Chương 4 LẬP CHƯƠNG TRÌNH BẰNG PASCAL

BÀI 1 TỔNG QUAN VỀ NGÔN NGỮ PASCAL

1.1. Giới thiệu

Pascal là ngôn ngữ lập trình bậc cao, vạn năng do giáo sư Niklaus Wirth ở trường đại học Zurich (Thuỵ sỹ) thiết kế và công bố vào năm 1971. Ông đặt tên cho ngôn ngữ này Pascal để tưởng nhớ đến Blaise Pascal nhà toán học, triết học người Pháp ở thế kỷ 17.

Ngôn ngữ Pascal được thiết kế nhằm phục vụ công tác giảng dạy cách tiếp cận hệ thống trong công việc lập trình và đặc biệt lần đầu tiên đưa ra khái niệm "Phương pháp lập trình cấu trúc". Ngày nay qua nhiều lần cải tiến, bổ sung Pascal đã có nhiều công cụ để giải quyết gần như bất cứ nhiệm vụ nào cho các loại máy tính.

Pascal là ngôn ngữ dùng để dạy học là chính. Nhiều hãng phần mềm dựa trên chuẩn mực của ngôn ngữ để thêm bớt và thương mại hoá với nhiều loại khác nhau:

ISO Pascal: Pascal chuẩn (International Standard organization)

ANSI Pascal: Pascal của viện tiêu chuẩn Hoa kỳ

Turbo Pascal: Pascal của hãng Borland

IBM Pascal: Pascal của Microsoft...

Từ khi ra đời đến nay Pascal đã trải qua nhiều Version khác nhau, mỗi Version được cải tiến và bổ sung thêm những chức năng mới, tốc độ dịch cũng được cải thiện. Ngày nay Version mới nhất của Pascal là 7.0 (bao gồm cả bản chạy trên DOS và bản chạy trên Windows)

Turbo Pascal gồm các file chính sau:

Turbo.exe: Là file chương trình soạn thảo, dịch và liên kết với bảng chọn.

Turbo.tpl: Là file thư viện lưu các đơn vị chương trình chuẩn để chạy với Turbo.exe.

Graph.tpu: Là file chương trình xử lý đồ hoạ.

*.chr: Là các file chứa các font chữ trong chế độ đồ hoạ.

*.bgi: Là file chứa các font chữ màn hình.

Ngoài ra còn các file khác với các chức năng riêng biệt.

1.2. Các khái niệm và các thành phần cơ bản

1.2.1 Bộ ký tự (dùng để soạn thảo chương trình)

Bao gồm các loại ký tự sau đây:

- Các chữ cái: a..z; A..Z (Tuy nhiên khi soạn thảo chương trình, Turbo Pascal không phân biệt chữ hoa hay chữ thường).
- Các ký tự số 0..9
- Các dấu toán: + - * / ^ = < >
- Các ký tự đặc biệt: ? ; . : ! [ ] { } # $ @.
- Dấu gạch nối: _
- Các ký tự điều khiển

1.2.2. Từ và từ khoá

a. Từ: Là một dãy liên tiếp các ký tự không chứa ký tự trắng và ký tự điều khiển.

b. Từ khoá: Là từ dành riêng của Pascal với chức năng và cú pháp được quy định sẵn. Vì vậy khi sử dụng phải theo đúng quy định, và không được sử dụng các từ khoá vào các công việc khác.

Ví dụ: Begin, End, If, Then, Const, Var, Function,…

1.2.3 Tên và tên chuẩn

a. Tên: Tên là một từ bao gồm tối đa 255 ký tự, chỉ được lấy trong các chữ cái, chữ số, và dấu gạch nối, nhưng không được bắt đầu bằng số. Tên dùng để đặt cho các đối tượng trong chương trình như hằng, biến, hàm, thủ tục, kiểu dữ liệu,…

b. Tên chuẩn: Là tên mà Turbo Pascal đã định nghĩa sẵn để chỉ các hàm, hằng, biến, thủ tục, thư viện của nó.

Chú ý: Turbo Pascal cho phép người sử dụng có thể định nghĩa lại các tên chuẩn để dùng vào các công việc khác.

1.3 Cấu trúc tổng quát của một chương trình.

Một chương trình của Turbo Pascal gồm 3 phần

- Tiêu đề
- Khai báo và định nghĩa
- Thân chương trình

Cụ thể:

Program	*Tiêu đề*
Uses *Const* *Type* *Var* *Procedure* *Funtion*	*Các khai báo và định nghĩa*
Begin *End*	*Thân chương trình*

1.3.1 Phần tiêu đề:

Từ khoá để khai báo là Program tiếp đến là tên của chương trình do người dùng tự đặt. Phần này không bắt buộc phải có.

(Chú ý: tên của chương trình phai theo đúng quy cách tên của Turbo Pascal)

1.3.2 Phần khai báo và định nghĩa

- *Uses*: Dùng để khai báo các Unit (đơn vị chương trình) của Turbo Pascal. Nếu có nhiều unit thì sử dụng dấu phẩy "," để ngăn cách.

- *Const*: Dùng để khai báo các hằng

Cú pháp: Tên_hằng: = Giá trị.

- *Type*: Dùng để định nghĩa các kiểu dữ liệu của người dùng.

Cú pháp: Tên_kiểu= định nghĩa cụ thể cho từng kiểu.

- *Var*: Khai báo biến.

Cú pháp: Tên biến: Kiểu_dữ liệu;

(Hoặc khai báo trực tiếp không thông qua kiểu).

Nếu có nhiều biến cùng kiểu thì sử dụng dấu phẩy "," để ngăn cách.

- *Procedure*: Định nghĩa chương trình con dạng thủ tục.

- *Function*: Định nghĩa chương trình con dạng hàm.

Chú ý: Tuỳ thuộc vào từng chương trình cụ thể mà trong chương trình có thể có các phần khai báo và định nghĩa phù hợp, có những chương trình có phần khai báo, định nghĩa này mà không có khai báo, định nghĩa kia hoặc ngược lại, thậm chí có những chương trình không cần đến một khai báo hay định nghĩa nào cả.

1.3.3. Phần thân chương trình: Được bắt đầu bằng từ khoá "Begin" và kết thúc bởi từ khoá "End". Giữa cặp từ khoá này là các câu lệnh của chương trình. Nếu có nhiều câu lệnh thì phải sử dụng dấu ";" để ngăn cách các câu lệnh. Phần này bắt buộc phải có.

Chú ý:

- Turbo Pascal cũng sử dụng dấu "**;**" để kết thúc phần này chuyển qua phần khác cũng như giữa khai báo này qua khai báo khác của chương trình.

- Khi soạn thảo chương trình cho phép đưa vào các chú thích nhưng phải được đặt trong cặp dấu móc {…} hoặc (*…*).

IV. Các bước lập trình

Bước 1: Tìm hiểu mục đích, yêu cầu của bài toán

Mỗi bài toán có những mục đích, yêu cầu mà người lập trình phải tìm hiểu thật kỹ mới không bị sai lầm khi viết chương trình. Sau khi nắm chắc nội dung của bài toán, người lập trình sẽ tìm được các biện pháp, các công cụ để giải bài toán đó.

Bước 2: Mô tả giải thuật và vẽ lưu đồ

Khi giải 1 bài toán thì ta dùng ngôn ngữ toán học để diễn đạt ý tưởng. Trong lập trình thì ta sẽ sử dụng ngôn ngữ giải thuật đó là những từ ngữ rất ngắn gọn để mô tả cách thức giải bài toán đó

Bước 3: Viết chương trình

Bước này ta sử dụng các câu lệnh, các cấu trúc của ngôn ngữ lập trình để viết chương trình. (có thể viết nháp ra giấy trước)

Bước 4: Nhập và chạy thử chương trình

Khởi động chương trình Pascal (C:\TP\BIN\Turbo.exe)

Soạn thảo chương trình đã viết nháp ấn tổ hợp Ctrl + F9 để chạy chương trình (nếu có lỗi thì máy sẽ thông báo và phải sửa cho đúng)

Nhập các số liệu để tính toán. Nếu chạy kết quả không đúng thì thoát về chương trình nguồn để kiểm tra.

Một số phím chức năng thường dùng

- **F2**: Lưu chương trình đang soạn thảo vào đĩa.
- **F3**: Mở file mới hoặc file đã tồn tại trên đĩa để soạn thảo.
- **Alt-F3**: Đóng file đang soạn thảo.
- **Alt-F5**: Xem kết quả chạy chương trình.
- **F8**: Chạy từng câu lệnh một trong chương trình.
- **Alt-X**: Thoát khỏi Turbo Pascal.
- **Alt-<Số thứ tự của file đang mở>**: Dịch chuyển qua lại giữa các file đang mở.
- **F10**: Vào hệ thống Menu của Pascal.
- Chọn khối văn bản: **Shift + <Các phím ←↑→↓>**
- **Ctrl-KY**: Xoá khối văn bản đang chọn
- **Ctrl-Insert**: Đưa khối văn bản đang chọn vào Clipboard
- **Shift-Insert**: Dán khối văn từ Clipboard xuống vị trí con trỏ.

BÀI 2 CÁC KIỂU DỮ LIỆU ĐƠN GIẢN

2.1 Tổng quan về các kiểu dữ liệu trong Turbo Pascal

Trong Turbo Pascal các kiểu dữ liệu được chia làm 2 loại:

- Các kiểu dữ liệu đơn giản
- Các kiểu dữ liệu có cấu trúc

2.1.1. Các kiểu dữ liệu đơn giản:

* *Kiểu chuẩn:*

- Logic - Số nguyên
- Số thực - Ký tự - Xâu

* *Kiểu do người dùng định nghĩa*

- Kiểu đoạn con - Kiểu liệt kê

2.1.2. Các kiểu dữ liệu có cấu trúc

- Mảng - Tập hợp
- Bản ghi - File

2.2. Các kiểu dữ liệu đơn giản chuẩn

2.2.1. Kiểu logic:

* *Từ khoá:* Boolean.

* *Miền trị:* Chỉ có 2 giá trị là True và False

* *Các phép toán:* AND, OR, NOT, XOR

A	B	A AND B	A OR B	NOT A	A XOR B
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	T
F	F	F	F	T	F

Phép so sánh:

Bằng nhau: "=" Nhỏ hơn: "<"

Nhỏ hơn hoặc bằng: "<=" Lớn hơn: ">"

Lớn hơn hoặc bằng: ">=" Khác nhau: "<>"

2.2.2. Kiểu số nguyên:

Có 5 kiểu số nguyên:

* *Miền trị* của các kiểu đó như sau:

Tên	Min	Max	Yêu cầu bộ nhớ
Shorint	-128	127	1 byte
Integer	-32.768	32.767	2 byte
Longint	-2.147.483.648	2.147.483.647	4 byte
Byte	0	255	1 byte
Word	0	65.535	2 byte

** Các phép toán:*

Phép cộng: "+" Phép trừ: "-" Phép nhân "*"

Phép DIV: Phép chia hai số nguyên cho nhau cho kết quả là phần nguyên của thương. Ví dụ: 12 div 5 = 2.

Phép MOD: Chia hai số nguyên cho nhau cho kết quả là số dư của phép chia đó. Ví dụ: 27 mod 8 = 3.

** Các phép so sánh*: Có đầy đủ các phép so sánh tương tự như kiểu logic. Tuy nhiên ở đây ta đã biết quan hệ thứ tự trên tập hợp các số nguyên.

*** Chú ý:**

(1). Trên dữ liệu kiểu số nguyên không tồn tại phép chia thực (/).

(2). Thông thường cá số nguyên được biểu diễn trong hệ cơ số thập phân. Nếu muốn biểu diễn trong cơ số 16 thì ta thêm dấu "$" phía trước. Ví dụ $A (tức là số 10 trong hệ thập phân).

(3). Với kiểu số nguyên còn có các phép toán AND, OR, NOT, XOR…

2.2.3. Kiểu số thực: Có 5 kiểu số thực.

** Miền trị* của chúng được cho bởi bảng sau:

Tên	**Min**	**Max**	**Yêu cầu bộ nhớ**
Real	$-2.9 * 10^{-39}$	$1.7 * 10^{38}$	6 byte
Single	$-1.5 * 10^{-47}$	$1.7 * 10^{38}$	4 byte
Double	$-5 * 10^{-324}$	$1.7 * 10^{380}$	8 byte
Extended	$-3.4 * 10^{-4932}$	$1.1 * 10^{4932}$	10 byte

*** Chú ý:**

(1). Miền trị của các kiểu dữ liệu được hiểu là lấy trong các đoạn:

* [-max, -min]

* [min, max]

* số 0.

Nếu một số <-max hoặc >max thì sẽ không biểu diễn được và xem như là tràn số.

Nếu một số >-min nhỏ hơn 0 hoặc lớn 0 nhỏ hơn min thì được xem là 0.

(2). Chế độ mặc định của Turbo Pascal là chỉ cho phép làm việc với kiểu số thực REAL. Muốn sử dụng các kiểu khác ta vào bảng chọn OPTION → COMPILER → NUMERIC PROCESING rồi đánh dấu [x] vào mục 8087/80287.

(3). Trong máy tính số thực được biểu diễn dưới hai dạng:

- Dấu chấm tĩnh, ví dụ: 3.14, 123.456…
- Dấu chấm động (chế độ mặc định), ví dụ: 3.1400000000E+02 (tức là $3.14*10^2$).

** Các phép toán:*

Cộng: "+" Trừ: "-"

Nhân "*" Chia "/"

*** Chú ý:** Với kiểu số thực không tồn tại các phép DIV và MOD

Các phép so sánh: Tương tự số nguyên gồm: =, <, >, >= (lớn hơn hoặc bằng), <= (nhỏ hơn hoặc bằng), <> (khác nhau).

2.2.4. Các hàm chuẩn

a. Các hàm trên kiểu số nguyên và số thực:

Giả sử x là một số nào đó khi đó ta có các hàm sau đây:

ROUND(x) ----→ Cho giá trị là số nguyên gần x nhất } Kiểu số nguyên

TRUNC (x) ----→ Cho giá trị là phần nguyên của số x

INT(x)----------→ Cho giá trị là phần nguyên của số x

ABS(x)----------→ Cho giá trị là gí trị tuyệt đối của số x

SIN(x)----------→ Cho giá trị là sinx.

COS(x)----------→ Cho giá trị là cosx

EXP(x)----------→ Cho giá trị là e^x

SQR(x)----------→ Cho giá trị là x^2

SQRT(x)--------→ Cho giá trị là căn bậc hai của số x (x>=0).

LN(x)----------→ Cho giá trị là lnx (x>0).

} Kiểu số thực

b. Các hàm khác

Giả sử var là biến có kiểu vô hướng đếm được.

INC(var, r)	DEC(var, r)	PRED(var)	SUCC(var)
ORD(ch)	CHR(n)	UPCASE(ch)	ODD(n)
RANDOM(n)			

c. Sử dụng hàm để viết các biểu thức bằng cú pháp của Turbo Pascal:

1. $a^b = e^{bln(a)}$ = exp(b*ln(a)). (a, b>0). 2. $Log_ab = log_ae*log_eb = 1/ln(a)*ln(b)$.

3. $x^3+sin(x^2y))^2$ = sqr(sqr(x)*x + sin(sqr(x)*y)).

2.2.5. Kiểu ký tự (CHAR):

- Từ khoá: CHAR.
- Miền trị: Các ký tự trong bảng mã ASCII bao gồm:

  0..31: Các ký tự điều khiển

  32..127: Các ký tự thông dụng

  128..255: Các ký tự đặc biệt (đồ hoạ).

Ví dụ: Ký tự ‘A’ có mã là 65; Ký tự ‘a’ có mã là 97.

Chú ý: Để phân biệt ký tự cũng như xâu ký tự với các đối tượng khác. Pascal quy định khi biểu diễn chúng phải đặt trong cặp dấu nháy đơn, ví dụ ‘a’, ‘abc’.

Phép so sánh: Muốn so sánh 2 ký tự ta so sánh các mã ASCII tương ứng của chúng, ký tự nào có mã ASCII lớn hơn được xem là lớn hơn. Ví dụ ‘a’ > ‘A’ vì 97 > 65.

Chú ý: Một kiểu dữ liệu được lọi là vô hương đếm được nếu miền trị của nó là một tập hợp đếm được và trên đó tồn tại quan hệ thứ tự. Ví dụ: Kiểu Byte, kiểu Integer; kiểu Char; kiểu Boolean…

2.2.6. ***Kiểu xâu (chuỗi ký tự)***

Chuỗi ký tự là kiểu dữ liệu không chuẩn hay còn gọi là kiểu dữ liệu có cấu trúc gồm một chuỗi các ký tự trong bảng mã ASCII. Đối với dữ liệu Kiểu chuỗi:

- Số ký tự trong một chuỗi có thể thay đổi từ 0 đến một giá trị xác định trong khi báo kiểu.
- Số ký tự trong một biến kiểu mảng luôn có chiều dài cố định.

BÀI 3 BIỂU THỨC VÀ CÂU LỆNH ĐƠN GIẢN

3.1. Biểu thức

3.1.1. Hằng: Hằng là đại lượng không thay đổi trong chương trình. Có các loại hằng: hằng số, hằng ký tự, hằng xâu, hằng logic…Từ khoá để khai báo hằng là CONST.

Ví dụ: Const max = 100; ch = 'A'; hoten = 'người van A';

3.1.2. Biến: Biến là đại lượng có thể thay đổi giá trị vào từng thời điểm khác nhau của chương trình. Biến dùng để lưu trữ dữ liệu. biến được khai báo sau từ khoá VAR. Việc khai báo các biến là nhằm cung cấp các vùng nhớ để lưu trữ dữ liệu trong quá trình thực hiện chương trình.

Ví dụ:

```
Var   a, b, c: Integer;
      Ch: Char;
      Ok: Boolean;
```

*** Chú ý:** Khi khai báo có nhiều biến cùng kiểu thì ta dùng dấu phẩy (",") để ngăn cách. Hết khai báo kiểu dữ liệu này chuyển qua khai báo kiểu dữ liệu khác ta dùng dấu ";"

3.1.3. Toán tử: Là các phép toán tác động lên dữ liệu (hay còn gọi là toán hạng).

Ví dụ: Các phép +, -, *, /, and, or, not…

*** Chú ý:** Thứ tự ưu tiên thực hiện các phép toán như sau:

- Các phép toán 1 ngôi.
- Các phép *, /, DIV, MOD, AND.
- Các phép +, -, OR, XOR
- Các phép toán so sánh =, <>, <=, >=, <, >

3.1.4. Toán hạng: Là một trong các đại lượng sau:

Hằng, biến, hàm, biểu thức

3.1.5. Biểu thức: Là một tập hợp gồm các toán tử và các toán hạng trong đó toán tử tác động phù hợp lên toán hạng.

Ví dụ: Biểu thức Readln*pi + abs(x + y) – 10

Các toán tử: *, +, -

Các toán hạng: Readln, x, y là các biến; pr, 10 là hằng; abs là hàm.

*** Chú ý:**

(1). Trong một biểu thức các toán tử trong dấu ngoặc "(…)" được ưu tiên thực hiện trước.

(2). Nếu có nhiều phép toán cùng cấp ưu tiên thì các phép toán được thực hiện tuần tự từ trái qua phải (trừ các phép toán tác động lên kiểu logic).

(3). Một biểu thức cho kết quả hằng số được gọi là biểu thức số học, cho kết quả kiểu logic thì được gọi là biểu thức logic.

Ví dụ: Tính giá trị của biểu thức sau:

((18 mod 4 div 2 < 3) >= false) = true

((2 div 2 < 3) >= false) = true

(1 < 3) >= false) = true

(true >= false) = true

True = true ---> true

3.2. Câu lệnh

3.2.1. Phân loại câu lệnh: Trong Pascal có hai loại câu lệnh đó là câu lệnh đơn giản và câu lệnh có cấu trúc.

Câu lệnh đơn giản gồm: Lệnh gán, lệnh xuất, nhập dữ liệu, lệnh nhảy và lời gọi chương trình con dạng thủ tục.

