CƠ SỞ LẬP TRÌNH

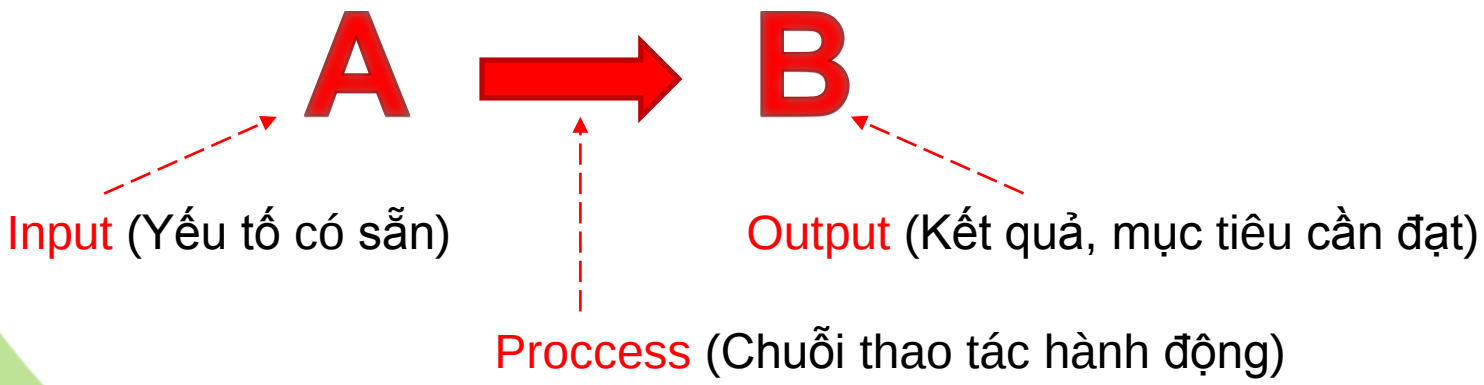
CÁC KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH

Nội dung

- Các khái niệm cơ bản
- Các bước xây dựng chương trình
- Tổng quan ngôn ngữ lập trình
- Giới thiệu ngôn ngữ lập trình C

