

TIN HỌC

MỘT TÀI LIỆU CỦA CLB TIN HỌC NHH

PHẦN 1: PASCAL

<IT/>
NHH IT CLUB



Mục lục

I. Chương trình máy tính	3
1. Chương trình máy tính là gì?	4
2. Tuân theo chỉ dẫn	4
3. Chương trình đầu tiên của chúng ta	4
II. Làm quen với chương trình và Ngôn ngữ lập.....	8
1. Ví dụ về 1 chương trình Pascal cơ bản	8
2. Từ khóa và tên	8
3. Cấu trúc một chương trình Pascal	9
III. Phần mềm Free Pascal.....	10
1. Các chức năng quan trọng.....	11
2. Soạn thảo chương trình	12
3. Một số phím tắt	14
4. Lưu ý	14
IV. Viết Pascal	15
Bài 1. Dữ liệu và kiểu dữ liệu	16
Bài 2. Giao tiếp giữa người và máy	20
Bài 3. Biến (Variable)	21
Bài 4. Câu lệnh điều kiện (Conditional Statements).....	24
Bài 5. Vòng lặp (Loop).....	28
Bài 6. Mảng một chiều (Array)	32
Bài 7. Hàm (Function) và Thủ tục (Procedure)	35

I.

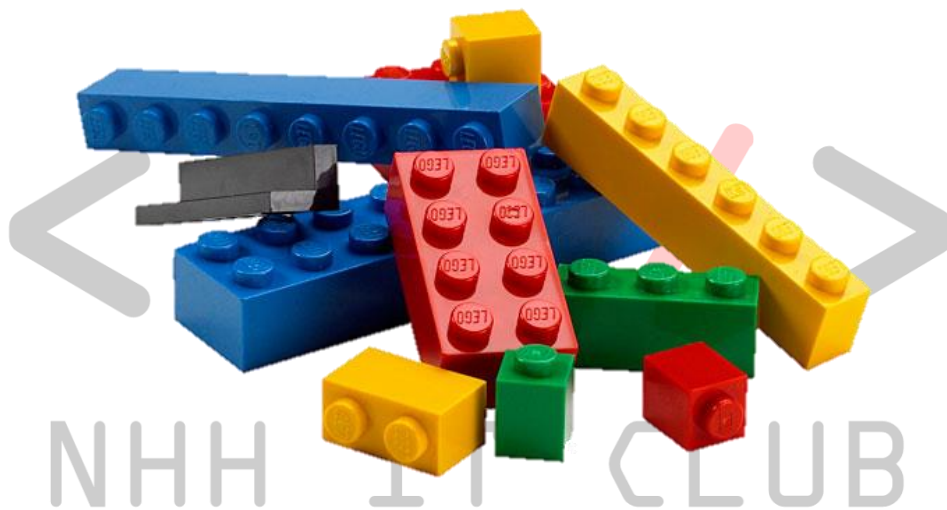
CHƯƠNG TRÌNH MÁY TÍNH

1. Chương trình máy tính là gì?

Hãy tưởng tượng xem bạn mua 1 chiếc máy tính và nó không thể làm gì được ngoài bật và tắt màn hình. Một chiếc máy tính mà không biết làm gì thì chỉ là một cục chặn giấy cực đắt tiền. Chương trình chính là thứ làm cho máy tính biết chơi trò chơi, tạo nhạc và vẽ hình. **Chương trình chỉ cho máy tính biết phải làm gì.**

2. Tuân theo chỉ dẫn

Một chương trình giống như 1 bảng danh sách các bước hướng dẫn để lắp ráp một đồng gạch nhựa đồ chơi thành 1 tòa lâu đài: là 1 tập hợp các chỉ thị riêng lẻ cho máy tính thực hiện. Mỗi chỉ thị riêng biệt là một bước rất nhỏ, như là nối 2 viên gạch với nhau, nhưng khi chương trình hoàn thiện sẽ tạo ra những điều tuyệt diệu.



Giống như 1 danh sách chỉ dẫn, 1 chương trình được đọc **từ trên xuống dưới**. Mỗi một dòng trong chương trình là 1 chỉ thị, yêu cầu máy tính thực hiện 1 hành động. Khi hành động đó đã hoàn thành, máy tính sẽ đi tới dòng tiếp theo trong chương trình, và cứ tiếp tục như thế cho đến dòng cuối cùng, chương trình sẽ dừng lại. Máy tính không buồn chán, cũng chẳng nhảy các dòng: nó tiếp tục chạy theo chỉ dẫn bất kể tốn bao nhiêu thời gian.

3. Chương trình đầu tiên của chúng ta

Các chương trình máy tính còn được gọi là mã lệnh, và 1 người viết chương trình máy tính còn được gọi là 1 lập trình viên. Đó có thể là một công việc, nhưng có rất nhiều người viết chương trình cho vui, kể cả trẻ em luôn :D. Khi đã viết 1 chương trình, bạn có thể yêu cầu máy tính thực hiện nó, tức là máy tính sẽ làm theo các chỉ thị và tiến hành các hành động.

Giờ hãy xem xét một đoạn chương trình sau:

```
writeln('Tôi vừa bắt đầu chương trình đầu tiên của mình!');  
writeln('Tôi vừa hoàn thành chương trình đầu tiên của mình!');
```

(IDE mà mình sử dụng là **Free Pascal** thật ra không hỗ trợ tiếng Việt, vì là ví dụ nên mình viết có dấu, mấy bài sau cũng có các ví dụ mình ghi có dấu :b);

Lệnh **writeln** (viết tắt của writeline í) yêu cầu máy tính in dòng văn bản (trên 1 dòng) ra màn hình.

Khi đoạn mã ở trên chạy, máy tính trước tiên sẽ in ra:

Tôi vừa bắt đầu chương trình đầu tiên của mình!

Tôi vừa hoàn thành chương trình đầu tiên của mình!

...bởi vì máy tính làm theo hướng dẫn của dòng lệnh đầu tiên và sau đó là dòng lệnh thứ 2. Và như vậy, bạn đã viết 1 chương trình máy tính rồi đó.

CÙNG BẮT ĐẦU NÀO!

Ghi nhớ

- Chương trình là tập hợp các câu lệnh chỉ cho máy tính biết phải làm gì.
- Giống như 1 danh sách chỉ dẫn, 1 chương trình được đọc từ trên xuống dưới.

pascal

II.

LÀM QUEN VỚI CHƯƠNG TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH PASCAL

Pascal

/ˈpæskəl/

“Pa-skal”

“Pát-xcao”

“A good designer must rely on experience, on precise, logic thinking;
and on pedantic exactness.”

- *Niklaus Emil Wirth*

1. Ví dụ về 1 chương trình Pascal cơ bản

```
program Chuong_Trinh_Dau_Tien;  
uses crt;  
begin  
    writeln('hello world');  
    readln;  
end.
```

“Program”: lệnh khai báo chương trình

“uses”: khai báo tên thư viện

“readln”: đọc chương trình

Hình trên minh họa 1 chương trình đơn giản được viết bằng pascal. Sau khi dịch, kết quả chạy chương trình là dòng chữ “hello world” được in ra màn hình

Chương trình chỉ có 6 dòng lệnh. Mỗi dòng lệnh mang những chỉ thị nhất định, tất cả kết hợp lại để tạo nên 1 chương trình hoàn thiện.

Trong các phần tiếp theo, chúng ta tìm hiểu xem các câu lệnh được viết như thế nào.