Câu lệnh có cấu trúc gồm: Lệnh ghép, lệnh rẽ nhánh, lệnh lựa chọn, các lệnh lặp và lệnh WITH.

3.2.2. Câu lệnh đơn giản.

a. Lệnh gán:

- Cú pháp: Ten_bien: = BT;
- Giải thích BT là ký hiệu cho biểu thức.
- Ý nghĩa: Lệnh sẽ thực hiện việc gán giá trị của BT cho biến Ten_bien.
- Chú ý:

(1). Biến Ten_bien và giá trị của BT phải cùng kiểu dữ liệu.

(2). Vế trái luôn là biến còn về phải là biểu thức, đảo ngược lại là sai cú pháp.

Ví dụ: a: = 5; S: = S + d * i; i: = i + 1;

b. Lệnh ghi dữ liệu lên màn hình.

* Cú pháp

(1) WRITE(Biểu thức 1, Biểu thức 2,…, Biểu thức n); (n≥1)

(2) WRITELN(Biểu thức 1, Biểu thức 2, ….Biểu thức n);

(3) WRITELN

* Ý nghĩa:

(1). Lần lượt ghi lên màn hình giá trị của các biểu thức tại vị trí con trỏ.

(2). Tương tự dạng (a) chỉ khác ở chỗ sau khi ghi xong con trỏ được đưa về đầu dòng tiếp theo.

(3). Không ghi gì lên màn hình cả, chỉ làm thao tác đưa con trỏ về đầu dòng tiếp theo.

Ví dụ:

```
Var a, b: integer;
Begin     a: = 5; b: =10;
          Writeln('tich cua', a, ' va', b, 'la :', a*b);
End.
```

*** Chú ý:**

(1). Giá trị của các biểu thức trong câu lệnh Write, Writeln phải có kiểu chuẩn.

(2). Có thể sử dụng cách ghi dữ liệu có quy cách như sau:

* Với dữ liệu kiểu số nguyên, ký tự, xâu, logic:

(a) WRITE (Biểu thức: Độ dài,…);

(b) WRITELN(BT: Độ dài,…);

Khi đó giá trị của biểu thức sẽ được ghi lên màn hình theo chế độ căn phải, trong đó Độ dài là số vị trí (cột) giành để ghi dữ liệu.

Ví dụ:

```
WRITE (123:5);        ---> ~~123
WRITE('abcd':8);      ---> ~~~~abcd
WRITE(3=5:9)          ---> ~~~~~False
```

** Với dữ liệu kiểu số thực*

Chế độ mặc định của Pascal là ghi theo dạng dấu chấm động.

Muốn ghi số thực theo dạng dấu chấm tĩnh ta dùng cách ghi sau đây:

(a) WRITE(Biểu thức: Độ dài: m,…); (Độ dài>m)

(b) WRITELN(Biểu thức: Độ dài: m,…);

Khi đó giá trị của biểu thức sẽ được ghi lên màn hình theo chế độ căn phải, trong đó Độ dài là số vị trí (cột) giành để ghi dữ liệu.

Ví dụ:

```
WRITE(123.456 :12 : 2);   ---> ~~~~~~123.47
```

*** Chú ý**: Khi ghi dữ liệu là số thực lên màn hình theo dạng có quy cách có thể số được ghi lên đã được quy tròn (ví dụ trên) nhưng giá trị thực của nó vẫn được lưu giữ trong bộ nhớ.

c. Lệnh nhập dữ liệu từ bàn phím:

* Cú pháp:

(1) READ (Bien 1, Bien 2,…, Bien n); (các biến đã được khai báo).

(2) READLN (Bien 1, Bien 2, …, Bien n);

(3) READLN;

* Giải thích: Bien 1, Bien 2,…Bien n ký hiệu cho biến 1, biến 2, …,biến n.

* Ý nghĩa:

(1): Cho phép lần lượt nhập dữ liệu cho các biến.

(2): Tương tự dạng (a) chỉ khác ở chỗ sau khi nhập xong con trỏ được đưa về đầu dòng tiếp theo.

(3): Chỉ làm thao tác tạm dừng chương trình chờ gõ bàn phím Enter để tiếp tục.

*** Chú ý:**

(1). Nguyên tắc nhập dữ liệu là trong câu lệnh có bao nhiêu biến thì phải nhập đủ giá trị cho bấy nhiêu biến. Giữa giá trị của biến này và biến kia được ngăn cách bởi ít nhất một ký tự trắng; kết thúc việc nhập bằng cách nhấn phím ENTER.(Nếu chưa nhập đủ giá trị cho các biến thì chương trình vẫn tiếp tục dừng lại chờ nhận tiếp, chừng nào nhập đủ mới tiếp tục, Ví dụ khi gặp lệnh READ(a, b, của); chương trình tạm dừng chờ nhập dữ liệu và khi đó ta nhập chẳng hạn 3~9~12)

(2). Thông thường khi thực hiện chương trình để nhập dữ liệu ta nên ghi lên màn hình câu hướng dẫn cho người thực hiện chương trình, nghĩa là trước câu lệnh READ, READLN, nên có một câu lệnh WRITE.

Ví dụ: Write(‘nhập n nguyên dương:’); readln(n);

(3). Câu lệnh READLN thường được sử dụng tạm dừng chương trình (màn hình) để xem kết quả, vì vậy thường được sử dụng ở cuối của một chương trình (trước từ khoá “END”).

(4). Với dữ liệu kiểu logic không cho phép nhập từ bàn phím.

3.2.3. Các ví dụ:

Ví dụ 1: Viết chương trình nhập vào chiều dài và chiều rộng của một hình chữ nhật rồi in lên màn hình chu vi và diện tích của hình chữ nhật đó.

Hướng dẫn:

- Khai báo: Các biến a, b kiểu số thực để lưu chiều dài, chiều rộng.
- Biến p để chứa chu vi, biến S để chứa diện tích.
- Nhập dữ liệu cho a, b.
- Tính chu vi theo công thức p: = 2*(a + b);
- Tính diện tích theo công thức S: = a * b;
- Ghi dữ liệu lên màn hình theo dạng có quy cách (số thực).

Chương trình:

```
Program hinh_chu_nhat;
Var a, b, p, s: real;
Begin
    Write ('Nhap chieu dai a:'); readln (a);
    Write ('Nhap chieu rong b:'); readln (b);
    p: = 2 * (a + b); s: = a * b;
    Writeln ('Chu vi cua hinh chu nhat la:', p: 10: 2);
    Writeln('Dien tich cua hinh chu nhat la:', s: 10:2);
    Readln
End.
```

Ví dụ 2: Viết chương trình nhập vào một số nguyên dương có ba chữ số rồi in lên màn hình số đảo ngược của số đó.

Hướng dẫn:

- Khai báo: Biến n để chứa dữ liệu nhập vào.
- Nhập n nguyên dương có 3 chữ số.
- Dùng các phép toán MOD và DIV lần lượt tách các chữ số hàng đơn vị, hàng trăm, hàng nghìn và ghi lên màn hình.

Chương trình

```
Program So_dao;
Var n: integer;
  Begin
      Write ('Nhap so nguyen co 3 chu so: '); readln (n);
      Write ('So dao nguoc của so', n ,' la: ');
      Writeln (n mod 10, n  mod 100 div 10, n div 100);
      Readln
  End.
```

Ví dụ 3: Viết chương trình tính tổng:

S = 1 + 2 + 3 + …+ n (n nhập từ bàn phím).

Hướng dẫn:

- Khai báo: Biến n kiểu số nguyên để chứa dữ liệu nhập vào.
- Biến S kiểu số nguyên để lưu kết quả.
- Nhập dữ liệu cho n.
- Tính S theo công thức S = n*(n+1) div 2;
- Ghi dữ liệu lên màn hình.

Chương trình.

```
Program Tinh_tong;
Var n, s: integer;
Begin
        Write ('Nhap so nguyen duong n:'); readln(n);
        s : = n*(n+1) div 2;
        Writeln ('Tong', n, 'so tu nhien dau tien la:', s);
        Readln
End.
```

BÀI 4 CÂU LỆNH CÓ CẤU TRÚC

4.1. Lệnh ghép

4.1.1. Khái niệm: Là tập hợp các lệnh được đặt giữa cặp từ khoá begin…end;

4.1.2. Cú pháp:

```
Begin
    CL1;
    CL2;
    …
End;
```

(Trong đó CL1, CL2,…ký hiệu cho câu lệnh 1, câu lệnh 2,…)

4.1.3. Ý nghĩa: Ghép nhiều lệnh thành một câu lệnh.

4.2. Lệnh IF (câu lệnh rẽ nhánh).

4.2.1. Cú pháp: Có 2 dạng:

(1) If <BTLG> then CL1
else CL2;

(2) If <BTLG> then CL;

Trong đó BTLG ký hiệu cho biểu thức logic.

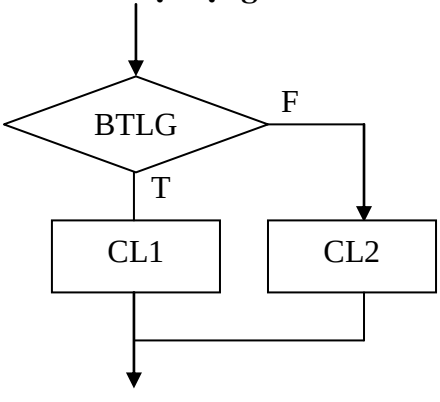
Dạng (1) còn được gọi là dạng đầy đủ, dạng (2) được gọi là dạng thiếu.

4.2.2. Ý nghĩa:

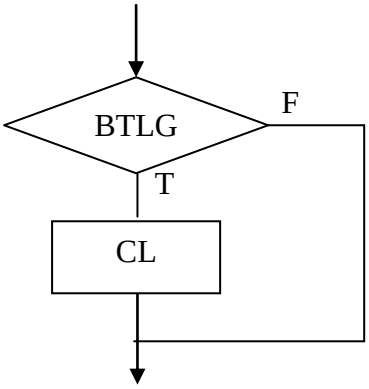
(1) Nếu BTLG nhận giá trị True thì máy sẽ thực hiện CL1, ngược lại máy sẽ thực hiện CL2.

(2) Nếu BTLG nhận giá trị True thì máy sẽ thực hiện CL, ngược lại thì không thực hiện.

4.2.3 Sơ đồ hoạt động:



Dạng (1)



Dạng (2)

4.2.4. Chú ý: Với lệnh If dạng (1) sau CL1 (trước else) không có dấu ";" vì nếu có thì máy sẽ hiểu là lệnh if dạng (2) sau đó gặp từ khoá else máy sẽ báo lỗi vì không có lệnh này.

4.2.5. Các ví dụ:

Ví dụ 1: Viết chương trình giải phương trình ax + b = 0

Hướng dẫn:

- Khai báo các biến a, b kiểu số thực để nhập dữ liệu, x để chứa nghiệm.
- Nhập a, b
- Biện luận nghiệm theo a và b
- Thông báo kết quả lên màn hình cho từng trường hợp

Chương trình

```
Program pt_bac_nhat;
Var a, b, x: real;
Begin
    Write ('nhap he so a, b:'); readln(a, b);
    If a<>0 then
        Begin
          x: = -b/a;
          Writeln('nghiem duy nhat la: ', x: 5: 2);
        End
    Else
        If b <> 0 then Writeln('phuong trinh vo nghiem')
        Else Writeln ('pt vo dinh – nghiem la moi x');
    Readln
End.
```

Ví dụ 2: Viết chương trình tính giá trị của hàm:

$$f(x) = \begin{cases} \dfrac{|x+1|}{\sqrt{x}} & x>0 \\ \sqrt[3]{x2-1} & x\le 0 \end{cases}$$

Hướng dẫn:

- Khai báo các biến x kiểu số thực để nhập đối số, For để chứa giá trị hàm.
- Nhập x
- Biện luận theo x để tính f.
- Thông báo kết quả lên màn hình số thực For theo dạng có quy cách.

Chương trình

```
Program Ham_so;
    Var x, f: real;
    Begin
        Write ('Nhap gia tri x:'); readln (x);
        If x > 0 then f: = (abs(x + 1))/ sqrt(x)
        Else f: = exp(1/3 * ln (sqr(x) -1);
        Writeln ('f(', x: 5: 2,' ) = ', f: 8: 2);
        Readln
    End.
```

4.3. Lệnh CASE (câu lệnh lựa chọn)

4.3. 1. Cú pháp: có 2 dạng:

(1)
```
Case <BT> of
   TH1: CL1;
   TH2: CL2;
    …
    THn: CLn;
  Else CLn+1;
  End;
```

(2)
```
Case <BT> of
   TH1: CL1;
   TH2: CL2;
   …
   THn: CLn;
End;
```

4.3.2. Giải thích:

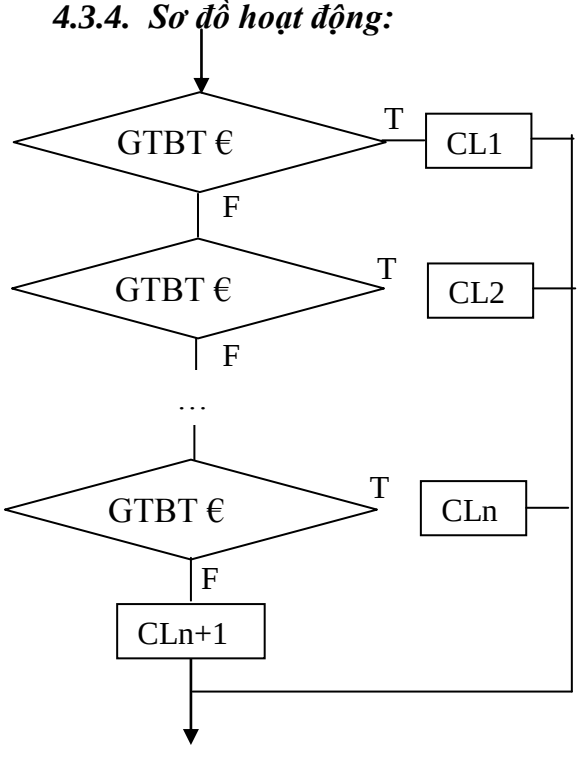
BT ký hiệu cho biểu thức; TH1, TH2, …THn ký hiệu cho các tập hằng; CL1, CL2,…CLn, CLn+1 ký hiệu cho các câu lệnh.

4.3.3. Ý nghĩa:

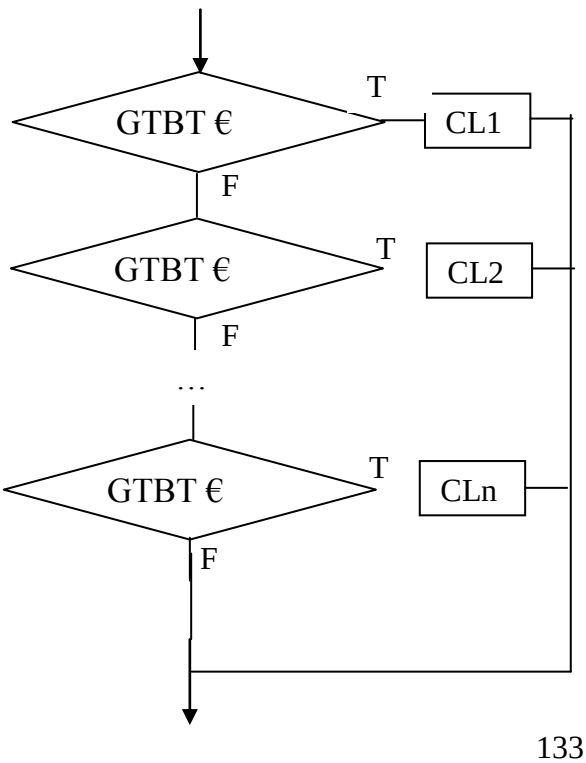
(1) Tuỳ thuộc vào giá trị của biểu thức <BT> thuộc vào tập hằng nào mà máy tính sẽ thực hiện câu lệnh tương ứng. Nếu không sẽ thực hiện CLn+1;

(2) Tương tự dạng (1), chỉ khác ở chỗ nếu giá trị của biểu thức không thuộc tập hằng nào cả thì máy sẽ không thực hiện câu lệnh nào cả trong thân lệnh CASE và sẽ thoát.

4.3.4. Sơ đồ hoạt động:



Dạng (1)



Dạng (2)

4.3.5 Chú ý:

(1). Giá trị của BT phải có cùng kiểu với các phần tử của tập hằng và phải là kiểu vô hướng đếm được.

(2). Tập hằng có thể là một giá trị, một tập hợp các giá trị, và thậm chí là một kiểu dữ liệu (kiểu đoạn con, liệt kê)

(3). Lệnh CASE có thể được thay thế bởi nhiều lệnh IF lồng nhau và ngược lại lệnh IF cũng có thể được thay thế bằng lệnh CASE. Ví dụ IF dạng (1) có thể được thay thế bằng:

```
CASE <BTLG> OF
    True: CL1;
    False: CL2;
END;
```

Tuy nhiên sử dụng lệnh Case hay If là tuỳ vào từng bài toán cụ thể.

4.3.6 Các ví dụ:

Ví dụ 1: Viết chương trình nhập vào một số n bất kỳ trong đoạn [0..255] và thông báo lên màn hình:

- Câu thông báo: 'Day la ky tu dieu khien' (nếu n<32)
- Câu thông báo: 'là ky tu thong dung' (nếu 127≥ n ≥32);
- Câu thông báo: 'la ky tu dac biet' (nếu n ≥ 128);

Hướng dẫn:

- Khai báo biến n kiểu byte để nhập dữ liệu.
- Nhập n
- Dùng lệnh CASE để phân chia các trường hợp theo n và thông báo kết quả lên màn hình theo yêu cầu.

Chương trình

```
Var n: byte;
Begin
    Write ('Nhap so n (0<=n<=255) : '); readln(n);
    CASE n of
        0..31: Writeln ('Day la ky tu dieu khien');
        32..127: Writeln (chr(n), 'la ky tu thong dung');
        128..255: Writeln (chr(n), 'la ky tu dac biet';
    END;
    Readln
End.
```

Ví dụ 2: Viết chương trình nhập vào tháng t (12 ≥t≥1), có thể nhập thêm năm sau đó in lên màn hình số ngày của tháng đó.

Hướng dẫn:

Khai báo biến t kiểu byte để nhập tháng; biến n kiểu số nguyên để nhập năm; biến sn kiểu byte để chứa số ngày của tháng.

- Nhập t.
- Dùng lệnh CASE để phân chia các trường hợp theo t:

- t € [1, 3, 5, 7, 8, 10, 12] thì sn: = 31;
- t € [2, 4, 6, 9] thì sn: = 30;
- t =2 thì nhập thêm năm n, nếu n là năm nhuận thì sn: = 29, ngược lại sn:= 28;

- Thông báo kết quả sn lên màn hình.

