Nội dung chính

1. Lịch sử phát triển
2. Các phần tử cơ bản của ngôn ngữ C
3. Cấu trúc cơ bản của chương trình C
4. Biên dịch chương trình C

Các phần tử cơ bản

1. Tập ký tự
2. Từ khóa
3. Định danh
4. Các kiểu dữ liệu
5. Hằng
6. Biến
7. Hàm
8. Biểu thức
9. Câu lệnh
10. Chú thích

1. Tập ký tự

Ký tự là các phần tử cơ bản tạo nên chương trình

- **Chương trình:** Tập các **câu lệnh** nhằm giải quyết nhiệm vụ đặt ra
- **Câu lệnh:** là các **từ** (*từ vựng*) liên kết với nhau theo cú pháp của ngôn ngữ lập trình
  - ***Ví dụ:*** while (i < N) do
- **Các từ:** Tổ hợp các ký tự theo nguyên tắc xây dựng *từ vựng*
  - ***Ví dụ:*** TenFile, BaiTap2…

1. Tập ký tự→ Tập ký tự trong C

- 26 chữ cái hoa:A B C ... X Y Z
- 26 chữ cái thường:a b c … x y z.
- 10 chữ số: 0 1 2 3 4 5 6 7 8 9.
- Các kí hiệu toán học:+ - * / = < >
- Các dấu ngăn cách: . ; , : space tab
- Các dấu ngoặc:( ) [ ] { }
- Các kí hiệu đặc biệt:_ ? $ & # ^ \ ! ‘ “ ~ ...

2. Từ khóa (keyword)

- Được định nghĩa sẵn trong mỗi NNLT
- Dành riêng cho các mục đích xác định
  - Đặt tên cho kiểu dữ liệu:
    - int, float, double…
  - Mô tả các lệnh, các cấu trúc lập trình
    - if, else, while, case, for…

2. Từ khóa→Từ khóa hay dùng trong Turbo C

break	case	char	const	continue	default
do	double	else	enum	float	for
goto	if	int	interrupt	long	return
short	signed	sizeof	static	struct	switch
typedef	union	unsigned	void	while	

Lưu ý: *Tất cả từ khóa trong C đều viết bằng chữ cái thường*

3. Định danh (*Identifier*)

- Định danh (*Tên*) là một dãy các kí tự dùng để gọi tên các đối tượng trong chương trình.
  - Các đối tượng trong chương trình
    - Biến
    - Hằng số
    - Hàm
    - Kiểu dữ liệu
- Định danh có thể được đặt bởi
  - Ngôn ngữ lập trình → các từ khóa
  - Người lập trình

3. Định danh→ Quy tắc đặt tên định danh trong C

- Định danh được bắt đầu bởi chữ cái hoặc dấu gạch dưới "_" (*underscore*)
- Các kí tự tiếp theo chỉ có thể là: chữ cái, chữ số hoặc dấu gạch dưới "_"
- Định danh do người lập trình đặt không được trùng với các từ khóa của C
- Độ dài định danh tùy thuộc phiên bản C
  - Turbo C++, không giới hạn độ dài tên, nhưng trình biên dịch chỉ sử dụng 32 ký tự đầu

Chú ý: *C là ngôn ngữ có phân biệt chữ hoa và chữ thường*

3. Định danh→Ví dụ

- Định danh hợp lệ:

  i, x, y, a, b, _function,

  _MY_CONSTANT, PI, gia_tri_1
- Định danh không hợp lệ

1_a, 3d, 55x	(bắt đầu bằng chữ số)
so luong, sin()	(có kí tự không hợp lệ, dấu cách, dấu ngoặc..)
int, char	(trùng với từ khóa của C)

3. Định danh→ Một số quy ước (code convention)

- Định danh nên có tính gợi nhớ
- Nên sử dụng dấu gạch dưới để phân tách các định danh gồm nhiều từ
  - Có thể dùng cách viết hoa chữ cái đầu mỗi từ
    - **Ví dụ:** sinh_vien, sinhVien, SinhVien
- Quy ước thường được sử dụng:
  - Hằng số dùng chữ cái hoa
    - **Ví dụ**: PI, EPSILON,…
  - Các biến, hàm, cấu trúc dùng chữ cái thường
    - *Biến điều khiển vòng lặp*: i, j, k…
    - *Hàm:* NhapDuLieu, TimKiem,…
    - *Cấu trúc:* SinhVien, MatHang,…

4. Các kiểu dữ liệu

- Một kiểu dữ liệu là một tập hợp các giá trị mà một dữ liệu thuộc kiểu dữ liệu đó có thể nhận được.
  - ***Ví dụ:*** Một đối tượng kiểu **int** của C sẽ là
    - Một số nguyên (*Số nguyên có dấu, 2 byte*)
    - Giá trị thuộc khoảng: [-32,768 (-2^{15}) …32,767 (2^{15}-1)]
- Trên một kiểu dữ liệu, xác định một số phép toán đối với các dữ liệu thuộc kiểu dữ liệu tương ứng.

4. Các kiểu dữ liệu→Ví dụ kiểu int

Một số phép toán được định nghĩa trên kiểu dữ liệu int của C

Tên phép toán	Ký hiệu	Ví dụ
Đảo dấu	-	
Cộng;Trừ;Nhân	+ ; - ; *	
Chia lấy nguyên	/	17/3→5
Chia lấy phần dư	%	17%3→2
So sánh	>, <, >=, <=, ==, !=	
Logic bit: AND; OR; XOR;NOT, Shift,…	& ; | ; ^ ; ~ ; <<; >>	3^17 →18 ~3 →-4

5. Hằng

- Hằng (*constant*) là đại lượng có ***giá trị không đổi trong chương trình.***
- Giá trị hằng do người lập trình xác định
- Các loại hằng
  - Hằng số nguyên
  - Hằng số thực
  - Hằng ký tự
  - Hằng chuỗi/xâu kỹ tự

5. Hằng → Hằng số nguyên

- Trong C, hằng số nguyên có thể biểu diễn dưới các dạng
  - Dạng thập phân
  - Dạng thập lục phân
  - Dạng bát phân

Giá trị thập phân	Giá trị thập lục phân	Giá trị bát phân
2011	0x7DB	03733
396	0x18C	0614

5. Hằng → Hằng số thực

- Trong C, hằng số thực có thể biểu diễn dưới các dạng
  - Dạng số thực dấu phẩy tĩnh
  - Dạng số thực dấu phẩy động

Số thực dấu phẩy tĩnh	Số thực dấu phẩy động
3.14159	31.4159 E-1
123.456	12.3456 E+1 hoặc 1.23456 E+2

5. Hằng → Hằng ký tự

- Hằng ký tự có thể biểu diễn theo hai cách
  - Đặt ký hiệu của ký tự giữa hai dấu nháy đơn
  - Dùng mã ASCII của ký tự:
    - Số thứ tự của ký tự đó trong bảng mã ASCII
    - Là số nguyên→tuân thủ quy tắc biểu diễn số nguyên

Ký tự	Dùng nháy đơn	Dùng mã ASCII
Chữ cái A	'A'	65, 0x41, 0101
Dấu nháy đơn	'\''	39, 0x27, 047
Ký tự tab	'\t'	0, 0x09, 011

5. Hằng → Hằng chuỗi/xâu kỹ tự

- Hằng chuỗi/xâu kí tự được biểu diễn bởi đặt dãy các kí tự trong xâu trong cặp dấu nháy kép.
- Ví dụ:
  - "ngon ngu lap trinh C"
  - "Tin hoc dai cuong"
  - "Dai hoc Bach Khoa Ha Noi"

6. Biến (*variable*)

- Biến là đại lượng mà ***giá trị có thể thay đổi trong chương trình.***
- Tên biến phải được đặt theo quy tắc đặt tên
  - Về thực chất, biến là các ô nhớ trong bộ nhớ máy tính dành cho 1 kiểu dữ liệu nào đó và được đặt tên để tiện tham khảo
    - ***Ví dụ:*** Biến kiểu int chiếm 2 ô nhớ
- **Lưu ý:**
  - Hằng số và biến được sử dụng để lưu trữ dữ liệu trong chương trình và phải thuộc một kiểu dữ liệu nào đó

7. Hàm (function)

- Hàm là chương trình con có chức năng
  - Nhận dữ liệu đầu vào (*các tham số vào*)
  - Thực hiện một công việc nào đó
  - Trả về kết quả ứng với tham số truyền vào
    - Ví dụ: hàm sin(x)
      - sin(3.14/2) →1.000
      - Sin(3.14/6) →0.499770
- Hàm không trả lại một giá trị: Thủ tục
  - Ví dụ: clrscr()

7. Hàm → Một số hàm toán học

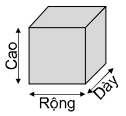
Hàm	Ý nghĩa	Ví dụ
sqrt(x)	Căn bậc 2 của x	sqrt(16.0) → 4.0
pow(x,y)	X mũ y (x^y)	pow(2,3) → 8
fabs(x)	Trị tuyệt đối của x (|x|)	fabs(-5.0) →5.0
exp(x)	E mũ x (e^x)	exp(1.0)→2.71828
log(x)	Logarithm tự nhiên của x (*ln x*)	Log(2.718)→0.999
log10(x)	Logarithm cơ số 10 của x (*logx*)	Log10(100) →2.00
sin(x) cos(x)/ tan(x)	Các hàm lượng giác	
ceil(x)	Số nguyên nhỏ nhất không nhỏ hơn x ($\lceil x \rceil$)	ceil(2.5)=3 ceil(-2.5)=-2
floor(x)	Số nguyên lớn nhất không lớn hơn x ($\lfloor x \rfloor$)	floor(2.5)=2 floor(-2.5)=-3

8. Biểu thức

- Biểu thức là sự kết hợp các các toán hạng (*operand*) bởi các toán tử (*operator*) theo một quy tắc xác định.
- Các toán hạng có thể là biến, hằng, hàm…
- Các toán tử rất đa dạng: cộng, trừ, nhân, chia

Ví dụ

- Thể tích hình hộp: **V=Rộng*Cao*Dày**
  - Phép nhân (*) là toán tử
  - Các toán hạng **Rộng, Cao, Dày**



9. Câu lệnh (*statement*)

- Câu lệnh diễn tả một hoặc một nhóm các thao tác trong giải thuật.
  - Chương trình được tạo thành từ dãy các câu lệnh.
- Các câu lệnh trong C, được kết thúc bởi dấu chấm phẩy (;)
  - Dấu chấm phẩy (;) dùng phân cách các lệnh

9. Câu lệnh → Phân loại

- Câu lệnh đơn:
  - Những câu lệnh không chứa câu lệnh khác.
    - ***Ví dụ:*** Phép gán, gọi hàm, vào/ra dữ liệu
- Các câu lệnh phức:
  - Những câu lệnh chứa câu lệnh khác.
    - ***Ví dụ***: Lệnh khối (Tập các lệnh đơn nhóm lại với nhau và đặt trong cặp ngoặc nhọn « { } »)
  - Các lệnh điều khiển cấu trúc chương trình
    - ***Ví dụ:*** Lệnh rẽ nhánh, lệnh lặp..

10. Chú thích (comment)

- Lời mô tả, giải thích vắn tắt cho một câu lệnh, một đoạn chương trình hoặc cả chương trình
  - Giúp việc đọc hiểu chương trình dễ dàng hơn
  - Chú thích không phải là câu lệnh ⇒ không ảnh hưởng tới chương trình
    - Khi gặp chú thích, trình biên dịch sẽ bỏ qua
- Cách viết chú thích
  - Chú thích một dòng: sử dụng « // »
  - Chú thích nhiều dòng: sử dụng « /* » và « */ »

Nội dung chính

1. Lịch sử phát triển
2. Các phần tử cơ bản của ngôn ngữ C
3. Cấu trúc cơ bản của chương trình C
4. Biên dịch chương trình C

Các phần cơ bản

Khai báo các tệp tiêu đề
#include

Khai báo các đối tượng toàn cục
- Định nghĩa kiểu dữ liệu mới
- Các biến, hằng
- Các hàm nguyên mẫu (prototype)

Định nghĩa hàm main()
{

}

Định nghĩa các hàm đã khai báo nguyên mẫu

1. Khai báo các tệp tiêu đề

- Liệt kê danh sách thư viện sẽ được sử dụng trong chương trình
  - Các hàm của C đều thuộc một thư viện nào đó
  - Không khai báo thư viện, trình biên dịch không hiểu được hàm (*có thể báo lỗi*)
- Cách thức (*cú pháp*) khai báo
  1. **#include<**ThuVien.h**>**
     - Thư viện phải nằm trong thư mục chứa các header file
     - Thường được sử dụng
     - ***Ví dụ:*** #include<stdio.h>
  2. **#include** "ThuVien.h"
     - Tìm kiếm thư viện tại thư mục hiện tại

2. Khai báo các đối tượng toàn cục

- Các đối tượng toàn cục có phạm vi sử dụng trong toàn bộ chương trình
  - Các kiểu dữ liệu mới
  - Các hằng, biến
  - Các nguyên hàm
- Tuân theo nguyên tắc khai báo đối tượng

2. Khai báo các đối tượng toàn cục

Định nghĩa kiểu dữ liệu

Cú pháp: typedef <ĐịnhNghĩaKiểu> <Tên kiểu

Ví dụ: **typedef** unsigned char byte;
typedef struct {float re, im;} complex;

Khai báo hằng

const float PI = 3.1415;
#define Max 50

Khai báo biến

int N;
float Delta, x1, x2;

2. Khai báo các đối tượng toàn cục (tiếp)

Khai báo các hàm nguyên mẫu

- Khai báo thông tin về các hàm của người dùng sẽ được sử dụng trong chương trình
  - Tên hàm
  - Danh sách các **kiểu** tham số sẽ truyền vào
  - Kiểu dữ liệu trả về
- ***Ví dụ***

float DienTichTamGiac(**float** a, **float** b, **float** c);
int getMax(**int** Arr []);
void swap(**int** * a, **int** * b);
void swap(**int** *, **int** *);

Có thể bỏ tên tham số

3. Định nghĩa hàm **main()**

- Bắt buộc phải có
- Là hàm đặc biệt trong C, đánh dấu điểm bắt đầu của mọi chương trình C
  - Khi thực hiện một chương trình C, hệ thống sẽ gọi tới hàm main đầu tiên, sau đó sẽ thực hiện lần lượt các câu lệnh (*bao gồm cả lời gọi tới các hàm khác*) nằm trong hàm main()
- **Cú pháp**

```
void main(){….}
void main(int argc, char * argv[ ]){….}
int main(){….; return 0;}
int main(int argc, char * argv[ ]){….; return 0;}
```


4. Định nghĩa các hàm đã khai báo

- Định nghĩa các hàm đã khai báo ở phần 3 (Phần khai báo nguyên mẫu - prototype)
  - Phần khai báo nguyên mẫu mới chỉ khai báo các thông tin cơ bản về hàm, chưa xác định rõ hàm hoạt động như thế nào
- **Ví dụ**

```
float DienTichTamGiac(float a, float b, float c){
  float p = (a+b+c)/2;
  return sqrt(p*(p-a)*(p-b)*(p-c));
}
```

$$S_\Delta = \sqrt{p(p-a)(p-b)(p-c)}$$

Chú ý

Các phần không bắt buộc phải theo đúng thứ tự

- Khi định nghĩa hàm được đặt trước hàm **main**(), không cần khai báo nguyên hàm
- **Nguyên tắc**:
  - Mọi đối tượng cần phải được khai báo trước khi sử dụng

Chương trình đầu tiên: Hello world!

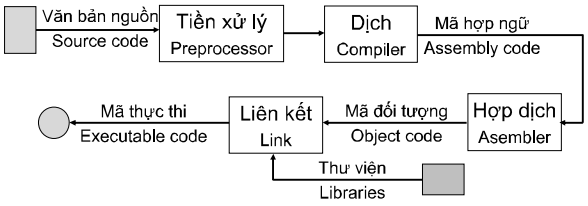
```
1.#include <stdio.h>
2.int main(){       //Không cần tham số dòng lệnh
3.     printf("Hello world! \n");
4.     return 0;    //Trả về giá trị 0
5.}
```

1. Nạp thư viện **stdio.h** vào, đây là thư viện vào ra chuẩn (*standard input output*) chứa khai báo nguyên hàm cho hàm ***printf***
2. Điểm bắt đầu thực hiện của chương trình. Máy tính thực hiện các câu lệnh nằm trong cặp ngoặc {} của main()
3. Hàm **printf** in ra một hằng chuỗi, có kết thuc bởi dấu xuống dòng (\n)
4. Trả về hệ điều hành một giá trị. Giá trị 0 thường dùng để thể hiện chương trình không có lỗi

Nội dung chính

Biên dịch chương trình

- Chương trình được viết bằng ngôn ngữ bậc cao phải được dịch ra mã máy để thực thi
  - Công việc dịch được thực hiện bởi ***trình biên dịch*** (compiler)
- Các giai đoạn dịch chương trình



Trình biên dịch Turbo C++

- Tồn tại nhiều trình biên dịch cho ngôn ngữ C
  - Turbo C++ của Borland Inc
    - Cho phép biên dịch cả C và C++
  - MSC của Microsoft, GCC của GNU
  - Dev-C, C-free,…
- Turbo C++ có nhiều phiên bản khác nhau
  - Sử dụng Turbo C++3.0 (***TC***)
    - Gọn nhẹ, đủ tính năng và dễ sử dụng

Cài đặt Turbo C++ 3.0

B1: Chuẩn bị bộ cài của Turbo C++ 3.0

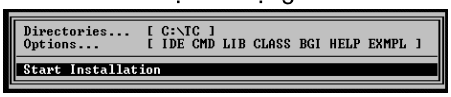
- Bộ cài tải trên mạng, kích thước khoảng 4M
- Copy bộ cài này vào máy (*giả sử C:\TC_Setup)*

B2: Cài đặt Turbo C

- Tìm đến thư mục chứa bộ cài (*C:\TC_Setup*)
- Kích hoạt file *INSTALL.EXE*
  - Chương trình sẽ yêu cầu chỉ ra ổ đĩa chứa bộ cài TC
- Enter the SOURCE drive to use
  - Nhập tên ổ đĩa (ổ C nếu đặt bộ cài tại *C:\TC_Setup*).
- Enter the SOURCE Path: Nhập đường dẫn tới thư mục chứa các file của bộ cài TC
  - Thông thường chương trình sẽ tự động tìm ra⇒ chỉ cần ấn Enter để chuyển sang bước tiếp theo.

Cài đặt Turbo C++ 3.0

B3: Xác định thư mục cài đặt. Thư mục này sẽ chứa các file của TC được sử dụng về sau.



Dùng các phím ↑ và ↓ để di chuyển hộp sáng đến phần **Start Installation** và ấn **Enter**. Chương trình sẽ tự động thực hiện và hoàn tất quá trình cài đặt

- Thư mục cài đặt mặc định sẽ là *\TC* nằm trên thư mục gốc của ổ đĩa chứa bộ cài.
- Nếu muốn thay đổi thư mục cài đặt, dùng các phím ↑ và ↓ để di chuyển hộp sáng đến **Directories**, gõ Enter và nhập đường dẫn mới, sau đó ấn phím Esc để trở về

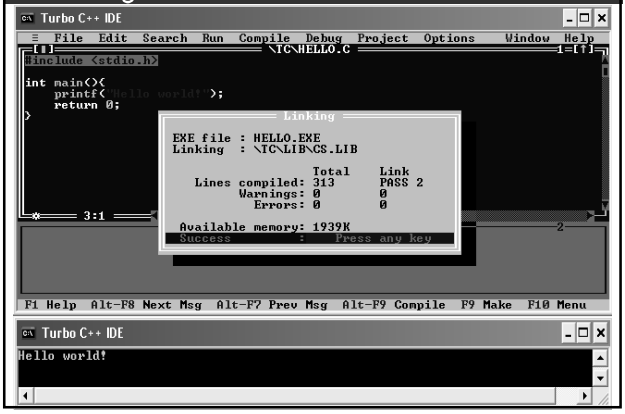
Lưu ý: Có thể copy toàn bộ thư mục **TC** để sử dụng

Màn hình giao diện Turbo C++ 3.0

Sử dụng Turbo C++ 3.0

- Khởi động chương trình:
  - Tìm đến thư mục *BIN* trong thư mục cài đặt
  - Chạy file *TC.EXE*
- Tạo cửa sổ soạn thảo mới
  - Chọn menu File (hoặc ấn Alt+F)→ chọn New
- Soạn thảo chương trình
  - Gõ chương trình nguồn vào cửa sổ soạn thảo
- Mở chương trình đã có: Alt+F → Open (**F3**)
- Lưu chương trình: Alt+F → Save (**F2**)
  - Nếu chưa có tên, sẽ được nhắc nhập tên file
- Biên dịch chương trình: Bấm phím **F9**
- Chạy chương trình: **Ctrl + F9**
- Xem lại kết quả thực hiện: **Alt+F5**

Chương trình Hello world!



Tóm tắt

1. Lịch sử phát triển của ngôn ngữ C
2. Các phần tử cơ bản của ngôn ngữ C
   - 10 phần tử cơ bản
3. Cấu trúc cơ bản của chương trình C
   - 4 phần
4. Thực hiện chương trình C với Turbo C

Nội dung chính

- Chương 1: Tổng quan về ngôn ngữ C
- Chương 2: Kiểu dữ liệu và biểu thức trong C
- Chương 3: Vào ra dữ liệu
- Chương 4: Cấu trúc điều khiển
- Chương 5: Mảng, con trỏ và xâu ký tự
- Chương 6: Cấu trúc
- Chương 7: Hàm
- *Chương 8: Tệp dữ liệu*

Nội dung chính

1. Các kiểu dữ liệu chuẩn trong C
2. Biểu thức trong C
3. Các toán tử trong C
4. Một số toán tử đặc trưng

Các kiểu đơn

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền dữ liệu
char	Kí tự; Số nguyên có dấu	1 byte	-128 ÷ 127
int short int	Số nguyên có dấu	2 byte	-32.768÷32.767
long long int	Số nguyên có dấu	4 byte	-2,147,483,648 ÷ 2,147,483,647
float	Số thực dấu phẩy động, độ chính xác đơn	4 byte	± 3.4E-38 ÷ ± 3.4E+38
double	Số thực dấu phẩy động, độ chính xác kép	8 byte	±1.7E-308 ÷ ± 1.7E+308

Các kiểu kết hợp

Với số nguyên, thêm từ khóa **unsigned** để chỉ ra số không dấu

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền dữ liệu
unsigned char	Số nguyên không dấu	1 byte	0 ÷ 255
unsigned short unsigned unsigned int	Số nguyên không dấu	2 byte	0÷65.535
unsigned long unsigned long int	Số nguyên không dấu	4 byte	0 ÷ 4,294,967,295
long double	Số thực dấu phẩy động,	10 byte	±3.4E-4932 ÷ ±1.1E+4932
void	Là kiểu rỗng, kích thước không		

Biểu diễn hằng số

Kiểu dữ liệu	Ví dụ	Ý nghĩa
Số nguyên	123, -12	Số thập phân
	012, 03777	Số bát phân
	0x7F, 0x3fe15	Số hệ 16
	39u 0267u, 0xFFu	Số không dấu
Số nguyên lớn	12L, 07723L 0xFFL, -10L 0xFFUL,0xFFLU	
Số thực	3.1415 -12.3, .327 10e-12, -15.3E12 3.1415F, -12.F	

Khai báo biến

- Một biến phải được khai báo trước khi sử dụng
- Cú pháp khai báo:

  KieuDuLieu TenBien;

  KieuDuLieu TenBien1, …, TenBien_N;
- ***Ví dụ***:

```
//Khai báo biến x là một số nguyên 2 byte có dấu
   int x;
//Khai báo các biến y, z là các số thực 4 byte
   float   y,z;
//Sau khi khai báo, có thể sử dụng
   x = 3;        y = x + 1;
```


Khai báo biến

- Sau khi khai báo, biến chưa có giá trị xác định.

  int n; m = 2 * n; ⇒ m=?
  - Biến cần được gán giá trị trước khi sử dụng
- C cho phép kết hợp khai báo và khởi tạo biến

  KieuDuLieu TenBien = GiaTriBanDau;

  KieuDuLieu $Bien_1$=$GiaTri_1$, $Bien_N$=Gia_Tri_N;
- ***Ví dụ:***

```
//Khai báo biến nguyên a và khởi tạo gia tri bằng 3
int a = 3;
//Khai báo biến thực x,y và khởi tạo giá tri bằng 5.0 và 7.6
float x = 5.0, y = 7.6;
```


Khai báo hằng

Dùng chỉ thị **#define**

Không có dấu chấm phây (;)

- Cú pháp:

  # define Tên_hằng Giá_trị
- Ví dụ:

```
#define  MAX_SINH_VIEN  50
#define  CNTT  “Cong nghe thong tin”
#define  DIEM_CHUAN  23.5
```


Khai báo hằng

Dùng từ khóa **const**

- Cú pháp:

  const Kiểu Tên_hằng = giá_trị;
- Ví dụ:

```
const int  MAX_SINH_VIEN  = 50;
const char  CNTT[20] = “Cong nghe thong tin”;
const float  DIEM_CHUAN = 23.5;
```


Khai báo hằng

Chú ý:

- Giá trị của các hằng phải được xác định ngay khi khai báo.
- Trong chương trình, **KHÔNG thể thay đổi** được giá trị của hằng.
- #define là chỉ thị tiền xử lý
  - Dễ đọc, dễ thay đổi
  - Dễ chuyển đổi giữa các nền tảng phần cứng hơn
  - Tốc độ nhanh hơn

Nội dung chính

Mục đích sử dụng

- Làm vế phải của lệnh gán.
- Làm toán hạng trong các biểu thức khác.
- Làm tham số thực sự trong lời gọi hàm.
- Làm biểu thức kiểm tra trong các cấu trúc điều khiển
  - Cấu trúc lặp: `for, while, do while`.
  - Cấu trúc rẽ nhánh: `if, switch`.