2. Từ khóa và tên

- Từ khóa (reserved keyword)

Trong chương trình trên, ta thấy có các từ như **program**, **uses**, **begin**, **end**,... đó là những từ khóa được quy định tùy theo mỗi ngôn ngữ lập trình. **Từ khóa** của 1 ngôn ngữ lập trình là những từ **dành riêng**, không được sử dụng các từ khóa này cho bất kì mục đích nào khác ngoài mục đích sử dụng do ngôn ngữ lập trình quy định.

Theo quy định, **program** là từ khóa dùng để khai báo tên chương trình, **uses** là từ khóa khai báo tên thư viện. Các từ khóa **begin** và **end** luôn đi thành cặp dùng để thông báo điểm bắt đầu và kết thúc phần thân của chương trình.

- Tên

Ngoài từ khóa, trong chương trình ở ví dụ trên còn thấy các từ như **Chuong_Trinh_Dau_Tien**, **crt**,... Đó là các **tên** được sử dụng trong chương trình. Ví dụ **Chuong_Trinh_Dau_Tien** là tên của chương trình. Còn **crt** là tên của thư viện.

Tên do người dùng đặt phải tuân thủ các quy tắc của ngôn ngữ lập trình cũng như phần mềm lập trình. Sau đây là 1 số quy tắc khi đặt tên ở Free Pascal:

- Tên có thể đặt tùy ý, nhưng để dễ sử dụng nên đặt tên sao cho *ngắn gọn, dễ nhớ và dễ hiểu*.
- Không được đặt tên bằng tiếng việt có dấu.
- Các biến khác nhau nên được đặt tên khác nhau.

- Tên không được trùng với các từ khóa.
- Không được đặt tên bắt đầu bằng số và không được chứa dấu cách.

3. Cấu trúc một chương trình Pascal

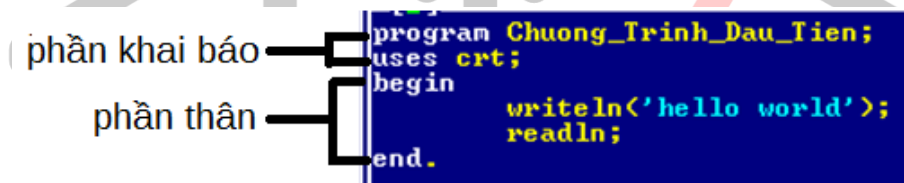
Cấu trúc chung của 1 chương trình gồm:

1. Phần khai báo thường gồm các câu lệnh dùng để:

- Khai báo tên chương trình.
- Khai báo các thư viện (chứa các lệnh viết sẵn có thể sử dụng trong chương trình) và một số khai báo khác.

2. Phần thân của chương trình gồm các câu lệnh mà máy tính thực hiện. Đây là phần bắt buộc phải có.

Phần khai báo có thể có hoặc không. Tuy nhiên, nếu có phần khai báo thì nó phải được nằm **trước** phần thân chương trình.



```
program Chuong_Trinh_Dau_Tien;
uses crt;
begin
    writeln('hello world');
    readln;
end.
```

Trở lại với ví dụ trên, ta có thể thấy:

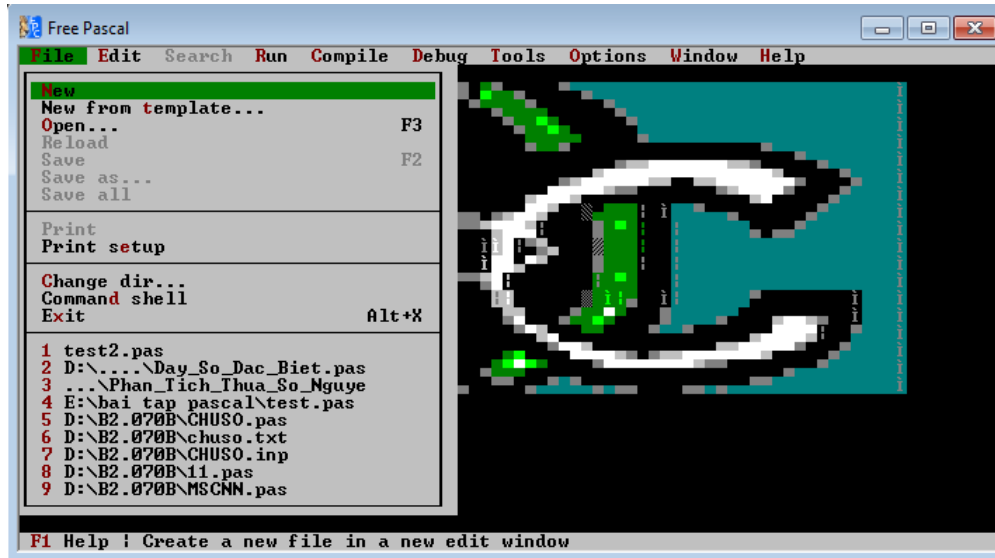
- Phần khai báo gồm 2 lệnh: khai báo tên chương trình là `Chuong_Trinh_Dau_Tien` với từ khóa `program` và khai báo thư viện `crt` với từ khóa `uses`.
- Phần thân gồm các từ khóa `begin` và `end` cho biết điểm bắt đầu và kết thúc phần thân, câu lệnh là `writeln('hello world');` để in ra màn hình dòng chữ "hello world".

III.

PHẦN MỀM FREE PASCAL

1. Các chức năng quan trọng

Ở màn hình chính của Free Pascal, các bạn chọn **File**



Dưới “Exit” là tên các chương trình đã mở gần đây.

“New”: tạo chương trình mới

“Open”: mở 1 chương trình có sẵn

“Save”: lưu chương trình

“Save as”: save chương trình như 1 file mới

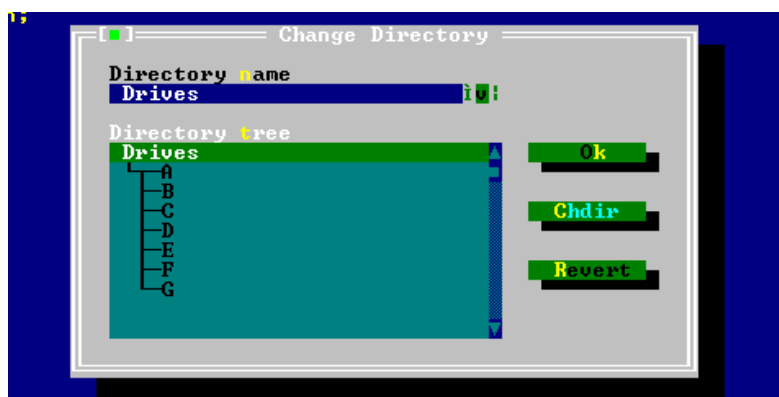
“Save all”: lưu tất cả chương trình đang mở

“Exit”: thoát khỏi Free Pascal

Địa chỉ lưu chương trình mặc định của Free Pascal là:

C:\FPC\3.0.2\bin\i386-win32. (có thể thay đổi theo từng máy)

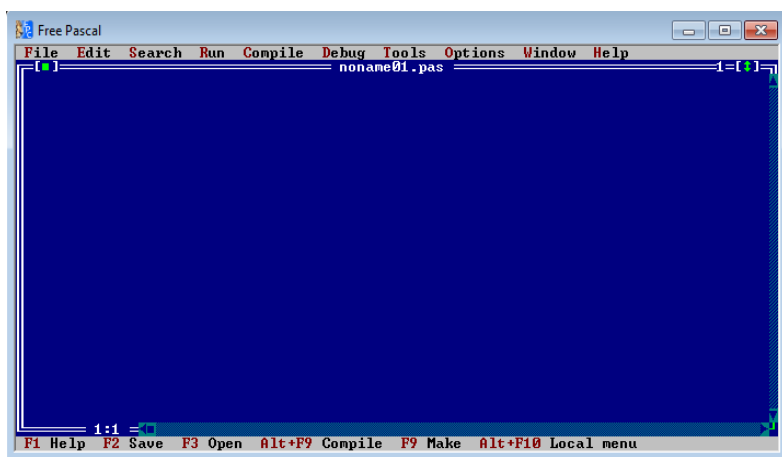
Change dir (change direction) giúp cho bạn có thể lưu chương trình ở thư mục mà mình mong muốn. Bạn chọn **Change dir**, ấn vào Drives và chọn ổ đĩa rồi tìm đến thư mục bạn muốn:




Lưu ý: Mỗi lần bạn tắt Free Pascal, chương trình sẽ tự động reset địa chỉ lưu chương trình về địa chỉ mặc định. Vì thế lời khuyên cho bạn là mỗi lần bật Free Pascal thì việc đầu tiên cần làm là **Change dir**.