Chương trình:

```
Var  t, sn: byte;
     n: Integer;
Begin
      Write ('Nhap so thang t (1<=t<=12) : '); Readln (t);
      CASE t  of
       1, 3, 5, 7, 8, 10, 12: sn:= 31;
       4, 6, 9, 11: sn:=30;
       2: Begin Write ('Nhap nam:'); Readln (n);
          If n mod 4 = 0 then sn:=29 Else sn:=28;
          End;
      ELSE Writeln('khong co thang do');
      END;
      If t<=12 then Writeln('so ngay cua thang', t,'la:',sn);
      Readln
End.
```

4.4. Lệnh FOR (Lặp biết trước số lần)

4.4.1. Cú pháp: Có 2 dạng:

(1) For Bdk: = Gtdau To Gtcuoi Do CL;

(2) For Bdk: = Gtdau Downto Gtcuoi Do CL;

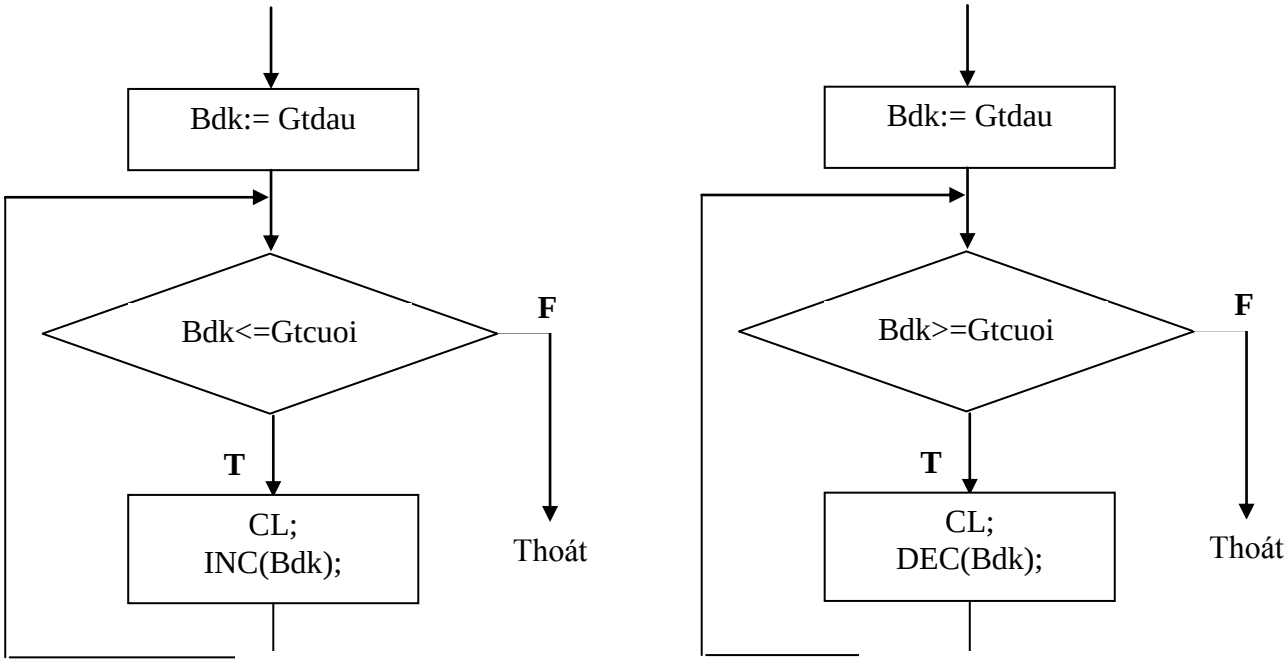
(Trong đó Bdk ký hiệu cho biến điều khiển; Gtdau ký hiệu cho giá trị đầu, Gtcuoi ký hiệu cho giá trị cuối)

4.4.2. Ý nghĩa

(1) Câu lệnh CL sẽ được lặp đúng bằng số lần mà Bdk được gán tăng dần lần lượt từ giá trị của Gtdau đến Gtcuoi.

(2) Câu lệnh CL sẽ được lặp đúng bằng số lần mà Bdk được gán giảm dần lần lượt từ giá trị của Gtdau đến Gtcuoi.

4.4.3. Sơ đồ hoạt động:



4.4.4. Chú ý:

(1). Biến điều khiển Bdk phải có kiểu vô hướng đếm được (thường là kiểu số nguyên).

(2). Trong thân câu lệnh FOR không được làm thay đổi giá trị của Bdk; khi thoát khỏi lệnh FOR biến điều khiển sẽ không có giá trị xác định.

(3). Có thể sử dụng lệnh Break để thoát khỏi lệnh FOR cũng như thoát khỏi các lệnh WHILE, REPEAT.

4.4.5. Các ví dụ:

Ví dụ 1: Viết chương trình hiển thị lên màn hình mã ASCII và các ký tự tương ứng, cứ 20 ký tự trên một trang màn hình.

Hướng dẫn:

- Khai báo biến điều khiển lệnh FOR là i kiểu byte;
- Dùng lệnh FOR dạng (1) với biến điều khiển i chạy từ 0 đến 255, với mỗi giá trị của i ta ghi lên màn hình ký tự có mã ASCII tương ứng. Sau đó kiểm tra xem nếu i+1 chia hết cho 20 thì dùng lệnh readln để tạm dừng màn hình.

Chương trình

```
Var i: byte;
Begin
  For i: = 0 to 255 do
      Begin
          Writeln ('ky tu tuong ung voi ma', i , ' la :', chr (i));
          If i+1 mod 20 = 0 then readln;
      End;
End.
```

Ví dụ 2: Viết chương trình tính tổng:

$S = 1^3 + 2^3 + 3^3 + \ldots + n^3$ (n nhập từ bàn phím)

Hướng dẫn:

- Khai báo: Biến n kiểu số nguyên integer để chứa dữ liệu nhập vào.
- Biến S kiểu longint để lưu kết quả.
- Nhập n.
- Khởi gán S: = 0
- Dùng lệnh FOR dạng (1) với biến điều khiển i chạy từ 1 đến n, với mỗi giá trị của i ta thực hiện lệnh cộng dồn S: = S + sqr(i) * I;
- Ghi kết quả S lên màn hình.

Chương trình

```
Program Tinh_tong;
   Var n, s, i : integer;
   Begin
       Write ('Nhap so nguyen duong n :'); Readln (n);
       S: = 0;
       For i: =  1 to n Do s: = s + sqr (i) * I;
       Write ('Tong lap phuong', n, 'so t.n dau tien la: ', s);
       Readln
   End.
```

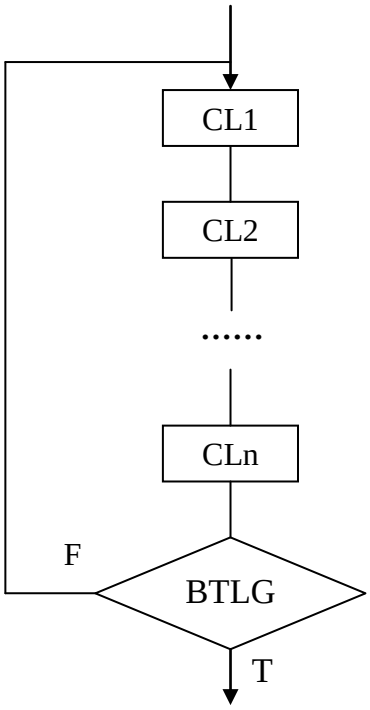
4.5. Lệnh REPEAT (lặp với số lần không biết trước).

4.5.1. Cú pháp:

```
Repeat
CL1;
CL2;
…
CLn;
Until <BTLG>;
```

4.5.2. Ý nghĩa: Thực hiện lần lượt các câu lệnh từ CL1 đến CLn dau đó kiểm tra giá trị của biểu thức logic, nếu sai thì lặp lại, nếu đúng thì thoát.

4.5.3. Sơ đồ hoạt động:



4.5.4. Chú ý:

(1). Các câu lệnh nằm giữa Repeat …Until không cần phải sử dụng cặp từ khoá Begin …End;

(2). Trong thân của lệnh Repeat thường phải có câu lệnh làm thay đổi giá trị của biểu thức logic nếu không dễ dẫn đến tình trạng lặp vô hạn.

(3). Khi thực hiện chương trình nếu bị lặp vô hạn thì nhấn tổ hợp phím CTRL – BREAK để dừng chương trình.

(4). Lệnh Repeat thường được sử dụng khi phải thực hiện ít nhất một lần các câu lệnh từ CL1 đến CLn và số lần lặp không xác định được trước. (Trong trường hợp biết trowcs số lần lặp ta nên dùng lệnh For).

4.5.5. Các ví dụ:

Ví dụ 1: Sử dụng lệnh Repeat để lọc dữ liệu vào theo yêu cầu, chẳng hạn cần nhập n nguyên dương ta viết như sau:

```
Repeat
    Write ('Nhap n > 0: ');
    Readln (n);
Until n > 0;
```

Ví dụ 2: Viết chương trình nhập điểm 6 môn thi sau đó tính điểm trung bình, với yêu cầu mỗi lần tính xong lại thông báo lên màn hình câu: "Tiep tuc hay khong (c/k) ?", nếu nhập 'c' thì lại tiếp tục, nếu nhập 'k' thì kết thúc.

```
Program Diem_trung_binh;
   Var a, b, c, d, e, f : Real;
   Ch: char;
      Begin
         Repeat
         Write ('nhap diem 6 mon thi a, b, c, d, e, f :');
         Readln (a, b, c, d,  e, f);
         Writeln('Diem trung binh la: ', (a + b + c + d + e + f)/6:4:2);
         Writeln ('Tiep tuc nua khong ?'); Readln (ch);
      Until Upcase (ch) = 'K';
   End.
```

Ví dụ 3. Viết chương trình nhập vào một số nguyên dương bất kỳ rồi in lên màn hình số chữ số của số đó.

Hướng dẫn:

- Khai báo biên n và biến dem kêiu Integer để nhập dữ liệu và đếm số chữ số.
- Nhập n.
- Khởi bán biến dem: =0;
- Mỗi lân tăng biến dem lên 1 thì loại bỏ đi chữ số sau cùng của n bằng lệnh gán lại n: = n div 10 và lại lặp lại cho đến khi n = 0 thì thôi.
- Thông báo kết quả là giá trị biến dem lên màn hình.

Chương trình.

```
Program Dem_so_chu_so;
Var n, dem: integer;
Begin
      Repeat
         Write ('Nhap n > 0 :'); Readln (n);
      Until n>0;
      Dem: = 0;
      Repeat
         Dem: = Dem + 1;
         n : = n div 10;
      Until n = 0;
      Writeln ('so', n, ' co ', dem, 'chu so');
      Readln;
End.
```

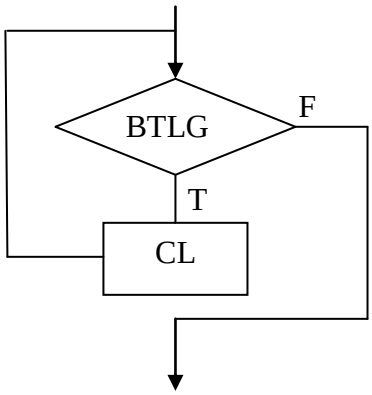
4.6. Lệnh WHILE (lặp với số lần không biết trước).

4.6.1. Cú pháp:

While <BTLG> Do CL;

4.6.2. Ý nghĩa: Chừng nào BTLG còn nhận giá trị True thì máy còn thực hiện câu lệnh CL;

4.6.3. Sơ đồ hoạt động:



4.6.4. Chú ý:

(1). Cũng giống lệnh Repeat, trong thân của lệnh While thường phải có câu lệnh làm thay đổi giá trị của BTLG nếu không dễ dẫn đến tình trạng lặp vô hạn.

(2). Khác với lệnh Repeat lệnh While kiểm tra điều kiện của biểu thức logic trước, nên có thể câu lệnh trong thân của nó không được thực hiện một lần nào cả khi mà ngay từ đầu BTLG nhận giá trị sai.

4.6.5. Các ví dụ:

Ví dụ 1: Sử dụng lệnh While, viết chương trình nhập vào một số nguyên dương bất kỳ rồi in lên màn hình số chữ số của số đó.

Chương trình:

```
Program Dem_so_chu_so;
   Var n, dem: integer;
   Begin
        Repeat
            Write ('Nhap n > 0: ');
            Readln (n);
        Until n > 0;
        Dem: = 0;
        While n>0 Do
            Begin
              Dem: = Dem + 1;
              n: = n div 10;
            End;
        Writeln ('so do co', dem, 'chu so');
   Readln;
   End.
```

Ví dụ 2: Viết chương trình nhập vào hai số nguyên dương rồi tìm ước số chung lớn nhất của chúng.

Hướng dẫn:

```
Program UCLN;
    Var m, n: integer;
    Begin
        Repeat
            Write (' Nhap m, n nguyen duong:');
            Readln (m, n);
        Until (m>0) and (n>0);
        While m <> n Do
            If m>n then m: = m –n
            Else n: = n – m;
        Writeln ('UCLN cua hai so do la:', m);
        Readln;
    End.
```

BÀI 5 DỮ LIỆU KIỂU TẬP HỢP KIỂU MẢNG - KIỂU XÂU

5.1. Kiểu tập hợp

5.1.1. Khái niệm:

Tập hợp là kiểu dữ liệu có cấu trúc bao gồm các phần tử có cùng kiểu vô hướng đếm được nào đó.

5.1.2. Định nghĩa kiểu và khai báo biến.

** Định nghĩa kiểu:* Type Ten_kieu = Set OF Tap_hop;

Giải thích: Type là từ khoá để định nghĩa kiểu;

Ten_kieu do người sử dụng đặt lấy đúng theo quy định của Turbo Pascal;

Set of là từ khoá để khai báo kiểu tập hợp.

Tap_hop Là một tập hợp gồm các phần tử có cùng kiểu vô hướng đếm được, ví dụ 1..10; byte; 'a' .. 'z';…

* Khai báo biến: Sau khi đã định nghĩa kiểu ta có thể khai báo biến thông qua tên kiểu

Ví dụ:

```
Type    Chu_cai = set of 'a' ... 'z';
            Tuoi = set of 0..250;
Var     ch: Chu_cai;
        t1, t2: tuoi;
```

Cũng có thể khai báo biên tập hợp trực tiếp không thông qua định nghĩa kiểu như sau:

Var ten_bien: Set of tap_hop;

Ví dụ: ch: set of ‘a’.. ‘z’;

t1, t2: set of 1..250;

Chú ý:

(1). Tap_hop cho phép tối đa là 256 phần tử.

(2). Gía trị của một biến kiểu tập hợp là một tập hợp con của tap_hop, khi biểu diễn các phần tử đó phải được đặt trong cập dấu móc vuông “[..]”.

(3). Tap_hop có thể biểu diễn dưới dạng kiểu đoạn con, kiểu liệt kê, các kiểu số như byte; char…

(4). Tập hợp rỗng được biểu diễn là “[]”.

5.1.3. Các phép toán:

a. Các phép quan hệ: Cho a, b là các biến có cùng một kiểu tập hợp khi đó ta có các phép toán:

a = b Khi a và b bằng nhau (theo nghĩa tập hợp).

a <> b Khi tập hợp a khác tập hợp b.

a <= b Khi a là tập con của b.

a >= b Khi b là tập con của a.

*** Chú ý:** Không tồn tại phép “>” hay nhỏ hơn “<”. Các phép này có thể được biểu diễn thông qua các phép quan hệ đã trình bày ở trên. Ví dụ để có được só sánh a<b ta có thể sử dụng cách viết: (a<=b) and (a<>b)…

b. Phép toán IN.

Phép toán IN dùng để kiểm tra xem một phần tử nào đó có mật trong một tập hợp hay không?

Ví dụ: ‘a’ In [‘a’.. ‘z’] cho giá trị là True.

c. Các phép toán tập hợp:

Phép hợp được biểu diễn bởi dấu “+”, ví dụ AU B được viết là A+B.

Phép giáo được biểu diễn bởi dấu “*”, ví dụ A B được viết là A*B

Phép hiệu được biểu diễn bởi dấu “-”, ví dụ A\B được viết là A-B

5.1.4. Ví dụ:

Viết chương trình nhập báo một chữ số rồi cho biết đó là nguyên âm hay phụ âm.

Hướng dẫn:

- Khai báo biến Chu_cai và biến Nguyen_am kiểu tập hợp mà các phần tử thuộc kiểu Char; khai báo biến Ch kiểu Char để chứa chữ cái nhập vào.
- Định nghĩa kiểu tập hợp chữ cái gồm các phần tử từ ‘a’ đến ‘z’ và ‘A’ đến ‘Z’ ; nguyên âm gồm các chữ cái ‘A, ‘E’, ‘I’, ‘O’, ‘U’.
- Nhập một chữ cái.
- Sử dụng phép toán IN để kiểm tra và thông báo kết quả lên màn hình.

Chương trình.

```
Program Nguyen_am;
Var chu_cai, nguyen_am: set of char;
Begin
    Chu_cai: = ['a'.. 'z'; 'A'.. 'Z'];
    Nguyen_am: = ['A', 'E', 'I', 'O' , 'U'];
    Repeat
        Write ('Nhap  mot chu cai: ');
        Readln(ch);
    Until ch in chu_cai;
    If upcase(ch) in nguyen_am
    Then Writeln(ch, 'la nguyen am')
    Else Writeln(ch, 'la phu am');
Readln;
End.
```

5.2. Kiểu mảng

5.2.1. Khái niệm:

Mảng là kiểu dữ liệu có cấu trúc bao gồm một số cố định các phần tử có cùng kiểu, cung một tên chung. Mỗi phần tử của mảng có thể xem như là một biến.

5.2.2. Định nghĩa kiểu và khai báo biến:

** Định nghĩa kiểu:* Type Ten_kieu = Array [CTCS] of KPT;

Giải thích:

Type là từ khoá để định nghĩa kiểu;

Ten_kieu do người sử dụng tự đặt lấy đúng teho quy định của Turbo Pascal.

CTCS ký hiệu cho các tập chỉ số.

KPT ký hiệu cho kiểu phần tử, và phải là kiểu cơ bản chuẩn, cũng có thể là kiểu mảng…

Array, of là từ khoá để khai báo kiểu mảng.

** Khai báo biến:* Sau khi đã định nghĩa kiểu ta có thể khai báo biến thông qua tên kiểu.

Ví dụ:

```
Type    Mang_Nguyen = array [1..100] of Integer;
        Mang_thuc = array [1..50] of real;
Var     m: Mang_nguyen;
        a, b: Mang_thuc;
```

Khi đó m là một biến kiểu mảng nên có các phần tử: m[1], m[2], …,m[100]. Mỗi phần tử như thế có thể xem như là một biến kiểu số nguyên và được biểu diễn thông qua tên biến và chỉ số (đặt trong cặp dấu móc vuông "[…]").

Tương tự a, biến cũng có các phần tử là: a[1], a[2], …a[50]; b[1], b[2], …,b[50].

Cũng có thể khai báo biến mảng trực tiếp không thông qua định nghĩa kiểu như sau:

Var Ten_bien: Array [CTCS] of KPT;

Ví dụ:

```
Var     m: array[1..100] of integer;
        a, b: array [1..50] of real;
```

Chú ý:

(1). Các CTCS phải là các không dữ liệu vô hướng đếm được và chỉ bồm hữu hạn phần tử. Ví dụ kiểu đoạn con, kiểu liệt lê, kiểu byte…

(2). Nếu có nhiều tập chỉ số thì dùng dấu phẩy "," để ngăn cách. Có bao nhiêu tập chỉ số thì mảng tương ứng được gọi là bấy nhiêu chiều và số phần tử của mảng bằng tích số phần tử của các tập chỉ số đó.

Ví dụ:

Var m: array [1..10, 1..5] of integer;

Khi đó m có 50 phần tử (10*5), các phần tử của biến mảng m là:

m[1,1], m[1,2],…, m[1,5].

m[2,1], m[2,2],…, m[2,5].

m[10,1], m[10,2],…,m[10,5].

(3). Không thể đọc và ghi dữ liệu cho toàn biến kiểu mảng mà phải thực hiện thông qua từng phần tử của biến mảng.

5.2.3. Các thao tác đối với biến mảng:

a. Gán một biến mảng cho một biến kiểu mảng khác:

Cho a, b là các biến có cùng một kiểu mảng và giả sử các phần tử của biến a đã được gán giá trị, khi đó ta có thể dùng lệnh gán b: =a. Sau lệnh gán này các phần tử của biến mảng b có giá trị tương ứng giống như các phần tử của biến mảng a.