Tính toán giá trị biểu thức

- Các toán hạng được thay thế bởi giá trị tương ứng
- Các phép tính được thực hiên

Ví dụ (alpha = 10, beta = 81)

Biểu thức: alpha + sqrt(beta)

: alpha + sqrt(81)

: alpha + 9.0

: 10 + 9.0

: 19.0

Các loại biểu thức

- Biểu thứ số học
- Biểu thức quan hệ
- Biểu thức logic

Biểu thức số học

- Là biểu thức mà giá trị của nó là các đại lượng số học (*số nguyên, số thực*).
  - Sử dụng các toán tử là các phép toán số học (*cộng, trừ, nhân, chia…*),
  - Các toán hạng là các đại lượng số học (*hằng số, biến, biểu thức khác*).
- ***Ví dụ:*** a, b, c là các biến thuộc kiểu số thực.
  - `3 * 3.7`
  - `8 + 6/3`
  - `a + b – c`

Biểu thức quan hệ

- Là những biểu thức có sử dụng các **toán tử quan hệ** như lớn hơn, nhỏ hơn, khác nhau…
- Chỉ có thể trả về một trong 2 **giá trị logic** Đúng (***TRUE***) hoặc Sai (***FALSE***)

Ví dụ

```
5 > 7          // có giá trị logic là sai, FALSE
9 != 10        // có giá trị logic là đúng, TRUE
2 >= 2         // có giá trị logic là đúng, TRUE
a > b          // giả sử a, b là 2 biến kiểu int
a+1 > a        // có giá trị đúng, TRUE
```


Biểu thức logic

- Là biểu thức trả về các giá trị logic Đúng/Sai
  - Các phép toán logic gồm có
    - **AND** VÀ logic, sử dụng toán tử `&&`
    - **OR** HOẶC logic, sử dụng toán tử `||`
    - **NOT** PHỦ ĐỊNH, sử dụng toán tử `!`
  - Biểu thức quan hệ là trường hợp riêng của biểu thức logic.
- Ngôn ngữ C coi các giá trị nguyên khác 0 (2, 8, -5,..) là giá trị logic đúng (TRUE), giá trị 0 là giá trị logic sai (FALSE)
  - Biểu thức logic cũng trả về một giá trị số học 0/1

Biểu thức logic →Ví dụ

```
(5 > 7)&&(9!=10)     // có giá trị logic là sai, FALSE
0 || 1               // có giá trị logic là đúng, TRUE
(5 > 7)||(9!=10)     // có giá trị logic là đúng, TRUE
0                    // có giá trị logic là sai, FALSE
!0                   // phủ định của 0, có giá trị logic là đúng, TRUE
3                    // có giá trị logic là đúng, TRUE
!3                   // phủ định của 3, có giá trị logic là sai, FALSE
(a > b)&&( a < b)    // Có giá trị sai, FALSE. Giả sử a, b là 2 biến kiểu int
```

5 * (12 > 6) →?

Nội dung chính

1. Các kiểu dữ liệu chuẩn trong C
2. Biểu thức trong C
3. Các toán tử trong C
4. Một số toán tử đặc trưng

Các toán tử chính

Các toán tử cho phép tạo nên các biểu thức từ các hằng và biến

- Toán tử số học
- Toán tử quan hệ
- Toán tử logic
- Toán tử logic bit
- Toán tử gán

Các toán tử số học

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ (*int a = 12; float x=3.0*)
-	Đảo dấu	float, double, int, long,.. (*Số nguyên hoặc thực*)	-12, -12.34, - a, - x - -a →12, **--a →?**
+	Cộng	float, double, int, long,..	12 + -x → 9.0
-	Trừ	float, double, int, long,..	12.0 - -3 →15.0
*	Nhân	float, double, int, long,.. (*Số nguyên hoặc thực*)	12 * 3.0 →36.0 12 * 3 → 36
/	Chia	Nếu có ít nhất 1 toán hạng là số thực	17.0/3.0 →5.666667 17/3.0 →5.666667 17.0/3 →5.666667
/	Chia lấy nguyên	**Số nguyên** int, long,..	17/3→5
%	Chia lấy dư	**Số nguyên:** int, long,..	17%3 →2

Các toán tử quan hệ

<, >, <=, >=, ==, !=

- Dùng cho phép so sánh giá trị 2 toán hạng
- Kết quả phép so sánh là một số nguyên
  - **1** nếu quan hệ có kết quả là đúng,
  - **0** nếu quan hệ có kết quả sai

Ví dụ:

6 > 4 → Trả về giá trị 1

6 < 4 → Trả về giá trị 0

int b =(x !=y);

Nếu x và y khác nhau, biểu thức đúng và b mang giá trị 1.
Ngược lại biểu thức sai và b mang giá trị 0

5 * (12 > 6) → 5

Các toán tử logic

Sử dụng để xây dựng các biểu thức logic

- Biểu thức logic có kết quả logic **đúng**
  → Trả về giá trị **1**
- Biểu thức logic có kết quả logic **sai**
  → Trả về giá trị **0**

Các toán tử

Và logic:	Op1 **&&** Op2
Hoặc logic:	Op1 **||** Op2
Phủ định logic:	**!** Op

Oprand: Toán hạng

Các toán tử logic (*tiếp*)

Và logic (&&) :
- Cho kết quả đúng (trả về giá trị 1) khi cả 2 toán hạng đều đúng (**khác 0**)
- ***Ví dụ:*** 3 < 5 && 4 < 6 →1; 3 < 5 && 5 > 6 → 0

Hoặc logic (||):
- Cho kết quả sai (trả về giá trị 0) chỉ khi cả 2 toán hạng đều sai (bằng 0)
- ***Ví dụ:*** 4 || 5 < 3 → 1; 5 < 5 || 2 > 6 → 0

Phủ định logic (!):
- Cho kết quả đúng (1) hoặc sai (0) khi toán hạng là sai (0) hoặc đúng (**khác 0**)
- ***Ví dụ:*** ! 3 → 0; !(2 > 3) → 1;

Toán tử logic bit

Toán tử bit được sử dụng với kiểu số nguyên

Và nhị phân: Op1 **&** Op2
Hoặc nhị phân : Op1 **|** Op2
Hoặc có loại trừ nhị phân: Op1 **^** Op2
Đảo bít nhị phân : **~** Op

Oprand: Toán hạng

Dịch trái: Op **<<** n (nhân với 2^n)
Dịch phải: Op **>>** n (Chia với 2^n)

Op là giá trị được dịch, n là số bit dịch

Toán tử logic bit (*tiếp*)

char Op1 = 83, Op2 = -38, Op = 3;

char r = Op1 **&** Op2;
0 1 0 1 0 0 1 1
1 1 0 1 1 0 1 0
r = 0 1 0 1 0 0 1 0 → (82)

char r = Op1 **|** Op2;
0 1 0 1 0 0 1 1
1 1 0 1 1 0 1 0
r = 1 1 0 1 1 0 1 1 → (-37)

char r = Op1 **^** Op2;
0 1 0 1 0 0 1 1
1 1 0 1 1 0 1 0
r = 1 0 0 0 1 0 0 1 → (-119)

char r = ~ Op2;
1 1 0 1 1 0 1 0
r = 0 0 1 0 0 1 0 1 → (37)

unsigned char r = Op1 **|** Op2; r = 1 1 0 1 1 0 1 1 → **219**

Toán tử logic bit (*tiếp*)

char Op1 = 83, Op2 = -38, Op = 3;

char r = Op1 **>>** Op;
0 1 0 1 0 0 1 1
r = **0 0 0** 0 1 0 1 0 → (10)

char r = Op2 **>>** Op;
1 1 0 1 1 0 1 0
r =**1 1 1** 1 1 0 1 1 → (-5)

unsigned char Op =218;
unsigned char r =Op >> 3;
1 1 0 1 1 0 1 0
r =**0 0 0** 1 1 0 1 1 → (27)

char r = Op2 **<<** 2;
r = 0 1 1 0 1 0 0 0 →(104)
int r = Op2<<2 →?

Toán tử gán

Biến = Biểu_thức;

- Ký tự "**=**" là toán tử gán
  - Biểu thức bên phải dấu bằng tượng tính toán
  - Giá trị của *biểu_thức* được gán cho *biến*
- ***Ví dụ:***

```
int a, b, c;
a = 3;
b = a + 5;
c = a * b;
```


Toán tử gán

- Biểu thức gán là biểu thức nên cũng có giá trị.
  - Giá trị của biểu thức gán bằng giá trị của biểu_thức bên phải toán tử
  - Có thể gán giá trị của biểu thức gán cho một biến khác
  - Có thể sử dụng như một biểu thức bình thường
- ***Ví dụ:***

```
int a, b, c;
a = b = 2007;
c = (a = 20) * (b = 30);// c→600
```

Toán tử gán → Dạng kết hợp

Var <op>= Exp ⇔ Var = Var <op> Exp

Toán tử số học:

	+=	**-=**	***=**	**/=**	**%=**
Ví dụ:	**a *= b**			**// a= a * b**

Toán tử logic bit:

	&=	**|=**	**^=**
Ví dụ:	x &= 0x3F		// x = x & 0x3F

Toán tử dịch:

	<<=	**>>=**	
Ví dụ:	s <<= 4		// s = s << 4

Chương 2: Kiểu dữ liệu và biểu thức trong C

Nội dung chính

1. Các kiểu dữ liệu chuẩn trong C
2. Biểu thức trong C
3. Các toán tử trong C
4. Một số toán tử đặc trưng

Các toán tử

- Tăng/giảm tự động một đơn vị
- Lấy địa chỉ
- Biểu thức điều kiện
- Toán tử phẩy

Tăng giảm tự động một đơn vị

++	Tăng tự động	++Var, Var++
--	Giảm tự động	--Var, Var–

Variable: Biến

- Tiền tố (hậu tố): biến được tăng/giảm trước (sau) khi sử dụng để tính toán biểu thức

Ví dụ:

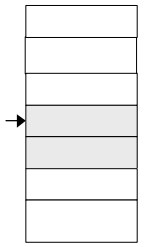
```
int   a = 5, b, c, d, e;
      b = a++;      // b = 5 sau đó a = 6
      c = ++a;      // a = 7 rồi tới c = 7
      d = a--;      // d = 7 rồi tới a = 6
      e = –a;       // a = 5 sau đó e= 5
```


Toán tử lấy địa chỉ

& Tên_biến

Ký tự **&** là toán tử lấy địa chỉ biến

- Biến thực chất là một vùng nhớ của máy tính được đặt tên → tên của biến
- Mọi ô nhớ trên bộ nhớ máy tính đều được đánh địa chỉ.
  → Mọi biến đều có địa chỉ



Ví dụ:

int a = 2006;

`&a` → Địa chỉ của ô nhớ dùng chứa giá trị biến a

Kiểu địa chỉ?

Toán tử phỏng điều kiện (*biểu thức điều kiện*)

exp1 ? exp 2 : exp3

expression: Biểu thức

- Nếu **exp1 ≠ 0** (*giá trị đúng*), biểu thức điều kiện trả về giá trị của **exp2**
- Nếu **exp1 = 0** (giá trị sai) biểu thức điều kiện trả về giá trị của **exp3**

Ví dụ:

```
float x= 5.2, y = 3.8, z;
z = (x < y) ? x : y;
  → z = 3.8    // z min{x, y}
  ⇔ if (x < y) z = x; else z = y;
```

Toán tử phẩy

biểu_thức_1, biểu thức_2,..

- Toán tử phẩy (**,**) cho phép sử dụng nhiều biểu thức tại nơi chỉ cho phép viết một biểu thức
- Các biểu thức được tính toán từ trái qua phải
- Giá trị và kiểu của biểu thức là giá trị và kiểu của biểu thức cuối cùng, bên phải

Ví dụ:

if (i = 0, a !=b)…

for(i = 0, j = 0; i < 100; i++, j++)….

Chuyển kiểu

(Kiểu) biểu thức

- Chuyển kiểu tự động
  - Chương trình dịch tự động chuyển đổi từ kiểu có phạm vi biểu diễn thấp tới kiểu có phạm vi biểu diễn cao

    `char → int → long int → float → double → long double`
- Ép kiểu
  - Bằng câu lệnh tường minh trong chương trình
  - Được sử dụng khi muốn chuyển sang kiểu có phạm vi biểu diễn thấp hơn

Chuyển kiểu→Ví dụ

```
#include <stdio.h>
#include <conio.h>
void main(){
    long L = 0xABCDEF; float f = 123.456;
    int i;
    clrscr();
    i = (int) L;
    printf("\n L = %ld; i = %d(%X)", L, i, i);
    i = (int) f; L = (long) f;
    printf("\n f = %f; L = %ld;  i = %d", f, L, i);
}
```

L = 11259375; i = -12817(CDEF)
f = 123.456001; L = 123; i = 123

Thứ tự ưu tiên các toán tử

Mức	Toán tử	Chức năng	Chiều
1	-> . [ ] ( ) ++ hậu tố --hậu tố	Lựa chọn, chỉ số…	→
2	++ -- ~ ! + - * & () sizeof	Toán tử 1 ngôi, ép kiểu,…	←
3	* / %	Toán tử số học lớp nhân	→
4	+ -	Toán tử số học lớp cộng	→
5	>> <<	Dịch bit	→
6	< <= > >=	Toán tử quan hệ	→
7	== !=	Bằng, khác	→
8	&	AND nhị phân	→
9	^	XOR nhị phân	→
10	|	OR nhị phân	→
11	&&	AND logic	→
12	||	OR logic	→
13	? :	Toán tử phỏng điều kiện	←
14	= *= += <<= &= ...	Toán tử gán	←

Chiều kết hợp với các toán hạng

Thứ tự ưu tiên các toán tử

Nguyên tắc

- Biểu thức con trong ngoặc được tính toán trước
- Phép toán một ngôi đứng bên trái toán hạng được kết hợp với toán hạng đi liền nó.
- Toán hạng đứng cạnh hai toán tử
  - Nếu hai toán tử có độ ưu tiên khác nhau thì toán tử nào có độ ưu tiên cao hơn sẽ kết hợp với toán hạng
  - Nếu hai toán tử cùng độ ưu tiên thì dựa vào trật tự kết hợp của các toán tử để xác định toán tử được kết hợp với toán hạng.

Ví dụ

a < 10 && 2 * b < c ≡ (a < 10) && ((2 * b) < c)

Chú ý: int x = 5, a = 5 * x++; → a = 25, x = 6

Ví dụ

```
const int N=10;
float S= 0.0;
int b;
S = N/3 +1;
b=(S>4);

S=…. b =…..
```

```
int a= 3, b=4, c;
c = a++  * ++b;

a=..  b=… c=…
```

```
int k ,num=30;
k =num>5 ? (num <=10 ? 100 : 200): 500;
k=?
```

Tóm tắt

- **Kiểu dữ liệu**
  - Nguyên : char, unsigned char, int, long, unsigned int, unsigned long
  - Thực : float, double, long double
- **Giá trị logic**
  - Đúng/TRUE : **1** (*Khác 0*)
  - Sai/FALSE : **0**
- **Toán tử**
  - Một ngôi : + -; ++ --; ~ !; &; (); &, ^, |
  - Hai ngôi : + - * / %; == != < <= > >=; << >>; && ||; = *= +=…
  - 3 ngôi : **? :**

Nội dung chính

- Chương 1: Tổng quan về ngôn ngữ C
- Chương 2: Kiểu dữ liệu và biểu thức trong C
- Chương 3: Vào ra dữ liệu
- Chương 4: Cấu trúc điều khiển
- Chương 5: Mảng, con trỏ và xâu ký tự
- Chương 6: Cấu trúc
- Chương 7: Hàm
- *Chương 8: Tệp dữ liệu*

Nội dung chính

1. Các hàm vào ra cơ bản:
   - **printf()**
   - **scanf()**
2. Các hàm vào ra khác
   - **gets()**
   - **puts()**
   - **getch()**

Các hàm vào ra cơ bản

- Đưa ra dữ liệu:
  - **printf()**
- Nhập dữ liệu
  - **scanf()**
- Cần nạp thư viện **stdio.h**
  - khai báo tệp tiêu đề :

```
#include <stdio.h>
```


Hàm đưa ra dữ liệu

printf()

Mục đích

- Hiển thị ra màn hình các loại dữ liệu cơ bản
  - Số nguyên, số thực, kí tự, xâu kí tự
- Tạo một số hiệu ứng hiển thị đặc biệt
  - Xuống dòng, sang trang,…

Cú pháp

printf(xau_dinh_dang [, DS_tham_so]);

Cú pháp

printf(xau_dinh_dang [, DS_tham_so]);

- **Xau_dinh_dang**: Là một xâu qui định cách thức hiển thị dữ liệu ra màn hình máy tính.
  - Bao gồm các nhóm kí tự định dạng
  - Nhóm kí tự định dạng thứ *k* xác định quy cách hiển thị tham số thứ *k* trong `DS_tham_số`
    - Số lượng tham số trong `DS_tham_số` bằng số lượng nhóm các kí tự định dạng trong `xâu_đị nh_dạng`.
- **DS_tham_so:** Danh sách các biến/biểu thức sẽ được hiển thị giá trị lên màn hình theo cách thức được qui định trong xau_dinh_dang.

Ví dụ

```
#include <stdio.h>
void main()
{ int a = 5;
  float x = 1.234;
  printf(" Hien thi mot bieu thuc nguyen %d và
    mot so thuc %f ",2 * a, x);
}
```

Kết quả:

Hien thi mot bieu thuc nguyen 10 va mot so thuc 1.234000

Xâu định dạng

- Các kí tự thông thường:
  - Được hiển thị ra màn hình.
- Các kí tự điều khiển:
  - Dùng để tạo các hiệu ứng hiển thị đặc biệt như xuống dòng (‘**\n**’)..
- **Các nhóm kí tự định dạng**:
  - Xác định quy cách hiển thị các tham số trong phần danh_sach_tham_so.

Nhóm ký tự định dạng

- Mỗi nhóm kí tự định dạng chỉ dùng cho một kiểu dữ liệu”

  Ví dụ: %d dùng cho kiểu nguyên

  %f dùng cho kiểu thực
- DS_tham_so phải phù hợp với các nhóm kí tự định dạng trong xau_dinh_dang về:
  - Số lượng;
  - Thứ tự
  - Kiểu dữ liệu;

Nếu không phù hợp sẽ hiển thị ra kết quả không như ý

printf(” %d ” ,3.14); →-31457

C-Free →1374389535 !?

Các ký tự định dạng

Ký tự	Kiểu dữ liệu	Kết quả
%i, %d	int, char	Số thập phân
%o	int, char	Số bát phân (không có 0 đằng trước)
%x %X	int, char	Số hexa (chữ thường/chữ hoa)
%u	unsigned int/char	Số thập phân

Các ký tự định dạng

Ký tự	Kiểu dữ liệu	Kết quả
%ld, %li	long	Số thập phân
%lo	long	Số bát phân (không có 0 đằng trước)
%lx, %LX	long	Số hexa (chữ thường/chữ hoa)
%lu	unsigned long	Số thập phân

Nhận xét:

Với kiểu long, thêm ký tự l ngay sau dấu %

Các ký tự định dạng

Ký tự	Kiểu dữ liệu	Kết quả
%f	float/double	Số thực dấu phẩy tĩnh
%e, %E	float/double	Số thực dấu phẩy động
%c	int, char	Kí tự đơn lẻ
%s	char []	Hiển thị xâu kí tự kết thúc bởi '\0'
%%		Hiển thị kí tự %

Độ rộng hiển thị → Số nguyên, Ký tự, Xâu ký tự

- Có dạng "%**m**",
  - **m** là một giá trị nguyên, không âm.
  - **m** cho biết số chỗ trống dành cho hiện thị biểu thức tương ứng

Ví dụ:

int a = 1234;

printf("%5d",a) → 1234

printf("%5d",34)→ 34

ký hiệu cho dấu trắng (space)

Độ rộng hiển thị→Ví dụ

```
printf("\n%3d %15s %3c", 1, "nguyen van a", 'g');
printf("\n%3d %15s %3c", 2, "tran van b", 'k');
```

```
  1       nguyen van a     g
  2         tran van b     k
```


Độ rộng hiển thị → Số thực

- Có dạng "%**m.n**",
  - **m, n** là 2 giá trị nguyên, không âm.
  - **m** cho biết kích thước để hiển thị số thực
  - **n** cho biết kích thước dành cho phần thập phân, nếu không đủ C sẽ làm tròn khi hiển thị

Ví dụ:

printf("\n%**f**",17.345); → 17.345000

printf("\n%**.2f**",17.345); → 17.35

printf("\n%**8.2f**",17.345); → ␣␣␣17.35

Độ rộng hiển thị→ Chú ý

- Nếu số chỗ cần để hiển thị dữ liệu lớn hơn được cung cấp trong định dạng ⇒ Tự động cung cấp thêm chỗ mới để hiển thị đầy đủ, không cắt bớt nội dung của dữ liệu.

Ví dụ:

printf("%2d", 1234); → 1234

printf("%6.3f", 123.456); → 123.456

printf("%12.6e", 123.456); →1.234560e+02

printf("%12.3e", 123.456); →␣␣␣1.235e+02

C-Free →␣␣1.235e+002

Căn lề trái - căn lề phải

%-

- Khi hiển thị dữ liệu có sử dụng tham số độ rộng, để căn lề trái cần thêm dấu trừ **-** vào ngay sau dấu %:
  - Ngầm định, căn lề phải

Ví dụ:

printf("%-3d%-10s%-5.2f%-3c",5,"Hello",7.5, 'g')

→5␣␣Hello␣␣␣␣␣7.50␣g␣␣

Hàm nhập dữ liệu

scanf()

Mục đích

Dùng để nhập dữ liệu từ bàn phím

- Ký tự đơn lẻ
- Chuỗi ký tự
- Số nguyên
  - Thập phân, Bát phân, Hexa
- Số thực
  - Dấu phẩy tĩnh; Dấu phẩy động

Cú pháp

scanf(xau_dinh_dang[,DS_dia_chi]);

Cú pháp

scanf(xau_dinh_dang [, DS_dia_chi]);

Xau_dinh_dang:

- Gồm các ký tự được qui định cho từng loại dữ liệu được nhập vào.
  - *Ví dụ: dữ liệu định nhập kiểu nguyên thì xâu định dạng là : %d*

DS_dia_chi:

- Bao gồm địa chỉ của các biến (*toán tử* **&**), phân tách nhau bởi dấu phẩy (**,**)
- Phải phù hợp với các kí tự định dạng trong **xau_dinh_dang** về số lượng, kiểu, thứ tự

Họa động

- Đọc các ký tự được gõ vào từ bàn phím
- Căn cứ vào xâu định dạng, chuyển thông tin đã nhập sang kiểu dữ liệu phù hợp
- Gán những giá trị vừa nhập vào các biến tương ứng trong DS_dia_chi

Ví dụ:

int a;

scanf("%d",&a); →1234_ → a = 1234

Ghi chú

Thông tin được gõ vào từ bàn phím, được lưu ở vùng đệm trước khi được xử lý bởi hàm scanf()→Hàm scanf() đọc từ vùng đệm

```
#include <stdio.h>
int main(){
    int a, b;
    scanf("%d",&a);
    scanf("%d",&b);
    printf ("%d %d", a, b);
    return 0;
}
```

```
123    456_
```

```
123 456
```


Các ký tự định dạng

Kí tự	Khuôn dạng dữ liệu nhập
%c	Đọc kí tự đơn lẻ
%d	Đọc số thập phân
%o	Đọc số bát phân
%x	Đọc số hexa
%u	Đọc số thập phân không dấu

Các ký tự định dạng

Kí tự	Chú thích
%s	Đọc xâu kí tự tới khi gặp dấu phân cách
%f	Đọc số thực dấu phẩy tĩnh (float)
%ld	Đọc số nguyên kiểu long
%lf	Đọc số thực dấu phẩy tĩnh (double)
%e	Đọc số thực dấu phẩy động
%%	Đọc ký tự %

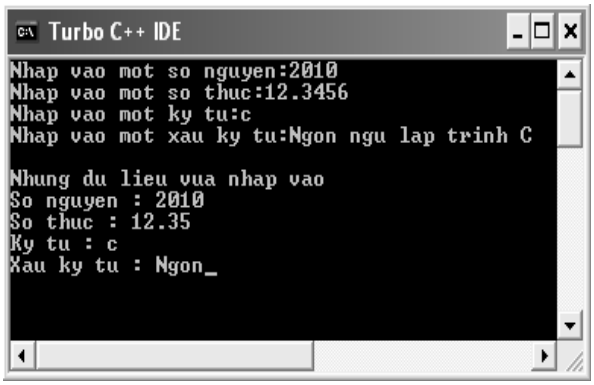
Ví dụ

```
#include <conio.h>
#include <stdio.h>
void main(){
    // khai bao bien
    int a; float x;
    char ch; char str[30];
    // Nhap du lieu
    printf("Nhap vao mot so nguyen:"); scanf("%d",&a);
    printf("\nNhap vao mot so thuc:"); scanf("%f",&x);
    printf("\n Nhap vao mot ki tu:");
    fflush(stdin); scanf("%c",&ch);
```


Ví dụ

```
    printf("\nNhap vao mot xau ki tu:");
    fflush(stdin); scanf("%s",str);

    // Hien thi du lieu vua nhap vao
    printf("\nNhung du lieu vua nhap vao");
    printf("\nSo nguyen : %d",a);
    printf("\nSo thuc : %5.2f",x);
    printf("\nKy tu : %c",ch);
    printf("\nXau ky tu : %s",str);
    getch();
}
```


Ví dụ→Kết quả thực hiện



Các quy tắc cần lưu ý

Khi đọc số

- Hàm scanf() quan niệm rằng mọi kí tự số, dấu chấm ('.') đều là kí tự hợp lệ.
  - Số thực dấu phẩy động, chấp nhận ký tự e/E
- Khi gặp các dấu phân cách như tab, xuống dòng hay dấu cách (space bar), scanf() sẽ hiểu là kết thúc nhập dữ liệu cho một số

Các quy tắc cần lưu ý

Khi đọc kí tự

Hàm **scanf()** cho rằng mọi kí tự có trong bộ đệm của thiết bị vào chuẩn đều là hợp lệ, kể cả các kí tự tab, xuống dòng hay dấu cách

Khi đọc xâu kí tự:

Hàm **scanf()** nếu gặp các kí tự dấu trắng, dấu tab hay dấu xuống dòng thì nó sẽ hiểu là kết thúc nhập dữ liệu cho một xâu kí tự.