2. Soạn thảo chương trình

Ở màn hình chính, bạn chọn **File**, rồi chọn **New**.



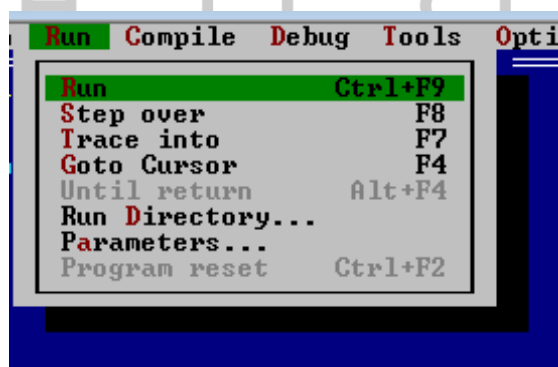
“noname01.pas”: tên của chương trình, ở đây bạn chưa đặt tên nên Free Pascal sẽ hiện là “noname01”.

 nút dấu chấm này dùng để tắt chương trình đang bật.

Màn hình soạn thảo mới

- Lưu và biên dịch chương trình

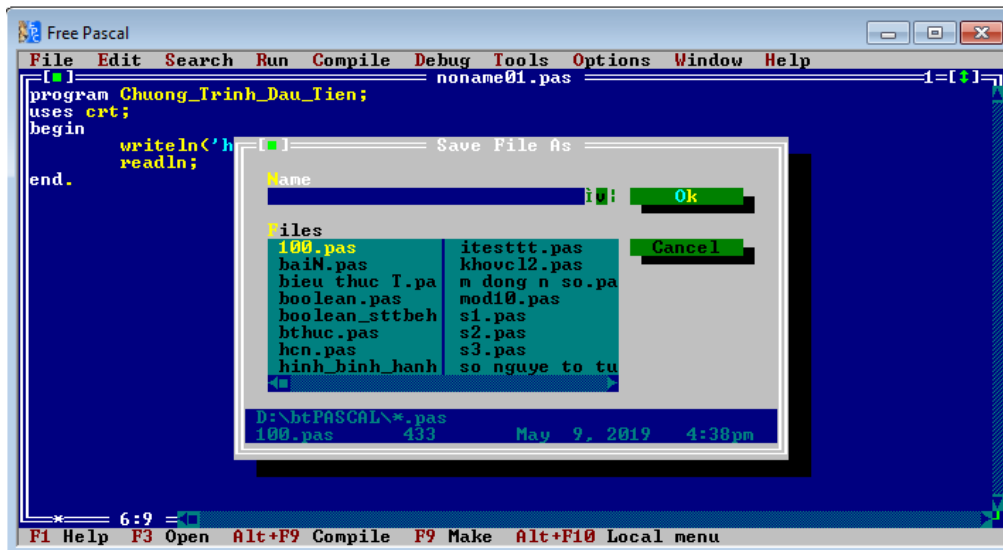
Sau khi hoàn thành chương trình, để chương trình chạy được, bước cuối cùng phải làm là lưu, kiểm tra lỗi và biên dịch chương trình:



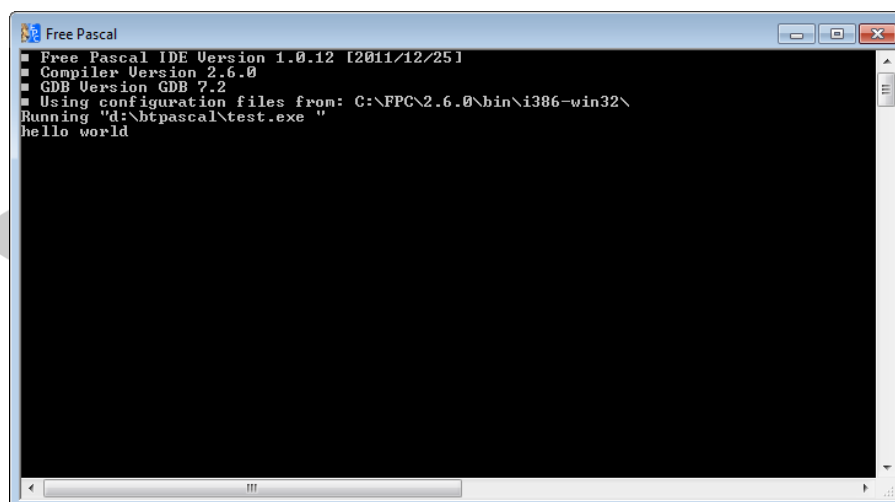
Các bạn chọn **Run**, rồi tiếp tục chọn **Run**, hoặc đơn giản là ấn tổ hợp phím **Ctrl+F9**, sau đó màn hình sẽ hiện lên:

“**Name**”: Tên đặt cho chương trình

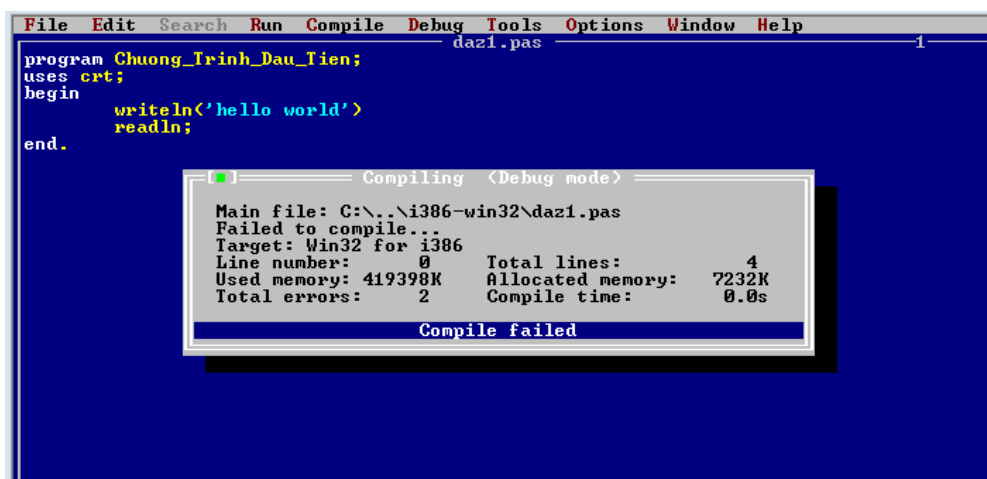
“**Files**”: Những chương trình có sẵn trong thư mục



Lúc này bạn gõ tên muốn đặt cho chương trình rồi ấn **Ok**, sau đó Free Pascal sẽ tiến hành kiểm tra lỗi của chương trình, nếu không có lỗi thì chương trình sẽ chạy được:



Nếu chương trình bị lỗi (ở đây là mình thiếu dấu ";" sau câu lệnh `writeln('hello world')`); thì Free Pascal sẽ hiện **Compile failed**:



3. Một số phím tắt

- Kiểm tra lỗi: **F9**
- Chạy chương trình **Ctrl+F9**
- Lưu lại chương trình **F2**
- Mở chương trình **F3**

4. Lưu ý

- Sau mỗi câu lệnh phải có dấu “;” (trừ **begin**, còn nếu **end** là câu lệnh kết thúc chương trình thì phải có dấu “.”).
- Các câu lệnh đi sau dấu “//” hay nằm trong cặp “{ }” đều bị vô hiệu hóa. Ở các bài sau, các từ mình ghi đi sau dấu “//” sẽ là chú thích.