Ví dụ:

```
Var a, b: array [1..5] of byte;
        i: byte;
Begin
    For i: = 1 to 5 do a[i]: = I;
    b: = a;
    …
End.
```

Thực hiện chương trình này ta có:

a[1]=b[1]=1;

a[2]=b[2]=2;

…

a[5]=b[5]=5;

b. Sắp xếp các phần tử của mảng:

Gỉa sử ta có một biến mảng chứa các phần tử nào đó có kiểu vô hướng đếm được. Vấn đề đặt ra là hãy sắp xếp các phần tử đó theo thứ tự tăng dần hay giảm dần.

Đây là bài toán sắp xếp cơ bản, người ta đã tìm ra nhiều thuật toán sắp xếp hay như thuật toán sắp xếp "nổi bọt", thuật toán "chèn",…Tuy nhiên ở đây ta đưa ra một cách sắp xếp đơn giản như thuật toán trình bày ở chương I.

Giả sử cần sắp xếp các phần tử a[1], a[2],…,a[n] theo thứ tự tăng dần ta dùng đoạn chương trình sau:

```
…
For i: = 1 to n-1 do
  Fof j: = i+1 to n do
    If a[i] > a[j] then
      Begin
        Tg: = a[i];
        a[i]: = a[j];
        a[j]: = tg;
      end;
…
```

Trong đoạn chương trình trên, với mỗi giá trị của i ta sử dụng lệnh for với biến điều khiển j để lần lượt so sánh a[i] với tất cả các phần tử a[j] đứng sau nó, nếu a[i] lớn hơn a[j] thì ta đổi chỗ a[i] và a[j] cho nhau, như vậy thì phần tử a[i] sẽ đứng đúng vị trí của nó trong danh sách sắp xếp. Cho i lần lượt đi từ 1 đến n-1 ta sẽ có danh sách được sắp xếp hoàn chỉnh.

Hoàn toàn ương tự cho vấn đề sắp xếp giảm dần.

5.2.4. Các ví dụ:

Ví dụ 1: Viết chương trình nhập vào n số nguyên dương rồi tính trung bình cộng và trung bình nhân của nó.

Hướng dẫn:

- Khai báo biên n, i kiểu số nguyên. Biến mảng kiểu số nguyên để chứa dãy số nhập vào. Các biến tbc, tbn kiểu số thực để chứa kết quả.
- Sử dụng lệnh for tính tổng, tích của n số nguyên dương đó từ đó tính trung bình cộng và trung bình nhân.
- Thông báo kết quả lên màn hình.

Chương trình.

```
Program Tinh_tb;
Var     n, i: integer;
        Tbc, tbn: real;
        a: array [1..100] of integer;
Begin
    Repeat
        Write ('Nhap n:');
        Readln (n);
    Until n > 0;
    Tbc: = 0; tbn: = 1;
    For i: = 1 to n  do
        Begin
            Repeat
                Write ('a[',i,']=');
                Readln (a[i]);
            Until a[i]>0;
            Tbc: = tbc+a[i];
            Tbn: = tbc+a[i];
        End;
    Tbc: = tbc/n; tbn: = exp(1/n*ln(tbn));
    Writeln ('Trung binh cong cua day la:', tbc:8:2);
    Writeln('Trung binh nhan cua day la:', tbn:8:2);
    Readln
End.
```

Ví dụ 2: Viết chương trình nhập vào hai ma trận vuông cấp n (với các phần tử nguyên) rồi tính tổng của hai ma trận đó.

Hướng dẫn:

- Khai báo b n, i, j, không kiểu số nguyên, Biến mảng a, b, c hai chiều các phần tử kiểu số nguyên để chứa các ma trận nhập vào và kết quả.
- Nhập cấp n của các ma trận vuông.
- Dùng hai lệnh for lồng nhau để nhập các phần tử cho ma trận a và b.
- Tính các phần tử của ma trận tổng của theo công thức.
- Thông báo kết quả lên màn hình.

Chương trình

```
Program Cong_ma_tran;
        Const
        Max=10;
  Type
        Matran=Array[1..Max,1..Max] Of Integer;
  Var
        A, B, C:Matran;
        i, j, m, n:Integer;
Begin
       Write('-So hang = '); Readln(m);
       Write('-So cot  = '); Readln(n);
       Writeln('MA TRAN A:');
       For i:=1 To m Do
          For j:=1 To n Do
             Begin
                Write('+Phan tu A[',i,',',j,' = ');
                Readln(a[i,j]);
             End;
       Writeln;
       Writeln('MA TRAN B:');
       For i:=1 To m Do
          For j:=1 To n Do
              Begin
                 Write('+Phan tu B[',i,',',j,' = ');
                 Readln(b[i,j]);
              End;
        For i:=1 To m Do
           For j:=1 To n Do
              C[i,j]:=A[i,j]+B[i,j];
        Writeln('Ma tran tong la: ');
        For i:=1 to m Do
            Begin
                 For j:= 1 To n Do
                   Write((c[i,j]):8);
                   Writeln;
            End;
         Readln;
End.
```

5.3. Kiểu xâu (chuỗi ký tự)

Chuỗi ký tự là kiểu dữ liệu không chuẩn hay còn gọi là kiểu dữ liệu có cấu trúc gồm một chuỗi các ký tự trong bảng mã ASCII. Kiểu chuỗi có nhiều điểm giống kiểu mảng nhưng có một số điểm khác biệt:

- Số ký tự trong một chuỗi có thể thay đổi từ 0 đến một giá trị xác định trong khi báo kiểu.

- Số ký tự trong một biến kiểu mảng luôn có chiều dài cố định.

5.3.1- Khai báo kiểu

```
Type
        Tên kiểu = String[n];
```

Trong đó Type là từ khoá khai báo kiểu, n là số phần tử tối đa mà chuỗi có thể nhận.

Ví dụ:
```
Type
        Hoten: String[24];
        Quequan: String;
```

5.3.2- Khai báo biến

Có 2 cách khai báo biến trong dữ liệu kiểu chuỗi:

- Khai báo trực tiếp:

```
Var
        Tên biến : String[n];
```

- Khai báo gián tiếp:

```
Type
        Tên kiểu = string[n];
Var
        Tên biến : Tên kiểu;
```

Đối với dữ liệu kiểu chuỗi thì phần tử đầu tiên (0) sẽ chứa ký tự biểu diễn độ dài thực của chuỗi do đó:

Độ dài của chuỗi = Ord(tên_tệp_ tin[0]);

5.3.3- Nhập/xuất dữ liệu kiểu chuỗi

Việc nhập xuất dữ liệu kiểu chuỗi tương tự như đối với các dạng dữ liệu chuẩn.

Để nhập dữ liệu từ bàn phím vào bộ nhớ ta cũng dùng các thủ tục Read hoặc readln, để xuất dữ liệu từ bộ nhớ ra ngoài màn hình ta cũng dùng thủ tục write hoặc writeln. Khi xuất dữ liệu ta có thể xuất theo khuôn dạng Writeln(biến: n). Nếu n>0 thì dữ liệu kiểu chuỗi chỉnh sát lề phải, ngược lại sẽ được chỉnh sát lề trái.

Để truy xuất đến từng ký tự của kiểu chuỗi ta có thể truy xuất thông qua chỉ số. Chỉ số của phần tử được đặt trong dấu móc vuông bắt đầu từ 1 riêng phần tử 0 sẽ chứa độ dài thực của chuỗi.

5.3.4- Các phép toán

- Phép cộng: Phép cộng đối với kiểu xâu ký tự thực chất là phép ghép nối 2 hoặc nhiều chuỗi ký tự với nhau.

Để nối các chuỗi ký tự với nhau ta có thể sử dụng toán tử cộng (+) hoặc hàm concat().

Ví dụ: Chuỗi 1:="Turbo"
Chuỗi 2:="Pascal"

Khi đó: Chuỗi 1+ Chuỗi 2 =" Turbo Pascal"
Concat(chuỗi 1, chuỗi 2) = " Turbo Pascal"

- Phép toán so sánh: Ta có thể sử dụng các phép toán quan hệ (>, <, =, >=, <=, <>) để so sánh 2 chuỗi với nhau. Kết quả trả về là một trong 2 trị của kiểu Boolean (True hoặc False). Khi so sánh 2 chuỗi ký tự thì các ký tự cùng vị trí của 2 chuỗi lần lượt được so sánh với nhau tù trái qua phải.

Nếu 2 chuỗi có độ dài khác nhau song kết quả so sánh của từng cặp ký tự là bằng nhau cho đến cặp ký tự cuối cùng thì chuỗi ngắn hơn sẽ nhỏ hơn.

2 chuỗi có độ dài bằng nhau nhưng có một cặp ký tự tương ứng giữa 2 chuỗi khác nhau khi đó chuỗi nào có ký tự tương ứng có mã ASCII lớn hơn thì lớn hơn.

5.3.5- Các hàm và thủ tục trên dữ liệu kiểu chuỗi

+ Các thủ tục

- Thủ tục Delete ()

Cú pháp: Delete(St, Pos, Num);

Trong đó: - St là chuỗi ký tự
- Pos là vị trí bắt đầu xoá
- Num: Số ký tự cần xoá

Công dụng: Dùng để xoá khỏi chuỗi St một chuỗi con chưa Num ký tự từ vị trí Pos trở đi.

Nếu Pos lớn hơn chiều dài của St thì không có ký tự nào bị xoá.

Nếu Pos + Num vượt quá chiều dài của St thì chỉ có các ký tự nằm trong phạm vi của chuỗi kể từ vị trí Pos mới bị xoá bỏ.

Trường hợp Pos nằm ngoài khoảng 0..255 thì hệ thống sẽ báo lỗi.

- Thủ tục Insert ()

Cú pháp Insert (St2, St1, Pos);

Trong đó: St2 là biểu thức chuỗi ký tự chèn, St1 là biến chuỗi ký tự, Pos là vị trí cần chèn.

Công dụng: Dùng để chèn chuỗi St2 vào vị trí Pos của chuỗi St1.

Nếu Pos lớn hơn chiều dài của St1 thì St2 được ghép vào cuối của St1

Nếu sau khi chèn mà chiều dài chuỗi mới lớn hơn chiều dài cho phép của chuỗi St1 thì những ký tự thừa sẽ bị cắt bỏ và chuỗi St1 chỉ chứa những ký tự bên trái nhất.

- Thủ tục Str()

Cú pháp: Str(Value, St);

Trong đó Value là các giá trị số (số nguyên hoặc số thực), st là chuỗi

Công dụng: Thủ tục Str() dùng để biến đổi giá trị số của Value thành chuỗi ký tự và lưu vào biến St.

- Thủ tục Val()

Cú pháp: Val (St, Var, Code);

Trong đó: St là chuỗi số cần đổi ra số

Var là biến số nguyên hoặc số thực

Code: Mã kết quả là một biến nguyên

Công dụng: Dùng để biến đổi một biểu thức chuỗi số St thành giá trị số nguyên hoặc số thực (tuỳ theo biến của Var) và gán giá trị kết quả cho biến Var. Code phải là một biến nguyên, nếu thủ tục thực hiện tốt thì Code = 0, ngược lại Code được gán một giá trị là vị trí thứ đầu tiên gây ra lỗi và khi đó giá trị của biến Var không được xác định.

+ Các hàm:

- Hàm Concat ()

Cú pháp: Concat(St1, St2,....., Stn)

Trong đó St1, St2..... là những biểu thức chuỗi ký tự

Công dụng: Dùng để ghép các chuỗi St1, St2..... thành một chuỗi mới.

- Hàm Copy()

Cú pháp: Copy(St, Pos, Num);

Trong đó: St là biểu thức chuỗi ký tự, Pos là biểu thức số nguyên chỉ định vị trí bắt đầu sao chép, Num là số ký tự cần sao chép.

Công dụng: Hàm Copy cho kết quả là một chuỗi con chứa Num ký tự của St kể từ vị trí Pos trở đi tính từ trái qua phải.

Nếu Pos lớn hơn chiều dài của St thì kết quả là một chuỗi rỗng.

Nếu Pos+Num lớn hơn chiều dài của St thì chỉ có các ký tự của St tính từ vị trí Pos đến cuối chuỗi mới được đưa vào kết quả.

- Hàm Length()

Cú pháp: Length(st);

Công dụng: Hàm Length() cho kết quả là một số nguyên xác định chiều dài thực của chuỗi St.

Chiều dài thực của chuỗi St có thể được xác định bằng hàm Ord(St[0]).

- Hàm Pos ();

Cú pháp: Pos(St2, St1);

Trong đó: St1 là biểu thức chuỗi thứ nhất

St2 là biểu thức chuỗi thứ hai

Công dụng: Hàm Pos cho kết quả là một số nguyen chỉ định vị trí xuất hiện của chuỗi St2 trong chuỗi St1. Nếu chuỗi St2 không có trong chuỗi St1 thì hàm pos có giá trị bằng 0.

5.3.6- Một số giải thuật trên xâu ký tự:

** Đổi một ký tư chữ hoa thành chữ thường.*

Trong bảng mã ASCII các ký tự từ "A" đến "Z' có thứ tự từ 65 đến 90

Các ký tự từ "a" đến "z" có số thứ tự từ 97 đến 122.

Như vậy công thức để đổi một chữ hoa ra chữ thường như sau:

Chữ thường = chữ hoa + 32

Khoảng cách giữa chữ thường và chữ hoa theo công thức

Khoảng cách = Ord('a') - Ord('A')

Giải thuật như sau:

If (Ch>=#65) and (Ch<=#90) then

Ch:=Chr(Ord(ch)+32)

Nếu quên các con số thì ta có thể dùng ký tự như sau:

If (Ch>="A") and (Ch<="Z') then

Ch:=Chr(Ord(ch) + Ord('a') - Ord('A'));

** Đổi một chuỗi ký tự thường thành hoa.*

Giải thuật: - Xác định chiều dài chuỗi

- Tạo vòng lặp và dùng hàm Upcase() để đổi ký tự chữ thường ra chữ hoa từ đầu đến cuối chuỗi rồi gán lại cho biến chuỗi ký tự.

```
For i:=1 To length(st) Do
St[i]:=Upcase(st[i]);
```

* Đổi một chuỗi ký tự hoa thành chữ thường

Giải thuật: - Tạo vòng lặp từ ký tự đầu đến ký tự cuối cùng của chuỗi

- Đổi một ký tự hoa thành ký tự thường.

```
For i:=1 to Length(St) Do
If (St[i]>=#65) and (St[i]<=#90) Then
St[i]:=Chr(ord(St[i]+32);
```

** Cắt các ký tự trống ở bên trái chuỗi ký tự.*

Giải thuật:

- Sử dụng vòng lặp While..Do để tìm ký tự trống đầu tiên (i=1) đến ký tự trống cuối cùng bằng cách tăng biến i lên 1 đơn vị, để xác định ký tự trống kế tiếp i:=i+1 hay Inc(i);

-Sử dụng thủ tục Delete() để xoá khỏi chuỗi i-1 ký tự.

```
I:=1
While St[i] = #32 Do
        I:=i+1;
Delete(St,1,i-1);
```

** Cắt các ký tự trống ở bên phải chuỗi*

Giải thuật: Dùng vòng lặp While .. Do tìm các ký tự trống kể từ ký tự ở bên phải nhất của chuỗi lần lượt trở về bên trái chuỗi cho tới ký tự trống cuối cùng. Giả sử đến vị trí thứ i không còn ký tự trống thì ký tự thứ i là chiều dài của chuỗi sau khi đã cắt bỏ các

ký tự trống ở bên phải. Thay ký tự thứ i vào phần tử thứ 0 để xác định chiều dài của chuỗi.

```
I:= Length(St);
While St[i] = #32 Do
    Dec(i);
    St[0]:=chr(i);
```

** Cắt các ký tự trống ở giữa chuỗi*

Giải thuật: Tìm chuỗi con có 2 khoảng trống trong chuỗi chính. Khi xác định được chúng sẽ xoá bớt một ký tự tại vị trí vừa tìm thấy. Tiếp tục làm như vậy cho tới khi không còn chuỗi con nào có 2 khoảng trống trong chuỗi (Sử dụng vòng lặp While.. Do)

```
I:=Pos(#32#32,St);
While i <> 0 Do
    Begin
        Delete(St, i ,1);
        I:=Pos(#32#32,St);
    End;
```

** Một số ví dụ đối với dữ liệu kiểu xâu ký tự:*

Ví dụ 1: Viết chương trình đổi một số nguyên hệ thập phân (cơ số 10)sang hệ nhị phân (cơ số 2).

```
Program Doi_thap_phan_ra_nhi_phan;
Var
   He10,N,Y:Word;
   He2,Tam:String;
Begin
    Writeln('DOI SO TU HE THAP PHAN SANG HE NHI PHAN');
    Writeln('       ----------------');
    Writeln;
    Write('-Nhap so nguyen he thap phan: ');
    Readln(He10);
    N:=He10;
    He2:=' ';
    Repeat
        Y:=He10 Mod 2;
        Str(Y, Tam);
```

```
            He2:=Tam + He2;
            He10:= He10 Div 2;
        Until He10 = 0;
        Writeln;
        Writeln('+So he 10 la     : ',N);
        Writeln('+Doi sang he 2 la: ',He2);
        Writeln;
        Writeln('      Bam phim <Enter> de ket thuc');
    Readln
End.
```

Ví dụ 2: Viết chương trình đảo ngược một chuỗi nhập vào từ bàn phím.

```
Program Dao_Chuoi;
Var
       St:String;
       i:Byte;
Begin
       Writeln('DAO NGUOC MOT CHUOI');
       Writeln('   -----------');
       Write('-Nhap chuoi: ');
       Read(St);
       Writeln;
       Write('+Chuoi dao : ');
       For i:= Length(St) DownTo 1 Do
              Write(St[i]);
       Writeln;
       Writeln;
       Writeln('      Bam phim <Enter> de ket thuc');
      Readln;
    Readln
End.
```

BÀI 6 CHƯƠNG TRÌNH CON HÀM VÀ THỦ TỤC

6.1. Đặt vấn đề.

Khi lập trình giải quyết một bài toán lớn, nếu viết chương trình một mạch từ trên xuống thì người lập trình sẽ rất vất vả và thậm chí có thể không thực hiện được. Mặt khác, nếu có viết được đi nữa thì việc sửa đổi và kiểm tra lỗi cũng sẽ rất là khó khăn vi phải đụng chạm đến toàn bộ chương trình.

Để khắc phục điều đó, người ta phân tích việc giải quyết bài toán thành những công việc tương đối độc lập. Sau đó chi tiết dần các công việc đó thành các công việc nhỏ hơn. Điều đó có nghĩa là chúng ta đã "phân mảnh" dần theo từng cấp. Turbo Pascal chop phép người lập trình "phân mảnh" chương trình nhằm phục vụ ý đồ trên. Có nghĩa là chia chương trình lớn thành nhiều phần nhỏ và giải quyết từng phần một (mỗi phần nhỏ như vậy được gọi là một chương trình con). Sau đó tuỳ nội dung công việc mà lắp ghép các chương trình nhỏ này lại với nhau để giải quyết bài toán.

Phương pháp lập trình có phân mảnh như trên được gọi là TOP-DOWN PROGRAMIN (Lập trình từ gốc đến ngọn hay còn gọi là lập trình có cấu trúc). Phân laọi và cấu trúc chung của chương trình con.

Turbo Pascal cho phép xây dựng hai loại chương trình con đó là thủ tục (Procedure) và hàm (Function). Trong đó:

- ***Thủ tục:*** là một chương trình con dùng để thực hiện một số thao tác xử lý nào đó để giải quyết một công việc cụ thể nào đó đã được phân mảnh.
- ***Hàm:*** là một chương trình con dùng để xác định một giá trị của đại lượng nào đó có kiểu dữ liệu đơn giản (số, kí tự, xâu kí tự, logic). Giá trị của đại lượng ra này được gọi là giá trị trả về của hàm.

Việc phân biệt này chỉ có tính cách tương đối nhằm phcụ vụ cho sự lựa chọn của người lập trình. Mỗi chương trình con để có cấu trúc chung theo tuần tự như sau:

- Phần khai báo chương trình con thuộc loại thủ tục hay hàm.
- Các định nghĩa và khai báo địa phương (hằng, kiểu, biến, chương trình con).