Ghi chú:

Trước khi nhập dữ liệu kí tự hay xâu kí tự nên dùng lệnh **fflush(stdin)** để xóa bộ đệm.

Ví dụ: Đọc 2 số nguyên, đưa ra tổng, hiệu, tích…

```
#include <stdio.h>
int main(){
    int A, B;
    printf("Nhap vao 2 so nguyen : "); scanf("%d %d",&A,&B);
    printf("\n");
    printf("Tong %d + %d = %d \n", A, B, A + B);
    printf("Hieu %d - %d = %d\n", A, B, A - B);
    printf("Tich %d x %d = %d\n", A, B, A * B);
    printf("Thuong %d / %d = %.3f\n", A, B, (float)A / B);
    printf("Chia nguyen %d / %d = %d\n", A, B, A / B);
    printf("Chia du %d  %% %d = %d\n", A, B, A % B);
    printf("\n");
    return 0;
}
```


Ví dụ: Đọc 2 số nguyên, đưa ra tổng, hiệu, tích…

```
Nhap vao 2 so nguyen : 17 5

Tong 17 + 5 = 22
Hieu 17 - 5 = 12
Tich 17 x 5 = 85
Thuong 17 / 5 = 3.400
Chia nguyen 17 / 5 = 3
Chia du 17  % 5 = 2
```


Bài tập

1. Viết chương trình nhập vào từ bàn phím chiều dài 3 cạnh của một tam giác, rồi đưa ra diện tích và các đường cao của tam giác
2. Nhập vào từ bàn phím tọa độ 3 điểm A,B,C rồi đưa ra độ dài các cạnh của tam giác ABC và của đường trung tuyến AM
3. Cho hàm số: $f(x) = x^7 + 5\sqrt[3]{x^5 + 3x^3 + 2} + 12$
   Viết chương trình nhập vào 3 số thực a,b,c và đưa ra trung bình cộng của f(a),f(b),f(c)
4. Nhập x vào từ bàn phím và tính giá trị của biểu thức $A = \frac{\cos 3a + \sqrt[5]{2x^3 + x + 1}}{\log_7(3^{x^2} + 2.14b)}$ trong đó $a = \sqrt{2^x + \pi}$ và $b = \ln(e^{x+1.23} + 1)$

Ví dụ: Đọc tọa độ 3 điểm A,B,C và đưa ra d/tích ΔABC

```
#include <stdio.h>
#include <math.h>
int main(){
    float Ax,Ay, Bx, By, Cx, Cy, AB, BC, CA,p;
    printf("Nhap vao toa do diem A  : "); scanf("%f %f",&Ax,&Ay);
    printf("Nhap vao toa do diem B  : "); scanf("%f %f",&Bx,&By);
    printf("Nhap vao toa do diem C  : "); scanf("%f %f",&Cx,&Cy);
    //Tinh do dai cac canh cua tam giac
    AB = sqrt((Ax-Bx)*(Ax-Bx)+(Ay-By)*(Ay-By));
    BC = sqrt((Bx-Cx)*(Bx-Cx)+(By-Cy)*(By-Cy));
    CA = sqrt((Cx-Ax)*(Cx-Ax)+(Cy-Ay)*(Cy-Ay));
    p = (AB +BC +CA)/2;
    printf("Dien tich tam giac ABC la: %f",sqrt(p*(p-AB)*(p-BC)*(p-CA)));
    printf("\n");
    return 0;
}
```


Ví dụ: Đọc tọa độ 3 điểm A,B,C và đưa ra d/tích ΔABC

```
Nhap vao toa do diem A  : 0 0
Nhap vao toa do diem B  : 6 0
Nhap vao toa do diem C  : 0 8
Dien tich tam giac ABC la: 24.000000
```

```
Nhap vao toa do diem A  : -1 -2
Nhap vao toa do diem B  : 5 -2
Nhap vao toa do diem C  : 6 6
Dien tich tam giac ABC la: 24.000000
```

```
Nhap vao toa do diem A  : 1 1
Nhap vao toa do diem B  : 2 2
Nhap vao toa do diem C  : 4 4
Dien tich tam giac ABC la: 0.000000
```


Nội dung chính

1. Các hàm vào ra cơ bản:
   - **printf()**
   - **scanf()**
2. Các hàm vào ra khác
   - **gets()**
   - **puts()**
   - **getch()**

⇐ Cần nạp thư viện conio.h
#include <conio.h>

gets()

- Mục đích:
  - Dùng để nhập vào từ bàn phím một xâu kí tự **bao gồm cả dấu cách**, điều mà hàm **scanf()** không làm được.
- Cú pháp :

  gets (xâu_kí_tự);
- Ví dụ:

```
char str [40];
printf("Nhap vao mot xau ki tu:");
fflush(stdin);
gets(str);
```


puts()

- Mục đích:
  - Hiển thị ra màn hình nội dung xâu_kí_tự và sau đó đưa con trỏ xuống dòng mới
- Cú pháp:

```
puts(xâu_kí_tự);
```

- Ví dụ:

  puts("Nhap vao xau ki tu:");

  Tương đương với lệnh:

  printf("%s\n","Nhap vao xau ki tu:").

getch()

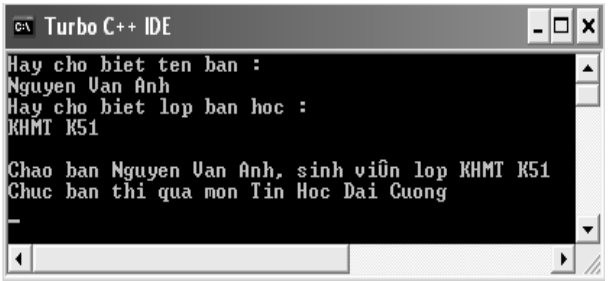
- Mục đích
  - Đợi đọc một ký tự bàn phím
  - Thường dùng để chờ người sử dụng ấn một phím bất kì trước khi kết thúc chương trình.
- Cú pháp

```
getch();
```


Ví dụ

```
#include <conio.h>
#include <stdio.h>
void main(){
   char ten[30], lop[10]; //Kieu chuoi, mang ky tu
   puts("Hay cho biet ten ban : ");
   fflush(stdin); gets(ten);
   puts("Hay cho biet lop ban hoc : ");
   fflush(stdin); gets(lop);

   printf("\nChao ban %s, sinh viên lop %s\n",ten,lop);
   puts("Chuc ban thi qua mon Tin Hoc Dai Cuong");
   getch();
}
```


Ví dụ → Kết quả thực hiện



Tóm tắt

- Các hàm cơ bản (stdio.h)
  - printf()/ scanf ()
  - Xâu định dạng:
    %[flags][width][.precision][l][t]
    - **flags** (+/-,#): Xác định sự căn lề
    - **l** (l/L): Biến ở dạng **long**
    - **t** (d/i/o/u/x/X/f/e/E/g/G/c/s/%): kiểu hiện thị
- Các hàm khác (conio.h)
  - puts() / gets() / getch()
- Tìm hiểu thêm
  - fprintf() / fscanf() ← Vào/ra từ file
  - sprintf(f) / sscanf() ← Vào ra từ xâu ký tự

Nội dung chính

- Chương 1: Tổng quan về ngôn ngữ C
- Chương 2: Kiểu dữ liệu và biểu thức trong C
- Chương 3: Vào ra dữ liệu
- Chương 4: Cấu trúc điều khiển
- Chương 5: Mảng, con trỏ và xâu ký tự
- Chương 6: Cấu trúc
- Chương 7: Hàm
- *Chương 8: Tệp dữ liệu*

Nội dung chính

1. Cấu trúc lệnh khối
2. Cấu trúc rẽ nhánh
   - Cấu trúc if, if … else
   - Cấu trúc lựa chọn switch
3. Cấu trúc lặp
   - Vòng lặp **for**
   - Vòng lặp **while** và **do while**
4. Các lệnh thay đổi cấu trúc lập trình
   - Câu lệnh **continue**
   - Câu lệnh **break**

Lệnh đơn >< Lệnh ghép

- Lệnh đơn:
  - Là biểu thức theo sau bởi dấu '**;**'
  - ***Ví dụ:*** x= 0; i++; printf("Hello");
- Lệnh ghép (khối lệnh)
  - Là tập hợp các câu lệnh (*đơn và **ghép***) được đặt trong cặp ngoặc nhọn **{ }**
  - **C** cho phép khai báo biến trong một khối lệnh
    - Phần khai báo phải nằm trước các câu lệnh
  - ***Chú ý:***
    - Lệnh ghép có thể đặt tại bất cứ chỗ nào mà cú pháp cho phép đặt 1 câu lệnh đơn
    - Không đặt dấu '**;**' sau một lệnh khối

Cấu trúc lồng nhau

- Trong lệnh ghép chứa lệnh ghép khác
  - Sự lồng nhau không hạn chế

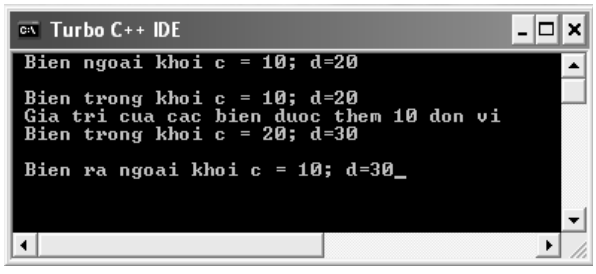
```
{//Khai báo đối tượng cục bộ trong khối
    lenh;
    {//Khai báo đối tượng cục bộ trong khối
        lenh;
        ...
    }
    ...
}
```

Nếu các đối tượng được khai báo trùng tên nhau ?

Ví dụ

```
#include <conio.h>
#include <stdio.h>
void main(){  // ham main() cung la mot khoi lenh
   int  c = 10,  d= 20;
   printf(" Bien ngoai khoi c = %d; d=%d ",c,d);
   {    int  c = 10;
        printf("\n Bien trong khoi c = %d; d=%d",c,d);
        printf("\n Gia tri cua cac bien duoc them 10 don vi");
        c = c + 10; d= d + 10;
        printf("\n Bien trong khoi c = %d; d=%d",c,d);
   }
    printf("\n Bien ra ngoai khoi c = %d; d=%d",c,d);
   getch();
} // ket thuc khoi lenh cua ham main()
```


Ví dụ→Kết quả thực hiện

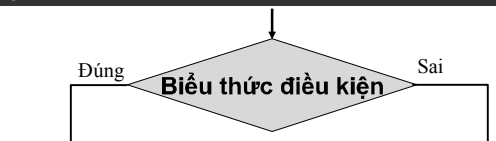


Biến địa phương / Biến toàn cục

Nội dung chính

1. Cấu trúc lệnh khối
2. Cấu trúc rẽ nhánh
   - Cấu trúc if, if … else
   - Cấu trúc lựa chọn switch
3. Cấu trúc lặp
   - Vòng lặp **for**
   - Vòng lặp **while** và **do while**
4. Các lệnh thay đổi cấu trúc lập trình
   - Câu lệnh **continue**
   - Câu lệnh **break**

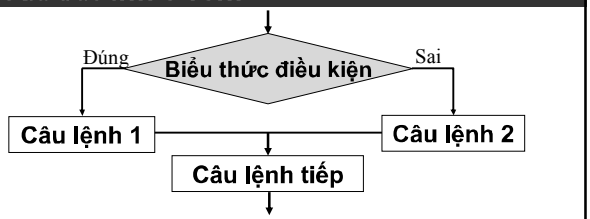
Cấu trúc **if …**




```
if (Biểu thức điều kiện)
    Câu lệnh;
Câu lệnh kế tiếp;
```

```
if (n % 2 == 0)
    printf("so chan");
```


Cấu trúc **if….else….**




```
if (Biểu thức điều kiện)
    Câu lệnh 1;
else
    Câu lệnh 2;
Câu lệnh kế tiếp;
```

```
if (x > y)
    z = x;
else
    z = y;
printf("max: %d", z);
```


Lưu ý

Biểu thức điều kiện:

- Là biểu thức trả về giá trị logic đúng/sai
  - Giá trị logic đúng/True : khác **0**
  - Giá trị logic sai/False: bằng **0**

Ví dụ

if (2+5) printf("Hello world! ");→Chấp nhận

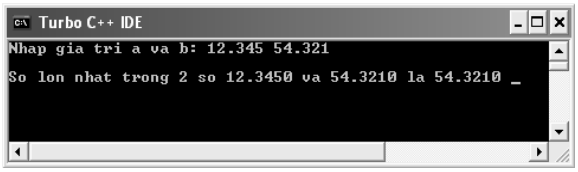
Câu lệnh:

Có thể là một lệnh khối (*Đặt trong cặp { }*)

Ví dụ: So sánh 2 số thực được nhập vào

```
#include <conio.h>
#include <stdio.h>
void main()
{
   float a, b; float max; // khai bao bien
   printf(" Nhap gia tri a va b: ");
   scanf("%f %f",&a,&b);
   if(a < b)
        max = b;
   else
        max = a;
   printf("\nSo lon nhat trong 2 so %.4f va %.4f la %.4f ",a,b,max);
   getch();
} //ket thuc ham main()
```


Ví dụ: So sánh 2 số thực được nhập vào




```
if(a < b)
  max = b;
else
  max = a;
```

⟺ **max = a > b ? a : b;**

Ví dụ: Giải phương trình **ax + b = 0**

```
#include <stdio.h>
#include <conio.h>
void main()
{ float a, b;
  printf("\nGiai phuong trinh bac nhat ax + b = 0");
  printf("\nCho biet he so a b : ");  scanf("%f%f", &a, &b);
  if (a==0)
        if (b!=0)
              printf("Phuong trinh vo nghiem");
        else
              printf("Phuong trinh vo so nghiem");
  else
        printf("Dap so cua phuong trinh tren = %f", -b/a);
  getch();
} //ket thuc ham main()
```


Giải phương trình **ax + b = 0** → Thực hiện

```
Giai phuong trinh bac nhat ax + b = 0
Cho biet he so a b : 3 17
Dap so cua phuong trinh tren = -5.666667

Giai phuong trinh bac nhat ax + b = 0
Cho biet he so a b : 0 0
Phuong trinh vo so nghiem

Giai phuong trinh bac nhat ax + b = 0
Cho biet he so a b : 0 6
Phuong trinh vo nghiem

Giai phuong trinh bac nhat ax + b = 0
Cho biet he so a b : 5 -123
Dap so cua phuong trinh tren = 24.600000
```


Ví dụ: Nhập x và tính hàm

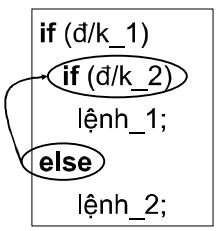
```
#include <stdio.h>
#include <math.h>
void main()
{
   float x, fx;
   printf("\nNhap x: "); scanf("%f",&x);
   if(x < 3)
        fx = x*x+pow(sin(2*M_PI*x),4)+1;
   else
     if (x==3)
        fx = 5;
     else
        fx = sqrt(x-3) +log10(x*x-3);
   printf("\n Ket quả: %.4f",fx);
}
```

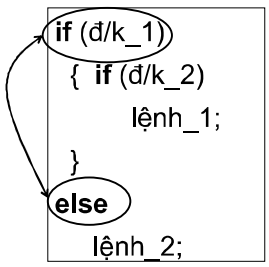
$$f(x)=\begin{cases} x^2+\sin^4 2\pi x+1 & \text{khi } x<3 \\ 5 & \text{khi } x=3 \\ \sqrt{x-3}+\log_{10}(x^2-3) & \text{khi } x>3 \end{cases}$$

```
Nhap x: 1.0
Ket qua: 2.0000

Nhap x: 3
Ket qua: 5.0000

Nhap x: 5.0
Ket qua: 2.7566
```


Cấu trúc **if / if… else** lồng nhau

Nếu cấu trúc **if..** và **if …else** có thể lồng nhau. Khi đó **else sẽ** tương ứng với **if** (phía trên, chưa có **else)** gần nhất





Cấu trúc **if / if… else** lồng nhau →Ví dụ

```
int a, b, c = 10;
```

```
if (a==0)
   if (b == 0)
      c = 20;
else
     c = 30;
```

a ≠ 0, b=? → c = 10

a=0, b=0 →c = 20

a=0, b ≠ 0 →c = 30

```
if (a==0){
   if (b == 0)
      c = 20;
}else
     c = 30;
```

a ≠ 0, b=? → c = 30

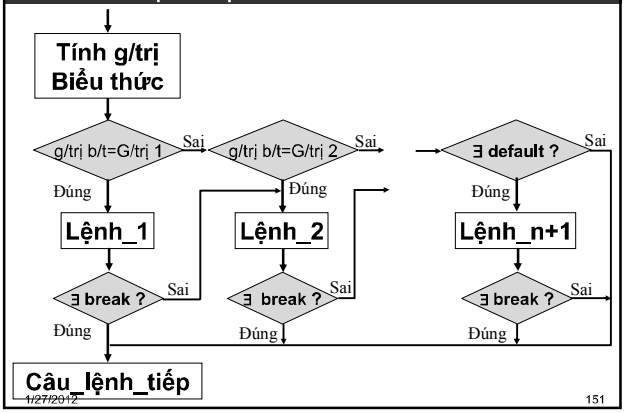
a=0, b=0 →c = 20

a=0, b ≠ 0 → c = 10

Cấu trúc lựa chọn **switch**

```
switch (bieu_thuc)
   {
     case gia_tri_1: lenh_1; [break];
     case gia_tri_2: lenh_2; [break];
     …
     [default: lenh_n+1; [break];]
   }
Câu_lệnh_tiếp
```

Cấu trúc lựa chọn **switch**



Cấu trúc lựa chọn **switch**

Cơ chế hoạt động

- Tính giá trị của **biểu_thức**,
- So sánh giá trị của **biểu_thức** với các **giá_trị_k** (*với k = 1, 2, … n*) nằm sau các từ khóa **case**
  - Xảy ra 2 khả năng

Cấu trúc lựa chọn **switch**→cơ chế hoạt động

- Tồn tại ***giá_trị_i*** bằng giá trị biểu thức.
  - Thực hiện ***lệnh_i***
    - Nếu tồn tại lệnh ***break***,
      - Nhảy tới tiếp tục thực hiện ***Câu_lệnh_tiếp*** nằm sau cấu trúc ***switch***
    - Nếu không tồn tại lệnh ***break***
      - Thực hiện các lệnh sau ***lệnh_i*** cho tới khi gặp ***break*** hoặc tới khi thoát khỏi cấu trúc ***switch***
      - Thực hiện ***Câu_lệnh_tiếp***

Cấu trúc lựa chọn **switch**→cơ chế hoạt động

- Không tồn tại ***giá_trị_i*** (*i = 1, 2, …n*) nào bằng giá trị biểu thức
  - Nếu có nhãn **default**:
    - Chương trình sẽ thực hiện ***lệnh_n+1***
    - Thực hiện ***Câu_lệnh_tiếp*** nằm ngay sau cấu trúc **switch**.
  - Nếu không có nhãn **default**:
    - Chương trình chuyển sang thực hiện lệnh tiếp theo nằm ngay sau cấu trúc **switch**: ***Câu_Lệnh_tiếp***

Cấu trúc lựa chọn **switch**→Ví dụ 1

Lập trình đọc từ bàn phím một số nguyên 1≤N ≤10 và đưa ra từ tiếng Anh tương ứng

Cấu trúc lựa chọn **switch**→Ví dụ 1

```
#include <stdio.h>
void main(){
    int N;
    printf("\nNhap mot gia tri so nguyen khong am: "); scanf("%d",&N);
    switch(N) {
          case 1: printf(" %d -> One \n",N); break;
          case 2: printf("%d -> Two \n",N); break;
          case 3: printf("%d -> Three \n",N); break;
          case 4: printf("%d -> Four \n",N); break;
          case 5: printf("%d -> Five \n",N); break;
          case 6: printf("%d -> Six \n",N); break;
          case 7: printf("%d -> Seven \n",N); break;
          case 8: printf("%d -> Eight \n",N); break;
          case 9: printf("%d -> Nine \n",N); break;
          case 10: printf("%d -> Ten \n",N); break;
          default : printf("Khong thoa man dieu kien [1..10] \n");
    }
}
```

Cấu trúc lựa chọn **switch**→ Thực hiện

```
Nhap mot gia tri so nguyen khong am: 7
7 -> Seven

Nhap mot gia tri so nguyen khong am: 3
3 -> Three

Nhap mot gia tri so nguyen khong am: -6
Khong thoa man dieu kien [1..10]
```

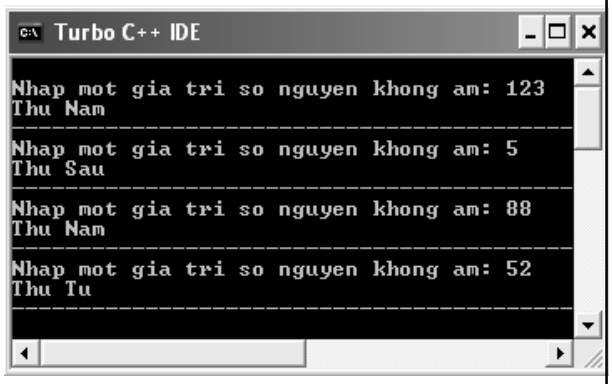

Cấu trúc lựa chọn **switch**→Ví dụ 2

Nhập vào số nguyên không âm, đưa ra ngày trong tuần tương ứng (*theo số dư khi chia cho 7*).

Cấu trúc lựa chọn **switch**→Ví dụ 2

```
#include <conio.h>
#include <stdio.h>
void main(){
   int a;
   printf("\nNhap mot gia tri so nguyen khong am: "); scanf("%d",&a);
   switch(a % 7)  {
        case 0: printf(" Chu nhat"); break;
        case 1: printf(" Thu Hai"); break;
        case 2: printf(" Thu Ba"); break;
        case 3: printf(" Thu Tu"); break;
        case 4: printf(" Thu Nam"); break;
        case 5: printf(" Thu Sau"); break;
        case 6: printf(" Thu Bay"); break;
   }
   getch();
}
```


Cấu trúc lựa chọn **switch** →Thực hiện



Cấu trúc lựa chọn **switch**

Có thể sử dụng đặc điểm Không có lệnh break chương trình sẽ tự động chuyển xuống thực hiên các câu lệnh tiếp sau *để viết chung mã lệnh cho các trường hợp khác nhau nhưng được xử lý như nhau*

Ví dụ: Trong một năm các tháng có 30 ngày là 4, 6, 9, 11 còn các tháng có 31 ngày là 1, 3, 5, 7, 8, 10, 12. Riêng tháng hai có thể có 28 hoặc 29 ngày. Hãy viết chương trình nhập vào 1 tháng, sau đó đưa ra kết luận tháng đó có bao nhiêu ngày

Cấu trúc lựa chọn **switch**→ Ví dụ

```
#include <conio.h>
#include <stdio.h>
void main () {
   int thang; clrscr();
   printf("\nNhap vao thang trong nam "); sanf("%d",&thang);
   switch(thang) {
       case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:  printf("\n Thang %d co 31 ngay ",thang);
                  break;
```

Cấu trúc lựa chọn **switch**

```
    case 4:
    case 6:
    case 9:
    case 11: printf("\n Thang %d co 30 ngay ",thang);
             break;
    case 2:  printf ("\ Thang 2 co 28 hoac 29 ngay");
             break;
    default : printf("\n Khong co thang %d", thang);
             break;
  }
  getch();
}
```


Cấu trúc lựa chọn **switch→Lưu ý**

- Giá trị của **biểu thức** trong cấu trúc **switch** phải là số nguyên (*kiểu đếm được*)
  - Phải có kiểu dữ liệu là **char**, **int**, **long**
- Các giá trị sau từ khóa **case** (*gia_tri_1, gia_tri_2,..*) cũng phải là số nguyên

Điều kiện trong cấu trúc **if / if..else** cho phép làm việc với các kiểu dữ liệu khác số nguyên

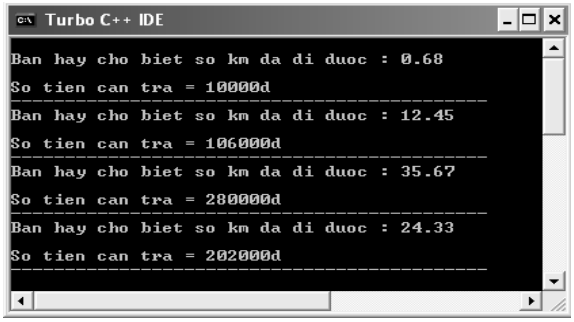
Các ví dụ

1. Viết chương trình tính cước Taxi theo công thức:
   - 1 km đầu tiên có cước là 10000đ,
   - 30 km tiếp theo có giá là 8000đ/1km
   - Các km sau đó có giá là 6000đ/1km.
2. Viết chương trình giải phương trình bậc hai $ax^2 + bx + c = 0$
3. Viết chương trình giải hệ phương trình bậc nhất $\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$

Ví dụ 1: Tính cước taxi

```
#include <stdio.h>
#include <math.h>
void main() {
    unsigned long sotien;
    float sokm;
    printf("\nBan hay cho biet so km da di duoc : ");
    scanf("%f", &sokm);
    if (sokm <= 1.0)
        sotien = 10000;
    else
      if (sokm <= 31.0)
            sotien = 10000 + (ceil(sokm) - 1.0 ) * 8000;
       else
            sotien = 250000 + (ceil(sokm) - 31) * 6000;
    printf("\nSo tien can tra = %lud", sotien);
}
```


Ví dụ 1 → Thực hiện chương trình




```
sotien= sokm <=1.0 ? 10000  : sokm <= 31 ? 10000 +
(ceil(sokm) - 1.0 ) * 8000 : 250000+(ceil(sokm) - 31) * 6000;
```