IV.

VIẾT PASCAL

Bài 1. Dữ liệu và kiểu dữ liệu

1. Dữ liệu và kiểu dữ liệu

Máy tính là công cụ xử lý thông tin, còn chương trình chỉ dẫn cho máy tính cách xử lý thông tin để có kết quả mong muốn. Thông tin rất đa dạng nên dữ liệu trong máy tính cũng rất khác nhau về bản chất. Để dễ dàng quản lý và tăng hiệu quả xử lý, các ngôn ngữ lập trình thường phân chia dữ liệu thành các **kiểu** khác nhau: chữ, số nguyên, số thập phân,...



Hình trên minh họa một chương trình in ra màn hình với các kiểu dữ liệu quen thuộc là chữ và số.

Các kiểu dữ liệu khác nhau thường được xử lý theo các cách khác nhau. Chẳng hạn, ta có thể thực hiện các phép toán số học với các số, nhưng với các ký tự (chữ cái) thì các phép toán đó không có nghĩa.

Các ngôn ngữ lập trình có định nghĩa sẵn một số kiểu dữ liệu cơ bản. Kiểu dữ liệu xác định miền giá trị của dữ liệu và các phép toán có thể thực hiện trên các giá trị đó. Dưới đây là một số kiểu dữ liệu thường dùng nhất trong Pascal:

- **Số nguyên (integer, longint, byte,...)**, ví dụ như là số học sinh, số bàn, ghế,...
- **Số thực (real)**, là phân số, số thập phân, và khi sử dụng phép chia trong Pascal ta phải dùng kiểu dữ liệu là real.
- **Kí tự (char)**, là một chữ trong bảng chữ cái, chữ số, hay các ký tự đặc biệt khác, ví dụ như: "a", "A", "+", "1" (chữ số 1 khác với số nguyên 1), " " (ký tự trống),...
- **Xâu ký tự (hay xâu, chuỗi) (string)** là dãy liên tiếp các ký tự (tối đa 255 ký tự), ví dụ: "Giao Hang", "Font chu", "NHH", "2/9/1945",...
- **Boolean** là kiểu dữ liệu của logic, một biến boolean chỉ có thể có 1 trong 2 giá trị: **True** hoặc **False**.

Trong các ngôn ngữ lập trình, kiểu dữ liệu số nguyên còn có thể được chia thành các kiểu nhỏ hơn hoặc lớn hơn theo các phạm vi giá trị khác nhau. Ở Pascal, ngoài integer ra còn các kiểu thông dụng khác như là longint (các số nguyên trong khoảng từ -2147483648 tới 2147483647), byte (các số nguyên trong khoảng 0 tới 255).

Tên kiểu	Phạm vi giá trị
integer	Số nguyên trong khoảng từ - 32768 đến 32767.
real	Số thực có giá trị tuyệt đối trong khoảng $2,9 \times 10^{-39}$ đến $1,7 \times 10^{38}$ và số 0.
char	Kí tự trong bảng chữ cái.
string	Xâu kí tự, tối đa gồm 255 kí tự.

Một số kiểu dữ liệu cơ bản trong Pascal

Trong Pascal, để phân biệt rõ cho chương trình hiểu dãy chữ số là kiểu Xâu (string) hay là số, ta phải đặt dãy số đó vào cặp nháy đơn (' '). Ví dụ '2020', '2005',...

`writeln('2020 + 5');` thì kết quả in ra màn hình là **2020 + 5**

Nếu không đưa vào dấu nháy đơn (`writeln(2020 + 5)`) thì chương trình sẽ hiểu là phép toán cộng giữa 2 số 2020 và 5, cho kết quả in ra màn hình là: **2025**

2. Các phép toán số học

Trong mọi ngôn ngữ lập trình ta đều có thể thực hiện các phép toán số học: cộng, trừ nhân chia với các kiểu số nguyên và số thực.

Chẳng hạn, bảng dưới đây là kí hiệu của các phép toán số học đó trong ngôn ngữ pascal:

Kí hiệu	Phép toán	Kiểu dữ liệu
+	cộng	số nguyên, số thực
-	trừ	số nguyên, số thực
*	nhân	số nguyên, số thực
/	chia	số nguyên, số thực
div	chia lấy phần nguyên	số nguyên
mod	chia lấy phần dư	số nguyên

Ở đây chúng ta có 2 phép toán mới là div và mod (sử dụng khá nhiều).

Vd:

$$9 \text{ div } 3 = 3$$

$$11 \text{ div } 3 = 3$$

$$9 \text{ mod } 3 = 0$$

$$11 \text{ mod } 3 = 2$$

Chúng ta đã quen thuộc với các phép toán cộng trừ nhân chia. Tuy nhiên, hãy lưu ý rằng các ngôn ngữ lập trình đều xem kết quả chia 2 số **n** và **m** (**n/m**) là số thực, cho dù **n** và **m** là số nguyên và **n** chia hết cho **m**.

Sử dụng dấu ngoặc, ta có thể kết hợp các phép tính số học nói trên để có các biểu thức số học phức tạp hơn.. Sau đây là một số ví dụ về biểu thức số học và cách viết chúng trong Pascal

Biểu thức số học	Cách viết trong Pascal
$a \times b - c + d$	<code>a*b-c+d</code>
$15 + 5 \times \frac{a}{2}$	<code>15+5*(a/2)</code>
$\frac{x+5}{a+3} - \frac{y}{b+5} (x+2)^2$	<code>(x+5)/(a+3)-y/(b+5)*(x+2)*(x+2)</code>

Chú ý rằng trong toán học, để dễ phân biệt, chúng ta có thể dùng các cặp dấu (), [], { } để gộp các phép toán. Tuy nhiên trong các ngôn ngữ lập trình, chúng ta chỉ sử dụng dấu () cho mục đích này.

Ví dụ, biểu thức $\frac{(a+b)(c-d)+6}{3} - a$

Khi viết trong Pascal sẽ có dạng: `((a+b)*(c-d) + 6)/3 - a`

3. Các phép so sánh

Ngoài các phép toán số học, ta còn có các phép so sánh số.

Khi viết chương trình, để so sánh dữ liệu (số, biểu thức,..) chúng ta sử dụng các kí hiệu do ngôn ngữ lập trình quy định.

Kí hiệu các phép toán và phép so sánh có thể khác nhau, tùy theo ngôn ngữ lập trình.

Bảng dưới đây cho biết các kí hiệu của các phép so sánh trong Pascal:

Phép so sánh	Kí hiệu toán học	Kí hiệu trong Pascal	Ví dụ trong Pascal
Bằng	=	=	5 = 5
Khác	≠	<>	6 <> 5
Nhỏ hơn	<	<	3 < 5
Nhỏ hơn hoặc bằng	≤	<=	5 <= 6
Lớn hơn	>	>	9 > 6
Lớn hơn hoặc bằng	≥	>=	9 >= 6

Kết quả của phép so sánh chỉ có thể là đúng hoặc sai. Ví dụ, phép so sánh $9 > 6$ cho kết quả đúng, $10 = 9$ cho kết quả sai hoặc $3 > 5$ cũng cho kết quả sai.

