trả kết quả về
- Khái niệm liên quan:
  - Truyền tham trị (pass-by-value)
  - Truyền tham chiếu (pass-by-reference)



Phần 7

Câu hỏi và bài tập

Bài tập

1. *Hãy xây dựng thuật toán để giải phương trình bậc nhất P(x): a x + b = c*
2. *Hãy xây dựng thuật toán để tính tổng các chữ số của một số nguyên **N** bất kỳ.*

   Ví dụ: N = 2015 thì thuật toán trả về 8 (2+0+1+5)
3. *Làm thế nào để tạo các số ngẫu nhiên trong máy tính? Hãy xây dựng một thuật toán để có thể tạo ra được các số như vậy.*
4. *Hãy xây dựng thuật toán để tính căn bậc 2 của số thực **N** không âm.*