Ví dụ 2: Giải phương trình bậc 2

```
#include <stdio.h>
#include <math.h> //Để sử dụng hàm toán học sqrt
void main(){
  float a, b, c, delta;
  printf("\n\nNhap he so a b c : ");  scanf("%f%f%f", &a, &b, &c);
  delta = b * b - 4 * a * c;
  if( a==0)  printf("P/trinh suy bien thanh p/trinh bac 1 %fx+%f=0",b,c);
  else  if (delta < 0)    printf("Phuong trinh vo nghiem");
  else  if (delta == 0)
     printf("Phuong trinh co nghiem kep x1 = x2 = %f", -b/(2*a));
  else
     printf("Phuong trinh co hai nghiem phan biet\n  x1=%f \n  x2=%f",
       (-b + sqrt(delta))/(2*a), (-b - sqrt(delta))/(2*a) );
}
```

Ví dụ 2→ Thực hiện chương trình

```
Nhap he so a b c : 0 3 2
P/trinh suy bien thanh p/trinh bac 1 3.000000x+2.000000=0

Nhap he so a b c : 1 2 3
Phuong trinh vo nghiem

Nhap he so a b c : 1 4 4
Phuong trinh co nghiem kep x1 = x2 = -2.000000

Nhap he so a b c : 1 -3 2
Phuong trinh co hai nghiem phan biet
  x1 = 2.000000
  x2 = 1.000000_
```


Ví dụ 3: Giải hệ phương trình

```
#include <stdio.h>
void main() {
    float a1,b1,c1,a2,b2,c2,x,y,dx,dy,d;
    printf("\n\nNhap cac so:\n");
    printf("a1,b1,c1=");scanf("%f%f%f",&a1,&b1,&c1);
    printf("a2,b2,c2=");scanf("%f%f%f",&a2,&b2,&c2);
    d  = a1 * b2 - a2 * b1;
    dx = c1 * b2 - c2 * b1;
    dy = a1 * c2 - a2 * c1;
    if (d != 0) {
        x = dx/d; y = dy/d;
        printf("He PT co nghiem x=%f, y=%f\n",x,y);
    }else
        if (dx==0) printf("He PT co vo so nghiem!\n");
        else printf("He phuong trinh vo nghiem!");
}
```


Ví dụ 3 → Thực hiện chương trình

```
Nhap cac so:
a1,b1,c1 = 3 5 8
a2,b2,c2 = 2 1 9
He PT co nghiem x=5.285714, y=-1.571429

Nhap cac so:
a1,b1,c1 = 1 2 3
a2,b2,c2 = 1 2 4
He phuong trinh vo nghiem!_

Nhap cac so:
a1,b1,c1 = 1 2 3
a2,b2,c2 = 2 4 6
He PT co vo so nghiem!
```


Bài tập

1. Viết chương trình nhập vào một ký tự hệ hexa và đưa ra giá trị hệ 10 tương ứng
2. Lập trình đọc tọa độ 4 điểm A,B,C,M rồi kiểm tra xem điểm M nằm trong, nằm trên cạnh hay nằm ngoài tam giác ABC.
3. Lập trình đọc vào từ bàn phím 2 giá trị a, b rồi tính y = 15 x^2+x+7.2 trong đó

$$x = \begin{cases} \dfrac{a+b}{3} & \text{nê'u } a < b \\ 1.5172 & \text{nê'u } a = b \\ \dfrac{a-b}{a^2+b^2} & \text{nê'u } a > b \end{cases}$$

Nội dung chính

1. Cấu trúc lệnh khối
2. Cấu trúc rẽ nhánh
   - Cấu trúc if, if … else
   - Cấu trúc lựa chọn switch
3. Cấu trúc lặp
   - Vòng lặp **for**
   - Vòng lặp **while** và **do while**
4. Các lệnh thay đổi cấu trúc lập trình
   - Câu lệnh **continue**
   - Câu lệnh **break**

Các cấu trúc lặp

- Vòng lặp **for**
- Vòng lặp **while**
- Vòng lặp **do while**

for

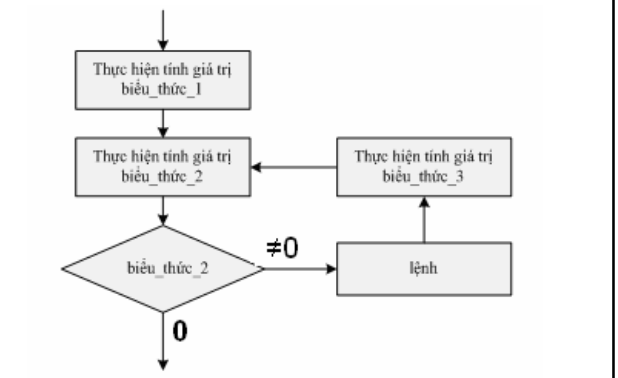
Mục đích và cú pháp

Dùng để lặp công việc một số chính xác lần đã định trước dựa vào sự biến thiên của biến điều khiển

for([b.thuc_1];[b.thuc_2];[b.thuc_3]) Lệnh;

- **b.thuc_1:** Khởi tạo giá trị ban đầu cho vòng lặp
- **b.thuc_2:** Điều kiện tiếp tục vòng lặp
- **b.thuc_3:** Thay đổi biến điều khiển của vòng lặp
- **Lệnh:** Có thể là lệnh đơn lệnh kép hoặc lệnh rỗng

Sơ đồ cú pháp



Sử dụng

```
int i;
for(i = 0; i < 100; i++) Câu_lệnh;
```

```
int i;
for(i = 0; i < 100; i+=2)Câu_lệnh;
```

```
int i;
for(i = 100; i > 0; i--) Câu_lệnh;
```

```
for(int i = 0; i < 100; i++) Lệnh;
for(int i = 100; i > 0; i--) Lệnh;
```

Turbo C++ 3.0, văn bản nguồn **.cpp** (**c++**)

Ví dụ 1 : Đưa ra các số nguyên lẻ nhỏ hơn 100

```
#include <stdio.h>
#include <conio.h>
void main(){
   int i;
   for(i = 1;i<100;i++)   {
        if(i%2 == 1) printf("%5d",i);
        if((i+1)%20 ==0) printf("\n");
   }
   getch();
}
```

```
  1   3   5   7   9  11  13  15  17  19
 21  23  25  27  29  31  33  35  37  39
 41  43  45  47  49  51  53  55  57  59
 61  63  65  67  69  71  73  75  77  79
 81  83  85  87  89  91  93  95  97  99
```


Ví dụ 2 : Đưa ra các số nguyên lẻ nhỏ hơn 100

```
#include <stdio.h>
#include <conio.h>
void main(){
   int i;
   for(i = 99;i > 0;i-=2)  {
        printf("%5d",i);
        if( (i-1) % 20 == 0) printf("\n");
   }
   getch();
}
```

```
 99  97  95  93  91  89  87  85  83  81
 79  77  75  73  71  69  67  65  63  61
 59  57  55  53  51  49  47  45  43  41
 39  37  35  33  31  29  27  25  23  21
 19  17  15  13  11   9   7   5   3   1
```

Ví dụ 3→ Nhập n và đưa ra n!

```
#include <stdio.h>
#include <conio.h>
void main()
{
  long P = 1;
   int n, i;
   printf("Nhap n : ");scanf("%d",&n);
  for(i = 1;i<=n;i++)
     P = P * i;
   printf("Ket qua là %ld ",P);
  getch();
}
```

Nhap n : 6
Ket qua là 720

Ví dụ 4 → Nhập n và tính tổng 1 +1/2+..+1/n

```
#include <stdio.h>
#include <conio.h>
void main()
{
  float S = 0.0;
   int n, i;
   printf("Nhap n : ");scanf("%d",&n);
  for(i = 1;i <= n;i++)
     S = S + (float)1/i;
   printf("Ket qua là %7.4f ",S);
  getch();
}
```

Nhap n : 10
Ket qua là 2.9290

Ví dụ 5→ Tìm số 3 chữ số thỏa mãn abc=$a^3+b^3+c^3$

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a, b, c;
  for(a = 1;a<=9;a++)
   for(b = 0;b<=9;b++)
    for(c = 0;c<=9;c++)
     if(a*a*a+b*b*b+c*c*c==100*a+10*b+c)
       printf("%d \n",100*a+10*b+c);
  getch();
}
```


Ví dụ 5→ Tìm số 3 chữ số thỏa mãn abc=$a^3+b^3+c^3$

```
#include <stdio.h>
#include <conio.h>
void main()
{int i, a, b, c;
  for(i = 100;i<1000;i++){
     a = i / 100;
     b = i % 100 / 10;
     c = i % 100 % 10;
     if(a*a*a+b*b*b+c*c*c == i)
       printf("%d \n",i);
  }//for
  getch();
}
```

153
370
371
407

Chú ý

Không nhất thiết phải có đầy đủ các thành phần trong vòng lặp **for**

int getchar(): đọc ký tự từ vùng đệm bàn phím. Nếu vùng đệm rỗng, đợi người dùng gõ dãy ký tự (*cho tới khi ấn phím Enter*), sẽ trả về ký tự đầu

putchar(int c): đưa ký tự ra màn hình

Chú ý

1. Biểu thức khởi tạo

```
char c; int i=0;
for( ; (c=getchar())! = '\n' ; i++)
    putchar(c);
printf("\nSo ky tu: %d",i);
```

Hello world
Hello world
So ky tu: 11

2. Biểu thức điều khiển

```
for(i=0 ; ; c=getchar(), i++)
    if(c=='\n') break;
printf("\nSo ky tu: %d",i);
```

Hello world

So ky tu: 12

3. Thân vòng lặp

```
for(i=0 ; getchar() != '\n', i++);
printf("\nSo ky tu: %d",i);
```

Hello world
So ky tu: 11

while

Mục đích & Cú pháp

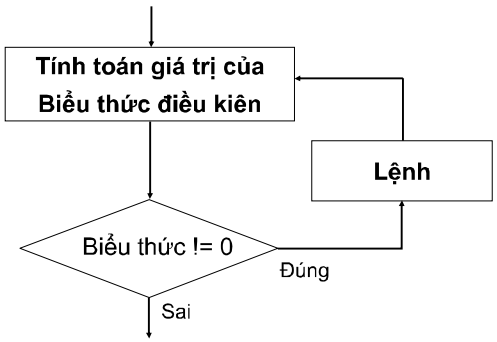
Dùng để thực hiện lặp đi lặp lại một công việc nào đó với số lần lặp **không** xác định.

Cú pháp:

```
while(bieu_thuc_dieu_kien)
    lenh;
```

- Chương trình kiểm tra điều kiện **trước** khi lặp
  - Giá trị của biểu thức điều kiện là đúng ⇒ thực hiện lệnh
- Các **lenh** của vòng lặp có thể không được thực hiện lần nào ⇐Biểu_thức_điều_kiện sai ngay từ đầu
- Biểu_thức_điều_kiện luôn đúng ⇒ lặp vô hạn lần

Sơ đồ cú pháp



Nhập n và đưa tổng của n số nguyên đầu tiên

```
#include <stdio.h>
#include <conio.h>
void main(){
  long S = 0;
   int n;
   printf("Nhap n : ");scanf("%d",&n);
  while (n > 0){
     S = S + n;
     n = n - 1;
   }
   printf("Ket qua la %ld ",S);
  getch();
}
```

while (n> 0)
S += n–;

```
Nhap n : 96
Ket qua la 4656
```


Tìm số nguyên lớn nhất thỏa mãn $3n^5-317n < 5$

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
  clrscr();
  int n=0;
  while (3* pow(n,5) - 317*n < 5) n++;
  printf("%4d",n-1);
  getch();
}
```

```
n =  10
while (3*pow(n,5)-317*n >= 5)
      n--;
```

n = 3

Cho biết kết quả thực hiện chương trình

```
#include <stdio.h>
#include <conio.h>
void main()
{
  clrscr();
  int i=3;
  while (i > 1){
    if(i % 2==0) i = i / 2;
    else i = i * 3 + 1;
    printf("%4d",i);
   }
  getch();
}
```

10 5 16 8 4 2 1

Nhập chuỗi và đếm số nguyên âm, phụ âm, khoảng trắng

```
#include <stdio.h>
#include <conio.h>
void main()
{   int na, pa, kt;          char c;
    na = pa = kt = 0;
    clrscr(); printf(">");
    while( (c=getchar()) !='\n'){
          switch(c){
             case 'a':  case 'e': case 'i': case 'o': case 'u' :
             case 'A':  case'E': case 'I': case'0':  case 'U' : na++; break;
             case ' ': kt++; break;
             default : pa++;
          }
    }
    printf("Chuoi co :%d nguyen am :%d phu am va %d khoang trang",na,pa,kt);
}
```

```
>Hello world
Chuoi co :3 nguyen am :7 phu am va 1 khoang trang_
```


do..while

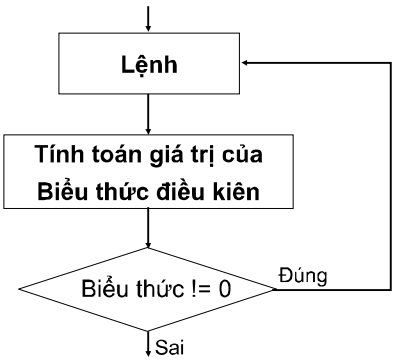
Mục đích & Cú pháp

- Dùng để thực hiện lặp đi lặp lại một công việc nào đó với số lần lặp **không** xác định.
- **Cú pháp:**

```
do{
    lenh;
}while(bieu_thuc_dieu_kien);
```

- Chương trình kiểm tra điều kiện **sau** khi lặp
- Các `lenh` được thực hiện ít nhất một lần
- Biểu thức luôn đúng, lặp vô hạn lần

Sơ đồ cú pháp



Nhập n và đưa tổng của n số nguyên đầu tiên

```
#include <stdio.h>
#include <conio.h>
void main(){
  long S = 0;
   int n;
   printf("Nhap n : ");scanf("%d",&n);
  do {
     S = S + n;
      n = n - 1;
   }while (n > 0);
   printf("Ket qua là %ld ",S);
  getch();
}
```

```
do
      S += n--;
while (n> 0);
```

```
Nhap n : 96
Ket qua là 4656
```


Ví dụ

- Nhập vào điểm của một sinh viên, nếu điểm đó không ∈ [0, 10] thì thông báo cho người dùng nhập lại.
- Cách làm:
  - Nếu dùng lệnh `if` Chỉ kiểm tra được 1 lần
  - Không dùng `for` được vì chưa biết trước số lần lặp.

    Sử dụng vòng lặp `while/ do while`

Dùng vòng lặp **while**

```
#include <stdio.h>
void main(){
   float diem; clrscr();
   printf("Chuong trinh nhap diem sinh vien\n");
   printf("Nhap diem (0<=diem<=10):");
    scanf("%f",&diem);
   while (diem < 0 || diem > 10)  {
      printf("\nBan nhap khong dung!\n");
      printf("Ban hay nhap lai (0<=diem<=10):");
      scanf("%f",&diem);
   }
   printf("\nDiem ban vua nhap la: %.2f", diem);
}
```


Dùng vòng lặp **while** → Kết quả

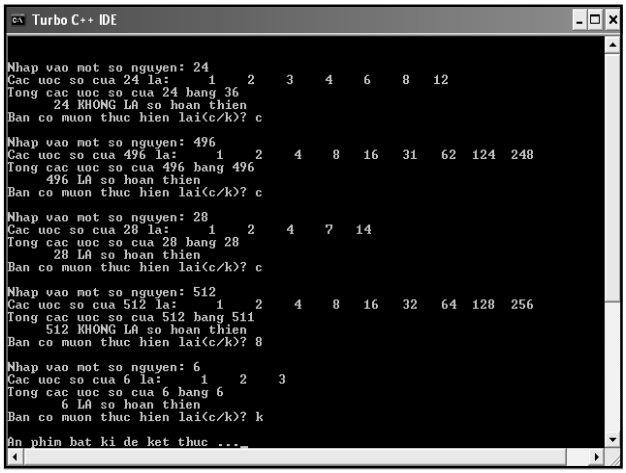
Dùng vòng lặp **do…while**

```
#include <stdio.h>
void main() {
   float diem; clrscr();
   printf("Chuong trinh nhap diem sinh vien\n");
   do {
      printf("Nhap diem (0<=diem<=10):");
      scanf("%f",&diem);
      if (diem < 0 || diem > 10)
           printf("\nBan nhap khong dung!\n");
   } while (diem < 0 || diem > 10);
   printf("\nDiem ban vua nhap la: %.2f", diem);
}
```


Nhập số và kiểm tra số hoàn thiện

fflush(): xóa vùng đệm bàn phím
toupper(): chuyển sang chữ hoa
getche(): Đọc ký tự từ vùng đệm & hiện thị lên màn hình (**getch()** thì không)

```
1.   #include <stdio.h>
2.   #include <ctype.h>
3.   #include <conio.h>
4.   void main(){
5.      long int n, tong, i;
6.      char ch;
7.      do      {
8.              tong = 0;
9.              printf("\n\nNhap vao mot so nguyen: "); scanf("%ld",&n);
10.             printf("Cac uoc so cua %ld la: ",n);
11.             for(i = 1;i<n;i++)
12.                if(n % i == 0){
13.                        printf("%5d",i);
14.                        tong = tong + i;
15.                }//if & for
16.             printf("\nTong cac uoc so cua %ld bang %ld",n,tong);
17.             if(tong == n)  printf("\n   %5ld LA so hoan thien",n);
18.             else    printf("\n   %5ld KHONG LA so hoan thien",n);
19.             printf("\nBan co muon thuc hien lai(c/k)? ");
20.             fflush(stdin);
21.     }while(toupper(getche()) !='K');
22.     printf("\n\nAn phim bat ki de ket thuc ...");       getch();
23.  }
```

Ví dụ

Viết chương trình thực hiện công việc

- Nhập vào từ bàn phím 2 số nguyên
- Nhập vào từ bàn phím một ký tự bất kỳ;
  - Nếu đây là một toán tử số học thì đưa ra giá trị tương ứng với toán tử.
  - Nếu không phải thì đưa ra thông báo sai
- Chương trình thực hiện cho tới khi ký tự nhập vào là ‘q’ hoặc ‘Q’

```
#include <stdio.h>
#include <conio.h>
void main() {
  int a, b;
  char ch;
  int Fin =  0;
  clrscr();
  printf("Nhap cac so a, b "); scanf("%d%d",&a,&b);
  do{
   printf("\nToan tu (+ ; - ; * ; / ; %) "); ch=getche();
   switch(ch){
      case '+': printf(" Co ket qua: %d\n",a+b); break;
      case '-': printf(" Co ket qua: %d\n",a-b); break;
      case '*': printf(" Co ket qua: %d\n",a*b); break;
```



```
      case '%':     if  (b==0) printf(" Chia cho 0\n");
                    else  printf(" Co ket qua: %d\n",a%b);
                    break;
      case '/':     if  (b==0) printf(" Chia cho 0\n");
                    else  printf(" Co ket qua: %d\n",a/b);
                    break;
      case 'q':
      case 'Q':     Fin=1;          break;
      default: printf(" khong co toan tu nay\n");
   }
  }while(Fin==0);
  printf("\nKet thuc, an mot phim...");
  getch();
}
```



```
Nhap cac so a, b 145 60

Toan tu (+ ; - ; * ; / ; %) + Co ket qua: 205

Toan tu (+ ; - ; * ; / ; %) - Co ket qua: 85

Toan tu (+ ; - ; * ; / ; %) * Co ket qua: 8700

Toan tu (+ ; - ; * ; / ; %) / Co ket qua: 2

Toan tu (+ ; - ; * ; / ; %) % Co ket qua: 25

Toan tu (+ ; - ; * ; / ; %) g khong co toan tu nay

Toan tu (+ ; - ; * ; / ; %) q
Ket thuc, an mot phim...
```


Ví dụ: Nhập xâu, đếm số ký tự của xâu

- **Khai báo:** *int n=0; char c;*
- **Dùng vòng for**
  - *for(n=0;;c=getchar(),n++) if(c=='\n') break;*
  - *for(n=0 ; getchar() != '\n' ; n++);*
- **Dùng vòng lặp while**

```
c = getchar();
while (c !='\n'){
        c= getchar();
        n++;
}
```

- **Dùng vòng lặp do… while**

```
do{
        c = getchar();
        n++;
}while(c!='\n');
```

- ***Đưa kết quả ra:*** *printf(«Chuoi chua %d ky tu »,n);*

Bài tập 1

1. Nhập vào 2 số a, b thỏa mãn 1 < a < a+100< b. Liệt kê các số nguyên tố trong khoảng a,b
2. Viết chương trình nhập vào 3 số nguyên a,b,n thỏa mãn 0<a<b+n và sinh ra n số ngẫu nhiên trong khoảng a, b
3. Viết chương trình nhập một số nguyên dương N và phân tích N thành các thừa số nguyên tố
4. Nhập vào dãy cho tới khi gặp số 0. Tính tổng của dãy và trung bình cộng các số lẻ
5. Nhập vào một dãy ký tự cho tới khi gặp dấu xuống dòng. Đếm số từ của dãy biết rằng các từ được phân cách chỉ bởi các ký tự trắng (1 hoặc nhiều)

Bài tập 2

1. Nhập vào một dãy ký tự cho tới khi gặp ký tự '*' đưa ra tần suất xuất hiện của các nguyên âm
2. Cho hàm số $f(x) = x^5 + \sqrt[5]{x}$ Lập trình tính và đưa ra màn hình các cặp giá trị x,f(x) với x lấy dãy giá trị -10;-9.9;-9.8; …..; 4.9;5.0.
3. Đọc vào dãy số cho tới khi gặp số 0; Tìm số lớn nhất, nhỏ nhất của dãy và số lần xuất hiện các giá trị đó
4. Đọc vào một dãy cho tới khi tổng của dãy lớn hơn 2011. Tính trung bình cộng các số lẻ đã đọc
5. Đọc x và eps và tính biểu thức sau với độ chính xác nhỏ hơn eps

$$S = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \text{L} + \frac{(-1)^n x^n}{n!} + \text{L}$$

Bài tập

Viết chương trình đọc x và n vào từ bàn phím rồi tính

$$S = \sqrt{x + \sqrt{x\sqrt{x + \mathrm{L} + \sqrt{x}}}} \quad n \text{ dâ'u c ăn}$$

$$S = 1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \mathrm{L} + \frac{x^n}{n}$$

$$S = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \mathrm{L} + \frac{x^n}{n!}$$

$$S = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \mathrm{L} + \frac{(-1)^n x^n}{n!}$$

Tính hàm $f(x) = x^5 + \sqrt[5]{x}$

```
1.  #include <stdio.h>
2.  #include <math.h>
3.  void main(){
4.      float x, fx;

5.      for(x=-10.0; x<=5.0; x+=0.1){
6.          if(x==0)
7.              fx = 0.0;
8.          else
9.              fx = pow(x,5)+x/fabs(x) * pow(fabs(x), 0.2);
10.     printf("<%4.1f,%7.2f>\n",x,fx);
11.     }
12. }
```

- **pow(x,y)** sinh ra lỗi khi x âm và y không là số nguyên
- **fabs(x)** trả về trị tuyệt đối của x khi là số thực

Có sai số khi x=0

```
for(int i=-100;i<50;i++){
        x=(float)i/10;
        .....
```


Đọc dãy số…., tìm và đếm số max

```
1.  #include <stdio.h>
2.  #include <limits.h>

3.  void main(){
4.      int a, d=0, max = INT_MIN;
5.      do {
6.              printf("Nhap mot so : "); scanf("%d",&a);
7.              if( a > max){
8.                      max = a;
9.                      d = 1;
10.             }else
11.                     if (a == max) d++;
12.     }while ( a!= 0);
13.     printf("Max: %d; Co %d gia tri",max,d);
14. }
```


Nội dung chính

1. Cấu trúc lệnh khối
2. Cấu trúc rẽ nhánh
   - Cấu trúc if, if … else
   - Cấu trúc lựa chọn switch
3. Cấu trúc lặp
   - Vòng lặp **for**
   - Vòng lặp **while** và **do while**
4. Các lệnh thay đổi cấu trúc lập trình
   - Câu lệnh **continue**
   - Câu lệnh **break**

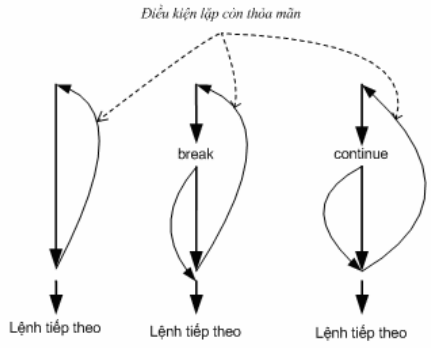
Mục đích

Các vòng lặp **while/ do ... while/ for** sẽ kết thúc quá trình lặp khi biểu thức điều kiện của vòng lặp không còn được thỏa mãn.

Tuy nhiên trong lập trình đôi khi ta cũng cần thoát khỏi vòng lặp ngay cả khi biểu thức điều kiện của vòng lặp vẫn còn được thỏa mãn.