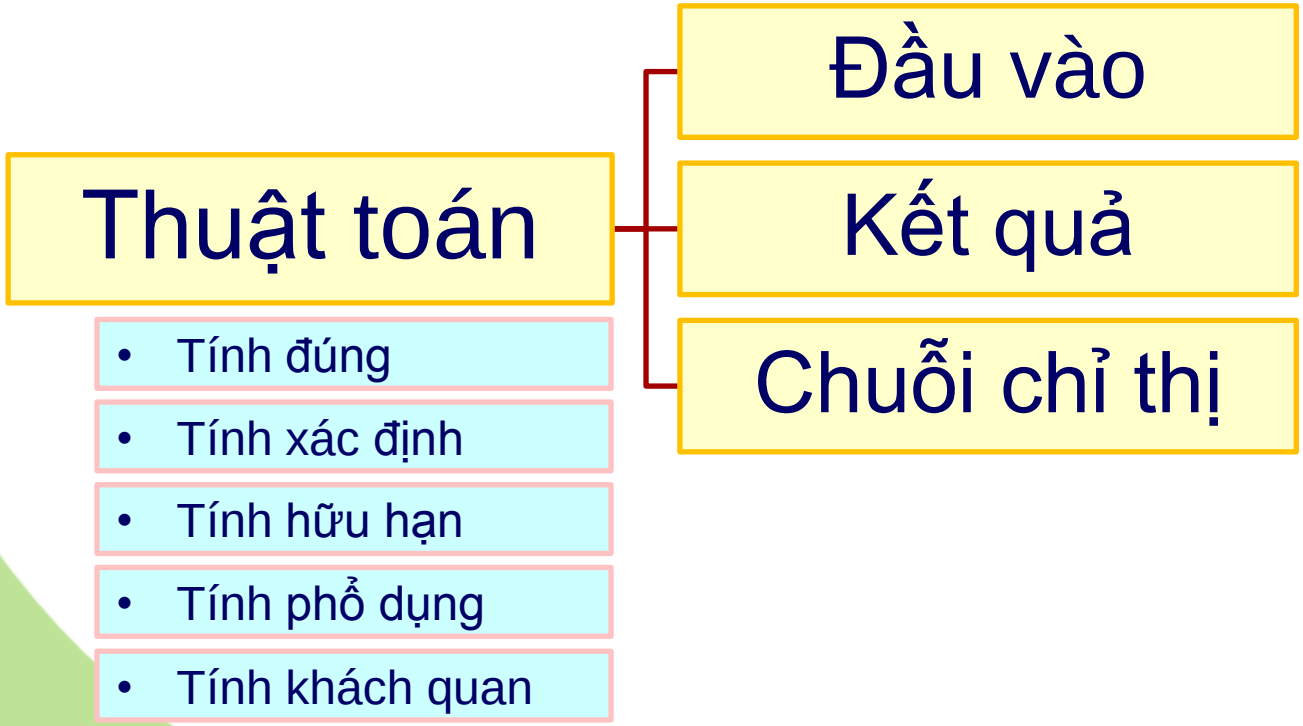
1. Các khái niệm cơ bản

- Vấn đề
  - Những vướng mắc, khó khăn cần giải quyết
- Bài toán
  - Vấn đề được giải quyết bằng tính toán
- Dạng tổng quát của bài toán



1. Các khái niệm cơ bản

- Thuật toán (Algorithm)
  - Dãy *hữu hạn* các *chỉ thị* được định nghĩa *rõ ràng* và *thực hiện được* nhằm giải quyết một *bài toán cụ thể* nào đó



1. Các khái niệm cơ bản

- Chương trình máy tính (computer program)
  - Tập hợp các chỉ thị được biểu thị qua ngôn ngữ lập trình nhằm mục đích thực hiện một số thao tác máy tính nào đó
- Lập trình máy tính (computer programming)
  - Quá trình *cài đặt* một hoặc nhiều *thuật toán* trừu tượng có liên quan với nhau bằng một *ngôn ngữ lập trình* để tạo ra một *chương trình máy tính* phục vụ cho việc giải quyết bài toán

2. Các bước xây dựng chương trình

Biểu diễn bằng
- Ngôn ngữ tự nhiên
- Lưu đồ
- Mã giả
- Ngôn ngữ lập trình

Lỗi phát sinh
- Lỗi cú pháp
- Lỗi ngữ nghĩa

2. Các bước xây dựng chương trình

Bài toán: Tìm nghiệm phương trình ax + b = 0
Input: a, b (số thực)
Output: Số nghiệm, giá trị nghiệm