Để so sánh giá trị của biểu thức, chúng ta cũng có thể sử dụng các phép toán ở phần 2. Ví dụ:

$5 \times 2 = 9$ (cho kết quả sai)

$15 + 7 > 20 - 3$ (cho kết quả đúng)

4. Một số hàm toán học khác

SQR(x):	Trả về bình phương của x
SQRT(x):	Trả về căn bậc hai của x ($x \geq 0$)
ABS(x):	Trả về x
SIN(x):	Trả về sin(x) theo radian
COS(x):	Trả về cos(x) theo radian
LN(x):	Trả về ln(x)
EXP(x):	Trả về e^x
TRUNC(x):	Trả về số nguyên gần với x nhất nhưng bé hơn x.
INT(x):	Trả về phần nguyên của x
FRAC(x):	Trả về phần thập phân của x
ROUND(x):	Làm tròn số nguyên x
ODD(n):	Cho giá trị TRUE nếu n là số lẻ.
INC(n):	Tăng n thêm 1 đơn vị ($n := n+1$).
DEC(n):	Giảm n đi 1 đơn vị ($n := n-1$).

Lưu ý: trong Pascal, để tính lũy thừa một số, bạn **không thể ghi x^y** . Để tính lũy thừa, bạn có thể dùng cú pháp **`exp(y*ln(x))`** để tính x^y .

Nhưng câu lệnh này sẽ trả về kiểu dữ liệu là real, nên bạn hãy thêm vào đó hàm làm tròn, là **`round(exp(y*ln(x)))`** để nhận về kiểu dữ liệu integer.

Bài 2. Giao tiếp giữa người và máy

1. Giao tiếp người – máy tính

Trong khi thực hiện chương trình máy tính, con người thường có nhu cầu can thiệp vào quá trình tính toán, thực hiện việc kiểm tra, điều chỉnh, bổ sung. Ngược lại, máy tính cũng cho thông tin về kết quả, tính toán,... Quá trình trao đổi dữ liệu 2 chiều ấy thường được gọi là giao tiếp hay tương tác giữa người với máy. Các giao tiếp này thường được thực hiện bằng chuột, bàn phím, màn hình. Dưới đây là một số ví dụ:

a. Thông báo kết quả tính toán

Thông báo kết quả tính toán là yêu cầu đối với mọi chương trình, ví dụ câu lệnh:

```
Writeln('Nam nay la nam ' , '2020');
```

Màn hình sẽ in ra kết quả:



b. Nhập dữ liệu

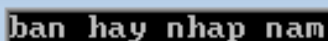
Một trong những tương tác thường gặp là chương trình sẽ yêu cầu nhập dữ liệu. Chương trình sẽ tạm ngưng để chờ người dùng nhập dữ liệu từ bàn phím. Hoạt động tiếp theo của chương trình sẽ tùy thuộc vào dữ liệu được nhập vào. Và Readln(biến) sẽ là câu lệnh giúp ta thực hiện điều này. Ví dụ:

Những lệnh dưới đây giúp đọc năm và lưu năm đó vào biến nam dùng nhập, sau đó in ra màn hình:

```
Writeln('Ban hay nhap nam:');
```

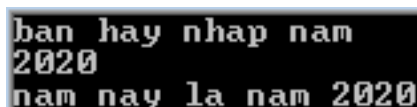
```
Readln(nam);
```

```
Writeln('Nam nay la nam ' , nam);
```



Chương trình sẽ tạm thời dừng lại và chờ cho đến khi bạn nhập dữ liệu vào mới thực hiện các câu lệnh khác.

Sau khi bạn nhập (ví dụ mình nhập 2020), chương trình sẽ in ra “Nam nay la nam 2020”.



Và trên đây là cách mà máy tính giao tiếp với bạn. Mình có giới thiệu sơ qua về “biến” (ví dụ trên là: nam), ở phần sau chúng ta sẽ tìm hiểu kỹ hơn về biến và cách sử dụng.

Bài 3. Biến (Variable)

1. Biến là công cụ lập trình

Biến nhớ, hay còn gọi là **biến** (variable) được sinh ra để ghi nhớ, lưu trữ dữ liệu. Nhưng lưu trữ các giá trị trong biến là chưa đủ, máy tính chỉ trở nên hữu dụng khi có thể thay đổi và kết hợp các giá trị đó. Máy tính được phát minh để xử lý những con số nên chúng rất giỏi việc này. Nhưng để cho máy biết lưu trữ và sử dụng biến, tất cả là ở bạn.

Một phần mềm thông minh chỉ khi người viết nên nó là 1 coder thông minh.

2. Khai báo biến

Tất cả các biến được sử dụng trong chương trình **phải** được khai báo ở phần khai báo đầu chương trình. Việc khai báo biến gồm:

- Khai báo **tên biến**.
- Khai báo **kiểu dữ liệu** của biến.

Tên biến phải tuân thủ theo qui tắc của ngôn ngữ lập trình (phần II.2)

```
var  
  a,b:integer;  
  s:real;  
  thong_bao:string;_
```

Ví dụ về cách khai báo biến trong Pascal

Trong ví dụ trên:

- **Var** là từ khóa của ngôn ngữ lập trình dùng để khai báo biến.
- **a, b** là các biến có kiểu **số nguyên** (integer).
- **s** là các biến có kiểu **số thực** (real).
- **thong_bao** là biến kiểu **chuỗi** (string).

3. Sử dụng biến trong chương trình

Sau khi khai báo, ta có thể sử dụng các biến trong chương trình. Các thao tác có thể thực hiện với biến là:

- Gán giá trị cho biến
- Tính toán với giá trị của biến

Kiểu dữ liệu của giá trị được gán cho biến thường phải trùng với kiểu của biến, và khi được gán 1 giá trị mới, giá trị cũ của biến bị xóa đi. Ta có thể thực hiện việc gán giá trị cho biến bất cứ lúc nào trong chương trình, do đó giá trị của biến có thể thay đổi.

Để nhập giá trị của biến từ bàn phím, ta dùng câu lệnh `Readln(biến);`

Câu lệnh dùng để gán giá trị cho biến là:

`biến := giá trị`

Ví dụ:

- muốn gán biến x có giá trị là 2: `x:=2;`
- muốn gán biến x có giá trị của biến y: `x:=y;`
- muốn gán biến x có giá trị lớn hơn y là 1: `x:=y+1;`
- muốn gán biến x có giá trị bằng một nửa y : `x :=y/2;`

4. Hằng

Ngoài công cụ chính để lưu trữ là biến, các ngôn ngữ lập trình còn có công dụng khác là **hằng (constant)**. Khác với biến, hằng có giá trị không đổi trong suốt chương trình. Giống với biến, ta cũng cần phải khai báo tên của hằng. Tuy nhiên hằng phải được gán giá trị ngay khi khai báo.

```
const  
    pi = 3.14;_
```

Khai báo hằng trong pascal

Trong đó :

- `const` là từ khóa để khai báo hằng.
- Hằng `pi` được gán giá trị tương ứng là 3.14.

Từ đó, để tính chu vi hình tròn có bán kính là biến `a`, ta có thể dùng câu lệnh sau

`chu_vi := 2*pi*a;`

Việc sử dụng hằng cũng rất hiệu quả cho những giá trị được sử dụng nhiều và không thay đổi trong chương trình. Và cần lưu ý thêm là bạn không thể thay đổi hay gán cho hằng 1 giá trị mới trong suốt phần thân chương trình như biến (ví dụ: câu lệnh `pi := 3.1416` không hợp lệ).