Để hỗ trợ người lập trình làm việc đó, ngôn ngữ C cung cấp 2 câu lệnh là **continue** và **break**

Continue >< break



continue

- Bỏ qua việc thực hiện các câu lệnh nằm sau lệnh **continue** trong thân vòng lặp.
- Chuyển sang thực hiện một vòng lặp mới

Ví dụ 1

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int i;
  int sum = 0;
  for(i = 1;i<=100;i++)
  {
     if(i % 5 == 0)
          continue;
     sum += i;
  }
}
```

Tính tổng 100 số nguyên đầu tiên ngoại trừ các số chia hết cho 5

```
for(i=1;i<=100;i++)
  if (i % 5 != 0)
     sum += i;
```


break

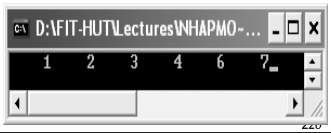
Thoát khỏi vòng lặp ngay cả khi biểu thức điều kiện của vòng lặp vẫn còn được thỏa mãn.

Chú ý:

- **break** dùng để thoát ra khỏi khối lặp hiện tại
- **break** cũng dùng để thoát ra khỏi lệnh rẽ nhánh **switch**

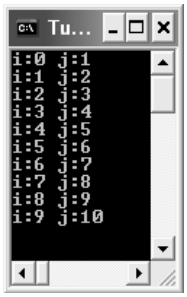
Ví dụ 2

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int i;
  for(i = 1;i<=10;i++)
  {
     if(i == 5) continue;
      printf("%5d",i);
     if(i==7) break;
  }
  getch();
}
```



Ví dụ 3

```
#include <stdio.h>
#include <conio.h>
void main()
{ int i,j;
  clrscr();
  for(i = 0;i<10;i++) {
     for (j=0; j < 10; j ++) {
        if(j > i){
           break;
        }//if
     }//for  j
    printf("i:%d j:%d\n",i,j);
  }//for_i
    getch();
}
```



Nhập một số nguyên, kiểm tra xem có là số nguyên tố không?

```
#includ <stdio.h>
#include <math.h>
void main()
{ int N, i, OK = 1;
  printf("\nNhap gia tri N : "); scanf("%d", &N);
  if (n<2) printf("\nSo %d khong phai so nguyen to",N);
  else {
     for(i=2;i<=(int)sqrt(N);i++)
     if (N%i == 0) {
          OK = 0;
          break;
     }
   if (OK)  printf("\nSo %d la so nguyen to.", N);
   else   printf("\nSo %d la hop so.", N);
  }
  getch();
}
```

i <= (int)sqrt(N)

```
i= 2;
while (N % i != 0) i++;
if (i == N) printf(« so ng to »)
```

Phân tích số nguyên ra thừa số nguyên tố

```
1.  #include <stdio.h>
2.  #include <conio.h>
3.  #include <ctype.h>
4.  void main(){
5.    int N, i;
6.    do{  printf("\n\nNhap vao so nguyen duong "); scanf("%d",&N);
7.                printf("%d = ",N);
8.                i = 2;
9.                while (i < N ){
10.                       if (N % i == 0){
11.                               printf("%d x ",i);
12.                               N = N/i;
13.                       } else   i++;
14.               }
15.               printf("%d \n",N);
16.               printf("Tiep tuc <C/K>?");  fflush(stdin);
17.   }while(toupper(getche()) != 'K');
18. }
```


Turbo C++ IDE

```
Nhap vao so nguyen duong 48
48 = 2 x 2 x 2 x 2 x 3
Tiep tuc <C/K>?c

Nhap vao so nguyen duong 1024
1024 = 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2
Tiep tuc <C/K>?c

Nhap vao so nguyen duong 1001
1001 = 7 x 11 x 13
Tiep tuc <C/K>?c

Nhap vao so nguyen duong 73
73 = 73
Tiep tuc <C/K>?k_
```

Nhập chuỗi ký tự cho đến khi gặp ký tự '*'
Tính tần suất xuất hiện nguyên âm 'a'

```
1.  #include <stdio.h>
2.  #include <conio.h>
3.  #include <ctype.h>
4.  void main(){
5.        char c; int n, d;
6.        do{
7.          printf("\n\n");
8.          d=0; n=0;
9.          while( (c=getche()) !='
10.               n++;
11.               if (c=='a') d++;
12.       }
13.       if(n==0)
14.         printf("\nChuoi ky tu rong\n");
15.       else
16.         printf("\ntan suat xuat hien ky tu 'a' la %5.2f%%\n",(float)100*d/n);
17.       printf("Tiep tuc <C/K>?: ");
18.     }while(toupper(getche()) != 'K');
19. }
```

```
Madam*
tan suat xuat hien ky tu 'a' la 40.00%
Tiep tuc <C/K>?: c

Hello world*
tan suat xuat hien ky tu 'a' la  0.00%
Tiep tuc <C/K>?: c

*
Chuoi ky tu rong
Tiep tuc <C/K>?: c

ABBA*
tan suat xuat hien ky tu 'a' la  0.00%
Tiep tuc <C/K>?: k
```


Viết chương trình đọc x và n vào từ bàn phím rồi tính

$$S = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \mathrm{L} + \frac{x^n}{n!}$$

```
1.  #include <stdio.h>
2.  #include <conio.h>
3.  void main(){
4.      int n, i;
5.      float x, u = 1.0,S=1.0;
6.      clrscr();
7.      printf("Nhap vao so nguyen n : "); scanf("%d",&n);
8.      printf("Nhap vao so thuc x : ");      scanf("%f",&x);
9.      for(i = 1; i <= n; i ++){
10.             u *= x/i;
11.             S += u;
12.     }
13.     printf("Ket qua la %.8f",S);
14.     getch();
15. }
```

```
Nhap vao so nguyen n : 20
Nhap vao so thuc x : 1
Ket qua la 2.71828198_
```

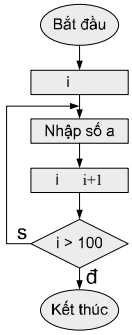
```
Nhap vao so nguyen n : 50
Nhap vao so thuc x : 2.31
Ket qua la 10.07442379_
```


Ví dụ tổng hợp

Viết chương trình thực hiện các công việc sau

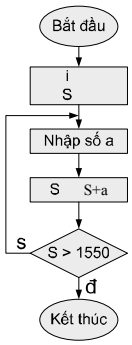
- Nhập vào một dãy số cho tới khi
  - Tổng của dãy lớn hơn 1550 hoặc là
  - Số phần tử trong dãy lớn hơn 100
- Đưa ra số phần tử nằm trong khoảng (35, 70)
- Đưa ra trung bình cộng của các phần tử chia hết cho 7

Nhập một dãy số cho tới khi số phần tử trong dãy lớn hơn 100



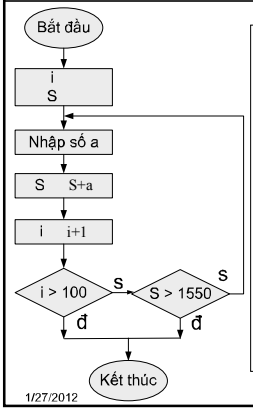

```
#include <stdio.h>
void main(){
   int a, i;
   i = 0;
   do{
      printf("Nhap vao so nguyen:");
      scanf("%d",&a);
      i++;
  }while (i <= 100);
}
```

Nhập một dãy số cho tới khi tổng của dãy lớn hơn 1550




```
#include <stdio.h>
void main(){
   int a, S;
   S = 0;
   do{
      printf("Nhap vao so nguyen:");
      scanf("%d",&a);
      S+=a;
  }while (S <= 1550);
}
```

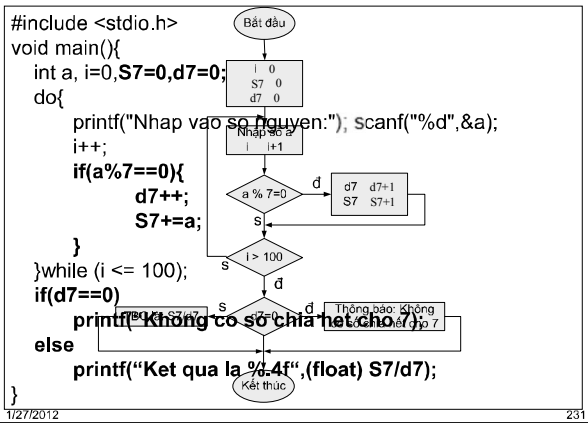

Nhập một dãy số cho tới khi thỏa mãn….




```
#include <stdio.h>
void main(){
   int a, i, S;
   S = 0; i=0;
   do{
      printf("Nhap vao so nguyen:");
      scanf("%d",&a);
      S+=a;
      i++;
   }while ( (i <=100)&&(S <= 1550) );
}
```

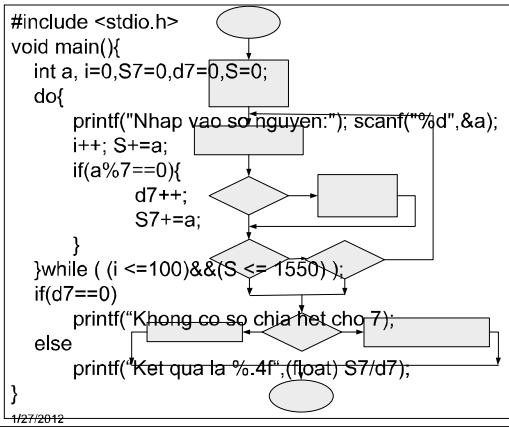

←0
←0

Đưa ra TBC của các phần tử chia hết cho 7(1)



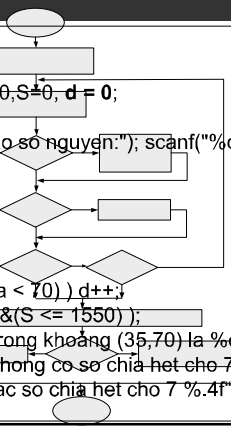

```
#include <stdio.h>
void main(){
   int a, i=0,S7=0,d7=0;
   do{
      printf("Nhap vao so nguyen:"); scanf("%d",&a);
      i++;
      if(a%7==0){
            d7++;
            S7+=a;
      }
   }while (i <= 100);
   if(d7==0)
      printf("Khong co so chia het cho 7");
   else
      printf("Ket qua la %.4f",(float) S7/d7);
}
```


Đưa ra TBC của các phần tử chia hết cho 7(2)



```
#include <stdio.h>
void main(){
   int a, i=0,S7=0,d7=0,S=0;
   do{
      printf("Nhap vao so nguyen:"); scanf("%d",&a);
      i++; S+=a;
      if(a%7==0){
            d7++;
            S7+=a;
      }
   }while ( (i <=100)&&(S <= 1550) );
   if(d7==0)
      printf("Khong co so chia het cho 7");
   else
      printf("Ket qua la %.4f",(float) S7/d7);
}
```


Ví dụ tổng hợp



```
#include <stdio.h>
void main(){
   int a, i=0,S7=0,d7=0,S=0, d = 0;
   do{
      printf("Nhap vao so nguyen:"); scanf("%d",&a);
      i++; S+=a;
      if(a%7==0){
            d7++;
            S7+=a;
      }
      if( (a>35) && (a < 70) ) d++;
   }while ( (i <=100)&&(S <= 1550) );
   printf("So phan tu trong khoang (35,70) la %d \n",d);
   if(d7==0)  printf("Khong co so chia het cho 7");
   else  printf("TBC cac so chia het cho 7 %.4f",(float) S7/d7);
}
```


Tổng kết

1. **Câu lệnh khối**

   Đặt trong cặp ngoặc nhọn **{ }**

2. **Cấu trúc rẽ nhánh**
   - **if** (bieu_thuc), **if** (bieu_thuc) … **else**
   - **switch** (bieu_thuc) {(**case**/**break**/**default**)}
3. **Cấu trúc lặp**
   - **for**(bieu_thuc_1; bieu_thuc_2; bieu_thuc_3) CauLenh;
   - **while** (bieu_thuc) CauLenh;
   - **do** Cau_Lenh **while** (bieu_thuc);
4. **Các lệnh thay đổi cấu trúc lập trình**
   - **continue**/ **break**

←
←
←
←
←

Nội dung chính

- Chương 1: Tổng quan về ngôn ngữ C
- Chương 2: Kiểu dữ liệu và biểu thức trong C
- Chương 3: Vào ra dữ liệu
- Chương 4: Cấu trúc điều khiển
- Chương 5: Mảng, con trỏ và xâu ký tự
- Chương 6: Cấu trúc
- Chương 7: Hàm
- *Chương 8: Tệp dữ liệu*

Nội dung chính

1. **Mảng**
   - Khái niệm
   - Khai báo và sử dụng
   - Các thao tác thường gặp
2. **Con trỏ**
   - Khái niệm và cách khai báo
   - Toán tử địa chỉ (&), toán tử nội dung (*)
   - Phép toán trên con trỏ
   - Con trỏ và mảng
3. **Xâu ký tự**
   - Khái niệm
   - Khai báo và sử dụng
   - Các hàm xử lý ký tự và xâu ký tự

Khái niệm mảng

- Trong thực tế, thường gặp các đối tượng có tính chất chung
  - Tháng trong năm
  - Điểm trung bình của sinh viên trong lớp
- Các đối tượng được nhóm lại dưới một tên
- Đối tượng được đặc trưng bởi **tên nhóm** và **thứ tự** trong nhóm
  - Tháng thứ 3 trong năm: Tháng 3
  - Sinh viên thứ 17 trong lớp:…
- Số thứ tự trong nhóm là **chỉ số phần tử**

Khái niệm mảng

- Kiểu mảng là một kiểu dữ liệu gồm
  - Một số hữu hạn thành phần.
  - Các thành phần có cùng một kiểu: kiểu cơ sở hay là kiểu thành phần.
- Mỗi phần tử của mảng được tham khảo thông qua
  - Tên mảng và
  - Chỉ số của phần tử trong mảng

Khai báo mảng

Kiểu_dữ_liệu Tên_Mảng[Kích thước];

- **Kiểu_dữ_liệu:** kiểu của các phần tử trong mảng (*nguyên, thực, ký tự, chuỗi, mảng,…*)
- **Tên_mảng:** tên của mảng
- **Kích_thước_mảng:** số phần tử trong mảng

Ví dụ

```
// khai báo mảng 10 phần tử có kiểu dữ liệu int
int Mang_so_nguyen[10];
float A[10]; //⇐ Mảng 10 phần tử kiểu số thực
```


Cấp phát bộ nhớ cho mảng

- Các phần tử trong mảng được cấp phát các ô nhớ kế tiếp nhau trong bộ nhớ
- Kích thước của mảng bằng kích thước một phần tử nhân với số phần tử

Ví dụ:

int A[10];//Mảng A gồm 10 phần tử nguyên

Kích thước của mảng A: 10 x 2 = 20 bytes

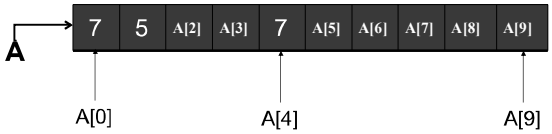
Truy nhập đến thành phần của mảng

- Biến mảng lưu trữ địa chỉ ô nhớ đầu tiên trong vùng nhớ được cấp phát
- Ngôn ngữ C đánh chỉ số các phần tử trong mảng bắt đầu từ **0**
- Các phần tử của mảng được truy nhập thông qua
  - Tên mảng và
  - Chỉ số của phần tử của phần tử trong mảng

Tên_Mang[Chỉ_số_phần_tử];

Truy nhập đến thành phần của mảng→Ví dụ

int A[10];//Mảng A gồm 10 phần tử nguyên



A[0] = 7;
A[1] = 5;
A[4] = 7;
int N = A[1] + A[4]; → N = 14

Ví dụ

int A[10];
for(int i = 0; i < 10; i++) A[i]= 2* i;

0	2	4	6	8	10	12	14	16	18

i : 10

Chú ý: C không kiểm tra vượt quá giới hạn của mảng khi truy nhập

int A[3], B[4], C[3];

A[0]	A[1]	A[2]	B[0]	B[1]	B[2]	B[3]	C[0]	C[1]	C[2]

A[5] ⇔ B[2] ⇔ C[-2] ←nếu c/cấp liên tiếp

Mảng nhiều chiều

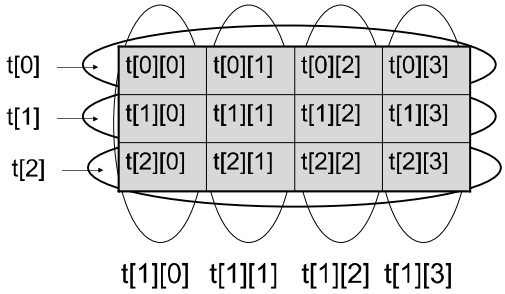
- Mỗi phần tử của mảng có thể là một mảng
  Mảng nhiều chiều

Kiểu Tên[Chiều_1] [Chiều_2]… [Chiều_N];

- **Kiểu**: Kiểu của mỗi phần tử trong mảng
- **Chiều_1, Chiều_2,…Chiều_N:** Các hằng số nguyên, cho biết kích thước (*số phần tử*) của mỗi chiều
- Mảng gồm: Chiều_1 x Chiều_2 x...x Chiều_N phần tử được lưu trữ trong vùng nhớ liên tục. Các phần tử thuộc kiểu **Kiểu**

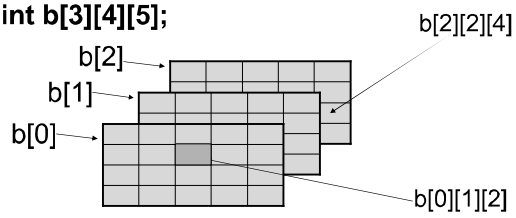
Mảng nhiều chiều

int t[3][4] ;



Mảng nhiều chiều→Ví dụ

int b[3][4][5];



- Mảng b gồm 3 phần tử b[0], b[1], b[2]
- Mỗi phần tử là mảng hai chiều gồm 4 hàng (hàng 0, 1, 2, 3) và 5 cột (0, 1, 2, 3, 4)
- Mỗi phần tử có kiểu nguyên có dấu, 2 byte

Khởi tạo giá trị cho mảng

Các phần tử của mảng có thể được khởi tạo giá trị ngay khi khai báo

Ví dụ

```
int a[4] = {1,4,6,2};
int b[2][3]={ {1,2,3}, {4,5,6} };
int t[3][4] = {
                {1, 2, 3, 4},
                {5, 6, 7, 8},
                {9, 10, 11, 12},
                };
```


Khởi tạo giá trị cho mảng → Chú ý

- Số lượng giá trị khởi tạo không được lớn hơn số lượng phần tử trong mảng
  - Nếu số lượng này nhỏ hơn, các phần tử còn lại được khởi tạo giá trị 0

```
int A[3][4] = { {1}, {4,5} };
int A[3][4] = { };←Tất cả đều mang giá trị 0
```

- Có thể xác định kích thước mảng thông qua số giá trị khởi tạo nếu để trống kích thước mảng

```
int A1 [8] = {2, 4, 6, 8, 10, 12, 14, 16};
int A2 [] = {2, 4, 6, 8, 10, 12, 14, 16};
```


Các thao tác thường gặp

- Nhập/Xuất dữ liệu cho mảng
  - Mảng 1 chiều, ma trận
- Bài toán đếm
  - Đếm số phần tử
  - Tính toán trên các phần tử..
- Tìm kiếm phần tử
  - Lớn nhất/nhỏ nhất/bất kỳ
- Sắp xếp phần tử trong mảng
  - Theo thứ tự, theo nguyên tắc
- Chèn thêm phần tử, xóa phần tử

Nhập dữ liệu

Dùng hàm scanf()

Ví dụ: int Tab[10];

- Nhập dữ liệu cho một phần tử

```
scanf("%d",&Tab[2]);←phần tử thứ 3 của mảng
```

- Nhập dữ liệu cho cả mảng
  - Dùng vòng lặp for

```
for(i = 1; i < 10; i++)
        scanf("%d",&Tab[i]);
```

  - Nên in ra chỉ số phần tử khi nhập

```
printf("Tab[%d] : "); scanf("%d",&Tab[i])
```


Nhập dữ liệu→Ví dụ 1

Nhập vào lượng mưa (mm) trong năm

```
#include <stdio.h>
#define  MONTHS 12
int main(){
    int rainfall[MONTHS], i;
    for ( i=0; i < MONTHS; i++ ){
        printf("Nhap luong mưa tháng %d: ", i+1);
        scanf("%d", &rainfall[i] );
    }
    return 0;
}
```


Nhập dữ liệu→Lưu ý

- Nếu số phần tử của mảng chỉ được biết tại thời điểm thực hiện chương trình (nhưng *biết số phần tử tối đa*)
  - Khai báo mảng với kích thước tối đa
  - Sử dụng biến nguyên lưu số phần tử thực sự của mảng.

Ví dụ:

- Nhập vào mảng không quá 100 số thực
  - Khai báo mảng thực Tab có tối đa 100 phần tử.
  - Nhập số phần tử thực sự của mảng
  - Nhập giá trị cho từng phần phần tử (dùng for)

Nhập dữ liệu→Ví dụ 2

```
#include<stdio.h>
#void main(){
   float A[100];
   int n, i;
   do{
        printf("\n Cho biet so phan tu cua mang: ");
        scanf("%d",&n);
   }while (n>100 || n<=0);
   for(i = 0; i < n; i++){
        printf("A[%d] = ", i); scanf("%f",&A[i]);
   }
}
```


Xuất dữ liệu trong mảng

Dùng hàm printf()

Ví dụ: int Tab[10];

- Hiện thị phần tử thứ 5:

  printf("%d",Tab[4]);

- Để hiển thị tất cả các phần tử:

  for(i = 0; i < 10; i++)

  printf("%4d",Tab[i]);

Các kiểu xuất dữ liệu

– Hiển thị tất cả/một phần theo dòng/cột..

– Hiển thị từng k phần tử trên một dòng…

Xuất dữ liệu trong mảng→Ví dụ 1

```
#include <stdio.h>
#define  MAX 12
void main(){
   int A[MAX], i;
   for ( i=0; i < MAX; i++ ){     //Nhập dữ liệu
         printf("A[%d]: ", i+1);   scanf("%d", &A [i] );
   }
    for ( i=0; i < MAX; i++ ){
          printf( "%4d",A[i]);
          if( (i+1) %4==0) printf("\n");
    }
}
```


Xuất dữ liệu trong mảng→Ví dụ 1→Thực hiện

```
A[1]: 12
A[2]: 14
A[3]: 15
A[4]: 26
A[5]: 27
A[6]: 48
A[7]: 56
A[8]: 68
A[9]: 50
A[10]: 30
A[11]: 19
A[12]: 14

  12   14   15   26
  27   48   56   68
  50   30   19   14

  12  14  15  26  27  48  56  68  50  30  19  14

12
14
15
26
27
48
56
68
50
30
19
14
```


Ví dụ 2: Nhập và đưa ra màn hình một ma trận

```
1.  #include <stdio.h>
2.  void main(){
3.      int A[20][20], n, m, i,j;
4.      printf("Nhap so hang : ");    scanf("%d",&n);
5.      printf("Nhap so cot : ");     scanf("%d",&m);
6.      printf("\n");
7.      for ( i=0; i < n; i++ )
8.        for(j=0; j < m; j++) {
9.            printf("Nhap phan tu A[%d,%d]: ", i+1,j+1);
10.           scanf("%d", &A[i][j] );
11.     }
12.     printf("\n\n MA TRAN DA NHAP \n\n");
13.     for ( i=0; i < n; i++ ){
14.       for(j=0; j < m; j++)
15.           printf( "%4d" ,A[i][j]);
16.       printf("\n");
17.     }
18. }
```


Ví dụ 2→Kết quả thực hiện

```
Nhap so hang : 2
Nhap so cot : 4

Nhap phan tu [1,1]: 12
Nhap phan tu [1,2]: 34
Nhap phan tu [1,3]: 54
Nhap phan tu [1,4]: 3
Nhap phan tu [2,1]: 123
Nhap phan tu [2,2]: 872
Nhap phan tu [2,3]: 12
Nhap phan tu [2,4]: 34

 MA TRAN DA NHAP

  12   34   54    3
 123  872   12   34
```

Đếm số phần tử thỏa mãn điều kiện

- Duyệt từng phần tử của dãy (*dùng **for***)
- Nếu phần tử xét thỏa mãn điều kiện
  - Ghi nhận
- Chuyển sang xem xét phần tử tiếp theo

Ví dụ: Đếm số tháng có lượng mưa lớn hơn 50mm

```
int dem = 0;
for(i = 0; i < MONTHS; i++)
    if(rainfall[i] > 50)
        dem++;
printf("\nThang mua nhieu hon 50mm: %d", dem);
```


Ví dụ: Nhập mảng, đưa ra TBC các số chia hết cho 7

```
#include<stdio.h>
void main(){
   int A[100];
   int n, i, d = 0, S=0;
   printf("\n So phan tu cua mang (<100) : "); scanf("%d",&n);
   for(i = 0; i < n; i++){
        printf("A[%d] = ", i); scanf("%d",&A[i]);
   }
    for(i = 0; i < n; i++)
        if(A[i] %7==0){
                d++;
                S+= A[i];
        }
   if(d > 0) printf("TBC so chia het cho 7: %7.2f",(float)S/d);
   else printf("Trong day khong co so chia het cho 7");
}
```


Tìm kiếm phần tử

Tìm phần tử lớn nhất (*nhỏ nhất*)

- Giả sử phần tử đó là phần tử đầu tiên
- Lần lượt so sánh với các phần tử còn lại
  - Nếu phần tử mới của dãy lớn hơn⇒ coi đây là phần tử lớn nhất và tiếp tục so sánh với phần tử kế
  - Nếu không đúng, so sánh tiếp với phần tử kế

Ví dụ: Tìm tháng có lượng mưa nhiều nhất trong năm

```
max = rainfall[0];
for(i = 1; i < MONTHS; i++)
        if(rainfall[i] > max)
                max = rainfall[i];
printf("\n Luong mua nhieu nhat la: %d", max);
```


Tìm kiếm phần tử

- Tìm kiếm các phần tử thỏa mãn điều kiện (*giống bài toán đếm*)
  - Dùng for duyệt toàn bộ
  - Nếu cần thiết, dùng thêm mảng ghi lại chỉ số

Ví dụ: Đưa ra danh sách các tháng có lượng mưa nhiều hơn 50mm

```
printf("Thang co luong mua lon hon 500mm")
for(i = 0; i < MONTHS; i++)
    if(rainfall[i] > 50)
        printf("\nThang %d", i+1);
```


Tìm kiếm phần tử (tiếp)

- Tìm phần tử đầu tiên của danh sách
  - Dùng vòn lặp **for** kết hợp với **break**;
  - Dùng vòng lặp **while**

Ví dụ

Đưa ra phần tử đầu của mảng có giá trị bằng k;

Tìm kiếm phần tử →Ví dụ

```
int Tab[100]
int N, i, k, f;//N: số phần tử, k phần tử cần tìm
```

Dùng for

```
for(i = 0; i < N; i++)
    if(Tab[i] == k) break;
if(i< N) printf("Tim thay tai vi tri %d",i);
```

Dùng while

```
i=0; f =0;        //f: found. f = 1 ⇔ k is found
while(i < N && f==0){
  if(Tab[i] == k) f = 1;
  else i++;}
if (f==1) printf("Tim thay tai vi tri %d",i);
```

Bài toán sắp xếp theo thứ tự

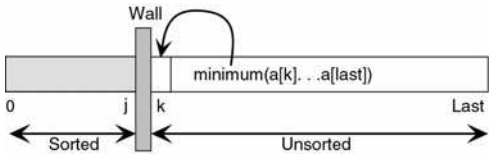
- Cho mảng phần tử, sắp xếp theo thứ tự tăng/giảm
- Các thuật toán
  - Sắp xếp thêm dần (insertion sort)
  - Sắp xếp lựa chọn (selection sort)
  - Sắp xếp nổi bọt (bubble sort)
  - Sắp xếp vun đống (heap sort)
  - Sắp xếp nhanh (quick sort)
  - Sắp xếp trộn (merge sort)
  - ….

