CHƯƠNG I : MỘT SỐ KHÁI NIỆM VỀ LẬP TRÌNH

BÀI 1 : KHÁI NIỆM LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

* Lập trình là sử dụng cấu trúc dữ liệu và các câu lệnh của ngôn ngữ lập trình (NNLT) cụ thể để mô tả dữ liệu và diễn đạt các thao tác của thuật toán.
* Chương trình viết bằng NNLT bậc cao nói chung không phụ thuôc vào loại máy, nghĩa là một chương trình có thể thực hiện trên nhiều loại máy tính khác nhau
* Chương trình viết bằng ngôn ngữ máy có thể nạp trực tiếp vào bộ nhớ và thực hiện ngay..
* Chương trình viết bằng NNLT bậc cao phải chuyển đổi thành chương trình trên ngôn ngữ máy mới có thể thực hiện được.

Chương trình có chức năng chuyển đổi chương trình được viết bằng NNLT bậc cao thành chương trình thực hiện được trên máy tính gọi là ***chương trình dịch.***

Chương trình dịch có 2 loại là **thông dịch** và **biên dịch**.

1. **Thông dịch**

Là dịch lần lượt từng câu lệnh ra ngôn ngữ máy rồi thực hiện ngay câu lệnh vừa dịch được hoặc thông báo lỗi nếu không dịch được câu lệnh.

1. **Biên dịch**

Là duyệt, kiểm tra, phát hiện lỗi, khi chương trình nguồn không còn lỗi thì dịch toàn bộ chương trình nguồn thành chương trình đích có thể thực hiện trên máy.

BÀI 2 : CÁC THÀNH PHẦN CỦA NGÔN NGỮ LẬP TRÌNH

1. **Các thành phần cơ bản**

Có 3 thành phần cơ bản:

1. ***Bảng chữ cái****:* là tập các kí tự được dùng để viết chương trình.

Trong Pascal, bảng chữ cái gồm các kí tự:

* Chữ cái thường và chữ cái in hoa trong bảng chữ cái tiếng Anh
* 10 chữ số thập phân Ả Rập: 0 1 2 3 4 5 6 7 8 9
* Các kí tự đặc biệt:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **+** | **-** | **\*** | **/** |  |  | **<** | **>** | **[** | **]** | **■** | *f* | *—* |
| ■ | **#** | **A** | **$** |  | **&** | **(** | **)** |  |  | **:** | *r* |  |

1. ***Cú pháp*** là bộ qui tắc để viết chương trình, cho biết cách viết một chương trình hợp lệ.

* Lỗi cú pháp được chương trình dịch phát hiện và thông báo cho người lập trình biết.
* Chỉ có các chương trình không còn lỗi cú pháp mới có thể được dịch sang ngôn ngữ máy.

1. ***Ngữ nghĩa****:* xác định ý nghĩa thao tác cần phải thực hiện, ứng với tổ hợp kí tự dựa vào ngữ cảnh của nó.

Ví dụ :

A + B (A, B là hai số thực) ý nghĩa thao tác là phép cộng, ngữ cảnh là hai số thực do đó ngữ nghĩa câu lệnh là cộng hai số thực

I + J (I, J là hai số nguyên)ý nghĩa thao tác là phép cộng, ngữ cảnh là hai số nguyên do đó ngữ nghĩa câu lệnh là cộng hai số nguyên

Vậy, hai biểu thức trên đều thực hiện phép cộng nhưng về mặt ngữ nghĩa là khác nhau.

1. **Tên**

Mọi đối tượng trong chương trình đều phải được đặt tên theo qui tắc của NNLT và từng chương trình dịch cụ thể.

Tên trong Pascal là một dãy liên tiếp *không quá 127 kí tự gồm chữ số, chữ cái hoặc dấu gạch dưới và bắt đầu bằng chữ cái hoặc dấu gạch dưới.*

1. ***Tên dành riêng-,*** còn gọi là **từ khoá**, các tên này được dùng với *ý nghĩa riêng xác định,* người lập trình không được dùng với ý nghĩa khác.
2. ***Tên chuẩn****:* là tên được dùng với *ý nghĩa nhất định nào đó,* được qui định trong các thư viện của NNLT, người lập trình có thể khai báo và dùng với ý nghĩa khác.

***c) Tên do người lập trình đặt.*** được dùng với ý nghĩa riêng, xác định bằng cách khai báo trước khi sử dụng và không được trùng với tên dành riêng.

1. **Hằng và biến**

***a) Hằng*** : *là đại lượng có giá trị không đôi trong quá trình thực hiện chương trình.*

- *Hằng số học* : là các số nguyên và số thực (dấu phẩy tĩnh hoặc dấu phẩy động).

- *Hằng logic* : là giá trị *đúng* hoặc *sa*/tương ứng với *true* hoặc *false.*

- *Hằng xâu* : là chuỗi kí tự trong bộ mã ASCII, được đặt trong cặp dấu nháy.

***b) Biến*** : *là đại lượng được đặt tên dùng để lưu trữ giá trị và giá trị có thể được thay đổi trong quá trình thực hiện chương trình.*

1. **Chú thích**

* Trong chương trình có thể đặt chú thích giúp cho người đọc nhận biết ý nghĩa của chương trình đó dễ hơn mà không ảnh hưởng đến nội dung của chương trình nguồn và được chương trình dịch bỏ qua.
* Trong Pascal chúng được đặt trong cặp dấu { và } hoặc (\* và \*).

xxxxx