Bài tập vận dụng về biến:

Viết 1 chương trình nhập các số nguyên x, y. Sau đó hoán đổi giá trị của x và y. In ra màn hình giá trị của x và y trước và sau khi hoán đổi.

Vd :

Input : 2 3

Output : 2 3

 3 2

```
program hoan_doi;
uses crt;
var
    x,y,tmp:integer;
begin
    readln(x,y);
    writeln(x,' ',y);

    tmp:=x;
    x:=y;
    y:=tmp;

    writeln(x,' ',y);
    readln;_
end.
```

Bài giải

Giải thích :

Sau khi thực hiện câu lệnh `x:=y` thì `x` sẽ có giá trị của `y` và giá trị ban đầu của `x` không còn nữa, vì thế chúng ta không thể gán giá trị của `x` cho `y`.

Do đó, chúng ta phải có 1 biến nhớ tạm (`tmp`) để ghi giá trị của `x`, sau khi gán giá trị `y` cho `x` thì gán giá trị của `tmp` cho `y`.

Bài 4. Câu lệnh điều kiện (Conditional Statements)

1. Câu lệnh điều kiện

Ở Bài 1, chúng ta đã học về biến boolean, đó là cách một chương trình máy tính theo dõi xem mọi thứ là True hay False. Sức mạnh thực sự của kiểu biến này, thực ra, lại là việc bạn sẽ làm gì với những kết quả kiểu boolean. Bạn có thể làm như vậy bằng các câu lệnh điều kiện.

Câu lệnh điều kiện có dạng như sau: `if <điều kiện> then <câu lệnh>;`

2. If - Else

Câu lệnh điều kiện rất đơn giản. Đó là cách ra lệnh cho chương trình làm một việc nếu (if) một cái gì đó là True, nếu không (else) thì làm một việc khác (nếu không có else thì câu lệnh sẽ bị bỏ qua).

Giống như khi bạn điền 1 tờ đơn, ở phần giới tính, nếu là nam thì bạn điền là nam, còn không thì là Nữ, còn không nữa thì là “Khác”. Máy tính sẽ hiểu điều đó qua câu lệnh minh họa sau:

```
If (gioi_tinh = nam) then  
    Writeln('nam')
```

```
Else  
    Writeln('nữ');
```

Lưu ý nhỏ : câu lệnh đứng trên *Else* không được có dấu “;” ở cuối, giống như ở ví dụ trên.

Những câu lệnh đó hoạt động như sau :

Nếu giới tính là nam thì

Viết “nam”

Nếu không

Viết “nữ”

Bạn có thể để câu lệnh *if* đứng một mình, nhưng cũng có thể ghép nó với một câu lệnh *else*. Câu lệnh *else* này không cần điều kiện, vì nó xác định điều phải làm nếu điều ở *if* trả về giá trị false.

Lưu ý to: lệnh if khi trả về giá trị True, chỉ thực hiện được duy nhất 1 câu lệnh sau nó. Và else cũng vậy.

Và nếu muốn thực hiện nhiều hơn 1 câu lệnh khi if trả về giá trị True, bạn phải thêm cặp *begin* và *end* trong đó (nhưng *end* ở đây đi với dấu “;”).

Ví dụ :

Chương trình dưới đây kiểm tra số N nhập vào từ bàn phím là số chẵn hay lẻ, nếu số chẵn thì in ra : “true” và “Đây là số chẵn”. Nếu là số lẻ thì in ra :

“false” và “Đây là số lẻ”.

```
If N mod 2 = 0 then
    Begin
        writeln('true');
        writeln('Day la so chan');
    End
else
    begin
        writeln('false');
        writeln('Day la so le');
    end;
```

Bạn có thể đưa bất kì loại câu lệnh nào vào khối mã lệnh theo sau câu lệnh điều kiện, thậm chí đưa thêm câu lệnh *if* khác.

Ví dụ:

```
If bien_1 = 0 then
    Begin
        If bien_2 = 0 then
            Begin
                If bien_3 = 0 then
                    If...
                End;
            End;
        End;
    End;
```

Đưa các câu lệnh if vào bên trong if như vậy gọi là điều kiện lồng nhau

And và or trong câu lệnh điều kiện

Nếu bạn muốn câu lệnh điều kiện trả về giá trị True khi đáp ứng nhiều điều kiện 1 lúc, thì bạn chỉ cần thêm and vào giữa các điều kiện, và bỏ các điều kiện vào trong dấu “()”;

Câu lệnh điều kiện sẽ có dạng:

```
if (điều kiện) and (điều kiện) and (điều kiện)... then
```

Ví dụ:

```
If (a = 2) and (b = 2) and (c = 2) then
```

..

Nếu bạn muốn câu lệnh điều kiện trả về giá trị True khi đáp ứng 1 trong những điều kiện bạn đưa, thì bạn chỉ cần thêm or và giữa các điều kiện, và hãy nhớ là bỏ các điều kiện vào trong dấu “()”

```
if (điều kiện) or (điều kiện) or (điều kiện)... then
```

lời khuyên dành cho bạn là, hãy luôn bỏ các điều kiện vào dấu “()”, vì khi bạn chuyển sang các ngôn ngữ khác, thì câu điều kiện của nó sẽ có dạng

```
if (điều kiện)
```

```
{
```

```
...
```

```
} //cặp dấu “{ }” ở đây thay thế cho begin và end;
```

Một lỗi mà các bạn hay mắc phải là 1 điều kiện của các bạn có 2 phép so sánh cùng lúc

Ví dụ như: If (a = b = c) then

Ở pascal, 1 điều kiện chỉ được phép có tối đa 1 phép so sánh (chỉ (a = b) hay (b = c)). Vì thế, nếu bạn muốn câu lệnh điều kiện của bạn trả về giá trị True khi 3 số bằng nhau, thì câu lệnh của bạn phải là:

```
if (a = b) and (a = c) then
```

3. Else if (Phần đọc thêm)

Cùng với *if* và *else* còn có một câu lệnh điều kiện thứ 3: *else if* (hay *elif* trong một số ngôn ngữ khác). Đây là câu lệnh dạng *if* chỉ chạy nếu các kiểm tra ở phía trên nó là False và giống như *if* nó yêu cầu 1 điều kiện:

```
If (bien_so = 3) then
```

```
    Writeln('Ban da nhap chinh xac 3')
```

```
//giống như else, câu lệnh trước else if không có dấu “;”
```

```
Else if (bien_so < 3) then
```

```
    Writeln('So cua ban nho hon 3')
```

```
Else
```

```
    Begin
```

```
        Writeln('Để đến đây, số của bạn phải lớn hơn 3');
```

```
        Writeln('nhớ là, bạn có thể đặt bao nhiêu câu lệnh trong 1 khối  
tùy thích');  
    End;
```

Bạn có thể có nhiều câu lệnh *else if* nối tiếp nhau nếu muốn. Các lệnh sẽ “rơi” như dòng nước chảy theo những điều kiện này cho đến khi đạt được một câu điều kiện True, và tại đây chương trình sẽ thực hiện các lệnh nằm trong khối lệnh điều kiện này

Bài tập vận dụng

Nhập một số, kiểm tra xem số đó có phải năm sinh của bạn không, nếu có thì in ra màn hình: “True”. Ngược lại in ra màn hình: “False” và “ban nhap chua dung”, nếu số nhập vào lớn hơn năm sinh của bạn thì in ra màn hình năm sinh của bạn.

Bài 5. Vòng lặp (Loop)

1. Lặp

Máy tính rất siêu đẳng trong việc lặp đi lặp lại các công việc. Nếu muốn chương trình của bạn làm gì đó lặp đi lặp lại, bạn có thể copy và paste mã lại nhiều lần, nhưng việc đó sẽ tốn khá nhiều thời gian đấy!