Bài toán sắp xếp tăng →Thuật toán lựa chọn

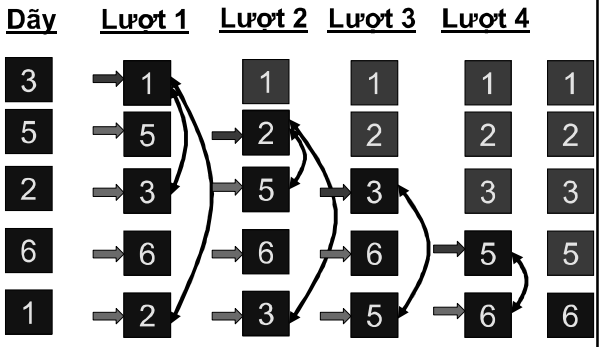
Nguyên tắc

Tại lượt sắp thứ k, tìm phần tử nhỏ nhất trong số các phần tử chưa được sắp xếp ([k..last]) và đổi chỗ cho phần tử thứ k (*có chỉ số k-1*)

- Khi k = 1, phần tử thứ nhất (chỉ số 0) đúng vị trí
- Khi k= 2, phần tử thứ hai (chỉ số 1) đúng vị trí…



Bài toán sắp xếp tăng →Thuật toán lựa chọn



Bài toán sắp xếp tăng →Thuật toán lựa chọn

```
//Khai báo các biến
   int A[100];       //Mảng chứa dữ liệu
   int N, i, j, tmp;
//Sắp xếp
   for(i = 0; i < N – 1; i++)
        for(j = i + 1; j < N; j++)
              if(A[i] > A[j]) {
                    tmp = A[i];
                    A[i] = A[j];
                    A[j] = tmp;
              }
```


Ví dụ

Nhập vào từ bàn phím một mảng các số nguyên không quá 100 phần tử

Hiển thị dãy số vừa nhập

Sắp xếp dãy theo thứ tự **giảm** dần

Hiện thị dãy tại mỗi lượt sắp xếp

Ví dụ

```
1.  #include<stdio.h>
2.  void main(){
3.     int A[100] ;
4.     int N, i, j , t;
5.     printf("So phan tu [< 100]: "); scanf("%d",&N);
6.     printf("Hay nhap day so...\n");
7.     for(i=0; i < N; i++){
8.           printf("A[%d] = ",i+1); scanf("%d",&A[i]);
9.     }
10.    printf("\nDay vua nhap...\n");
11.    for(i=0; i < N; i++)
12.          printf("%4d", A[i]);
13.    printf("\n\n");
```

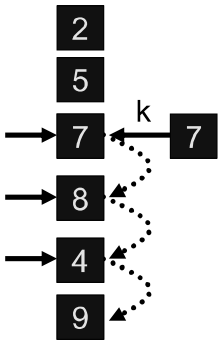
Ví dụ

```
1.      printf("Sap xep day theo thuat toan lua chon");
2.      for(i=0; i < N-1; i++){
3.              for(j=i; j < N; j++)
4.                      if(A[i] < A[j]) {
5.                              t = A[i];
6.                              A[i] = A[j];
7.                              A[j] = t;
8.                      }//if & for_j
9.              printf("\nLuot %d : ",i+1);
10.             for(j=0; j < N; j++)
11.                     printf("%4d", A[j]);
12.     }//for_i
13. }//main
```


Ví dụ→Kết quả

```
So phan tu [< 100]: 6
Hay nhap day so...
A[1] = 5
A[2] = 32
A[3] = 67
A[4] = 12
A[5] = 24
A[6] = 16

Day vua nhap...
   5  32  67  12  24  16

Sap xep day theo thuat toan lua chon
Luot 1 :   67   5  32  12  24  16
Luot 2 :   67  32   5  12  24  16
Luot 3 :   67  32  24   5  12  16
Luot 4 :   67  32  24  16   5  12
Luot 5 :   67  32  24  16  12   5
```


Bài toán chèn phần tử **a** vào vị trí **k**




```
for(i = N; i > k; i--)
   A[i] = A[i-1];
A[k] = a;
N= N + 1;
```

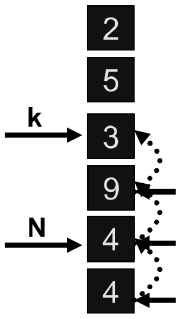
Chú ý:

N = MAX: không chèn được

k > N → Chèn vào vị trí N;

k< 0 → Chèn vào vị trí 0

Bài toán xóa phần tử ở vị trí **k** (0≤k<N)




```
for(i = k+1; i < N; i++)
   A[i-1] = A[i];
N= N - 1;
```


Bài tập 1

1. Nhập vào dãy số, tính và đưa ra màn hình
   - Tổng và tích của dãy số
   - Các số chia hết cho 3 và lớn hơn 10
   - Đếm các số nằm trong đoạn [100,1000]
2. Nhập vào một dãy số; tìm số chẵn nhỏ nhất dãy
3. Nhập dãy số; đếm xem có bao nhiêu bộ 3 số thỏa mãn điều kiện $x_i=(x_{i-1}+x_{i+1})/2$
4. Đọc vào dãy số có n phần từ (n<100). Đọc số x và số k nguyên. Chèn x vào vị trí k của dãy. Nếu k>n, chèn x vào vị trí n+1.
5. Nhập vào n và dãy số $(x_1,x_2,\ldots x_n)$;$(y_1,y_2,..y_n)$ rồi tính

$$a)\ \sum_{i=1}^{n}\cos x_i \sin x_i \qquad b)\ \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \qquad c)\ \sum_{1}^{n-1} x_2^{i+1} y_{i+1}$$

Bài tập 2

1. Nhập vào từ bàn phím một dãy số nguyên (<100 phần tử). Sắp xếp dãy theo nguyên tắc: Bên trên là số chẵn chia hết cho 3. Bên dưới là số lẻ chia hết cho 3. Giữa là các số còn lại. Đưa cả 2 dãy ra màn hình.
2. Viết chương trình nhập vào từ bàn một dãy số (<100 phần tử). Đưa ra số bé nhất và vị trí những số bằng số bé nhất
3. Nhập vào một dãy số (<100 phần tử) và sắp xếp theo thứ tự tăng dần. Nhập thêm vào một số và chèn số mới nhập vào đúng vị trí
4. Nhập vào một dãy (<100 phần tử); xóa đi các phần tử chia hết cho 5 và đưa kết quả ra màn hình

Bài chữa

```
#include<stdio.h>
void main(){
    int A[100];
    int N, i;
    //Nhập dữ liệu
    printf("So phan tu : "); scanf("%d",&N);
    for(i=0; i < N; i ++){
      printf("A[%d] = ",i);scanf("%d",&A[i]);
    }
    //Các thao tác xử lý mảng: chèn, xóa, sắp xếp,…
    //Đưa Dữ liệu ra
    for(i=0; i < N; i ++)
      printf("%4d",A[i]);
}
```


Bài chữa → Sắp xếp số chẵn chia hết 3 lên đầu dãy..

```
1.  { int d = 0, t;
2.    for(i=0;i < N; i++)
3.        if(A[i]%6==0){
4.            t=A[i];  A[i]=A[d]; A[d] = t;
5.            d++;
6.        }
7.    for(i=d; i < N; i++)
8.        if(A[i]%3 != 0){
9.            t=A[i];   A[i]=A[d];  A[d] = t;
10.           d++;
11.       }
12. }
```


Bài chữa → Sắp xếp tăng dần và chèn đúng vị trí

```
1.  {  int k = 0,i, j, t;
2.     for(i=0;i < N - 1; i++)//Sắp xếp lựa chọn
3.          for(j=i+1;j < N ; j++)
4.            if(A[i] > A[j]){
5.                t = A[j];  A[j] = A[i];  A[i]=t;
6.            }
7.     printf("Phan tu moi:"); scanf("%d",&k);//Nhập p/tử mới
8.     i = N;  //Chèn đúng vị trí
9.     while( (i > 0) &&(A[i-1] > k) ){
10.         A[i] = A[i-1];
11.         i--;
12.    }
13.    A[i] = k;
14.    N++;
15. }
```


Bài chữa → Xóa các phần tử chia hết cho 5..

```
1.  { // PP: Giữ lại các phần tử không chia hết cho 5
2.        int d = 0, i;
3.        for(i=0;i < N; i++)
4.            if(A[i] % 5 != 0){
5.                A[d] = A[i];
6.                d++;
7.            }
8.        N = d;
9.  }
```


Xóa các phần tử chia hết cho 5 →Kết quả

```
So phan tu : 12
A[0] = 5
A[1] = 1
A[2] = 21
A[3] = 15
A[4] = 10
A[5] = 34
A[6] = 23
A[7] = 45
A[8] = 54
A[9] = 32
A[10] = 12
A[11] = 50
Day ban dau :   5   1  21  15  10  34  23  45  54  32  12  50
Day sau khi xoa :   1  21  34  23  54  32  12
```


Bài tập 3 : Ma trận

1. Viết chương trình nhập vào một ma trận vuông, các phần tử nguyên, sau đó
   - Đưa ra ma trận tam giác dưới
   - Đưa ra ma trận tam giác trên
2. Nhập M, N (M, N < 30) và một ma trận MxN. Đưa ma trận ra màn hình
   - Tìm hàng/cột có tổng các phần tử lớn nhất
   - Tìm số lớn nhất/nhỏ nhất và vị trí trong ma trận
   - Đưa ra ma trận S cùng kích thước thỏa mãn

$$s_{i,j} = \begin{cases} 1 & \text{nếu } u_{i,j} > 0 \\ 0 & \text{nếu } u_{i,j} = 0 \\ -1 & \text{nếu } u_{i,j} < 0 \end{cases}$$

Nhập vào một ma trận vuông,..

```
#include <stdio.h>
void main(){
      int A[20][20], N,i,j;
      printf("Nhap kich thuoc : "); scanf("%d",&N);
      printf("\n");
      for ( i=0; i < N; i++ )
        for(j=0; j < N; j++)     {
            printf("Nhap phan tu [%d,%d]:", i+1,j+1);
            scanf("%d", &A[i][j] );
      }
      printf("\n\n MA TRAN DA NHAP \n\n");
      for ( i=0; i < N; i++ ){
        for(j=0; j < N; j++)
            printf( "%4d" ,A[i][j]);
        printf("\n");
      }
}
```


Đưa ra ma trân tam giác trên, dưới

```
      printf("\n\n MA TRAN TAM GIAC TREN \n\n");
      for ( i=0; i < N; i++ ){
        for(j=0; j < N; j++)
         if(j >= i)
                printf( "%4d" ,A[i][j]);
           else
             printf("%4c",32);  //32 là mã ASCII của dấu cách
        printf("\n");
      }
      printf("\n\n MA TRAN TAM GIAC DUOI \n\n");
      for ( i=0; i < N; i++ ){
        for(j=0; j <= i; j++)
            printf( "%4d" ,A[i][j]);
        printf("\n");
      }
}//main
```


Thực hiện

```
Nhap kich thuoc : 4

Nhap phan tu [1,1]: 12
Nhap phan tu [1,2]: 34
Nhap phan tu [1,3]: 52
Nhap phan tu [1,4]: 9
Nhap phan tu [2,1]: 10
Nhap phan tu [2,2]: 52
Nhap phan tu [2,3]: 54
Nhap phan tu [2,4]: 75
Nhap phan tu [3,1]: 8
Nhap phan tu [3,2]: 12
Nhap phan tu [3,3]: 45
Nhap phan tu [3,4]: 7
Nhap phan tu [4,1]: 11
Nhap phan tu [4,2]: 42
Nhap phan tu [4,3]: 22
Nhap phan tu [4,4]: 34
```

```
MA TRAN DA NHAP

 12  34  52   9
 10  52  54  75
  8  12  45   7
 11  42  22  34
```

```
MA TRAN TAM GIAC TREN

 12  34  52   9
     52  54  75
         45   7
             34
```

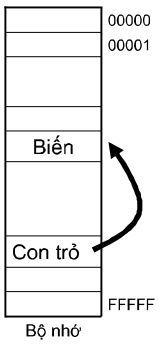
```
MA TRAN TAM GIAC DUOI

 12
 10  52
  8  12  45
 11  42  22  34
```


Nội dung chính

1. **Mảng**
   - Khái niệm
   - Khai báo và sử dụng
   - Các thao tác thường gặp
2. **Con trỏ**
   - Khái niệm và cách khai báo
   - Toán tử địa chỉ (&), toán tử nội dung (*)
   - Phép toán trên con trỏ
   - Con trỏ và mảng
3. **Xâu ký tự**
   - Khái niệm
   - Khai báo và sử dụng
   - Các hàm xử lý ký tự và xâu ký tự

Giới thiệu

- Là một khái niệm "mạnh" trong C
  - Cho phép tính toán trên con trỏ
  - Sử dụng con trỏ hàm
- Cho phép truy nhập gián tiếp tới một đối tượng có địa chỉ (*biến, hàm*)
  - Truy nhập trực tiếp→ thông qua tên

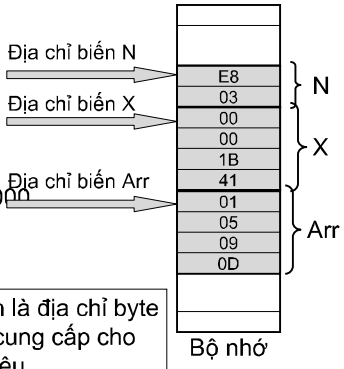


Địa chỉ

- Bộ nhớ gồm dãy các ô nhớ
  - Mỗi ô nhớ là một byte
  - Mỗi ô nhớ có một địa chỉ riêng
- Các biến trong chương trình được lưu tại vùng nhớ nào đó trong bộ nhớ
- Khi khai báo biến, tùy thuộc vào kiểu, biến sẽ được cấp một số ô nhớ liên tục nhau
  - Biến int được cấp 2 bytes, float được cấp 4 bytes,..
  - Địa chỉ của biến, là địa chỉ của byte đầu tiên trong số các byte được cấp
  - Khi gán giá trị cho biến, nội dung các byte cung cấp cho biến sẽ thay đổi

Địa chỉ →Ví dụ

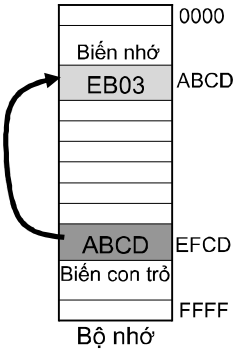
```
int N;
float x;
char Arr[4];

N = 1000;//03E8
X=9.6875;//411B0000
for(i=0;i<4;i++)
   Arr[i]= 4*i+1;
```



Địa chỉ của một biến là địa chỉ byte nhớ đầu tiên được cung cấp cho biến để lưu trữ dữ liệu

Con trỏ

- Con trỏ là **một biến** mà **giá trị của nó là địa chỉ** của một vùng nhớ
  - Vùng nhớ này có thể dùng để chứa các biến có kiểu cơ bản (*nguyên, thực, ký tự,…*) hay có cấu trúc (*mảng, bản ghi,..*)
- Con trỏ dùng "***trỏ tới***" một biến nhớ
  - Có thể trỏ tới một hàm
  - Có thể trỏ tới con trỏ khác



Con trỏ →Khai báo

Kiểu * Tên;

- **Tên:** Tên của một biến con trỏ
- **Kiểu:** Kiểu của biến mà con trỏ "Tên" trỏ tới
  - Giá trị của con trỏ có thể thay đổi được
    - Trỏ tới các biến khác nhau, có cùng kiểu
  - Kiểu biến mà con trỏ trỏ tới không thay đổi được
    - Muốn thay đổi phải thực hiện "***ép kiểu***"

Ví dụ:

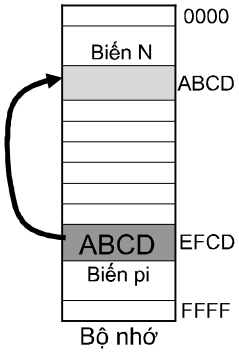
```
int * pi;     //Con trỏ, trỏ tới một biến kiểu nguyên
char * pc; //Con trỏ, trỏ tới một biến kiểu ký tự
```


Toán tử địa chỉ (&)

- Ký hiệu: **&**
- Là toán tử một ngôi, trả về địa chỉ của biến
  - Địa chỉ biến có thể được gán cho một con trỏ, trỏ tới đối tượng cùng kiểu

Ví dụ

```
int N; // &N→ ABCD
int * pi;
pi = &N; // pi←ABCD
```

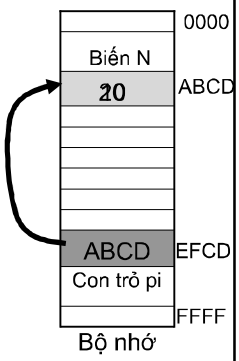


Toán tử nội dung (*)

- Ký hiệu: *
- Là toán tử một ngôi, trả về giá trị (nội dung) của vùng nhớ mà con trỏ đang trỏ tới

Ví dụ

```
int N;
int * pi;
pi = &N; ⇔ N= 10; ⇔ *pi = 10;
N = 10;//Vùng nhớ mà pi trỏ tới
   mang giá trị 10; Vậy *pi=10
*pi = 20;// Vùng nhớ pi trỏ tới được
   gán giá trị 20; Vậy N= 20
```



Gán giá trị cho con trỏ

- Con trỏ được gán địa chỉ của một biến
  - Biến cùng kiểu với kiểu mà con trỏ trỏ tới
    - Nếu không, cần phải ép kiểu
- Con trỏ được gán giá trị của con trỏ khác
  - Hai con trỏ sẽ trỏ tới cùng một biến(do cùng địa chỉ)
  - Hai con trỏ nên cùng kiểu trỏ đến
    - Nếu không, phải ép kiểu
- Con trỏ được gán giá trị NULL

  Ví dụ: int *p; p = 0;
- Gán nội dung vùng nhớ 2 con trỏ trỏ tới.

  Ví dụ: int *p1, *p2; *p1 = *p2;

Ví dụ

```
#include <stdio.h>
void main(){
   int N=5, M=10;
   int *p1 = &N;
   int *p2 = &M;
   *p1 = *p2;
   printf("%d %d",*p1,*p2);
}
```

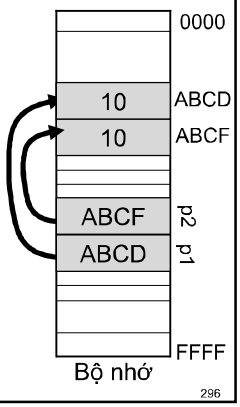
10 10

```
#include <stdio.h>
void main(){
   int N=5, M=10;
   int *p1 = &N;
   int *p2 = &M;
   p1 = p2;
   printf("%d %d",*p1,*p2);
}
```

10 10

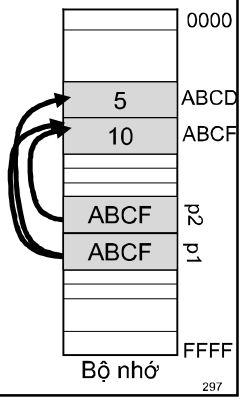
Ví dụ

```
1. #include <stdio.h>
2. void main(){
3.      int N=5, M=10;
4.      int *p1 = &N;
5.      int *p2 = &M;
6.      *p1 = *p2;
7.      printf("%d %d",*p1,*p2);
8. }
```



Ví dụ

```
1. #include <stdio.h>
2. void main(){
3.      int N=5, M=10;
4.      int *p1 = &N;
5.      int *p2 = &M;
6.      p1 = p2;
7.      printf("%d %d",*p1,*p2);
8. }
```



Các phép toán trên con trỏ

- Cộng con trỏ với một số nguyên
  - Kết quả: Con trỏ cùng kiểu
- Trừ con trỏ với một số nguyên
  - Kết quả: Con trỏ cùng kiểu
- Trừ 2 con trỏ cùng kiểu cho nhau
  - Kết quả: Một số nguyên
    - Khoảng cách giữa 2 con trỏ được đo bằng số phần tử thuộc kiểu dữ liệu mà con trỏ trỏ tới

Các phép toán trên con trỏ→Ví dụ

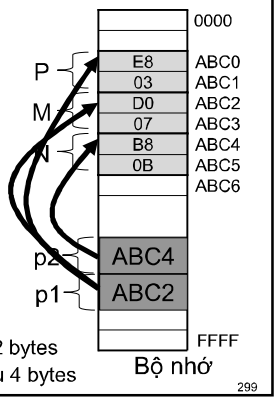
int N=1000, M=2000,P=3000;

int * p1 = &P, *p2 = &N;

p1 - p2 → **-2**

* (p2-1) → 2000

* ++ p1 → 2000



Ghi chú:

- Kiểu **int**, các phần tử cách nhau 2 bytes
- Kiểu **float**, các phần tử cách nhau 4 bytes

Mối quan hệ giữa con trỏ và mảng một chiều

- Nếu **Tab** là tên một mảng ⇒ **Tab** là một **con trỏ hằng** chứa địa chỉ của phần tử đầu tiên của mảng **Tab** (&Tab[0])
  - Không tồn tại phép tính trên tên mảng, hoặc gán giá trị cho tên mảng (*VD: Tab=…; Tab++*)
- Có thể sử dụng một con trỏ để duyệt mảng nếu nó được gán giá trị bằng địa chỉ của mảng (*địa chỉ của phần tử đầu tiên*)

Ví dụ

```
int Tab[10];
int * p = Tab;  // int *p = &Tab[0]
```

```
for(i = 0; i < 10; i ++)
   printf("%d ", *(p + i) );
```

```
for(i = 0; i < 10; i ++)
   printf("%d ", p[i]);
```

```
for(i = 0; i < 10; i ++)
   printf("%d ", *(p++) );
```


Con trỏ void

```
void * Tên_con_trỏ
```

- Là một con trỏ đặc biệt: con trỏ không có kiểu
- Có thể nhận giá trị là địa chỉ của một biến có kiểu dữ liệu bất kỳ
  - Thường dùng làm đối số trong lời gọi hàm để có thể nhận bất kỳ kiểu địa chỉ nào của tham số được truyền

Ví dụ:

```
void * p, *q;
int n;  float x;
p = & n;  q= & x; \\←Các câu lệnh hợp lệ
```


Ví dụ

```
#include<stdio.h>
int main()
{
   int a=3, *p;
   p = &a;
   printf("%d\n", a * *p * a + *p);
   return 0;
}
```

30

Ví dụ

```
#include<stdio.h>
int main(){
   int arr[2][2][2] = {10, 2, 3, 4, 5, 6, 7, 8};
   int *p, *q;
   p = &arr[1][1][1];
   q = (int *) arr;
   printf("%d, %d\n", *p, *(q+4) );
   return 0;
}
```

8, 5

Nội dung chính

1. **Mảng**
   - Khái niệm
   - Khai báo và sử dụng
   - Các thao tác thường gặp
2. **Con trỏ**
   - Khái niệm và cách khai báo
   - Toán tử địa chỉ (&), toán tử nội dung (*)
   - Phép toán trên con trỏ
   - Con trỏ và mảng
3. **Xâu ký tự**
   - Khái niệm
   - Khai báo và sử dụng
   - Các hàm xử lý ký tự và xâu ký tự

Khái niệm xâu ký tự

- Xâu kí tự (string) là một dãy các kí tự viết liên tiếp nhau
  - Độ dài xâu là số kí tự có trong xâu
  - Xâu không có kí tự nào: Xâu rỗng
- Ví dụ: "Tin hoc", "String"
- Lưu trữ: kết thúc xâu bằng kí tự '\0' hay NULL (mã ASCII là 0)

'T'	'i'	' n '	' '	'h'	'o'	'c'	'\0'

Khái niệm xâu ký tự→Lưu ý

- Xâu kí tự >< mảng kí tự
  - Tập hợp các kí tự viết liên tiếp nhau
    - Truy nhập một phần tử của xâu ký tự (*là một ký tự*) giống như truy nhập vào một phần tử của mảng: Tên[Chỉ_số]
  - Xâu kí tự có kí tự kết thúc xâu, mảng kí tự không có kí tự kết thúc xâu
- Xâu kí tự độ dài 1 >< kí tự *("A" ='A' ?)*
  - 'A' là 1 kí tự, được lưu trữ trong 1 byte
  - "A" là 1 xâu kí tự, ngoài kí tự 'A' còn có kí tự '\0' => được lưu trữ trong 2 byte

Khai báo

char tên_xâu [số_kí_tự_tối_đa];

- Để lưu trữ một xâu có **n** kí tự chúng ta cần một mảng có kích thước **n+1**
  - Phần tử cuối cùng chứa ký tự NULL

Ví dụ

- Để lưu trữ xâu "Tin hoc" chúng ta phải khai báo xâu có số phần tử tối đa ít nhất là 8

char str[8] = "Tin hoc";

'T'	'i'	' n '	' '	'h'	'o'	'c'	'\0'

Truy nhập phần tử của xâu

Giống như truy nhập tới một phần tử của mảng ký tự

tên_xâu [chỉ_số_của_kí_tự]

Ví dụ: char Str[10] = "Tin hoc";

T	i	n	-	h	o	c	\0	?	\0

Str[0] → 'T' Str[3] = '-';
Str[3] → ' ' Str[7] = ' ';
Str[7] → '\0 ' Str[8] = '1 ' ;
Str[8] → ? Str[9] = '\0';

Str: Tin-hoc 1

Ví dụ: Nhập xâu và đếm số ký tự '*'

```
#include <stdio.h>
void main(){
  char Str[100];
  int d=0, i=0;
  printf("Nhap xau ky tu: "); gets(Str);
  while(Str[i] != '\0'){
    if(Str[i]=='*')
      d++;
    i++;
  }
  printf("Ket qua : %d",d);
}
```

Tính chiều dài của xâu
```
d=0;
while(Str[d] != '\0') d++;
```

```
Nhap xau ky tu: **  abc** dd*e**e*dd
Ket qua : 8_
```

```
Nhap xau ky tu: *****  *****  **
Ket qua : 12
```


Các hàm xử lý ký tự

Tệp tiêu đề : ctype.h

#include <ctype.h>

Các hàm xử lý ký tự→Chuyển đổi chữ hoa/thường

- int **toupper**(char ch):
  - Chuyển kí tự thường thành kí tự hoa
    toupper('a') => 'A'
- int **tolower**(char ch)
  - Chuyển kí tự hoa thành kí tự thường
    tolower('B') => 'b'

Ví dụ

```
do{
  …….
  printf("Tiep tuc <C/K>? :"); fflush(stdin);
}while(toupper(getche()) !='K');
```

Các hàm xử lý ký tự →Kiểm tra chữ hoa/thường

- int **islower**(char ch)
  - –Kiểm tra chữ thường:
    - Hàm trả về giá trị khác 0 nếu ch là chữ thường, ngược lại trả về 0
    - Ví dụ: printf("%d ",islower('A')); ⇒ 0
- int **isupper**(char ch):
  - –Kiểm tra chữ hoa:
    - Hàm trả về giá trị khác 0 nếu ch là chữ hoa, ngược lại trả về 0
    - Ví dụ: printf("%d ",isupper('A')); ⇒ ≠ 0 (1 !?)