6.2. Thủ tục

6.2.1 Định nghĩa thủ tục:

Một thủ tục được định nghĩa theo cú pháp sau:

```
PROCEDURE TênThủTục [(Danh sách các tham số hình thức)];
        Các định nghĩa và khai báo địa phương;
        BEGIN
                Các câu lệnh xử lý; {thân của thủ tục}
        END;
```

Trong đó:

- TênThủTục là một định danh do người sử dụng đặt theo nguyên tắc đặt tên của Turbo Pascal.

- Danh sách các tham số hình thức: là tên của các đối tượng đóng vai trò nhân thông tin vào cho thủ tục hoạt động. Tham số hình thứuc có hai loại: tham biến và tham trị. Các tham số hình thức đều được xác định rõ kiểu dữ liệu của chúng và nếu la kiểu dữ liệu của người dùng thì kiểu này phải được định nghĩa trước đó chư không thể định nghĩa trực tiếp. Nếu là tham biến thì phải có từ khoá Var đứng trước.

Ví dụ:

1. PROCEDURE UCLN (m, n: Word; Var a: Word);
2. TYPE MANG = Array [1..100] of Integer;

   ……;

   PROCEDURE SapXep (Var A: Mang; N: Word);

   ……;

Một thủ tục có thể không có tham số hình thức nào. Khi đó phần khai báo sau tên thủ tục sẽ không có cặp dấu ngoặc.

Ví dụ: PROCEDURE TO_MAU;

- ***Các định nghĩa và khai báo địa phương:*** là các khai báo về hằng, kiểu, biến, chương trình con của nó. Các đối tượng được khai báo ở đây chỉ được dùng cho thủ tục này và các chương trình con chứa trong nó.

- ***Các câu lệnh xử lý*** là hệ thống câu lệnh được cài đặt nhằm thực hiện giải thuật đã thiết kế cho thủ tục.

Ví dụ:

Để viết tìm UCLN của hai số nguyên dương m và n cho trước ta chọn m, n đóng vai trò tham trị, giá trị UCLN tìm được sẽ lưu ở tham biến a. Thủ tục được viết như sau:

```
PROCEDURE UCLN (m, n: Word; Var a: Word);
    BEGIN
        If m>n  Then a: = n Else a:= m;
        Whle (m mod a <> 0) or (n mod a <> 0) Do a: = a -1;
    END;
```

6.2.2. Lời gọi thủ tục:

Sau khi thủ tục đã được định nghĩa, muốn sử dụng thủ tục đó thì phải thực hiện lời gọi thủ tục. Lời gọi thủ tục được thực hiện như sau:

- Nếu thủ tục có tham số hình thức thì lời gọi thủ tục phải có tên thủ tục và các tham số thực tế kèm theo như sau:

```
…;
TênThủTục [(danh sách tham số thực tế)];
…;
```

- Nếu thủ tục không có tham số hình thức thì lơig gọi thủ tục chỉ có tên thủ tục.

```
…;
TênThủTục;
…;
```

Nếu định nghĩa thủ tục có bao nhiêu tham số hình thức thì lời gọi thủ tục có bấy nhiêu tham số thực thế. Các tham số thực thế theo thứ tự sẽ lần lượt thay thế các tham số hình thức, các tham số thực thế được phân cách bởi một dấu phẩy. Tham số thực tế thay thế cho tham số hình thức nào thì phải có đúng kiểu dữ liệu của tham số hình thức đã khai báo. Ngoài ra cần chú ý:

- Tham số thực thế thay cho tham trị có thể là hằng, biến, biểu thức hay lời goọ hàm (vì dây là các đại lượng có giá trị xác định đóng vai trò nhận thông tin vào cho thủ tục hoạt động).
- Tham số thực thế thay cho tham biến bắt buộc phải là biến đã được khai báo trong phạm Ví dụ thủ tục có thể sử dụng được (vì chỉ có biến được khai báo ngoải thủ tục mới có thể lưu giữ thông tin ra).

Khi thực hiện lời gọi thủ tục, nếu có truyền biến cho tham biến hoặc tham trị thì trong quá trình xử lý giá trị của biến có thể bị thay đổi. Tuy nhiên, khi thủ tục hoạt động xong thì:

- Nếu tham số thực thế là biến truyền theo kiểu tham trị thì biến đó sẽ lấy lại giá trị của nó trước khi truyền cho thủ tục.
- Nếu tham số thực thế là biến truyền theo kiểu tham biến thì biến đó sẽ giữ lại giá trị của lần thay đổi cuối cùng.

Ngoài ra Turbo Pascal cho phép trong thân các chương trình con có thể có lời gọi đến các chương trình con đã được định nghĩa trước nó.

Ví dụ:

```
USES CRT;
Var m, n: Word;
Procedure Wait;
Begin
    Writeln ('ấn phím bất kì để tiếp tục');
    Repeat Until Keypressed;
End;
Procedure P1 (m: Word; Var n: Word);
Begin
    If m>n Then m:= m – n Else m: = n – m;
    n: = 2 * m;
    Writeln ('m=',m , 'n=', nhữngng);
End;
BEGIN
    m: = 5; n: = 8; {1}
    Writeln('m=', m, '=', n); {1}
    P1(m, n); {1}
    Wait; {1}
    Writeln ('m=', m, 'n=', n); {1}
    Wait;
END.
```

Khi chay chương trình trên thì kết quả thực hiện của các dòng lệnh được giải thích như sau:

{1} Gán giá trị cho biến m và n, ta được m = 5 và n = 8.

{2} Viết giá trị của biến m = 5 và n = 8 ra màn hình.

{3} Thực hiện lời gọi thủ tục P1. Trong thủ tục này có sự thay đổi giá trị của m và n, vì m>n có giá trị False nên thực hiện phép gán m: = n – m, tức là m = 3 và n = 6 tại thời điểm này ra màn hình.

Khi kết thúc thủ tục này, vì m truyền theo kiểu tham trị và n truyền theo kiểu tham biến nên m lấy lại giá trị ban đầu của nó, tức m = 5 và n lưu giữ giá trị của lần thay đổi sau cùng, tức n = 6.

{4} Thực hiện thủ tục Wait;

{5} Viết giá trị của m và n tại thời điểm này ra màn hình, tức m = 5 và n = 6.

{6} Thực hiện thủ tục Wait.

Như vậy, khi thực hiện chương trình trên sẽ cho ta kết quả in lên màn hình như sau:

```
m = 5    n = 8
m = 3    n = 6
ấn phím bất kỳ để tiếp tục
m = 5    n = 6
ấn phím bất kỳ để tiếp tục
```

6.2.3. Biến toàn cục và biến địa phương:

* ***Biến toàn cục*** là các biến được khai báo sau từ khoá VAR của chương trình chính.

**** Biến địa phương*** là các biến được khai báo sau từ khoá VAR trong các chương trình con.

```
USES CRT;
VAR  Khai báo các biến toàn cục;
PROCEDURE AAA(danh sách các tham số hình thức);
        Var Khai báo các biến địa phương;
        Begin
        ....;
        End;
BEGIN
....;
END.
```

Phạm Ví dụ sử dụng của biến địa phương là trong thân của chương trình con khai báo chúng và trong các chương trình con chứa trong chương trình con này. Thời gian tồn tại của chúng là từ khi chương trình con được gọi thực hiện cho đến khi thực hiện xong. Còn phạm Ví dụ sử dụng của biến toàn cục là trong toàn bộ chương trình (trong chương trình chính và trong tất cả các chương trình con của nó) và thời gian tồn tại của chúng là khi chương trình đang hoạt động.

6.2.4. Chú ý:

(1). Tên của các tham số hình thức và tên của các biến địa phươg trong cùng một chương trình con không được trùng nhau.

(2). Tên của các biến địa phương có thể trùng trên với tên biến toàn cục. Tuy nhiên khi chương trình con được gọi, nếu có sử dụng bến trung tên thì trong qúa trình chương trình con hoạt động, Turbo Pascal sẽ hiểu đó là biến địa phương, khi đó biến toàn cục tạm thời bị che dấu cho đến khi chương trình con hoạt động xong.

(3). Biến toàn cục có thể tham gia trong các chương trình con, mọi tác động ảnh hưởng đến nó vẫn giống như khi chúng tham gia trong chương trình chính. Ngược lại, biến địa phương chỉ tham gia trong chương trình con khai báo chúng, không thể tham gia trong chương trình chính.

<u>Ví dụ 1:</u>

```
        Var a, b, c: Word;
        PROCEDURE P1(m: Word; Var n: Word);
                Begin
                        If m>n Then m: = m – n Else m: = n – m;
                        n: = 2 * m;
                End;
        BEGIN
                a: = 5; b: = 8; c: = 10;
                P1(b+c, a);
                Writeln ('a=',a , 'b=', b, 'c=', c);
        END.
```

Các bước thực hiện của chương trình được giải thích như sau:

{1} Gán giá trị cho các biến toàn cục a, b, c.

{2} Thực hiện lời gọi thủ tục P1, vì m=b+c=8+10=18 và n=a=5 nên m>n do đó phép gán m:=m-n, tức là m:=13 được thực hiện. Sau đó phép gán n:=2*m được thực hiện, tức là n=26. Sau khi kết thúc, vì a được truyền theo kiểu tham biến cho tham số hình thức n nên a lưu giữ giá trị của lần thay đổi sau cùng là 26.

Như vậy, khi thực hiện chương trình trên sẽ cho ta kết quả in lên màn hình như sau:

a = 26 b = 8 c = 10

Ví dụ 2.

```
USES CRT;
Var a, b, c: Word;
PROCEDURE P1(m: Word; Var n: Word);
        Begin
                If m>n Then m: = m –n Else m: = n –m;
                n: = 2 * m;
                a: = m+n; {phép gán này ở Ví dụ 1 không có}
        End;
BEGIN
        a : = 5; b: = 8; c: = 10;
        P1(b+c, a);
        Writeln (‘a=’, a, ‘b=’, b, ‘c=’, c);
END.
```

Như vậy, khi thực hiện của trên sẽ cho ta kết quả in lên màn hình như sau:

a = 39 b = 8 c = 10

Vấn đề khác ở trên là do có tác động phép gán a: = m+n, tức là a = 13+26 = 39. Vì a là biến toàn cục (không trùng với tên tham số hình thức và tên biến địa phương nào) nên mọi tác động thay đổi giá trị trên nó có ý nghĩa.

Ví dụ 3:

```
USES CRT;
Var a, b, c: Word;
PROCEDURE P1(m, n: Word);
        Var a: Word;
        Begin
                If m>n Then a: = m Else a: = n;
                Writeln (‘a=’, a, ‘b=’, b, ‘c=’, c);
        End;

BEGIN
        a: = 5; b: = 8; c: = 10;
        P1(b,c);
        Writeln (‘a=’, a, ‘b=’, b, ‘c=’, c);
END.
```

Các bước thực hiện c chương trình được giải thích như sau:

{1} Gán giá trị cho các b toàn cục a, b, c.

{2} Thực hiện lời gọi thủ tục P1 với m=b=8 và n=c=10. Trong thủ tục này có biến địa phương a trung tên với biến toàn cục. Vì m>n có giá trị False nên phép gán a: = n được thực hiện, tức là n: = 10 được thực hiện (chú ý ở đây là biến địa phương). Sau đó viết ra màn hình giá trị của a (biến địa phương) và b, c (biến toàn cục).

Như vậy, khi thực hiện c trên sẽ cho ta kết quả in lên màn hình như sau:

a = 10 b = 8 c = 10

a = 5 b = 8 c = 10

Trong đó dòng đầu viết bởi câu lệnh Writeln trong thủ tục P1 với a là b địa phương và dòng sau được viết bởi câu lệnh Writeln trong chương trình chính với a là biến toàn cục.

6.3. Hàm

6.3.1. Định nghĩa hàm:

Một hàm được định nghĩa theo cú pháp sau:

FUNCTION TênHàm [(Danh sách các tham số hình thức)]: <Kiểu>;

{đầu chương trình con hàm}

Các khai báo địa phương;

Định nghĩa các chương trình con của hàm;

BEGIN

Các câu lệnh xử lý; {thân của hàm}

END;

Trong đó:

- ***Tên Hàm*** là một định danh do người sử dụng đặt theo nguyên tắc đặt tên của Turbo Pascal.
- ***Danh sách các tham số hình thức*** giống như đối với thủ tục
- ***Kiểu*** là kiểu của giá trị trả về của hàm.

Ví dụ: FUNCTION UCLN (m, n: Word): Word;

Một hàm có thể không có tham số hình thức nào. Khi đó phần khai báo sau tên hàm sẽ không có cặp dấu ngoặc.

Ví dụ: FUNCTION SO_NGAU_NHIEN: Word;

- ***Các khai báo địa phương*** giống như đối với thủ tục/
- ***Các câu lệnh xử lý*** giống như đối với thủ tục. Tuy nhiên đối với thân của hàm bắt buộc phải có câu lệnh gán tên hàm bằng giá trị trả về của hàm, giá trị này có thể dưới dạng một biểu thức có kiểu là kiểu của giá trị trả về của hàm theo cú pháp:

TênHàm: = Biểu thức;

Trong đó kiểu của Biểu thức phải trùng với kiểu của giá trị trả về của hàm.

6.3.2. Lời gọi hàm:

Khi sử dụng hàm thì phải thực hiện lời gọi hàm. Thực hiện lời gọi hàm là để lấy giá trị trả về của hàm. Vì vậy lời gọi hàm phải được đặt trong một biểu thức có xử lý giá trị của nó. Khi thực hiện lời gọi hàm thì cũng phải truyền tham số thực thế cho nó giống như đối với thủ tục.

Ví dụ:

```
USES CRT;
Var m, n, a: Word;
FUNCTION SO_NGAU_NHIEN: Word;
    BEGIN
        Rrandomize;
        SO_NGAU_NHIEN: = Random(100) + 1;
    END;

FUNCTION UCLN (m, n: Word): Word;
    Var a: Word;
    BEGIN
        If m>n then a: = n Else a: = m;
        While (m mod a <> 0) or (n mod a <> 0) Do a: = a – 1;
        UCLN: = a;
    END;
BEGIN
    m: = SO_NGAU_NHIEN; n: = SO_NGAU_NHIEN;
    Writeln('USCLN của', m, 'và', n, 'là:', UCLN (m, n));
END.
```

Chương trình trên gồm có hai hàm:

- Hàm SO_NGAU_NHIEN có giá trị trả về của hàm là lấy ngẫu nhiên một số nguyên dương bé hơn hoặc bằng 100.
- Hàm UCLN(m, n) có giá trị trả về của hàm là ước số chung lớn nhất của hai số nguyên dương m và n.

(độc giả tự tìm hiểu các giá trị đã sử dụng trong các hàm trên).

6.4. Chương trình con lồng nhau

Trong một chương trình con, ở phần các khai báo địa phương, ta có thể định nghĩa các chương trình con phục vụ riêng cho nó. Và việc này có thể tiếp tục được thực hiện theo nhiều mức của các chương trình con lồng nhau. Ta có thể minh hoạ như sau:

```
PROCEDURE P1(các tham số hình thức);
    FUNCTION F1(các tham số hình thức): Word;
        Begin
            {Thân của chương trình con F1}
        End;
    PROCEDURE P2(các tham số hình thức);
        PROCEDURE P3(các tham số hình thức);
            Begin
                {Thân của chương trình con P3}
            End;
      Begin
            {Thân của chương trình con P2}
      End;
Begin
    {Thân của chương trình con P1}
End;
```

Theo cấu trúc như trên, thì:

- F1, P2 là các chương trình con được định nghĩa trong chương trình con P1. Như vậy trong thân của chương trình con P1 có thể có lời gọi đến các chương trình con F1. P2 và dĩ nhiên không thể gọi chương trình con P3 được.
- P3 là chương trình con được định nghĩa trong chương trình con P2. Như vậy trong thân của chương trình con P2 có thể có lời gọi đến chương trình con P3.

Chú ý: Trong quá trình xây dựng CTC, khi nào thì nên dùng thủ tục/hàm?

Dùng hàm	**Dùng thủ tục**
- Kết quả của bài toán trả về ***1 giá trị duy nhất (kiểu vô hướng, kiểu string hoặc kiểu con trỏ).*** - Lời gọi CTC cần nằm trong các biểu thức tính toán.	- Kết quả của bài toán ***không trả về giá trị nào*** hoặc trả về ***nhiều giá trị*** hoặc ***trả về kiểu dữ liệu có cấu trúc (Array, Record, File).*** - Lời gọi CTC không nằm trong các biểu thức tính toán.

Ví dụ 1: Viết CTC để tính n! = 1.2...n.

Ý tưởng: Vì bài toán này trả về 1 giá trị duy nhất nên ta dùng hàm.

```
Function GiaiThua(n:Word):Word;
Var P, i:Word;
Begin
    P:=1;
    For i:=1 To n Do P:=P*i;
    GiaiThua:=P;
End;
```

Ví dụ 2: Viết chương trình con để tìm điểm đối xứng của điểm (x,y) qua gốc tọa độ.

Ý tưởng: Vì bài toán này trả về tọa độ điểm đối xứng (xx,yy) gồm 2 giá trị nên ta dùng thủ tục.

```
Procedure DoiXung(x,y:Integer; Var xx,yy:Integer);
Begin
    xx:=-x;
    yy:=-y;
End;
```

CHÚ Ý: Trong 2 ví dụ trên:

- **n, x, y** được gọi là **tham trị** (*không có từ khóa* **var** *đứng trước*) vì sau khi ra khỏi CTC giá trị của nó ***không bị thay đổi***.
- **xx, yy** được gọi là **tham biến** (*có từ khóa* **var** *đứng trước*) vì sau khi ra khỏi CTC giá trị của nó ***bị thay đổi***.

Chương 5. MICROSOFT POWERPOINT 2010

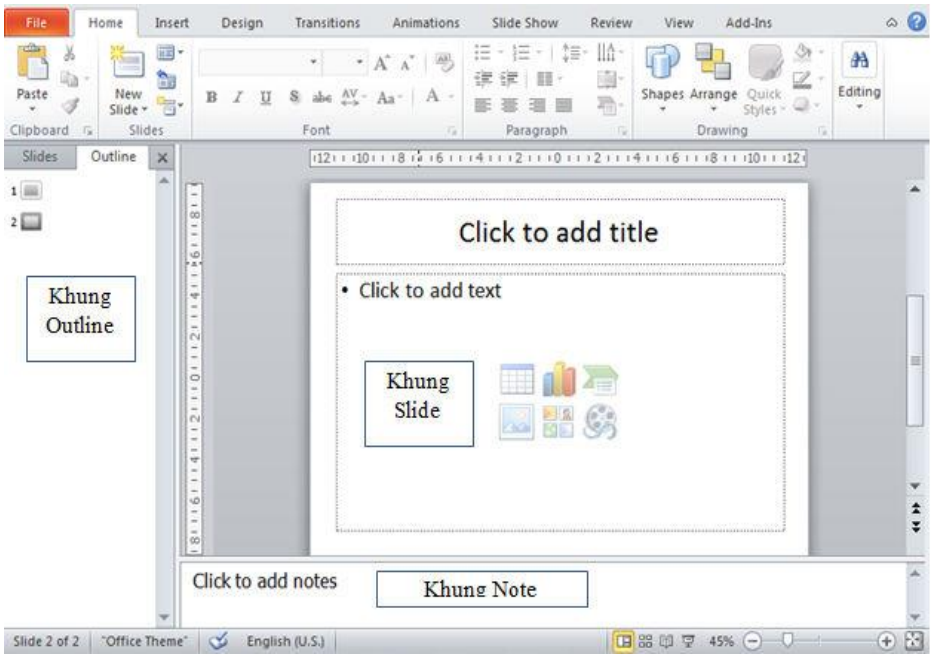
5.1. Giới thiệu Microsoft Powerpoint

PowerPoint là phần mềm tạo trình diễn (presentation) chuyên nghiệp có nhiều tiện ích giúp trình bày, thuyết trình các vấn đề trở nên sinh động và hiệu quả hơn, được sử dụng trong các báo cáo, hội nghị, giảng dạy....Phần này chỉ đề cập các tính năng thuộc riêng về PowerPoint, những điểm giống với Word và Excel học viên xem ở các chương trên.

