Phần lý thuyết:
Các kiến thức cơ bản về công nghệ thông tin

(Tuần 4)

Chương 4. Tổng quan về thuật toán và lập trình	
	Một số khái niệm cơ bản
	Thuật toán
	Ngôn ngữ lập trình

Chương 4. Tổng quan về thuật toán và lập trình

A.Một số khái niệm cơ bản

1) Khái niệm về lập trình

Lập trình là việc sử dụng cấu trúc dữ liệu, các lệnh của ngôn ngữ lập trình cụ thể để mô tả dữ liệu và diễn đạt các thao tác của thuật toán.

2) Ngôn ngữ lập trình

Là ngôn ngữ dùng để diễn tả thuật toán sao cho máy tính hiểu và thực hiện được. Có 3 loại NNLT:

- Ngôn ngữ máy :
Các lệnh được mã hóa bằng các bít 0 – 1. Chương trình được viết trên ngôn ngữ máy có thể được nạp vào bộ nhớ và thực hiện được ngay. Với các ngôn ngữ khác máy không hiểu nên không thực hiện được ngay, nên phải có trình dịch (**Compiler)** tương ứng ra mã máy.

- Hợp ngữ (hay **assembly**) thường được viết tắt là **asm** là **ngôn ngữ lập trình bậc thấp** người đọc hiểu được tương ứng với tập lệnh trong ngôn ngữ mã máy. Hợp ngữ là mã máy tượng trưng (*symbolic machine code)*. Trình dịch tương ứng là **assembler.**

- Ngôn ngữ bậc cao :
Các lệnh được mã hóa bằng một ngôn ngữ gần với ngôn ngữ Tiếng Anh. Chương trình viết trên ngôn ngữ bậc cao này cũng phải được chuyển

đổi thành mã máy nhờ **chương trình dịch** tương ứng để chuyển đổi.
Lập trình bằng ngôn ngữ bậc cao dễ viết vì gần với ngôn ngữ tự nhiên.

B. Thuật toán

Thuật toán là một khái niệm cơ sở của Toán học và Tin học. Hiểu một cách đơn giản, thuật toán là một dãy các hướng dẫn nhằm thực hiện một công việc nào đó.
Như vậy, thuật toán là một phương pháp thể hiện lời giải của bài toán.

Các đặc trưng khác của thuật toán

- ✓ Tính "thực thi được"
- ✓ Tính "dừng"
- ✓ Có đầu vào và đầu ra (*input/output*) :
- ✓ Tính "đúng", hiệu quả (*effectiveness*) :
- ✓ Tính phổ dụng (*generalliness*) :

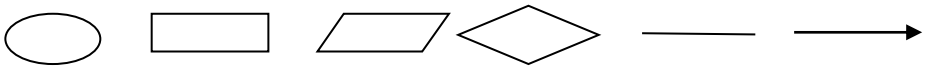
Phương pháp biểu diễn thuật toán

✓ ***Ngôn ngữ tự nhiên***

Trong cách biểu diễn thuật toán theo ngôn ngữ tự nhiên, người ta sử dụng ngôn ngữ thường ngày để liệt kê **các bước** của thuật toán

✓ ***Lưu đồ - sơ đồ khối***

Lưu đồ hay sơ đồ khối là một công cụ trực quan để diễn đạt các thuật toán.



Bắt đâu/Kết thúc, Công việc, Đầu vào, Điều kiện, đường đi.

✓ ***Mã giả (Giả lập trình)***

Ví dụ 1. Máy giặt, Luộc trứng lòng dào. Hoạt độ của con chó.

Cấu trúc thuật toán

1. Tuần tự
2. Nhảy tự do đến bước xác định

3. Rẽ nhánh
   (1) Đầy đủ
   (2) Không đầy đủ
4. Lặp
   (1) Có số lần xác định
   (2) Điều kiện trước while - do
   (3) Điều kiện sau do – while hoặc repeat - until
5. Nhóm gộp vài chỉ thị { … }
6. Dùng tên chương trình con
   (1) Hàm f(…)
   (2) Thủ tục …