Các hàm xử lý ký tự →Kiểm tra chữ cái/chữ số

- int **isalpha**(char ch):
  - – Kiểm tra kí tự trong tham số có phải chữ cái không (‘a’…’z’,’A’,..’Z’). Hàm trả về khác 0 nếu đúng, ngược lại trả về giá trị bằng 0
  - – Ví dụ: printf("%d ",isalpha('A')); ⇒ ≠ 0 (1 !?)
- int **isdigit**(char ch):
  - – Kiểm tra kí tự trong tham số có phải chữ số (‘0’,‘1‘,..‘9‘) không. Hàm trả về khác 0 nếu đúng, ngược lại trả về giá trị bằng 0
  - – Ví dụ: printf("%d ",isdigit('A')); ⇒ 0

Khái niệm xâu ký tự→Kiểm tra ký tự đặc biệt

- int **iscntrl**(char ch)
  - – Kiểm tra kí tự điều khiển (0-31).
  - – Hàm trả về khác 0 nếu đúng, ngược lại trả về giá trị bằng 0
- int **isspace**(char ch)
  - – Kiểm tra kí tự dấu cách (mã 32), xuống dòng (‘\n’ 10), đầu dòng (‘\r’ 13), tab ngang (‘\t’ 9), tab dọc (‘\v’ 11).
  - – Hàm trả về khác 0 nếu đúng, ngược lại trả về giá trị bằng 0

Ví dụ: Nhập xâu và đếm từ, phân cách bởi dấu trắng

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
int main(){
    char Str[100]; int d=0, i=0;
    printf("Nhap xau ky tu: "); gets(Str);
    if(Str[0] == '\0') printf(" Xau rong ");
    else{
        if( ! isspace(Str[0]) ) d=1;
        i=1;
        while(Str[i] != '\0'){
            if(  isspace(Str[i-1] ) && (! isspace(Str[i])) )  d++;
            i++;
        }
        printf("Ket qua : %d",d);
    }
}
```

```
Nhap xau ky tu: Hello   world
Ket qua : 2_
```

```
Nhap xau ky tu:          Hello
Ket qua : 1
```

```
Nhap xau ky tu:    Hello
Ket qua : 1_
```


Các hàm xử lý xâu ký tự

Vào/ra xâu ký tự

- Tệp tiêu đề: **stdio.h**
- Nhập xâu kí tự
  - – gets(tên_xâu);
  - – scanf(“%s”,&tên_xâu);
- Hiển thị xâu kí tự
  - – puts(tên_xâu);
  - – printf(“%s”,tên_xâu);

Sự khác nhau giữa gets và scanf?

Các hàm xử lý xâu ký tự

Tệp tiêu đề: string.h

#include <string.h >

Chú ý:

char str[100] = “Hello world”;

char * p = str;

p là con trỏ tới mảng các ký tự/ xâu ký tự

- – p+6 cũng là xâu ký tự : world
- Xâu ký tự, có thể được khai báo **char ***

Các hàm xử lý xâu ký tự

size_t strlen(char * xâu)

– Trả về độ dài xâu

printf("%d ",strlen("Hello world")); ⇒ 11

char * strcpy(char * đích, char * nguồn)

– sao chép xâu, trả về giá trị xâu nguồn

char Str[20];

printf("%s ",strcpy(Str,"Hello")); ⇒ Hello

printf("%s", Str); ⇒ Hello

Chú ý: Phép gán Str = "Hello" là không hợp lệ

Các hàm xử lý xâu ký tự

int strcmp(char * xâu_1, char * xâu_2)

– So sánh hai xâu.
– Trả về giá trị 0 nếu hai xâu giống nhau;
– Giá trị < 0: xâu_1 < xâu_2
– Giá trị >0: xâu_1 > xâu_2

Ví dụ

char Str[20];

strcpy(Str,"hello");

printf("%d", strcmp(Str,"hello"));→0

printf("%d", strcmp(Str,"hello!"); →-1 (!?)

printf("%d", strcmp(Str,"Hello"); → 1 (!?)

Các hàm xử lý xâu ký tự

char * strcat(char * xđích, char * nguồn)

– Ghép nối xâu nguồn vào ngay sau xâu đích, trả lại xâu kết quả

Ví dụ

char Str[20];

strcpy(Str,"Hello ");

printf("%s ",strcat(Str,"world")); ⇒ Hello world

printf("\n%s",Str); ⇒ Hello world

Các hàm xử lý xâu ký tự

char * strchr (char * s, int c)

– Trả về con trỏ trỏ tới vị trí xuất hiện đầu tiên của ký tự c trong s. Nếu không có trả về con trỏ null

strcpy(Str,"Hello world");

printf("%s ",strchr(Str,'o')); ⇒ o world

char* strstr(char * s1, char * s2)

– Trả về con trỏ trỏ tới vị trí xuất hiện đầu tiên của chuỗi s2 trong s1. Nếu không tồn tại, trả về con trỏ null

printf("%s ",strstr(Str,"llo")); ⇒ llo world

Các hàm xử lý xâu ký tự (tiếp)

Tệp tiêu đề: stdlib.h

- int **atoi**(char * str):
  – Chuyển một xâu kí tự thành một số nguyên tương ứng
  – ***Ví dụ:*** atoi("1234") →1234
- int **atol**(char * str):
  – Chuyển xâu kí tự thành số long int
- float **atof**(char * str):
  – Chuyển xâu kí tự thành số thực
  – ***Ví dụ:*** atof("123.456E-2") →1.23456
- **Thất bại cả 3 hàm: trả về 0**

Ví dụ: Đảo ngược xâu ký tự

```
#include<stdio.h>
#include<string.h>
main(){
    char s[100],c;
    int i, n;
    printf("Nhap xau: ");gets(s);
    n =strlen(s);
    for(i=0;i <n/2;i++){
          c = s[i];
          s[i] = s[n-i-1];
          s[n-i-1]=c;
    }
    printf("%s",s);
}
```

Ví dụ: Kiểm tra xâu đối xứng

```
#include<stdio.h>
#include<string.h>
main(){
    char s[20];
    int i,n;
    printf("Nhap vao xau ki tu: ");gets(s);
    n=strlen(s);
    for(i=0;i<n/2;i++)
        if(s[i]!=s[n-1-i])
            break;
    if(i==n/2)
        printf("Xau doi xung");
    else
        printf("Xau khong doi xung");
}
```


Đếm số lần xuất hiện chữ cái trong xâu

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
main(){
   char s[20];
   int dem[26] = {};
   int i,n;
   puts("Nhap vao xau ki tu:");gets(s);
   n=strlen(s);
   for(i=0;i<n;i++)
      if(isalpha(s[i]))
         dem[ tolower(s[i ]) - 'a' ]++;
   for(i=0;i<26;i++)
     if(dem[i]!=0)
       printf("Ki tu %c xuat hien %d lan\n",'a'+i,dem[i]);
}
```


Đếm số lần xuất hiện chữ cái trong xâu

```
Nhap vao xau ki tu:
study, study more, study forever
Ki tu d xuat hien 3 lan
Ki tu e xuat hien 3 lan
Ki tu f xuat hien 1 lan
Ki tu m xuat hien 1 lan
Ki tu o xuat hien 2 lan
Ki tu r xuat hien 3 lan
Ki tu s xuat hien 3 lan
Ki tu t xuat hien 3 lan
Ki tu u xuat hien 3 lan
Ki tu v xuat hien 1 lan
Ki tu y xuat hien 3 lan
```


Mảng xâu ký tự

- Xâu ký tự có thể là kiểu phần tử của mảng
- Khai báo

  ```
  char DS[100][30];
  ```

  Mảng có tối đa 100 phần tử, các phần tử là xâu có độ dài tối đa 30
- Sử dụng
  - Như một mảng bình thường
  - Mỗi phần tử mảng được sử dụng như một xâu ký tự

Ví dụ: Nhập vào DSSV cho tới khi gặp tên rỗng, in DS

```
#include <stdio.h>
#include <string.h>
void main(){
  int i, n;
  char DS[100][30];
  printf("Nhap DSSV (<100), go Enter de thoat..\n");
    n =0;
    do{
     printf("Ten sinh vien[%d]: ",n+1);  gets(DS[n]);
     if(DS[n][0] !='\x0') n++;
     else break;
     if(n==100) break;
    }while(1);
    printf("\n\nDS sinh vien vua nhap \n");
    for(i=0;i<n;i++)  printf("%s\n",DS[i]);
}
```

```
if(strcmp(DS[n],""))n++
if(strlen(DS[n])>0)  n++
```


Ví dụ→Kết quả thực hiện

```
Turbo C++ IDE
Nhap DSSV (<100),go Enter de thoat..
Ten sinh vien[1]: Pham Viet Linh
Ten sinh vien[2]: Ha Minh Duc
Ten sinh vien[3]: Nguyen Binh An
Ten sinh vien[4]: Bui Thanh Trung
Ten sinh vien[5]: Du Duc Sang
Ten sinh vien[6]: Vu Tu Anh
Ten sinh vien[7]: Phung Nhat Huy
Ten sinh vien[8]: Le Quang Nam
Ten sinh vien[9]:

DS sinh vien vua nhap
Pham Viet Linh
Ha Minh Duc
Nguyen Binh An
Bui Thanh Trung
Du Duc Sang
Vu Tu Anh
Phung Nhat Huy
Le Quang Nam
```

Ví dụ: Nhập vào DS sinh viên, in ra DS đã sắp

```
#include <stdio.h>
#include <string.h>
void main(){
   int i, j, N;
   char  DS[100][30], str[30];
   //Nhap DS lop
   printf("So sinh vien : ");      scanf("%d",&N);
    fflush(stdin);
   for(i=0;i < N;i++){
      printf("Ten sinh vien[%d]: ",i);
      gets(DS[i]);
    }
```


Ví dụ

```
   //So sánh theo Họ+đệm+tên
   for(i = 0; i < N - 1; i ++)
      for(j = i +1; j < N; j ++)
        if(strcmp(DS[i],DS[j]) > 0){
                strcpy(str,DS[i]);
                strcpy(DS[i],DS[j]);
                strcpy(DS[j],str);
        }
   //In danh sách đã sắp xếp
   printf("\nDS sinh vien vua nhap \n");
   for(i=0;i < N;i++)
      printf("%s\n",DS[i]);
}//main
```


Ví dụ

```
So sinh vien : 10
Ten sinh vien[0]: Nguyen Hoai Nam
Ten sinh vien[1]: Le Tu Anh
Ten sinh vien[2]: Tran Thai Ha
Ten sinh vien[3]: Nguyen Van Tuan
Ten sinh vien[4]: Tran Thanh Son
Ten sinh vien[5]: Tran Hoai Nam
Ten sinh vien[6]: Bui Trong Son
Ten sinh vien[7]: Le Thanh Tuan
Ten sinh vien[8]: Le Thanh Son
Ten sinh vien[9]: Nguyen Nam Giang

DS sinh vien vua nhap
Bui Trong Son
Le Thanh Son
Le Thanh Tuan
Le Tu Anh
Nguyen Hoai Nam
Nguyen Nam Giang
Nguyen Van Tuan
Tran Hoai Nam
Tran Thai Ha
Tran Thanh Son
```


Ví dụ

```
//Sap xep theo tên
  char ten_i[30],ten_j[30];
   for(i = 0; i < N - 1; i ++)
     for(j = i +1; j < N; j ++){
        strcpy(ten_i,strrchr(DS[i],32));
        strcpy(ten_j,strrchr(DS[j],32));
        if(strcmp(ten_i,ten_j) > 0){
                strcpy(str,DS[i]);
                strcpy(DS[i],DS[j]);
                strcpy(DS[j],str);
       }
     }
```


Ví dụ

```
So sinh vien : 10
Ten sinh vien[0]: Nguyen Hoai Nam
Ten sinh vien[1]: Le Tu Anh
Ten sinh vien[2]: Tran Thai Ha
Ten sinh vien[3]: Nguyen Van Tuan
Ten sinh vien[4]: Tran Thanh Son
Ten sinh vien[5]: Tran Hoai Nam
Ten sinh vien[6]: Bui Trong Son
Ten sinh vien[7]: Le Thanh Tuan
Ten sinh vien[8]: Le Thanh Son
Ten sinh vien[9]: Nguyen Nam Giang

DS sinh vien vua nhap
Le Tu Anh
Nguyen Nam Giang
Tran Thai Ha
Tran Hoai Nam
Nguyen Hoai Nam
Bui Trong Son
Le Thanh Son
Tran Thanh Son
Nguyen Van Tuan
Le Thanh Tuan
```


Nội dung chính

Nội dung chính

1. **Khái niệm cấu trúc**
   - Khái niệm
2. **Khai báo cấu trúc**
   - Khái báo kiểu cấu trúc
   - Khai báo biến cấu trúc
   - Định nghĩa kiểu dữ liệu với **typedef**
3. **Xử lý dữ liệu cấu trúc**
   - Truy nhập các trường dữ liệu
   - Phép gán giữa các biến cấu trúc
   - Một số ví dụ

Ví dụ → Bài toán quản lý thí sinh thi đại học

Để quản lý cần lưu trữ các thông tin
- Số báo danh: Số nguyên không dấu
- Họ tên sinh viên: Chuỗi ký tự không quá 30
- Khối thi: Ký tự (A,B,C..)
- Tổng điểm 3 môn thi: kiểu thực

Do vậy với mỗi sinh viên cần các biến

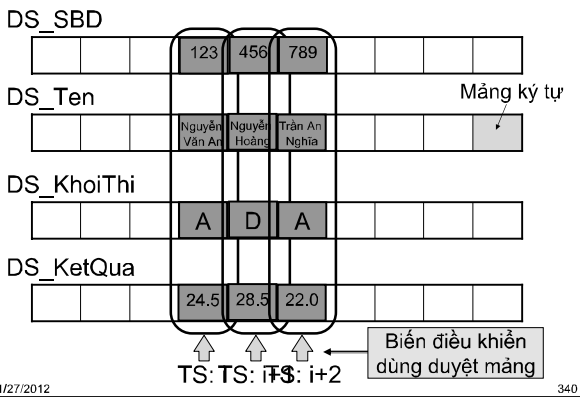
```
unsigned    SBD;
char        Ten[30];
char        KhoiThi;
float       KetQua;
```


Ví dụ → Bài toán quản lý thí sinh thi đại học (tiếp)

Để quản lý danh sách (dưới 1000) thí sinh dự thi, cần nhiều mảng rời rạc

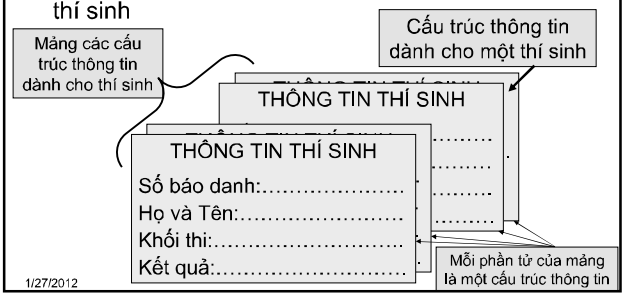
```
#define MAX 1000
unsigned    DS_SBD[MAX];
char        DS_Ten[MAX][30];
char        DS_KhoiThi[MAX];
float       DS_KetQua[MAX];
```


Ví dụ → Bài toán quản lý thí sinh thi đại học (tiếp)



Ví dụ → Vấn đề & giải pháp

Dùng nhiều mảng
- Khó quản lý, dễ nhầm lẫn
- Không thể hiện cấu trúc thông tin dành cho từng thí sinh



Khái niệm

- Cấu trúc là kiểu dữ liệu phức hợp, do người dụng tự định nghĩa
  - Kiểu cấu trúc bao gồm nhiều thành phần có thể thuộc các kiểu dữ liệu khác nhau
  - Các thành phần: gọi là trường dữ liệu (*field*)
  - Các thành phần, không được truy nhập theo chỉ số (*như mảng*) mà theo tên của trường.

Có thể coi một biến cấu trúc là một tập hợp của một hay nhiều biến rời rạc, thường có kiểu khác nhau thành một biến có một tên duy nhất để dễ dàng quản lý và sử dụng

Khái niệm → Ví dụ

- Kết quả học tập của sinh viên
  - TenSV: Chuỗi ký tự
  - MaSV: Chuỗi số/ số nguyên
  - Điem: Số thực
- Điểm trong mặt phẳng
  - Tên điểm: Ký tự (A, B, C..)
  - Hoành độ: Số thực
  - Tung độ: Số thực

Nội dung chính

1. **Khái niệm cấu trúc**
   - Khái niệm
2. **Khai báo cấu trúc**
   - Khái báo kiểu cấu trúc
   - Khai báo biến cấu trúc
   - Định nghĩa kiểu dữ liệu với **typedef**
3. **Xử lý dữ liệu cấu trúc**
   - Truy nhập các trường dữ liệu
   - Phép gán giữa các biến cấu trúc
   - Một số ví dụ

Khai báo kiểu cấu trúc

```
struct Tên_kiểu_cấu_ trúc {
   <Khai báo các trường dữ liệu>
};
```

- **struct:** từ khóa, cho phép người dùng khai báo kiểu dữ liệu mới: kiểu cấu trúc
- **Tên_kiểu_cấu_truc:** Tên của kiểu cấu trúc do người dùng tự định nghĩa
  - Tuân theo nguyên tắc đặt tên đối tượng trong C
- **Khai báo các trường dữ liệu:** Danh sách các khai báo thành phần (*trường:field*) của cấu trúc
  - Giống khai báo biến
  - Các trường có thể có kiểu bất kỳ

Khai báo kiểu cấu trúc → Ví dụ

Thẻ sinh viên

Số hiệu:…(*Chuỗi ký tự*)..
Tên sinh viên: (*Chuỗi ký tự*)
Năm sinh:…(*Số nguyên*)…
Khóa:……(*Số nguyên*)…….
Lớp:…… :.(*Chuỗi ký tự*). …

```
struct SinhVien{
   char SHSV[10];
   char Ten[30];
   int NS;
   int Khoa;
   char Lop [10];
};
```

Point2D

Hoành độ (x)…(*Số thực*)..
Tung độ (y)…..(*Số thực*)..

```
struct Point{
   float x, y;
};
```


Khai báo biến cấu trúc

- Khai báo **kiểu** cấu trúc nhằm tạo định nghĩa toàn thể cho các cấu trúc sẽ được dùng sau này
  - Không cung cấp không gian nhớ cho kiểu
- Khai báo **biến** cấu trúc nhằm yêu cầu chương trình tạo vùng nhớ để lưu trữ các dữ liệu cho biến cấu trúc
  - Chứa dữ liệu của các trường của cấu trúc

Khai báo biến cấu trúc → Cú pháp

Tồn tại định nghĩa kiểu cấu trúc

```
struct Kiểu_cấu_ trúc Tên_biến;
```

Khai báo trực tiếp

```
struct {
   <Khai báo các trường dữ liệu>
}Tên_biến;
```

Kết hợp với khai báo kiểu

```
struct Kiểu_cấu_ trúc {
   <Khai báo các trường dữ liệu>
}Tên_biến;
```

Khai báo biến cấu trúc → Ví dụ

Tồn tại định nghĩa kiểu cấu trúc

```
struct SinhVien SV1, SV2, Thu khoa;
```

Khai báo trực tiếp

```
struct {
  float x, y; //Tọa độ trên mặt phẳng
}A, B;      //Khai báo 2 điểm A, B
```

Kết hợp với khai báo kiểu

```
struct Point_3D{
  float x, y, z;// Tọa độ không gian
}A, B;
```


Khai báo biến cấu trúc →Chú ý

Các cấu trúc có thể được khai báo lồng nhau

```
struct diem_thi {
      float dToan, dLy, dHoa;
}
struct thi_sinh{
      char SBD[10];
      char ho_va_ten[30];
      struct  diem_thi   ket_qua;
} thi_sinh_1, thi_sinh_2;
```


Khai báo biến cấu trúc →Chú ý

Có thể khai báo trực tiếp các trường dữ liệu của một cấu trúc bên trong cấu trúc khác

```
struct thi_sinh{
      char SBD[10];
      char ho_va_ten[30];
      struct{
            float dToan, dLy, dHoa;
      } ket_qua;
} thi_sinh_1, thi_sinh_2;
```


Khai báo biến cấu trúc →Chú ý

Có thể gán giá trị khởi đầu cho một biến cấu trúc, theo nguyên tắc như kiểu mảng

Ví dụ:

```
struct Date{
  int day;
  int month;
  int year;
};
struct SinhVien{
  char Ten[30];
  struct Date NS;
} SV = {"Tran Anh", 20, 12, 1990 };
```

```
struct SinhVien{
  char Ten[20];
  struct Date{
        int day;
        int month;
        int year;
        } NS;
} SV = {"Tran Anh", 20,12,1990 };
```


Định nghĩa kiểu dữ liệu với **typedef**

```
typedef <tên_cũ> <tên_mới>;
```

Mục đích

- Đặt tên mới đồng nghĩa với tên của một kiểu dữ liệu đã được định nghĩa
  - Thường được sử dụng cho kiểu cấu trúc
    - Giúp cho khai báo trở nên quen thuộc và ít bị sai hơn

Ví dụ

```
typedef char Str80[80] ;
typedef long mask;
Str80 str="Bonjour tout le monde !";
mask a, b;
```


Định nghĩa kiểu dữ liệu với **typedef**

Thường được kết hợp với kiểu cấu trúc để khai báo một bí danh cho một cấu trúc

- Giúp khai báo trở nên quen thuộc và ít bị sai hơn

```
typedef struct {  //Định nghĩa một cấu trúc
  char SHSV[10];
  char Ten[30];
  int NS;
  int Khoa;
  char Lop [10];
} SinhVien;          //Đặt tên cho cấu trúc là SinhVien
SinhVien SV;         //Tạo một biến cấu trúc
```

Định nghĩa kiểu dữ liệu với **typedef** →Chú ý

Cho phép đặt tên mới trùng với tên cũ

Ví dụ

```
struct point_3D{
    float x, y, z;
 }
struct point_3D  M;
typedef struct point_3D point_3D;
point_3D  N;
```

```
typedef struct {
    float x, y, z;
 }point_3D;
point_3D M;
point_3D N;
```


Định nghĩa kiểu dữ liệu với **typedef** →Chú ý

```
typedef struct point_2D   {
        float x, y;
}point_2D, diem_2_chieu, ten_bat_ki;
point_2D   X;
diem_2_chieu   Y;
ten_bat_ki    Z;
```

Chú ý:

point_2D, diem_2_chieu, ten_bat_ki là các tên cấu trúc, không phải tên biến

Chương 6: Cấu trúc

Nội dung chính

1. **Khái niệm cấu trúc**
   - Khái niệm
2. **Khai báo cấu trúc**
   - Khái báo kiểu cấu trúc
   - Khai báo biến cấu trúc
   - Định nghĩa kiểu dữ liệu với **typedef**
3. **Xử lý dữ liệu cấu trúc**
   - Truy nhập các trường dữ liệu
   - Phép gán giữa các biến cấu trúc
   - Một số ví dụ

Truy cập các trường dữ liệu

- Cú pháp

  tên_biến_cấu_trúc.tên_trường
- Lưu ý
  - Dấu "**.**" là toán tử truy cập vào trường dữ liệu trong cấu trúc
  - Nếu trường dữ liệu là một cấu trúc => sử dụng tiếp dấu "**.**" để truy cập vào thành phần mức sâu hơn

Ví dụ

```
#include <stdio.h>
void main(){
   struct{
     char Ten[20];
     struct Date{
                int day;
                        int month;
                        int year;
                } NS;
    } SV = {"Tran Anh", 20,12, 1990 };
   printf(" Sinh vien %s (%d/%d/%d)",
   SV.Ten,SV.NS.day,SV.NS.month,SV.NS.year);
 }
```

```
Sinh vien Tran Anh (20/12/1990)
```


Ví dụ

Xây dựng một cấu trúc biểu diễn điểm trong không gian 2 chiều. Nhập giá trị cho một biến kiểu cấu trúc này, sau đó hiển thị giá trị các trường dữ liệu của biến này ra màn hình.

- Cấu trúc: tên điểm, tọa độ x, tọa độ y
- Nhập, hiển thị từng trường của biến cấu trúc như các biến dữ liệu khác

Ví dụ

```
#include<stdio.h>
#include<conio.h>
typedef struct{
   char ten[5];
   int x,y;
}toado;
void main(){
  toado t;
  printf("Nhap thong tin toa do\n");
  printf("Ten diem: ");gets(t.ten);
  printf("Toa do x: ");scanf("%d",&t.x);
  printf("Toa do y: ");scanf("%d",&t.y);
  printf("Gia tri cac truong\n");
  printf("%-5s%3d%3d\n",t.ten,t.x,t.y);
  getch();
}
```


Phép gán giữa các biến cấu trúc

- Muốn sao chép dữ liệu từ biến cấu trúc này sang biến cấu trúc khác cùng kiểu
  - gán lần lượt từng trường trong hai biến cấu trúc
  - C cung cấp phép gán hai biến cấu trúc cùng kiểu:

    biến_cấu_trúc_1 = biến_cấu_trúc_2;

Phép gán giữa các biến cấu trúc

- Ví dụ
  - Xây dựng cấu trúc gồm họ tên và điểm TĐC của sinh viên
  - a, b, c là 3 biến cấu trúc.
  - Nhập giá trị cho biến a.
  - Gán b=a,
  - gán từng trường của a cho c.
  - So sánh a, b và c ?