Một file trình diễn PowerPoint 2010 được tạo từ nhiều trang, gọi là slide, có đuôi là pptx (có thể có đuôi là ppsx -PowerPoint Show dùng cho việc trình chiếu, không cho phép soạn thảo).

5.1.1. Giao diện chính

Khởi động chương trình Microsoft PowerPoint từ nhóm Mi- crosoft Office.



Màn hình PowerPoint có 3 khung chính: khung Slide là nơi soạn thảo trình diễn, khung Note là nơi nhập ghi chú của slide, và khung Slide/Outline dùng để liệt kê danh sách các Slide và soạn thảo outline.

5.1.2. Các chế độ View

PowerPoint có nhiều chế độ View khác nhau tùy mục đích sử dụng. Chế độ View được thay đổi từ thẻ Tab View-Group Presentations Views, chọn chế độ thiển thị tương ứng: Normal view, Slide Sorter view, Reading View, Slide Show.

Normal View: Chế độ soạn thảo Slide được sử dụng khi người dùng soạn thảo nội dung, note của từng slide của file trình diễn.

Slide Sorter: hiển thị nhiều Slide cùng lúc, không cho phép soạn thảo nội dung slide mà được dùng để thay đổi thứ tự, sắp xếp, dời (bằng cách drag-drop), chép, xóa các Slide. Kiểu này cho một cái nhìn tổng thể về toàn bộ slide.

Reading View: Hiển thị từng Slide ở dạng đọc tài liệu.

Slide Show: Hiển thị từng Slide ở chế độ trình chiếu, toàn màn hình với các hiệu ứng hoạt hình. Để thoát khỏi chế độ Slide Show, nhấn phím ESC.

5.1.3. Tạo file trình diễn mới

Trong PowerPoint, ta có thể tạo một trình diễn mới bằng cách dựa trên các khuôn mẫu (template) có sẵn, tuỳ theo chủ đề. Chọn Tab File-New xuất hiện màn hình tạo mới file từ nhiều mẫu (template) chia làm 2 nhóm: Home và Office.com. Nhóm Office.com Template chứa các Template được Microsoft cung cấp sẵn, máy tính phải có kết nối Internet để tải về.

Một Theme thiết kế trước cho một file trình chiếu về: màu chủ đề, font chữ, và hiệu ứng hoạt hình (effects). Người dùng có thể chọn một theme có sẵn, sau đó thiết lập lại từng phần về màu sắc, font chữ, và hiệu ứng hoạt hình. Tài liệu này chỉ tập trung ở nhóm Home, dựa trên các template đã có sẵn trong máy, bao gồm các mẫu như sau:

+Blank Presentaion: Tạo trình diễn trắng. Người dùng phải thiết kế mẫu và nhập nội dung mới cho file trình chiếu.

+Recent Templates: Những template vừa được sử dụng trước đó.

Sample Templates: Tạo trình diễn bằng một số mẫu có sẵn bao gồm cả nội dung, theo các chủ đề khác nhau. Người dùng chỉ việc chọn một mẫu trong nhóm Sample ứng với chủ đề cần tạo, sau đó chỉnh sửa lại nội dung.

Themes: Tạo trình diễn từ các mẫu được thiết kế Themes, không có nội dung. Đây là loại template được sử dụng nhiều nhất.

My Templates: Tạo trình diễn từ Template do người dùng định dạng trước.

New from exsiting: Tạo trình diễn từ file Template hoặc file PowerPoint có sẵn nào đó.

Lưu ý là người dùng có thể thay đổi lại theme khác bất cứ lúc nào, bằng cách chọn thẻ Design- Group Themes, chọn một mẫu thích hợp (có thể mở rộng danh sách Themes từ mũi tên phía dưới bên phải), sau đó có thể tùy chỉnh về màu, font và Effect của theme. Có thể áp dụng một theme cho toàn bộ slide, hay các slide đang chọn bằng cách R_click trên theme đó, và chọn Apply to All Slides hay Apply to Select- ed Slides.

5.1.4. Tạo Slide mới

Thêm Slide mới: Chọn thẻ Home-Group Slide-New Slide chọn Slide có layout thích hợp (từ danh sách các slide master có sẵn của file trình diễn) hoặc bấm phím Ctrl+N. Slide sau khi tạo xong có thể thay đổi Layout bằng cách chọn thẻ Tab Home-Group Slides-Layout và chọn mẫu có sẵn.

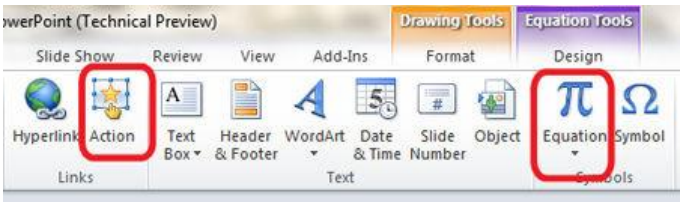
Xoá bỏ/Copy/Paste Slide: Chọn một hay nhiều Slide từ khung Slide/Outline, và thực hiện thao tác như các đối tượng khác. Ngoài ra người dùng có thể tạo bản Copy bằng cách R_Click lên Slide cần Copy chọn Duplicate Slide.

Sắp xếp lại các Slide: tốt nhất nên dùng chế độ View là Slide

Sorter, sau đó kéo thả các slide về vị trí mong muốn.

Như vậy, chúng ta lưu ý rằng: tạo file mới dựa trên các Design template, còn tạo silde mới dựa trên các Slide outline.

5.1.5. Soạn thảo nội dung Slide



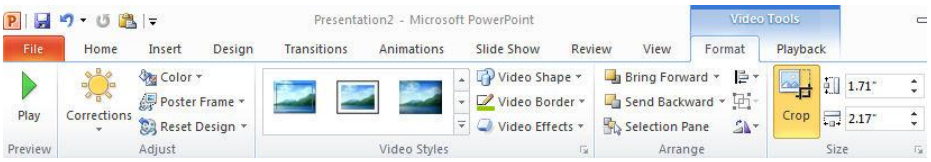
Trong PowerPoint, văn bản được đặt trong các Place holder hoặc các Textbox. Chú ý: các thao tác định dạng văn bản (font, đậm, nghiêng, gạch dưới, Bullets and Number- ing…) và chèn các đối tượng (table, biểu đồ, wordart, hình ảnh, header, footer…) được thực hiện tương tự như khi sử dụng MS Word. So với các phiên bản trước, PowerPoint 2010 bổ sung them chức năng Insert Equation để chèn và soạn thảo công thức toán như MS Word.

PowerPoint cho phép đổi thay đổi font chữ toàn bộ tài liệu với chứng năng Replace Font (được mở từ Tab Home-Group Edit- ing-Replace-Replace Font). Sau khi chọn font đang sử dụng cần thay thế (ở hộp chọn Replace) và font mới (ở hộp chọn With), bấm nút Re- place để thực hiện thay thế.

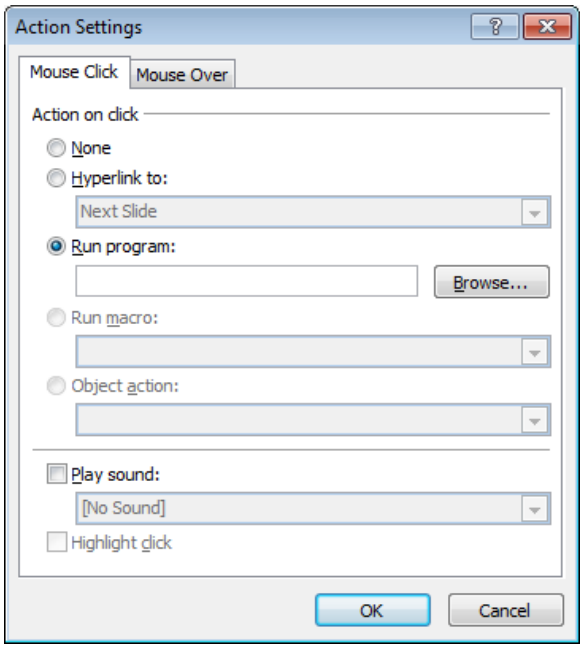
Chèn âm thanh/Video vào slide: thêm âm thanh/video giúp demo tốt hơn ý tưởng trình bày. Âm thanh/Video hiển thị trong slide dưới dạng biểu tượng và được phát khi click vào biểu tượng đó. Để chèn âm/video thành vào slide, chọn Tab Insert-Group Media- Au- dio/Video, chọn Audio from file (chèn từ file có sẵn), hoặc Clip Art Audio (để chèn từ thư viện của Of- fice).

Một cải tiến đáng giá trong PowerPoint 2010 là cho phép hiệu chỉnh Video đã insert vào tài liệu. Video có có thể chỉnh sửa màu sắc (Color, Coreections), đóng khung (video style), hiệu chỉnh hình dạng (video shape, video border, video effects….) rất nhanh và tiện lợi mà không cần dùng bất kỳ phần mềm xử lý video cồng kềnh nào khác. Ngoài ra PowerPoint 2010 cho phép chèn video từ các link

(như youtube, myspace) vào file trình chiếu.



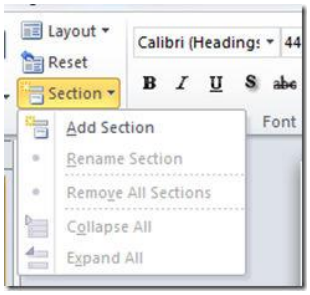
Chèn hành động (action) cho một đối tượng



Một action cho phép nhảy đến một slide khác, thực thi một chương trình khác, mở một trang Web, … khi người dùng bằng cách click chuột hoặc đưa trỏ chuột "đi" ngang qua đối tượng được chọn. Sau khi chọn đối tượng muốn chèn action, từ thẻ In- sert-Group Links- Ac- tion xuất hiện hộp thoại Action Settings.

Thẻ Mouse Click: hành động thi hành khi Click chuột lên đối tượng. Thẻ Mouse over: Chọn hành động khi đưa chuột ngang qua đối tượng. Ứng với mỗi hành động xảy ra, chúng ta có thể chọn: Hy- perlink to (nhảy đến Slide khác), Run program (khởi động một chương trình ứng dụng khác), Play sound (phát âm thanh tùy chọn).

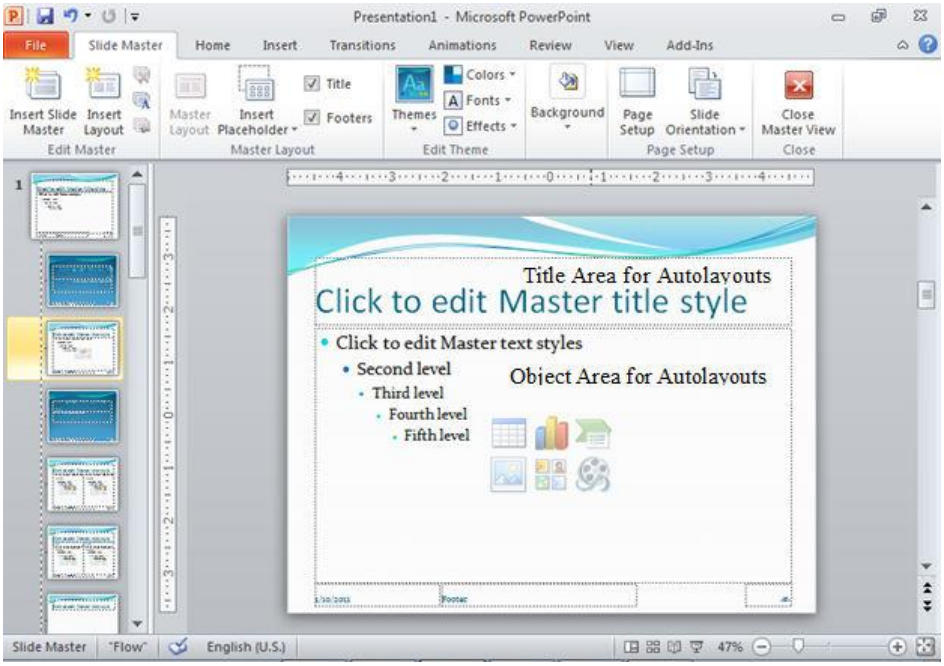
Section: Khi soạn thảo một file với hàng trăm slide ở các phiên bản PowerPoint cũ là rất bất tiện khi không thể gom nhóm cho các slide. PowerPoint 2010 cung cấp một tính năng mới – section – giúp tổ chức các slide vào các nhóm gọi là section. Chúng ta có thể Insert them sec- tion mới (từ thẻ Home->Slides->nút Section), đổi tên section đó (dùng context menu), xóa section và các slide thuộc section, kéo thả các slide qua lại giữa các section giúp tổ chức tài liệu một cách khoa học hơn.



5.2. Slide Master

Slide Master là nơi qui định tất cả các mẫu định dạng của các Slide trong bản trình bày. Khi thay đổi các thuộc tính của Slide Master thì những thay đổi đó sẽ có hiệu lực với tất cả các Slide trong bản trình diễn. Việc sử dụng tốt Slide Master giúp tạo ra một bản trình diễn có tính nhất quán cao về mặt định dạng, giảm kích thước file trình chiếu. Lấy ví dụ, nếu để logo trường ĐH Công Nghiệp Tp.HCM ở slide master, thì hang trăm slide của file sẽ chứa logo này, thay vì phải chép hàng trăm hình vào các slide.

Để mở chế độ hiệu chỉnh slide Master, chọn thẻ View-Group Master Views-Slide Master. Trong màn hình soạn thảo Slide Master, có nhiều mẫu slide khác nhau trong file trình diễn. Người dùng chọn từng mẫu slide, hiệu chỉnh khuôn dạng để thay đổi mẫu cho tất cả slide của file dựa trên mẫu này. Người dùng có thể thêm Slide Master. Định dạng các thuộc tính của Slide Master giống như định dạng một Slide thông thường.



Về cơ bản, khi thay đổi một Slide Master, có các thành phần như sau: Title Area for Autolayouts: vùng áp dụng cho tiêu đề slide; Object Area for Autolayouts: vùng áp dụng cho danh sách đánh dấu đầu mục, ở vùng này có thể thiết lập cho từng level text khác nhau; Date Area, Foot- er Area, Number Area: thay đổi các thuộc tính cho ngày giờ, tiêu đề đầu và cuối trang, số trang…

Sau khi hiệu chỉnh xong phải tắt chế độ soạn thảo Slide Master, quay về chế độ soạn slide bằng cách bấm nút Close Master View.

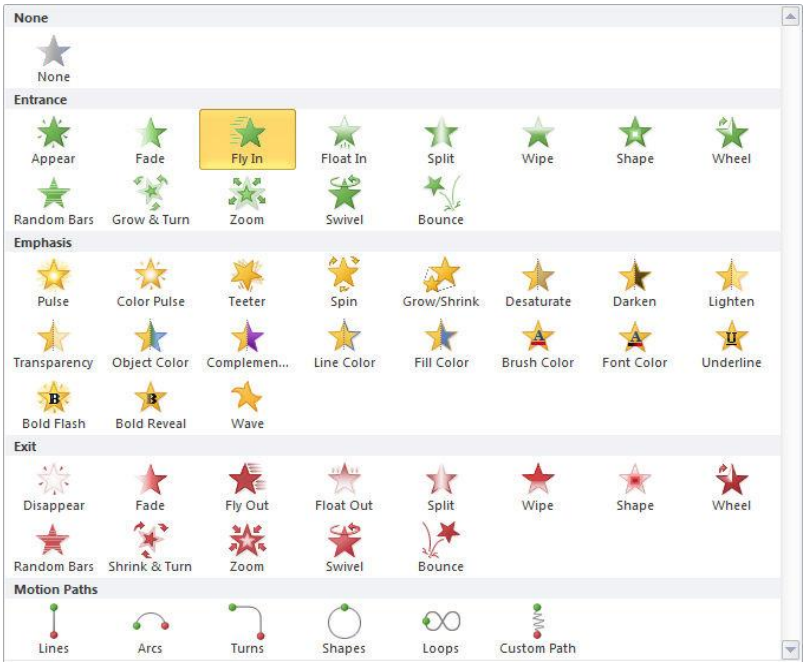
5.3. Animation

Để phần trình diễn trở nên sinh động, PowerPoint cung cấp nhiều hiệu ứng hoạt hình (animation) để minh họa ý tưởng một cách trực quan dễ hiểu, hoặc gây sự chú ý.

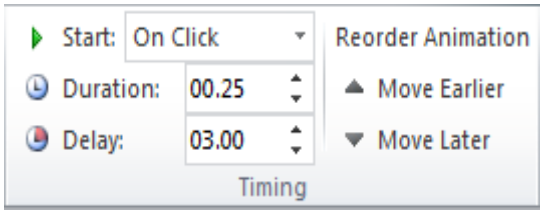
Tạo animation

Một đối tượng trong slide (textbox, table, picture…) có thể gán nhiều animation khác nhau, bằng cách chọn đối tượng đó, chọn thẻ An-imations- Group Animation, và chọn animation thích hợp (có thể mở rộng để chọn), hoặc có thể add animation từ group Advanced Animation.

Animation được chia làm 4 nhóm: Entrance (Cách thể hiện hiệu ứng khi đối tượng bắt đầu xuất hiện); Emphasis (Cách thể hiện hiệu ứng khi đối tượng đã cố định, thường dùng để nhấn mạnh nội dung cần biểu diễn); Exit: (Hiệu ứng thể hiện khi kết thúc Slide); Motion Paths: Cách thể hiện hiệu ứng di chuyển.



Danh sách các animation hiển thị trong cửa sổ Animation Pane (có thể bật hay tắt từ group Advanced Animation), và các tùy chỉnh hiển thị ở Group Timing. Ứng với mỗi animation, chúng ta có thể tùy chỉnh điều kiện để animation kích hoạt ở mục Start: Start On Click (bắt đầu animation khi click chuột), Start With Previous: bắt đầu song song với animation trước nó trong Animation Pane, Start After Previous(bắt đầu sau khi animation đứng trước nó đã hoàn tất).



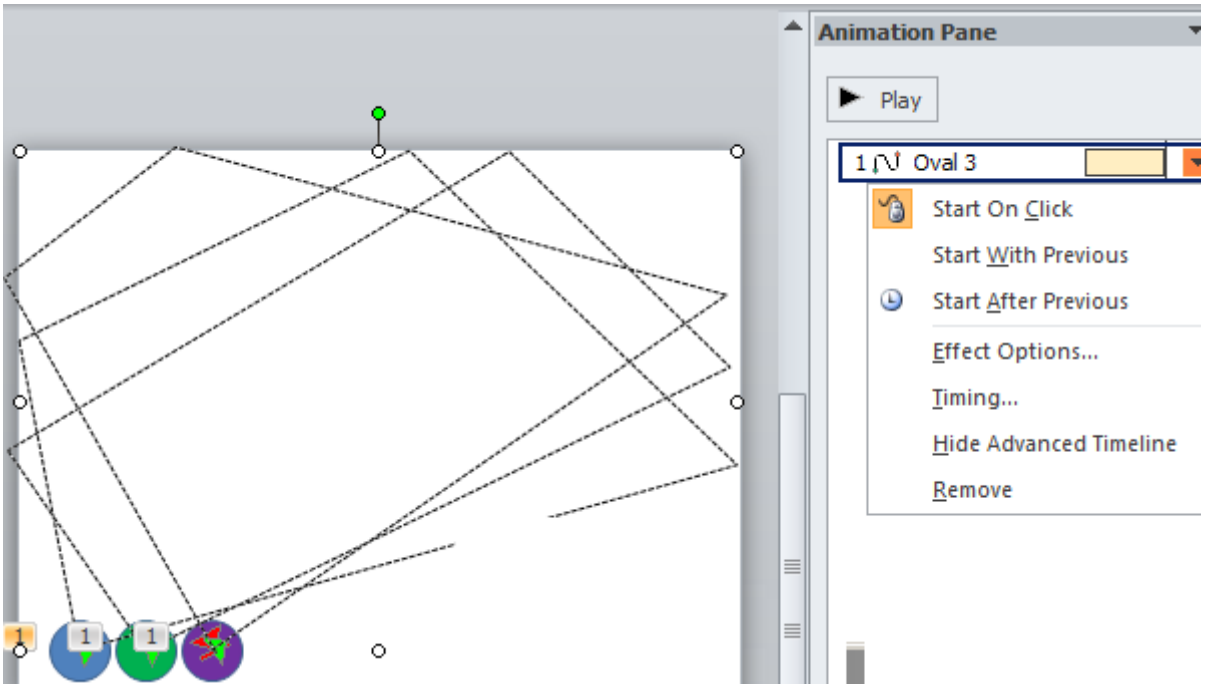
Ngoài ra, một animation còn có thể tùy chỉnh Duration (thời gian thực hiện hiệu ứng) và Delay (thời gian chờ cho hiệu ứng).