C. Ngôn ngữ lập trình

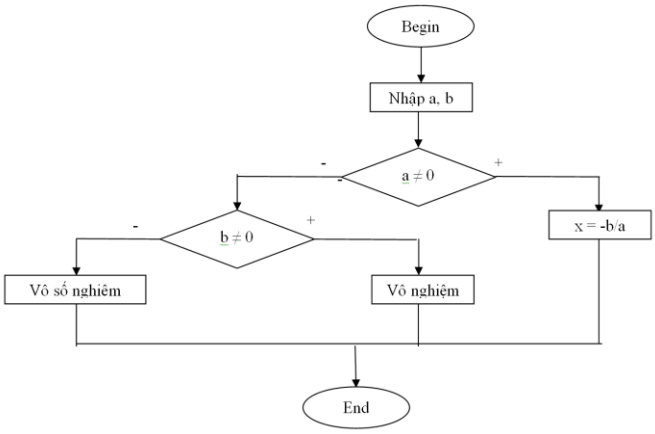
Để hiểu nhanh về lập trình ta xét một ví dụ đơn giản

Ví dụ 1. Giải phương trình bậc nhất một ẩn $ax + b = 0$.
Ta hiểu đầu vào là hai số thực a, b nhờ máy giải quyết đưa ra đáp số.
Thuật toán rất đơn giản:
Nhập *a* và *b* vào bộ nhớ Ram
Xét các trường hợp của *a* và *b* để đi đến quyết định.
Và hết!

Nếu đúng đắn ta **Biên dịch** ra mã máy, có thể thực hiện ở các nơi.

Chú ý:
Lập trình web ta nhờ phương pháp **Thông dịch** không cần Biên dịch!

Có thể vẽ sơ đồ khối bài toán ở Ví dụ 1 như sau:



Cách 1, Mã nguồn ở ngôn ngữ Pascal

```
Uses crt;
Var a,b,x : real;
Begin
   Clrscr;
   Writeln('Nhap a = ') Readln(a);
   Writeln('Nhap b = ') Readln(b);
   write(' Dap so: ');
   If a<>0 then
     Begin
        x:=-b/a;
        writeln(' x=',x)
     end
   else
     if b<>0 then writeln('Vo nghiem')
     else writeln('Vo so nghiem');
   readln
end.
```

Tiến hành dịch, sửa lỗi để thu được mã máy, đặt tên và cho chạy…

Cách 2, Mã nguồn ở ngôn ngữ C/C++

```
#include <conio.h>
#include <stdio.h>
main()
{
    float a,b,x;
```

```
        printf("Nhap a, b : ");
        scanf("%f%f",&a,&b);
        printf("Dap so: ");
        if (a!=0)
            {
              x=-b/a;
              printf("%f",x);
            }
        else
            {
                    if (b!=0) printf("Vo Nghiem");
                    else printf("Vo So Nghiem");
            }
    }
```

Biên dịch, sửa lỗi để thu được mã máy, đặt tên và cho chạy thử …

Cách 3, Mã nguồn ở ngôn ngữ Python

```
def giai_phuong_trinh_bac_nhat(a, b):
    if a == 0:
        if b == 0:
            return "Phương trình vô số nghiệm"
        else:
            return "Phương trình vô nghiệm"
    else:
        x = -b / a
        return f"Nghiệm của phương trình là: x = {x}"

# Thử nghiệm với một số giá trị
a = 2
b = 4
print(giai_phuong_trinh_bac_nhat(a, b))
```

Như vậy để lập trình phải trải qua nhiều công đoạn:

- *Bước 1*. Xác định vấn đề - bài toán. Thiết kế thuật toán...
- *Bước 2*. Soạn thảo văn bản mã nguồn trong đó có bộ tiền sử lý và các câu lệnh người lập trình viết ra!
- *Bước 3*. Chạy trình biên dịch (Compiler)
  Nếu có lỗi Cú pháp phải quay về *Bước 2* sửa ngay! Cho đến khi hết lỗi.

- *Bước 4.* Chạy chương trình đích,
  Nếu có lỗi Ngữ nghĩa phải về *Bước 2* sửa ngay! Cho đến khi hết.
  Nếu có lỗi Ngữ nghĩa hoặc thuật toán phải về *Bước 1* và *Bước* 2 sửa ngay!
  Trong quá trình sử dụng ta cũng cần hiệu chỉnh và sửa lại, thậm chí đưa lên phiên bản (***Version***) mới!

Bài tập

1/- Lập trình giải phương trình bậc 2
2/- Tìm max và min của dãy số nhập từ bàn phím.
3/- Tính giá trị của một đa thức bậc n và các hệ số nhập từ ban phim!