Ví dụ

```
#include<stdio.h>
#include<conio.h>
typedef struct{
   char hoten[20];
   int diem;
}sinhvien;
void main(){
  sinhvien a,b,c;
  printf("Nhap thong tin sinh vien\n");
  printf("Ho ten: ");gets(a.hoten);
  printf("Diem:");scanf("%d",&a.diem);
```


Ví dụ

```
   b=a;
   strcpy(c.hoten,a.hoten);
    c.diem=a.diem;
   printf("Bien a: ");
   printf("%-20s%3d\n",a.hoten,a.diem);
   printf("Bien b: ");
   printf("%-20s%3d\n",b.hoten,b.diem);
   printf("Bien c: ");
   printf("%-20s%3d\n",c.hoten,c.diem);
   getch();
}
```


Ví dụ →Kết quả

```
Nhap thong tin sinh vien
Ho ten: Nguyen Van Anh
Diem:9
Bien a: Nguyen Van Anh          9
Bien b: Nguyen Van Anh          9
Bien c: Nguyen Van Anh          9
```

Bài tập

1. Lập trình đọc vào một danh sách không quá 100 sinh viên gồm: Họ tên, năm sinh
   1. Đưa ra DS những sinh viên sinh năm 1990
   2. Đưa ra DSSV đã sắp xếp theo thứ tự ABC
2. Lập trình đọc vào DS thí sinh gồm Họ tên, điểm thi 3 môn Toán, Lý,Hóa, kết thúc nhập khi gặp sinh viên có tên rỗng
   1. Đọc tiếp vào một điểm chuẩn; đưa ra danh sách thí sinh trúng tuyển (*không có điểm liệt - 0*)
   2. Đưa ra thí sinh cao điểm nhất
   3. Tìm điểm chuẩn, nếu chỉ lấy K SV, K nhập vào. Nếu có nhiều người bằng điểm nhau; loại cả

Bài tập 1

```
#include <stdio.h>
#include <string.h>
typedef struct{
   char Ten[30];
   int NS;
}SinhVien;
void main(){
   SinhVien DS[100], SV;
   int N,i,j;
   printf("Nhap So sinh vien : "); scanf("%d",&N);
   fflush(stdin);
   for ( i=0; i < N; i++ ){
      fflush(stdin);
      printf("Nhap du lieu cho sinh vien %d: \n", i+1);
      printf("Ho ten : "); gets(DS[i].Ten);
      printf("Nam sinh :");scanf("%d", &DS[i].NS);
   }
```


Bài tập 1 (*tiếp*)

```
   printf("\n\n DANH SACH SINH VIEN\n\n");
   for(i = 0; i < N; i ++)
      if(DS[i].NS ==1990)
         printf("%s\n",DS[i].Ten);

   for(i = 0; i < N - 1; i ++)
      for(j = i+1; j < N; j ++)
            if(strcmp(DS[i].Ten,DS[j].Ten) > 0){
                  SV= DS[i];
                  DS[i]=DS[j];
                  DS[j] = SV;
            }

   printf("\n\n DANH SACH SINH VIEN DA SAP XEP\n\n");
   for(i = 0; i < N; i ++)
         printf("%s\n",DS[i].Ten);
}//main
```


Bài tập 1 →Kết quả

```
Nhap So sinh vien : 7
Nhap du lieu cho sinh vien 1:
Ho ten : Nguyen Thanh Nga
Nam sinh :1991
Nhap du lieu cho sinh vien 2:
Ho ten : Tran Hoai Thanh
Nam sinh :1990
Nhap du lieu cho sinh vien 3:
Ho ten : Le Nam Anh
Nam sinh :1990
Nhap du lieu cho sinh vien 4:
Ho ten : Bui Duc Cuong
Nam sinh :1992
Nhap du lieu cho sinh vien 5:
Ho ten : Trinh Quoc Anh
Nam sinh :1989
Nhap du lieu cho sinh vien 6:
Ho ten : Le Danh Tra
Nam sinh :1990
Nhap du lieu cho sinh vien 7:
Ho ten : Hoang Mai Huynh
Nam sinh :1991
```

```
DANH SACH SINH VIEN

Tran Hoai Thanh
Le Nam Anh
Le Danh Tra
```

```
DANH SACH SINH VIEN DA SAP XEP

Bui Duc Cuong
Hoang Mai Huynh
Le Danh Tra
Le Nam Anh
Nguyen Thanh Nga
Tran Hoai Thanh
Trinh Quoc Anh
```


Bài tập 2 (*1/4*)

```
#include <stdio.h>
#include <string.h>

typedef struct{
   char Ten[30];
   struct{
     int T, L, H, S;
   } DT;
}SinhVien;

void main(){
   SinhVien DS[100], TK, SV;
   int N,i,j,K;
   float C;
```


Bài tập 2 (*2/4*)

```
N = 0;
do{
  printf("\nNhap DL cho sv thu %d\n",N+1);
  printf("Ten SV : "); gets(DS[N].Ten);
  if(strlen(DS[N].Ten)==0)
     break;
  else{
     printf("Diem thi T L H cua SV %s : ",DS[N].Ten);
      scanf("%d%d%d",&DS[N].DT.T,&DS[N].DT.L,&DS[N].DT.H);
     fflush(stdin);
     DS[N].DT.S = DS[N].DT.T + DS[N].DT.L + DS[N].DT.H;
     N++;
  }
}while(1);
printf("\n\n DANH SACH SINH VIEN\n\n");
printf("      Ten SV          Toan  Ly   Hoa Tong \n");
for(i = 0; i < N; i ++)
   printf("%-20s%5d%5d%5d%6d\n",DS[i].Ten,
         DS[i].DT.T,DS[i].DT.L,DS[i].DT.H,DS[i].DT.S);
```

Bài tập 2 (3/4)

```
printf("\n\nDiem Chuan : ");scanf("%f",&C);
printf("\n\n DANH SACH SINH VIEN TRUNG TUYEN \n\n");
for(i = 0; i < N; i ++)
  if( (DS[i].DT.S >= C)&&
      (DS[i].DT.T*DS[i].DT.L*DS[i].DT.H > 0))
      printf("%s\n",DS[i].Ten);

TK = DS[0];
for(i = 1; i < N; i ++)
   if(DS[i].DT.S > TK.DT.S)
        TK = DS[i];
for(i = 0; i < N; i ++)
   if(DS[i].DT.S == TK.DT.S)
     printf("\n\n THU KHOA: %s \n\n",TK.Ten);
```


Bài tập 2 (4/4)

```
   printf("\nSo nguoi trung tuyen:"); scanf("%d",&K);
   for(i = 0; i < N - 1; i ++)
       for(j = i+1; j < N; j ++)
          if(DS[i].DT.S < DS[j].DT.S ){
               SV= DS[i];
               DS[i]=DS[j];
               DS[j] = SV;
          }
    while((K>0)&&(DS[K-1].DT.S==DS[K].DT.S))K--;
    if(K>0){
     printf("Diem Chuan La: %4d",DS[K-1].DT.S);
     printf("\n\n Danh Sach sinh vien trung tuyen \n");
     for(i=0; i < K; i++)
          printf("%s\n",DS[i].Ten);
    }
}//main
```


Câu hỏi 1: Kết quả đưa ra màn hình

```
#include<stdio.h>
 typedef struct {
   int SHSV;
   char Ten[25];
 }SV;
void main(){
  SV DS[] = { {12, "Mai"},
              {13, "Nam"},
              {14, "Minh"}};
   printf("%d ", DS[1].SHSV);
   printf("%s\n", (*(DS+2)).Ten);
}
```

a	12 Mai
b	12 Nam
c	13 Nam
d	13 Minh
e	14 Minh

Bài tập 1

Lập trình thực hiện các công việc sau

- Đọc vào từ bàn phím một danh sách thuốc gồm
  - Tên thuốc (chuỗi không quá 20 ký tự)
  - Năm hết hạn
  - Số lượng còn
  - Đơn giá

  Kết thúc nhập khi gặp thuốc có tên »*** »
- Đưa danh sách thuốc ra màn hình
- Đưa ra danh sách các thuốc đã hết hạn
- Xóa khỏi danh sách những thuốc đã hết hạn. Đưa danh sách mới ra màn hình
- Tính tổng giá trị các thuốc đã hết hạn

Bài tập 2

Cho một danh sách thành tích thi đấu bóng đá của 32 đội tuyển bao gồm: Tên đội bóng, số bàn thắng, số bàn thua, số thẻ đỏ, số thẻ vàng

Viết chương trình thực hiện

- Nhập dữ liệu vào từ bàn phím
- Nhập vào tên đội bóng,
  - đưa ra thành thích của đội này
  - Nếu không tồn tại, thông báo: không tìm thấy
- Tính và đưa ra màn hình số điểm của các đội nếu
  - Mỗi bàn thắng được tính 10 điểm
  - Mỗi bàn thua bị phạt 5 điểm, mỗi thẻ vàng trừ 2 điểm, thẻ đỏ trừ 5 điểm

Nội dung chính

Chương 7: Hàm

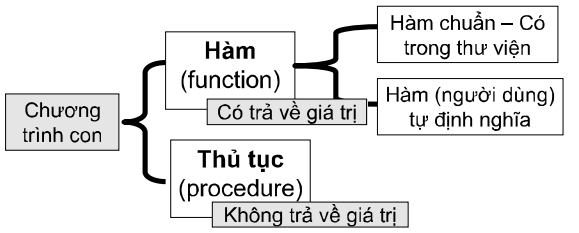
Nội dung chính

1. **Khái niệm hàm**
   - Khái niệm chương trình con
   - Phân loại: hàm và thủ tục
2. **Khai báo và sử dụng hàm**
   - Khai báo và sử dụng
3. **Phạm vi của biến**
   - Toàn cục và địa phương
4. **Truyền tham số**
   - Truyền theo giá trị, truyền theo địa chỉ
   - Biến static, biến register

Khái niệm & Vai trò

- Khái niệm
  - Là một chương trình nằm trong một chương trình lớn hơn nhằm thực hiện một nhiệm vụ cụ thể
- Vai trò
  - Chia nhỏ chương trình ra thành từng phần để quản lý
    - Phương pháp lập trình có cấu trúc
  - Có thể sử dụng lại nhiều lần: printf(), scanf()…
  - Chương trình dễ dàng đọc và bảo trì hơn

Phân loại



Ngôn ngữ lập trình C

- Chỉ cho phép khai báo chương trình con là hàm.
- Sử dụng kiểu "void" với ý nghĩa "không là kiểu dữ liệu nào cả" để chuyển thủ tục về dạng hàm

Chương 7: Hàm

Nội dung chính

1. **Khái niệm hàm**
   - Khái niệm chương trình con
   - Phân loại: hàm và thủ tục
2. **Khai báo và sử dụng hàm**
   - Khai báo và sử dụng
3. **Phạm vi của biến**
   - Toàn cục và địa phương
4. **Truyền tham số**
   - Truyền theo giá trị, truyền theo địa chỉ
   - Biến static, biến register

Ví dụ

Khai báo chương trình con

Gọi chương trình con ra thực hiện

```
#include<stdio.h>
int bp(int x){
   int y;
   y =  x * x;
   return y;
}
void main(){
   int i;
   for (i=1; i< 20; i+=2)
        printf("%4d\n",bp(i));
   printf("\n");
}
```

```
   1
   9
  25
  49
  81
 121
 169
 225
 289
 361
```


Định nghĩa hàm

Cú pháp

Dòng đầu hàm

```
Kiểu_hàm Tên_hàm(DS khai báo tham số)
{
    [<Các khai báo cục bộ>]
    [<Các câu lệnh>]
}
```

Thân hàm

Dòng đầu hàm

Kiểu_hàm Tên_hàm(DS khai báo tham số)

- Mô tả các thông tin được trao đổi giữa bên trong và bên ngoài hàm.
  - Tên của hàm,
  - Các tham số đầu vào
    - Hàm cần những thông tin gì để hoạt động
  - Tham số đầu ra và giá trị trả về
    - Hàm cung cấp những thông tin gì cho môi trường
- Dùng phân biệt các hàm với nhau,
  - không tồn tại 2 hàm có dòng đầu hàm giống nhau.

Dòng đầu hàm→Tên hàm

Là tên do người sử dụng tự định nghĩa

- Tuân theo quy tắc đặt tên đối tượng
- Nên mang ý nghĩa gợi ý chức năng của hàm

Dòng đầu hàm→Khai báo các tham số hình thức

- Khai báo các thông tin cần cho hoạt động của hàm và các thông tin, kết quả tính toán được hàm trả lại.
  - Tham số chứa dữ liệu vào cung cấp cho hàm
  - Tham số chứa dữ liệu ra mà hàm tính toán được.
- Các tham số sử dụng trong khai báo hàm là tham số hình thức.
  - Nguyên tắc khai báo tham số hình thức như giống như khai báo một biến
    kiểu_dữ_liệu_của_tham_số tên_của_tham_số

Dòng đầu hàm→Khai báo các tham số hình thức

- Các tham số cung cấp cho hàm trong quá trình thực hiện hàm là tham số thực sự
  - Kiểu dữ liệu của tham số thực phải giống kiểu dữ liệu của tham số hình thức tương ứng với tham số thực sự đó,.
- Một hàm có thể có một, nhiều hoặc không có tham số nào cả
  - Nếu có nhiều tham số, phải được phân cách với nhau bằng dấu phẩy.
  - không có tham số vẫn phải có cặp dấu ngoặc đơn sau tên hàm

Dòng đầu hàm→Kiểu dữ liệu trả về

- Thông thường hàm sau khi được thực hiện sẽ trả về một giá trị kết quả tính toán nào đó.
- Để sử dụng được giá trị đó cần phải biết nó thuộc kiểu dữ liệu gì.
  - Kiểu dữ liệu của đối tượng tính toán được hàm trả về được gọi là kiểu dữ liệu trả về của hàm.

Dòng đầu hàm→Kiểu dữ liệu trả về

- Trong C, kiểu dữ liệu trả về của hàm có thể là kiểu dữ liệu bất kì (kiểu dữ liệu có sẵn hoặc kiểu dữ liệu do người dùng tự định nghĩa) nhưng **không được là kiểu dữ liệu mảng**.
- Nếu kiểu dữ liệu trả về là kiểu **void** thì hàm không trả về giá trị nào cả.
- Nếu không khai báo kiểu dữ liệu trả về thì chương trình dịch của C sẽ ngầm hiểu rằng kiểu dữ liệu trả về của hàm là kiểu **int**.

Thân hàm

- Danh sách các câu lệnh
- Thường có ít nhất một lệnh return

Hoạt động của hàm

- Thực hiện lần lượt các lệnh cho đến khi
  - Thực hiện xong tất cả các câu lệnh có trong thân hàm
  - Gặp lệnh return
    - Cú pháp chung

      return [biểu_thức];

Thân hàm (tiếp)

Khi gặp lệnh *return biểu_thức*

- Tính toán giá trị của ***biểu_thức***,
- Lấy kết quả tính toán được làm giá trị trả về cho lời gọi hàm
- Kết thúc việc thực hiện hàm, trở về chương trình đã gọi nó.

Nếu **return** không có phần ***biểu_thức***,

- Kết thúc thực hiện hàm mà không trả về giá trị nào cả.
  - Dùng khi hàm được khai báo có kiểu trả về là **void**

Sử dụng hàm

Tên_hàm (DS_tham_số_thực _sự);

Ví dụ:

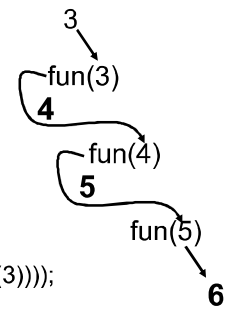
N = bp(1);N= bp(3);,…

Lưu ý:

- Gọi hàm thông qua tên hàm và các tham số được cung cấp thực sự cho hàm (*tham số thực sự*).
- Nếu hàm nhận nhiều tham số thì các tham số ngăn cách nhau bởi dấu phẩy
- Các tham số hình thức của hàm sẽ nhận các giá trị từ tham số truyền vào
- Sau khi thực hiện xong, trở về điểm mà hàm được gọi

Ví dụ: Cho biết kết quả thực hiện chương trình

```
#include<stdio.h>

int fun(int a){
   a++;
   return a;
}

int main(){
    printf("%d\n", fun(fun(fun(3))));
   return 0;
}
```



Ví dụ: Cho biết kết quả thực hiện chương trình

```
#include<stdio.h>

int fun(int n){
   if(n==0) return 1;
   else return n*fun(n-1);
}

int main(){
    printf("%d\n", fun(5));
   return 0;
}
```

fun(5)
5 * fun(4)
4 * fun(3)
3 * fun(2)
2 * fun(1)
1 * fun(0)
1

120

Ví dụ 1: Tính TBC f(a),f(b), f(c) nếu f(x) = $f(x) = x^5 + \sqrt[3]{x}$

```
#include <stdio.h>
#include <math.h>
float f(float x){
  if(x==0.0)
    return 0;
  else
      return pow(x,5)+x/fabs(x) * pow(fabs(x), 0.2);
}
void main(){
    float a, b, c;
    printf("So 3 so thuc : "); scanf("%f%f%f",&a,&b,&c);
    printf("Ket qua %f \n",(f(a)+f(b)+f(c))/3);
}
```

```
So 3 so thuc : 1 2 3
Ket qua 93.131475
```

VD Đọc tọa độ 3 điểm A,B,C và đưa ra d/tích ΔABC

```
#include <stdio.h>
#include <math.h>
typedef struct{
    float x, y;
}Point;

float kc(Point A, Point B){
    return sqrt(pow(A.x-B.x,2)+pow(A.y-B.y,2));
}
```


VD Đọc tọa độ 3 điểm A,B,C và đưa ra d/tích ΔABC

```
void main(){
  Point A, B, C;
  float AB,BC,CA,p,S;
  printf("Toa do A (x,y) :"); scanf("%f%f",&A.x,&A.y);
  printf("Toa do B (x,y) :"); scanf("%f%f",&B.x,&B.y);
  printf("Toa do C (x,y) :"); scanf("%f%f",&C.x,&C.y);
  AB = kc(A,B);    BC = kc(B,C);    CA = kc(C,A);
  p= (AB + BC + CA)/2;
  S = sqrt(p*(p-AB)*(p-BC)*(p-CA));
  printf("Dien tich ABC %f",S);
}
```


Ví dụ 3: Tìm ƯSCLN của dãy số

```
1.  # include <stdio.h>
2.  int uscln(int a, int b) {
3.    while (a !=b){
4.      if(a > b) a = a- b;
5.      else b = b - a;
6.    }
7.    return a;
8.  }
9.  void main(){
10.   int A[100], N, i, r;
11.   printf("So phan tu : "); scanf("%d",&N);
12.   for(i=0; i < N; i++){
13.     printf("A[%d] = ",i+1); scanf("%d",&A[i]);
14.   }
15.  r = A[0];
16.  for(i = 1; i < N; i++)
17.     r = uscln(r,A[i]);
18.  printf("Ket qua %d \n",r);
19. }
```

```
So phan tu : 6
A[1] = 120
A[2] = 48
A[3] = 96
A[4] = 72
A[5] = 36
A[6] = 180
Ket qua 12
```


Nội dung chính

1. **Khái niệm hàm**
   - Khái niệm chương trình con
   - Phân loại: hàm và thủ tục
2. **Khai báo và sử dụng hàm**
   - Khai báo và sử dụng
3. **Phạm vi của biến**
   - Toàn cục và địa phương
4. **Truyền tham số**
   - Truyền theo giá trị, truyền theo địa chỉ
   - Biến static, biến register

Phạm vi

- Phạm vi:
  - Khối lệnh, chương trình con, chương trình chính
- Biến chỉ có tác dụng trong phạm vi được khai báo
- Trong cùng một phạm vi các biến phải có tên khác nhau.

Tình huống

- Trong hai phạm vi khác nhau có hai biến cùng tên. Trong đó một phạm vi này nằm trong phạm vi kia?

```
#include<stdio.h>
#include<conio.h>
int i;
int binhphuong(int x){
   int y;
   y =  x * x;
   return y;
}
void main(){
  int y;
  for (i=0; i<= 10; i++){
     y = binhphuong(i);
     printf("%d  ", y);
  }
}
```


Phân loại biến

- **Biến toàn cục:**
  - Biến được khai báo trong chương trình chính, được đặt sau khai báo tệp tiêu đề.
- **Biến cục bộ:**
  - biến được khai báo trong lệnh khối hoặc chương trình con, được đặt trước các câu lệnh.

Ghi chú

- Hàm main() cũng là một chương trình con nhưng là nơi chương trình được bắt đầu
- Biến khai báo trong hàm main() cũng là biến cục bộ, chỉ có phạm vi trong hàm main().

Biến static

- Biến cục bộ ra khỏi phạm vi thì bộ nhớ dành cho biến được giải phóng
- Yêu cầu lưu trữ giá trị của biến cục bộ một cách lâu dài => sử dụng từ khóa *static*

Cú pháp:

static <kiểu_dữ_liệu> tên_biến;

Ví dụ→Kết quả

```
#include <stdio.h>
# include <conio.h>
void fct() {
   static int count = 1;
   printf("\n Day la lan goi ham fct lan thu %2d", count++);
}
void main(){
   int i;
   for(i = 0;  i < 10; i++)  fct();
   getch();
}
```


Ví dụ

```
Day la lan goi ham fct lan thu  1
Day la lan goi ham fct lan thu  2
Day la lan goi ham fct lan thu  3
Day la lan goi ham fct lan thu  4
Day la lan goi ham fct lan thu  5
Day la lan goi ham fct lan thu  6
Day la lan goi ham fct lan thu  7
Day la lan goi ham fct lan thu  8
Day la lan goi ham fct lan thu  9
Day la lan goi ham fct lan thu 10
```


Biến register

- Thanh ghi có tốc độ truy cập nhanh hơn RAM, bộ nhớ ngoài
- Lưu biến trong thanh ghi sẽ tăng tốc độ thực hiện chương trình

Cú pháp

register <kiểu_dữ_liệu> tên_biến;

Lưu ý:

– số lượng biến register không nhiều và thường chỉ với kiểu dữ liệu nhỏ như int, char

Nội dung chính

Ví dụ

```
# include <stdio.h>
void swap(int a, int b) {
    int x = a;
      a = b;
      b = x;
      return;
}
void main(){
   int a = 5, b = 100;
   printf("Truoc: a=%d, b=%d \n\n",a,b);
   swap(a,b);
   printf("Sau  : a=%d, b=%d\n\n",a,b);
   return;
}
```

```
Truoc: a=5, b=100

Sau  : a=5, b=100
```

Truyền theo giá trị và truyền theo biến

- **Truyền theo trị**
  - Dựa trên nguyên tắc truyền những bản sao của biến được truyền
  - Những câu lệnh thay đổi giá trị tham số hình thức sẽ không ảnh hưởng tới biến được truyền
- **Truyền theo biến**
  - Tham số được truyền sẽ thực sự là biến và các thao tác sẽ thi hành trực tiếp với biến
  - Những câu lệnh thay đổi giá trị tham số hình thức sẽ ảnh hưởng tới biến được truyền

Truyền theo biến

- Thực chất là truyền theo **địa chỉ** của biến
- Khai báo hàm [tham số phải chứa "*địa chỉ*"]:
  - Khai báo là một con trỏ, trỏ tới một đối tượng có kiểu muốn truyền vào
  - Ví dụ: void swap (int *pa, int *pb);
- Truyền tham số
  - Địa chỉ của biến được truyền
  - Ví dụ: swap(&a,&b)

Ví dụ

```
# include <stdio.h>
# include <conio.h>
void swap(int * pa, int * pb) {
int x = *pa;
   *pa = *pb;
   *pb = x;
   return;
}
void main(){
   int a = 5, b = 100;
   printf("Truoc: a=%d, b=%d \n\n",a,b);
   swap(&a,&b);
   printf("Sau  : a=%d, b=%d\n\n",a,b);
   return;
}
```

```
Truoc: a=5, b=100

Sau  : a=100, b=5
```


Câu hỏi 1: Kết quả đưa ra màn hình

```
#include<stdio.h>
void fun(int n){
   if(n > 0)   {
      fun(--n);
      printf("%d ", n);
      fun(--n);
   }
}
int main(){
    fun(3);
   return 0;
}
```

a	0 2 1 0
b	0 1 0 2
c	1 1 2 0
d	0 1 2 0
e	0 2 0 1

Câu hỏi 2: Kết quả đưa ra màn hình

```
#include<stdio.h>
void fun(int *i, int *j){
   *i = *i * *i;
   *j = *j * *j;
}
int main(){
   int i=5, j=2;
   fun(&i, &j);
   printf("%d, %d", i, j);
   return 0;
}
```

a	5, 2
b	2, 5
c	10, 4
d	4, 25
e	25, 4

Câu hỏi 3: Kết quả đưa ra màn hình

```
#include<stdio.h>
void fun(char *a){
   printf("%c", *++a);
   a++;
   printf("%c", *a);
}
int main(){
   void fun(char*);
   char a[10]="ABCDEF";
   fun(&a[0]);
   return 0;
}
```

a	AB
b	AC
c	BC
d	BD
e	CD

Nội dung chính

- Chương 1: Tổng quan về ngôn ngữ C
- Chương 2: Kiểu dữ liệu và biểu thức trong C
- Chương 3: Vào ra dữ liệu
- Chương 4: Cấu trúc điều khiển
- Chương 5: Mảng, con trỏ và xâu ký tự
- Chương 6: Cấu trúc
- Chương 7: Hàm
- *Chương 8: Tệp dữ liệu*