Từ cửa sổ Animation Pane cho phép xóa animation, hiệu chỉnh lại animation, thay đổi thứ tự animation bằng cách kéo-thả.

Ví dụ tạo animation

Trong ví dụ này chúng ta sẽ tạo một slide chứa 3 trái banh di chuyển xung quanh màn hình. Sử dụng chức năng Inser-Shape để tạo 3 trái banh có màu xanh (b1), đỏ (b2), vàng (b3).

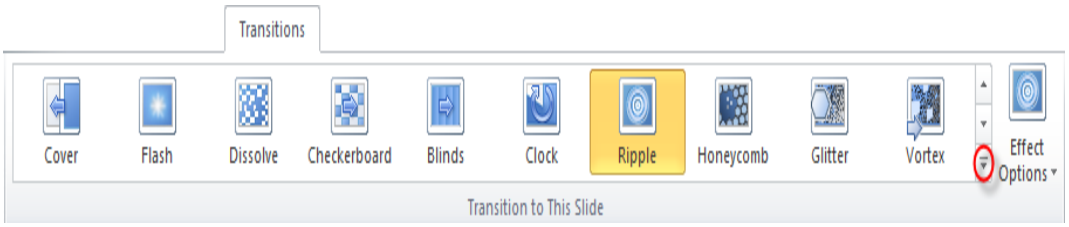
Chọn b1, chọn animation, chọn Custom Path, click chuột để vẽ 4 điểm dội của banh, double click để kết thúc. Tương tự, thêm animation cho b2 và b3. Trên Animation Pane sẽ xuất hiện 3 hàng. Chọn hàng đầu tiên, và để mặc định là Start On Click, chọn repeat là Until End of Slide (ở mục Effect Options- Timing), hàng thứ 2 và chọn Start After Previous, hàng thứ 3 và chọn là Start With Previous. Repeat tất cả đều là repeat là Until End of Slide. Bấm F5 để xem thử kết quả của bạn.



5.4. Slide Transition

Silde Transition là chức năng tạo hiệu ứng chuyển tiếp giữa các Slide trong chế độ Slide Show. PowerPoint 2010 cung cấp thêm rất nhiều hiệu ứng transition mới, đặt biệt là nhóm Exciting với các hiệu ứng 3D rất đẹp mà các phiên bản trước không có.

Để chọn Slide Transition, từ thẻ Transitions-Group Transition to This Slide và chọn hiệu ứng.

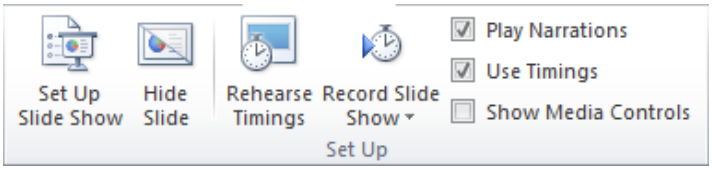


Với mỗi hiệu ứng Slide transition, từ mục Effect Options cho phép chọn hướng xuất hiện của slide mới. Cũng như Animation, Slide Transition cho phép hiệu chỉnh các thiết lập ở nhóm Timing, và cho phép áp dụng transition vào một nhóm hay toàn bộ slide.

5.5. Trình chiếu

Sau khi soạn thảo các Slide, bước cuối cùng là thực hiện trình chiếu slide trước thính giả..

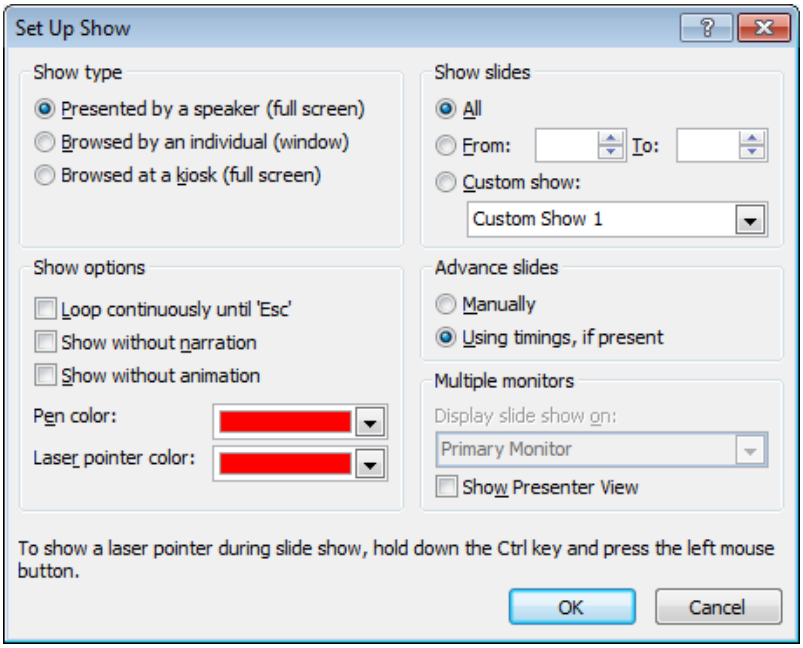
Thiết lập trước khi trình chiếu



Chọn thẻ Slide Show-Group Set Up.

Mục Set Up Slide Show: mở cửa sổ thiết lập trình chiếu với các tùy chọn.

Show type: Cách thực hiện trình diễn, bao gồm: Presented by a speaker (full screen): Cho phép thực hiện trình diễn ở chế độ toàn màn hình; Browsed by an individual (Windows): Cho phép trình diễn với thanh cuộn khi slide thiết kế ở độ phân giải cao khiến việc hiển thị không đầy đủ trang, Browsed at a kiosk (full screen): Cho phép thực hiện trình diễn ở chế độ toàn màn hình, diễn ra một cách tự động, và được lặp đi lặp lại.

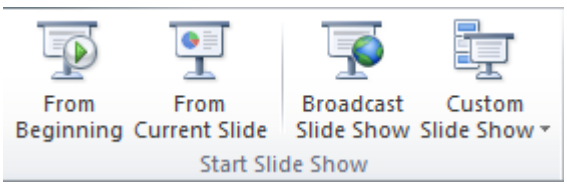


Show options: với các chọn lựa: Loop continuously until 'Esc': Cho phép trình diễn lặp đi lặp lại liên tục đến khi nhấn phím Esc, tùy chọn này là mặc định khi sử dụng chế độ Browsed at a kiosk; Show without animation: tắt các hiệu ứng.

Show Slides: Chọn các Slide cần trình diễn: All (trình diễn tất cả các Slide); From … To … (trình diễn một nhóm Slide liên tục);

Advanced Slides: Các tùy chọn nâng cao, bao gồm: Manually (chế độ chuyển trang thủ công - nhấn phím hoặc Click chuột để qua Slide khác); Using timings, if present: Chế độ chuyển trang tự động (sau một khoảng thời gian nào đó).

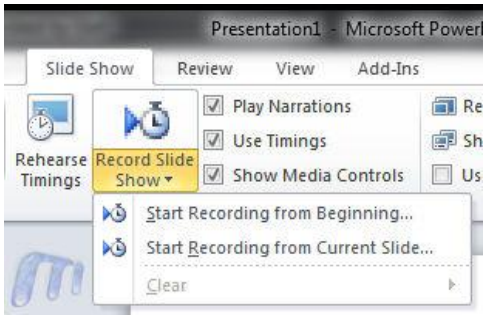
Thực hiện trình chiếu



Chọn thẻ Slide Show-Group Start Slide Show, với các tùy chọn như sau: From Beginning (trình chiếu từ slide đầu);

From Current Slide (Bắt đầu trình

diễn từ slide hiện hành); Broadcast Slide Show (Thực hiện trình chiếu qua Internet, đây là chức năng mới của PowerPoint 2010); Custom Slide Show (tạo trình chiếu từ việc chọn lựa một số slide nào đó thay vì chiếu tuần tự toàn bộ).



PowerPoint 2010 cho phép chụp thành phim (hoặc tại file pdf) khi trình chiếu Slide sử dụng menu Record Slide Show. Hoặc từ thẻ File->Save &Send -> Create Video (hoặc chọn menu Create pdf/xps document để tạo tài liệu pdf). PowerPoint 2010 tạo file wmv chứa toàn bộ quá trình trình chiếu tự động để người dung có thể ghi thành đĩa phim.

5.6. Một số lưu ý

Để việc trình chiếu thành công, nên lưu ý một số đặc điểm sau đây:

Không nên đọc slide, vì người nghe có thể tự đọc lấy. Người nghe quan tâm và thích thú với sự hiểu biết của người diễn thuyết về chủ đề hiện tại.

Không "trốn" sau các slide, mà nên nhìn thẳng vào người nghe. Người nghe muốn nhìn thấy người diễn thuyết như hai người đang nói chuyện với nhau vậy.

Các phim minh họa là các "slide" sống động nhất.

Sử dụng một hay hai font cho bài trình chiếu, không dùng quá nhiều font. Sử dụng font chữ rõ ràng và dễ đọc (font Roman và Gothic được đề nghị).

Font size không nên nhỏ hơn 22 pt. Các tiêu đề có font cỡ 28 pt. Không nên nhập quá nhiều text cho một slide.

Sử dụng chữ hoa cho các keyword muốn nhấn mạnh.

Chú ý màu nền của file trình diễn. Trên máy tính, màu nền đen sẫm với font chữ sáng thường khá đẹp, tuy nhiên khi trình chiếu trong môi trường sáng và khoảng cách xa thì rất khó nhìn rõ.

Chương 6 MẠNG MÁY TÍNH VÀ INTERRNET

BÀI 1: KHÁI NIỆM MẠNG MÁY TÍNH

I- Khái niệm mạng máy tính.

Mạng máy tính bao gồm nhiều thành phần, chúng được nối với nhau theo một cách thức nào đó và cùng sử dụng chung một ngôn ngữ:

+ Thiết bị đầu cuối (**End system**) kết nối với nhau tạo thành mạng, có thể là các máy tính, hoặc các thiết bị khác.

+ Môi trường truyền (**Media**) mà truyền thông được thực hiện qua đó. Môi trường truyền có thể là dây dẫn (cáp), sóng (đối với mạng không dây),…

+ Giao thức (**Protocol**) là quy tắc quy định cách thức trao đổi dữ liệu giữa các thực thể.

Tóm lại, **Mạng máy tính** là một tập hợp các máy tính và các thiết bị khác (các nút), chúng sử dụng một giao thức mạng chung để chia sẻ tài nguyên với nhau nhờ các phương tiện truyền thông mạng.

II- Phân loại mạng máy tính

Có rất nhiều mạng máy tính khác nhau. Việc phân loại chúng thường dựa trên các tiêu chí khác nhau như: theo vùng địa lý, theo topo gép nối mạng hay kiểu truyền thông mà mạng sử dụng…

1. Phân loại theo hoạt động

- Mạng cục bô (Local Area Network – LAN)

Liên kết tài nguyên máy tính trong một vùng hạn chế, đó có thể là một phòng, vài phòng trong một toà nhà. Phạm vi của mạng LAN chỉ nằm trong bán kính vài chục mét đến vài km.

Ví dụ một số mạng LAN: Ethenet/802.3, Token ring…

- Mạng diện rộng (WAN)

Liên kết các tài nguyên máy tính trong một phạm vi rộng (bán kính có thể đến 100km) như thị xã, tỉnh, thành phố…

Có thể coi mạng WAN là mạng gồm nhiều mạng LAN kết nối với nhau.

Ví dụ về mạng WAN: ISDN, Frame relay…

- Mạng toàn cầu (GAN): Mạng này là mạng của các mạng WAN trải rộng trên phạm vi toàn cầu.

Ví dụ mạng Inter net cũng là mạng GAN…

2. Phân loại the mô hình ghép nối.

- Mô hình điểm - điểm (Point to Point): một mạng Point to Point gồm các nút, một nút chỉ có thể liên lạc với một nút liền kề.

- Mô hình sao (Star): mạng được nối theo hình sao, có một HUB sử lý trung tâm –HUB này là trung tâm truyền tin cho tất cả các nút.

- Mô hình cây (Tree): mô hình phân cấp, nó bao gồm một nút gốc hay một HUB nối đến các nút mức hai hoặc HUB mức hai. Các thiết bị ở mức hai lại được nối đến mức 3…

- Mô hình điểm – nhiều điểm (Broadcast): mô hình này gồm nhiều các nút dùng chung một kênh chuyền thông.

3. Phân loại mạng theo kiểu chuyển

- Chuyển mạch ảo (Circuit-Swictched):
- Mạng chuyển gói (Packet-Switched network)

BÀI 2: INTERNET VÀ CÁC DỊCH VỤ TRÊN INTERNET

I. Giới thiệu Internet

Internet là hệ thống thông tin toàn cầu có thể được truy nhập công cộng gồm các mạng máy tính được liên kết với nhau. Hệ thống này truyền thông tin theo kiểu chuyển mạch gói dữ liệu (packet switching) dựa trên bộ giao thức đã được chuẩn hóa (giao thức TCP/IP). Hệ thống này bao gồm hàng ngàn mạng máy tính nhỏ hơn của các doanh nghiệp, của các viện nghiên cứu và các trường đại học, của người dùng cá nhân, và các chính phủ trên toàn cầu.

Tiền thân của mạng Internet ngày nay là mạng ARPANET do bộ quốc phòng Mỹ liên kết với một số trường ĐH của Mỹ thực hiện.

Hiện nay nhu cầu khai thác thông tin trên mạng Internet ngày càng tăng. Vì vậy trong chương này sẽ giới thiệu những khái niệm cơ bản về Internet, các dịch vụ trên Internet và cách sử dụng trình duyệt In- ternet Explorer để khai thác thông tin trên Internet.

Mạng Internet có thể chỉ dừng lại ở đấy nếu như không có sự ra đời của World Wide Web, được phát minh bởi Tim Berners-Lee trong thời gian làm việc tại CERN (Tổ chức nghiên cứu nguyên tử của Châu Âu đặt tại Thụy Sĩ). Ông Tim đã tìm ra cách thức để máy tính ở các vị trí, địa điểm khác nhau có thể hiển thị những văn bản có liên kết đến các tập tin văn bản khác. Kết quả nghiên cứu của ông Tim đó là ngôn ngữ đánh dấu siêu văn bản HTML (Hyper Text Markup Language).

Để hiểu nguyên lý hoạt động của Internet theo cách đơn giản nhất, ta có thể hình dung thông tin gởi và nhận từ các máy tính giống như việc gởi thư qua hệ thống bưu điện. Khi muốn gởi thư, người ta thường đến một trạm bưu điện gần nhất để bỏ thư vào thùng. Tại đây thư của chúng ta và của nhiều người khác nữa sẽ được phân loại theo địa chỉ rồi tiếp tục được gởi lên tuyến cao hơn. Quy trình cứ tiếp tục như thế cho đến khi thư của chúng ta tới được địa chỉ của người nhận.

Tương tự như vậy, khi nhận và gởi thông tin trên Internet, thông tin cần phải được xác định địa chỉ duy nhất. Địa chỉ Internet của các tư liệu được quản lý bằng bộ định vị tài nguyên đồng dạng URL (Uniform Resource Locator). Mỗi trang Web khi được đưa lên Internet sẽ có ít nhất một địa chỉ URL tham chiếu đến nó.

II. Một số khái niệm

Địa chỉ IP (Internet Protocol Address)

Khi tham gia vào Interntet, mỗi máy tính gọi là host, phải có một địa chỉ IP dùng để nhận dạng. Địa chỉ IPv4 được chia làm 4 số thập phân có giá trị từ 0-255, phân cách nhau bằng dấu chấm (ví dụ như:

172.16.19.5 hoặc 172.16.0.3). Để truy xuất trang chủ của ĐH Thái Bình thì gõ IP là 210.245.80.121 vào trình duyệt.

Cách đánh IP gồm 4 số thập phân như trình bày ở trên gọi là IPv4. Hiện nay do tốc độ phát triển quá nhanh của Internet, IPv4 (có chiều dài 32 bit) đã cạn kiệt. Internet đang chuyển đổi dần sang IPv6 (128 bit).

Hệ thống tên miền DNS (Domain Name System)

Đối với con người, cách truy xuất một website (như trang chủ trường ĐH Thái Bình nói trên) bằng IP như trên là rất khó nhớ. Do đó hệ thống tên miền DNS giúp ánh xạ giữa tên và IP giúp dễ dàng sử dụng hơn. Ví dụ, www.tbu.edu.vn dễ nhớ hơn nhiều so với IP là 220.231.93.18. Tên miền DNS do các DNS server đảm trách, do đó chúng ta phải trả phí hàng năm cho việc đăng ký tên miền.

Các phần com, edu hay vn trong địa chỉ tên miền ở trên được gọi là tên miền cấp 1, chia làm 2 nhóm chính:

- Domain mang tính tổ chức:

Domain	Tổ chức
com (Commercial)	Thương mại
edu (Educational)	Giáo dục
gov (Governmental)	Nhà nước
int (International)	Tổ chức quốc tế
mil (Military)	Quân đội
net (Networking)	Tài nguyên trên mạng
org (Organizational)	Các tổ chức khác

− Domain mang tính địa lý: Gồm 2 ký tự tắt đại diện cho từng quốc gia, như vn (Việt Nam), au (Úc), in (Ấn Độ), …

IAP (Internet Access Provider) là nhà cung cấp dịch vụ đường truyền để kết nối với Internet, quản lý cổng (gateway) nối với quốc tế. Các máy tính ở các quốc gia khác muốn kết nối với nhau phải có cơ sở hạ tầng, do đó phải nhờ đến IAP. Tại Việt Nam, IAP là công ty dịch vụ truyền số liệu VDC thuộc tổng công ty bưu chính viễn thông, cơ quan thực hiện trực tiếp là VNN.

ISP (Internet Service Provider) là nhà cung cấp dịch vụ Internet cho các tổ chức và cá nhân. ISP phải thuê đường truyền và cổng của một IAP. Việt Nam có nhiều ISP thương mại: Công ty dịch vụ truyền số liệu VDC, Công ty FPT, …

Khi đăng ký với một ISP, người dùng được cung cấp một tài khoản để quản lý truy cập và tính phí. Mỗi tài khoản bao gồm tên (User name) và mật khẩu đăng nhập (Password). Tài khoản này thường được lưu sẵn trong các thiết bị (như các modem) được cấu hình bởi ISP nên người dùng không sử dụng trực tiếp.

III. Các dịch vụ thông dụng trên Internet

Dịch vụ truy xuất từ xa (Remote Login)

Dịch vụ truy xuất từ xa cho phép nối kết, và sử dụng một máy tính ở vị trí khác. Ví dụ, Telnet là một dịch vụ cho phép remote login. Ngày nay, nhiều dịch vụ an toàn hơn đã ra đời thay thế cho telnet như ssh, vpn…

Dịch vụ thư điện tử (Mail Service)

Để có thể gửi hoặc nhận các thư điện tử (Electronic Mail - Email) từ bất cứ một nơi nào với điều kiện là người nhận và người gửi phải có một địa chỉ Email và máy tính sử dụng có nối mạng Internet.