Một cách tốt hơn là việc đặt khối mã (hãy nhớ rằng, một khối chính là mã được đặt trong begin - end;) ở bên trong một cấu trúc gọi là vòng lặp (loop), ở đó, chúng sẽ chạy nhiều lần

2. Vòng lặp For: số lần lặp xác định

Thông thường, bạn sẽ muốn lặp lại với một số lần cố định rồi mới làm tiếp; loại vòng lặp này gọi là vòng lặp for. Một vòng lặp for chỉ lặp lại một số lần nhất định. Vòng lặp for có một chỉ số - chỉ số này giúp theo dõi số lần chương trình đã tiến hành vòng lặp (thường mọi người sẽ gọi nó là i).

```
for i:=a to b do (a và b phải luôn là số nguyên)
```

```
Begin
```

```
...
```

```
End;
```

Chương trình minh họa:

Đếm từ 1 đến 10 rồi in ra màn hình chữ "Let's get it".

```
Writeln('bat dau dem nguoc');
```

```
For i:=1 to 10 do
```

```
Begin
```

```
Writeln(i); //ở đây i sẽ chạy từ 1 đến 10, chương trình sẽ  
in ra i mỗi lần chương trình tiến hành lặp, tức là in ra màn  
hình lần lượt từ 1 tới 10
```

```
End;
```

```
Writeln('let's get it'); //dòng này ở bên ngoài vòng lặp, nó sẽ  
chạy khi vòng lặp hoàn thành
```

Bạn có thể cho vòng lặp chạy từ số lớn về số bé, khi đó bạn chỉ cần thay chữ **to** thành **downto**.

Ví dụ:

Đếm ngược từ 10 đến 1, rồi in ra màn hình "Run".

```
For i:=10 downto 1 do
    Begin
        Writeln(i);
    End;
Writeln("run");
```

Có một vài các giúp mã lệnh trong vòng lặp có thể biến đổi. Lệnh *continue* yêu cầu chương trình bỏ qua các dòng lệnh còn lại và quay về khởi đầu vòng lặp trong lần chạy kế tiếp. Lệnh *break* sẽ yêu cầu chương trình dừng vòng lặp ngay tức khắc.

Ví dụ về lệnh *break*:

In ra màn hình từ 1 đến 9

```
For i:=1 to 10 do
    Begin
        Writeln(i);
        If i = 9 then
            Break; //ở đây, nếu i = 9 thì chương trình
                sẽ dừng lại ngay lập tức
    End;
```

3. Vòng lặp While: số lần lặp tùy thuộc điều kiện

Loại vòng lặp đơn giản nhất được gọi là vòng lặp while. Nó được đặt tên như vậy là vì nó tiếp tục lặp đi lặp lại khi thực hiện cùng một mã lệnh trong khi một số điều kiện là true. Khi sử dụng nó, bạn có thể viết mã lệnh cho đến khi điều gì đó hay ho xuất hiện. Well, bạn có thể hiểu vòng lặp while giống như câu lệnh điều kiện lặp vậy.

Vòng lặp while có dạng như sau:

While (điều kiện) do

Begin

...

End;

Chương trình minh họa:

```
Writeln('Những số nhỏ sẽ lớn rất nhanh nếu bạn nhân đôi nó liên  
tục');
```

```
x:=1
```

```
while (x <= 10000) do
```

```
begin
```

$x:=x*2$; (ở đây, chương trình sẽ tiến hành lặp $x*2$ liên tục cho đến khi $x > 10000$. Nếu không có câu lệnh này, giá trị của x sẽ mãi bằng 1 và vòng lặp sẽ chạy đến vô tận, chương trình sẽ bị lỗi. Nên bạn hãy nhớ khi sử dụng while , luôn có một câu lệnh để thoát khỏi vòng lặp (điều kiện dừng) nhé.)

```
writeln(x);
```

```
end;
```

```
writeln('bây giờ x đã lớn hơn 10000 rồi');
```

BÀI TẬP VẬN DỤNG VỀ VÒNG LẶP

Bài 1: Nhập 2 số a và b ($a < b$), tìm xem từ a đến b có bao nhiêu số chia hết cho 3.

Gợi ý:

Gọi thêm 1 biến đếm là “count”, cho giá trị của count là 0. Cho vòng lặp for chạy từ a đến b , nếu gặp số chia hết cho 3 thì $\text{count} := \text{count} + 1$.

Bài 2: Nhập các số nguyên, thao tác nhập chỉ dừng lại cho đến khi gặp số 0.

In ra tổng các số chẵn trong các số vừa nhập.

Bài giải:

```
uses crt;
var
    so_nhập, sum: integer;
begin
    so_nhập := 0;
    readln(so_nhập);
    if so_nhập mod 2 = 0 then
        sum := so_nhập
    else
        sum := 0;
    while (so_nhập <> 0) do
    begin
        readln(so_nhập);
        if (so_nhập mod 2 = 0) then
            sum := sum + so_nhập;
        end;
    end;
    writeln(sum);
    readln; _
end.
```

Giải thích code:

Gọi thêm 1 biến sum để tính tổng các số chẵn. Ở đây chúng ta không biết chính xác được chúng ta sẽ nhập bao nhiêu số nên sẽ dùng vòng lặp while, điều kiện là cho đến khi số nhập vào khác 0. Biến sum sẽ chạy trong vòng lặp, nếu gặp số chẵn thì nó sẽ cộng vào. Nhưng mà do biến sum chạy trong vòng lặp nên nếu số đầu tiên nhập vào là số chẵn, sum của chúng ta sẽ không cộng vào được, vì thế cần xác định số đầu tiên là số chẵn hay lẻ, nếu số chẵn thì gán cho giá trị của biến sum , nếu số lẻ thì cho biến sum bằng 0.

Bài 6. Mảng một chiều (Array)

1. Mảng một chiều

Mảng một chiều có vẻ lúc đầu nghe qua nó là một khái niệm gì đó có đôi chút phức tạp nhỉ. Nên bạn có thể hiểu nôm na: Mảng một chiều là một dãy hữu hạn các phần tử cùng kiểu dữ liệu.

Ví dụ: Xét mảng:

10 20 30 5 6 7

Mảng trên là 1 dãy bao gồm 6 số nguyên, trong đó:

Phần tử thứ 1 là: 10

Phần tử thứ 2 là: 20

Phần tử thứ 3 là: 30

Phần tử thứ 4 là: 5

Phần tử thứ 5 là: 6

Phần tử thứ 6 là: 7

Cách tham chiếu đến mảng: tên mảng[số thứ tự của phần tử trong mảng]

Ví dụ: mảng trên của chúng ta tên là A, thì cách tham chiếu đến mảng là A[i] với i là số thứ tự của phần tử trong mảng, từ đó ta có:

A[1] = 10; A[2] = 20; A[3] = 30; A[4] = 5; A[5] = 6; A[6] = 7;

2. Khai báo mảng N phần tử

Trong pascal, có 2 cách để khai báo mảng 1 chiều:

Cách đầu tiên là khai báo **trực tiếp**, cách này thì giống như khi bạn khai báo một biến vậy:

Var

<tên mảng>:array[1..n] of <kiểu phần tử>

Vd:

m1c:array[1..100] of integer;


```
a,b:array[1..100] of integer;
```

Cách thứ 2 là khai báo **gián tiếp**:

Type

<tên kiểu mảng> [1..n] of <kiểu dữ liệu>;

Var

```
<tên mảng>:<tên kiểu mảng>;
```

Ví dụ:

Type

```
mang_1_chieu = array[1..6] of integer;
```

var

```
m1c,a,b:mang_1_chieu;
```

3. Truy xuất mảng một chiều

```
for i:=1 to n do
begin
writeln('nhap phan tu thu ',i);
readln(m1c[i]);
end;
```

Nhập mảng 1 chiều

Bạn có thể bỏ câu lệnh `writeln('nhap phan tu thu ',i);` câu lệnh này chỉ giúp bạn biết là bạn đang nhập phần tử thứ mấy.

```
for i:=1 to n do
begin
writeln('phan tu thu ',i);
writeln(m1c[i])
end;
```

Xuất mảng 1 chiều

Tương tự như trên, bạn có thể bỏ câu lệnh `writeln('phan tu thu ',i);` vì nó chỉ giúp bạn biết là chương trình đang xuất ra phần tử thứ mấy.

```
program Nhap_Xuat_M1C;  
uses crt;  
type  
    mang_1_chieu = array[1..100] of integer;  
var  
    mic:mang_1_chieu;  
    n,i:integer;  
begin  
    writeln('nhap gioi han mang');  
    readln(n);  
  
    for i:=1 to n do  
    begin  
        writeln('nhap phan tu thu ',i);  
        readln(mic[i]);  
    end;  
  
    for i:=1 to n do  
    begin  
        writeln('phan tu thu ',i);  
        writeln(mic[i]);  
    end;  
  
    readln;  
end.
```

Một chương trình hoàn chỉnh để nhập và xuất mảng 1 chiều

Kết quả khi chạy chương trình và nhập dữ liệu:

```
nhap gioi han mang  
6  
nhap phan tu thu 1  
1  
nhap phan tu thu 2  
2  
nhap phan tu thu 3  
3  
nhap phan tu thu 4  
4  
nhap phan tu thu 5  
5  
nhap phan tu thu 6  
6  
phan tu thu 1  
1  
phan tu thu 2  
2  
phan tu thu 3  
3  
phan tu thu 4  
4  
phan tu thu 5  
5  
phan tu thu 6  
6
```

Bài tập vận dụng

Nhập mảng 1 chiều, in ra màn hình số số lẻ và số chẵn có trong mảng.

Bài 7. Hàm (Function) và Thủ tục (Procedure)

1. Chương trình con

Trong chương trình, có những chức năng hoặc chuỗi lệnh cần được thực hiện ở nhiều chỗ khác nhau. Để tránh phải viết lại các đoạn đó, người ta thường phân chương trình ra thành nhiều module, mỗi module giải quyết một công việc nào đó, các module như vậy là những chương trình con (subprogram).

Một tiện lợi khác của việc sử dụng module là ta có thể dễ dàng kiểm tra tính đúng đắn của nó trước khi ráp nối vào chương trình chính. Do đó việc xác định sai sót và tiến hành điều chỉnh trong chương trình sẽ thuận lợi hơn.

Trong Pascal chương trình con được viết dưới dạng hàm (Function) hoặc thủ tục (Procedure). Hàm và thủ tục đều là những chương trình con, nhưng hàm khác thủ tục ở chỗ hàm trả về một giá trị, còn thủ tục thì không.

Cấu trúc chung của một chương trình có sử dụng chương trình con:

```

USES CRT;
CONST .....;
VAR .....;
PROCEDURE THUTUC(Các tham số);
[Khai báo Const, Var]
BEGIN
    .....
END;
FUNCTION HAM(Các tham số):<Kiểu dữ liệu>;
[Khai báo Const, Var]
BEGIN
    .....
    HAM:=<Giá trị>;
END;
BEGIN //Chương trình chính
    .....
    THUTUC(...);
    .....
    A:=HAM(...);
    .....
END.

```

2. Hàm (function)

Hàm là một chương trình con tính toán trả về cho ta một giá trị kiểu vô hướng (Standard scalar types). Cấu trúc hàm như sau:

```
FUNCTION <tên hàm>(Danh sách các tham số):<Kiểu dữ liệu      (của
kết quả trả về)>;
```

[Khai báo Const, Type, Var] // Lưu ý : các kiểu, biến, hằng này chỉ có thể sử dụng trong hàm này.

```
BEGIN
```

```
  <các lệnh trong thân hàm>;
  <tên hàm>:=<Giá trị>;
```

```
END;
```

Chú ý luôn có phép gán tên hàm cho giá trị để hàm trả về giá trị khi được gọi

Ví dụ: kiểm tra 1 số có phải số chẵn không:

```
function IsEvenNum(n:integer):boolean;
var
  flag:Boolean;
begin
  flag:=false;
  if n mod 2 = 0 then
    flag:=true;
  IsEvenNum:=flag;
end;
```

3. Thủ tục

Thủ tục không trả về một giá trị kiểu vô hướng.

Khai báo thủ tục:

```
PROCEDURE <tên thủ tục>(Danh sách các tham số);//không có giá trị
trả về
```

[Khai báo Const, Type, Var] // Lưu ý : các kiểu, biến, hằng này chỉ có thể sử dụng trong hàm này.

```
BEGIN
```

```
  <các câu lệnh>;
```

```
END;
```

Ví dụ: in ra trên màn hình 2 số “x” và “y” 5 lần:

```

Procedure Print(x,y:integer);
Var
    i:integer;
begin
    for i:=1 to 5 do
        writeln(x,' ',y);
end;

```

4. Phân biệt các sử dụng hàm và thủ tục

Hàm khác thủ tục ở chỗ hàm trả về một giá trị cho lệnh gọi thông qua tên hàm còn thủ tục thì không.

Dùng function:

- Hàm trả về 1 giá trị duy nhất (kiểu vô hướng, kiểu string hoặc kiểu con trỏ).
- Lời gọi chương trình con cần nằm trong các biểu thức tính toán.

Ví dụ:

```
If IsEvenNum = True then...
```

Dùng procedure:

- Thủ tục không trả về giá trị nào.
- Lời gọi CTC không nằm trong các biểu thức tính toán.

Chú ý: Nếu một công việc có thể làm bằng hàm thì chắc chắn sẽ làm được bằng thủ tục (tuy nhiên sẽ phức tạp hơn khi dùng hàm) nhưng một chương trình làm bằng thủ tục thì chưa chắc ta đã làm được bằng hàm.

Một chương trình minh họa có sử dụng chương trình con

“Kiểm tra 1 số có phải số lẻ hay không, nếu có thì in ra màn hình “yes”, ngược lại in ra màn hình “no””:

```

uses crt;
var
    n:integer;

function IsEvenNum(n:integer):Boolean;
var
    flag:Boolean;
begin
    flag:=false;
    if n mod 2 = 0 then
        flag:=true;
    IsEvenNum:=flag;
end;

```

procedure PrintTrue; //nếu không sử dụng các biến khai báo ở đầu chương trình thì không cần khai báo danh sách tham số

begin

 writeln('true');

end;

procedure PrintFalse;

begin

 writeln('false');

end;

Begin

 readln(n);

 if IsEvenNum(n) = true then

 PrintTrue;

 else

 PrintFalse;

 readln;

End.

Ngoài ra, để kiểm tra 1 số bất kì có phải là số chẵn ovr chương trình chính, chỉ cần đưa số đó vào danh sách tham số, ví dụ: IsEvenNum(x) = true, IsEvenNum(2) = true,...

NHH IT CLUB

<IT/>
NHH IT CLUB



TÀI LIỆU TIN HỌC – PHẦN 1: PASCAL

Biên soạn bởi CLB Tin học THPT Nguyễn Hữu Huân.

Không sao chép dưới mọi hình thức mà không có sự cho phép.

Thực hiện năm 2020.