Dịch vụ tin điện tử (News)

Cho phép trao đổi các bài báo và bản tin điện tử (bulletin) xuyên suốt Internet. Các bài báo được đặt tại trung tâm cơ sở dữ liệu trên suốt không gian Internet. Người dùng truy cập vào cơ sở dữ liệu đó để lấy về các bài báo họ cần. Điều này làm giảm tắc nghẽn trên mạng và loại trừ việc lưu trữ nhiều bản sao không cần thiết của mỗi bài báo trên từng hệ thống của người dùng.

Dịch vụ truyền tập tin (FTP: File Transfer Protocol)

FTP là một hệ thống chính yếu để chuyển tải file giữa các máy vi tính vào Internet. File được chuyển tải có dung lượng rất lớn. FTP hầu hết được sử dụng cho việc chuyển tải những dữ liệu mang tính cá nhân.

Thông thường, bạn sẽ dùng FPT để chép file trên một máy chủ từ xa vào máy của bạn, việc xử lý này gọi là nạp xuống (downloading). Tuy nhiên, bạn cũng có thể chuyển file từ máy của bạn vào máy chủ từ xa, điều này gọi là nạp lên (uploading). Thêm vào đó, nếu cần thiết, FTP sẽ cho phép bạn chép file từ một máy chủ từ xa tới một máy khác.

Dịch vụ Web (World Wide Web – WWW)

Đây là dịch vụ được dùng nhiều nhất. Người ta viết tắt là WWW hay gọi ngắn gọn là Web. Web chứa thông tin bao gồm văn bản, hình ảnh, âm thanh và thậm chí cả video được kết hợp với nhau. Web cho phép bạn có thể thâm nhập vào mọi ngõ ngách trên Internet, là những điểm chứa dữ liệu gọi là Web site. Nhờ có Web, nên dù không phải là một chuyên gia, bạn vẫn có thể sử dụng Internet.

Phần mềm sử dụng để nạp, dịch và hiển thị Web gọi là trình duy- ệt Web (Web browser). Hiện nay, trình duyệt thông dụng nhất là là In- ternet Explorer của Microsoft, ngoài ra còn có Mozilla Firefox, Chrome của Google, Opera, Safari của Apple …

IV.Trình duyệt Web IE 9.0

Để khởi động trình duyệt: Chọn Start/ All Programs/ Internet Explorer.

Các thành phần trong màn hình Internet Explorer

Các nút công cụ cho phép thực hiện các thao tác nhanh hơn. Chức năng của các nút trên thanh công cụ như sau:

− Back: Trở về trang trước đó.

− Forward: Hiển thị trang kế tiếp trong lịch sử duyệt.

− Stop: Ngưng tải trang web hiện hành từ máy chủ.

− Refresh: Tải lại nội dung trang Web hiện hành.

− Home: Hiển thị trang home, địa chỉ trang này được config theo mỗi máy khác nhau, là website tự động nạp mỗi khi khởi động IE.

−Favorites: Danh sách những trang Web ưa thích.

−New Tab: Thêm một Tab mới.

− Address: Dùng để nhập địa chỉ của trang Web muốn truy cập. Thanh địa chỉ cũng hiện lên địa chỉ của trang Web hiện hành. Bạn có thể Click vào hộp thả xuống để chọn địa chỉ của những trang Web thường truy cập.

− Current Tab: Tab hiện đang hiển thị.

− Tools: Chứa các nhóm công cụ của Internet Explorer.

− Trạng thái nạp : Dùng để hiển thị tiến trình nạp trang Web khi trang Web đang được nạp. Sau khi nạp xong biểu tượng nạp sẽ được chuyển thành biểu tượng Internet Explorer.

Lưu nội dung trang Web

Người dùng có thể lưu nội dung của các trang Web trên máy cục bộ, sau đó ta có thể xem nội dung của các trang này mà không cần phải có kết nối Internet. Sau khi truy cập đến trang Web cần lưu nội dung, chọn Tools-File-Save As, và chọn vị trí cùng với tên file cần lưu.

Lưu các địa chỉ thường dùng

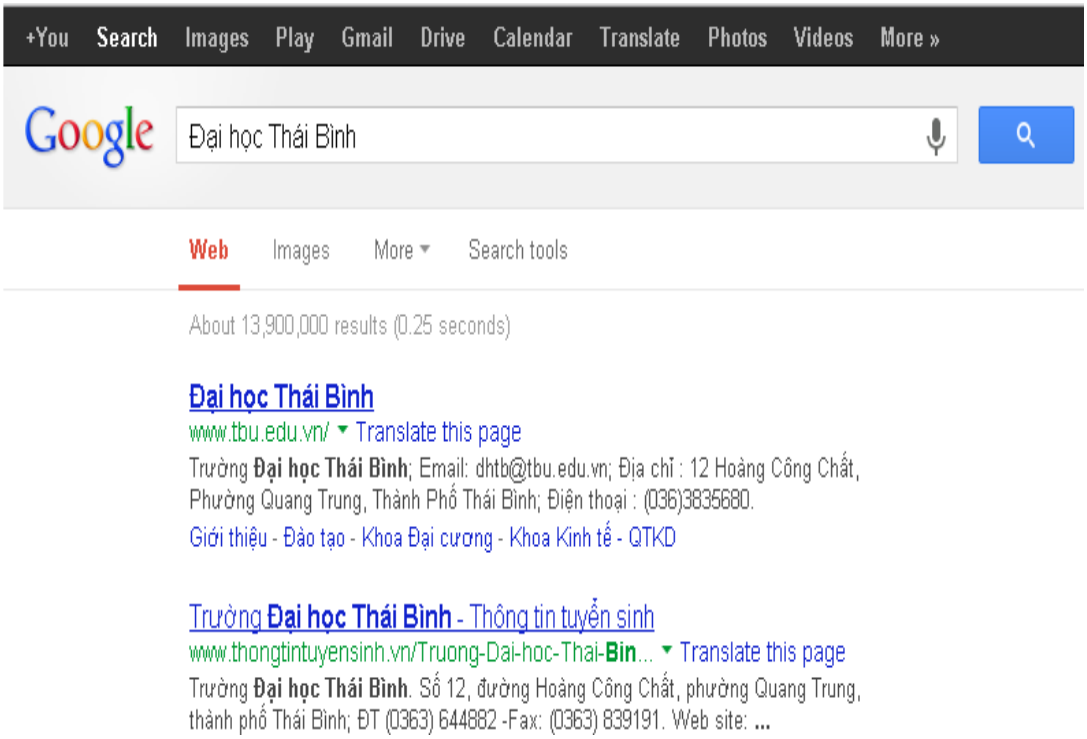
Chức năng này giúp người dùng lưu địa chỉ của các trang Web thường sử dụng, giúp người dùng không cần phải nhớ chỉ của nó. Sauk hi truy cập đến trang Web cần lưu địa chỉ, Chọn Favorites/ Add to Fa- vorites, nhập tên cho địa chỉ muốn lưu->OK.

V.Tìm kiếm thông tin trên Internet

Tìm kiếm thông tin là hoạt động phổ biến đối với người sử dụng Internet. So với thông tin được lưu trữ trên những phương tiện khác, thông tin được lưu trữ trên Internet phong phú và tìm kiếm dễ dàng hơn. Ngoài ra, kết quả tìm kiếm đạt được nhiều hơn so với việc tìm kiếm thông tin được lưu trữ trên các phương tiện khác. Đây là điểm mạnh nhưng đôi khi cũng là điểm yếu của Internet vì khi tìm được quá nhiều thông tin liên quan đến thông tin cần tìm, ta phải tốn thời gian để lọc lại những thông tin phù hợp.

Hiện tại có nhiều hãng cung cấp dịch vụ tìm kiếm khác nhau: google (www.google.com), microsoft (www.bing.com), yahoo (www.yahoo.com)… Mỗi hãng có những điểm mạnh và điểm yếu riêng, tuy nhiên dẫn đầu hiện nay là www.google.com.

Nhập thông tin muốn tìm vào hộp văn bản rồi nhấn Enter hoặc Click vào nút lệnh Google Search. Thông tin tìm kiếm thường là các từ khóa (Keyword) hoặc một cụm từ đặc trưng nhất. Việc chọn từ khóa chính xác cho giúp cho việc tìm kiếm dễ dàng, nhanh chóng.



Chú ý:

- Đặt dấu " trước từ khóa để yêu cầu tìm kiếm chính xác cụm từ cần tìm (liên kết AND).
- Không nên nhập vào những từ khóa có nội dung tổng quát vì kết quả tìm kiếm sẽ rất nhiều, thông tin được trả về sẽ không gần với nội dung muốn tìm.

Tìm kiếm nâng cao với Google

Trong trường hợp muốn có kết quả tìm kiếm gần với thông tin đang tìm, ta sử dụng chức năng tìm kiếm nâng cao của Google (Ad- vanced Search).

- Find results: Đặt điều kiện cho kết quả tìm kiếm:
- With all of the Words: Kết quả tìm kiếm phải bao gồm tất cả các từ cần tìm
- With the exact phase: Kết quả tìm kiếm phải có chính xác cụm từ cần tìm.
- With at least one of the Word: Kết quả tìm kiếm phải có ít nhất một trong những từ cần tìm
- Without the Word: Kết quả tìm kiếm không chứa những từ này

Hiển thị kết quả trả về: chọn số kết quả trả về từ hộp kê thả, mặc nhiên sẽ trả về 10 kết quả nếu tìm được từ 10 trang Web trở lên.

Ngoài ra ta có thể đặt thêm điều kiện vào những tùy chọn khác như ngôn ngữ, thời gian, định dạng, …

VI. Dịch vụ Email

Giới thiệu

Thư điện tử (Email) là phương tiện liên lạc vô cùng tiện lợi trong thời đại công nghệ thông tin ngày nay. Sử dụng Email ta có thể trao đổi thông tin với bạn bè, đồng nghiệp trên toàn cầu mà không quan tâm đến việc người nhận ở đâu.

Địa chỉ Email (Email Address), gọi tắt là email, là một định danh duy nhất trên Internet cho phép người sử dụng giao dịch với nhau. Cấu trúc một địa chỉ Email: <Tên tài khoản>@<Tên miền>. Tên miền: Tên của máy chủ cung cấp dịch vụ Email (Email Server), tên tài khoản là tên được đăng ký, để phân biệt với các địa chỉ Email khác có cùng tên miền.

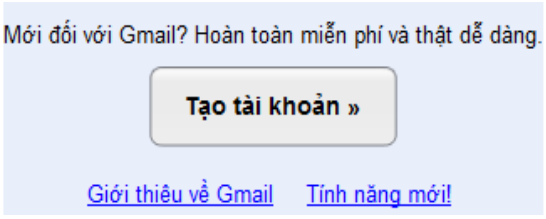
Ví dụ: ttcntt@yahoo.com; email@hui.edu.vn; list@hotmail.com; … Muốn gởi email thì người dùng phải có nối kết Internet (hoặc nối

kết vào Mail Server. Khi người A gởi Email cho người B, nội dung thư sẽ được lưu trữ tại Email Server. Chỉ duy nhất người B có thể đăng nhập vào tài khoản của mình và đọc tất cả Email gởi cho mình. Có nhiều cách để làm việc với email: sử dụng Webbrowser để sử dụng Webmail (nếu nhà cung cấp email có dịch vụ này), hoặc dùng công cụ khác.

Webmail là hệ thống cung cấp các dịch vụ Email thông qua một Website. Người dùng sử dụng Webbrowser truy cập vào địa chỉ Website của nhà cung cấp dịch vụ, sử dụng tài khoản Email đã được cung cấp để sử dụng các dịch vụ Email. Một số webmail thông dụng: www.gmail.com, www.hotmail.com, www.mail.yahoo.com... Cách sử dụng các dịch vụ email của các hang lớn gần như tương tự, do đó trong giáo trình này chỉ tập trung mô tả về dịch vụ email của Google. Google tự động hiển thị ngôn ngữ theo vùng truy xuất email, tức là tiếng Việt, và cung cấp nhiều dịch vụ khác dựa trên tài khoản email mà thế giới rất ưa chuộng.

Đăng ký tài khoản email

Để đăng ký một tài khoản Email mới: từ Webmail (chẳng hạn, từ trang www.gmail.com), chọn mục Tạo tài khoản.



Sau đó điền các thông tin vào biểu mẫu, và bấm nút tạo tài khoản.

Lưu ý là gmail yêu cầu bạn phải có số điện thoại di động để gởi mã xác nhận nhằm bảo vệ người dùng.

Bắt đầu sử dụng Gmail

Tên:

Họ:

Tên Đăng nhập mong muốn: @gmail.com

Ví dụ: JSmith, John.Smith

kiểm tra tính khả dụng!

Chọn mật khẩu: Độ mạnh của mật khẩu:

Dài tối thiểu 8 ký tự.

Nhập lại mật khẩu:

☑ Duy trì trạng thái đăng nhập

☑ Bật Lịch sử Web Tìm hiểu thêm

Câu hỏi Bảo mật: Chọn một câu hỏi ...

Nếu bạn quên mật khẩu của bạn chúng tôi sẽ yêu cầu bạn trả lờ
hiểu thêm

Trả lời:

Email phục hồi:

Địa chỉ này được sử dụng để xác thực tài khoản của bạn khi bạ
mật khẩu. Nếu bạn không có một địa chỉ email khác, bạn có thể
thêm

Vị trí: Việt Nam

Xác minh Từ: Nhập các ký tự bạn nhìn thấy trong bức hình dưới đây.

Google tài khoản

Xác minh tài khoản của bạn

Mã xác minh của bạn đã được gửi đến [redacted]

Nếu bạn không nhận được thư, hãy thử gửi lại.

Nhập mã của bạn

Xác minh

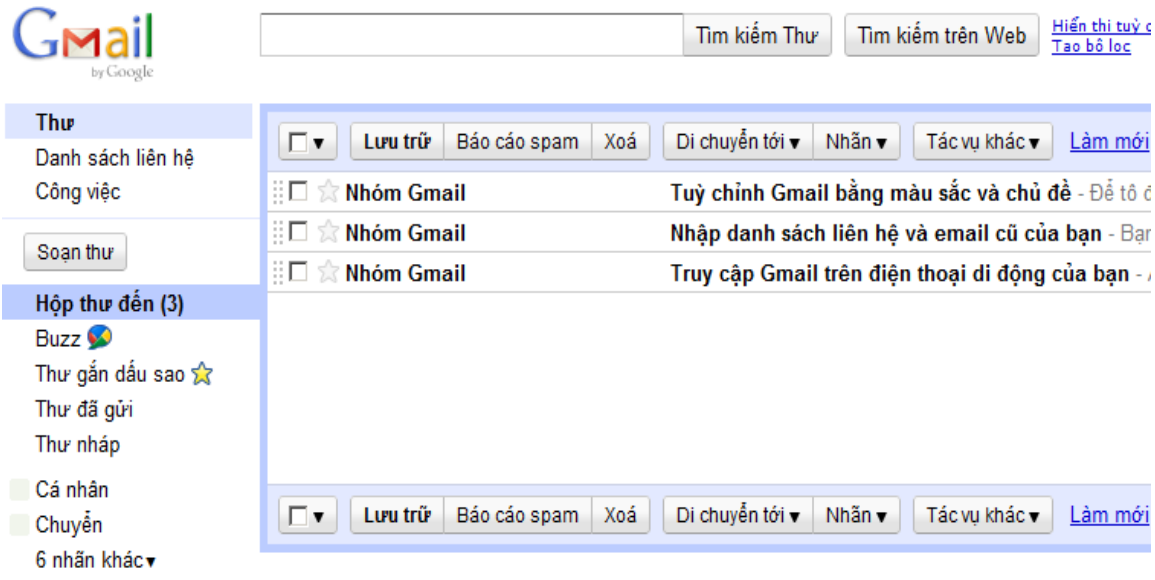
Sử dụng email với Webmail

Sau khi đã có tài khoản Email, trước khi sử dụng Email phải đăng nhập từ trang www.gmail.com.

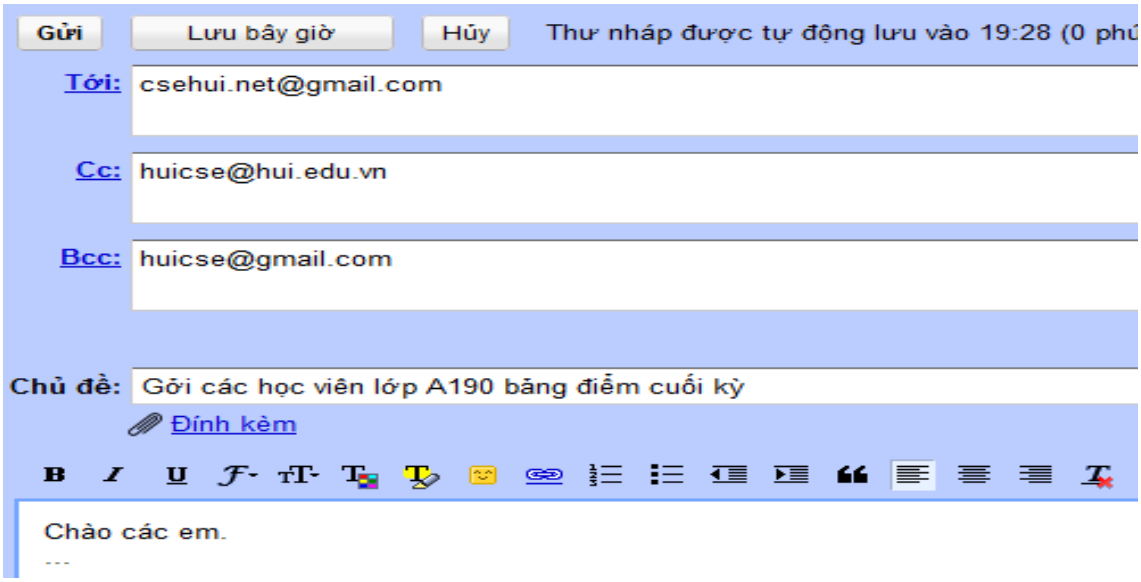
Các email nhận được sẽ hiển thị khi chọn mục Thư.

Để soạn Email mới, chọn nút Soạn thư. Nhập nội dung Email với các mục như sau: Tới (Email một hay nhiều người nhận -cách nhau bằng dấu ;); CC

(Email người đồng nhận); BCC (Email người đồng nhận nhưng không hiển thị cho biết cùng nhận Email với hai nhóm Email ở trên); Chủ đề (Subject - tiêu đề Email); Body (nội dung Email);



Để gửi file đính kèm Email, chọn nút Đính kèm. Sau đó chọn file cần gửi đi.



Chú ý: sau khi sử dụng Email xong, chúng ta phải bấm nút Sign- out để tránh việc mất tài khoản Email, bảo vệ thông tin riêng tư.

Sử dụng email với Outlook Express.

Outlook Express là phần mềm quản lý Email của hãng Microsoft Windows (tương tự với Thunderbird của Mozilla). Phần mềm này cung cấp hầu hết các tính năng cần thiết để người dùng có thể quản lý Email một cách hiệu quả. Outlook Express sẽ giúp giao tiếp với Email Server để chuyển-nhận Email với một số tính năng quản lý rất tiện dụng. Khởi động Outlook Express từ mục All Pro- grams- Outlook Express.

Để sử dụng email với Oulook, đầu tiên chúng ta phải khai báo tài khoản email cho Outlook. Từ menu Info, chọn Add Account.

Sau đó nhập thông tin email

Thanh công cụ New E-Mail: Soạn Email mới.

Reply: trả lời.

Email đang mở cho người gửi.

Forward: Gửi Email nhận được cho những người khác.