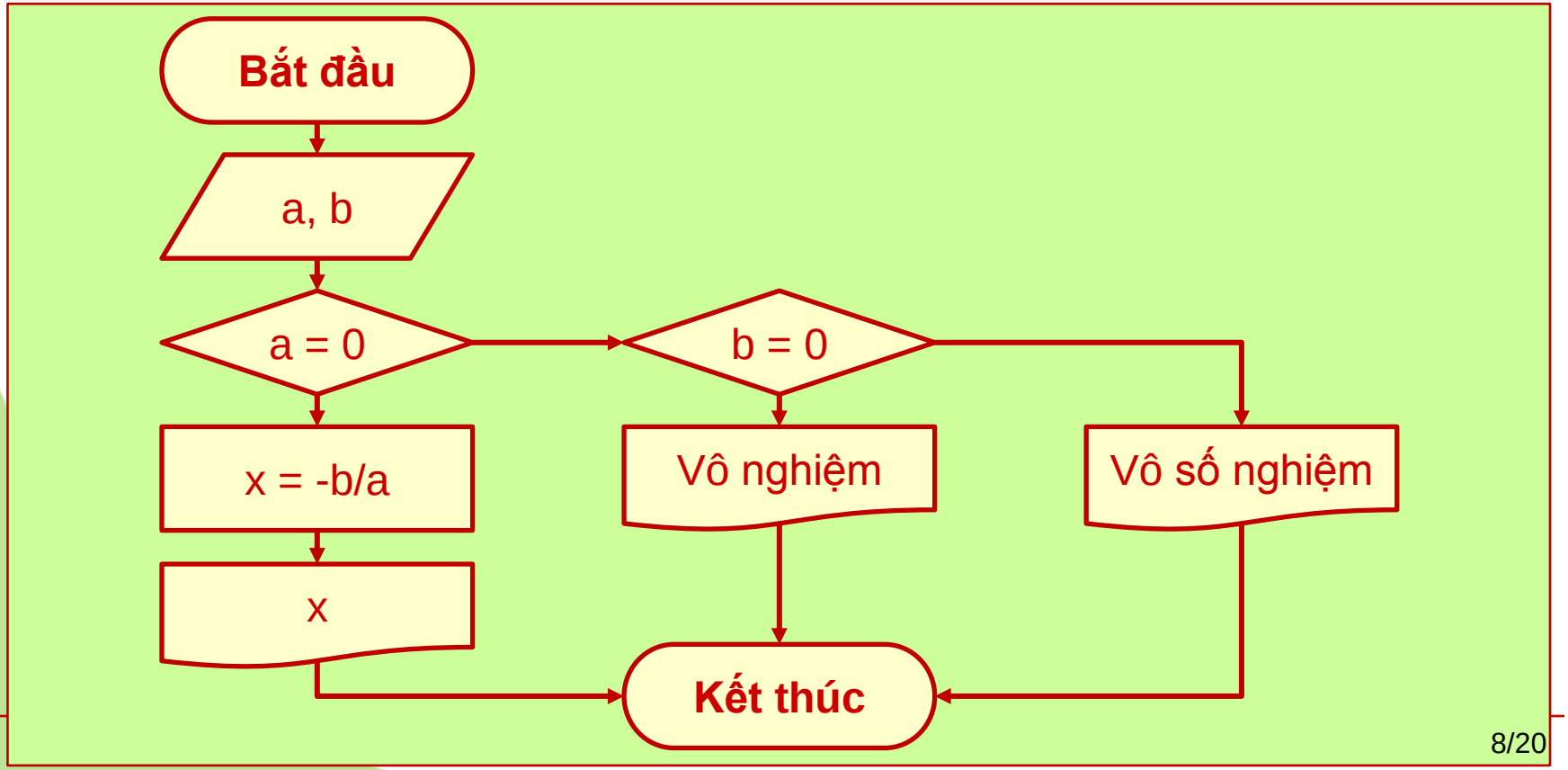
Bắt đầu
    Nhập a, b.
    Nếu a = 0, **thì**
        Nếu b = 0, **thì** Xuất "Vô số nghiệm"
        Ngược lại, **thì** Xuất "Vô nghiệm"
    Ngược lại, **thì**
        Tính nghiệm x = -b/a, xuất x
Kết thúc

2. Các bước xây dựng chương trình

Bài toán: Tìm nghiệm phương trình ax + b = 0

Input: a, b (số thực)

Output: Số nghiệm, giá trị nghiệm



2. Các bước xây dựng chương trình

Bài toán: Tìm nghiệm phương trình ax + b = 0

Input: a, b (số thực)

Output: Số nghiệm, giá trị nghiệm

```
Float a, b, x;
{
    Nhập (a, b);
    If  (a == 0)
            If (b == 0)  xuất( “Vô số nghiệm”);
            else xuất(“Vô nghiệm”);
    else
            {
                    x = -b/a;
                    xuất (“Một nghiệm x = “, x);
            }
}
```

3. Tổng quan về ngôn ngữ lập trình

- Ngôn ngữ lập trình
  - Hệ thống các ký hiệu được dùng để mô tả các tính toán mà cả con người và máy tính đều có thể đọc và hiểu được
- Yêu cầu đối với ngôn ngữ lập trình
  - Dễ hiểu và dễ sử dụng ài
  - Mô tả đầy đủ và rõ ràng các tiến trìn y

3. Tổng quan về ngôn ngữ lập trình

Thế hệ 1	• Ngôn ngữ máy (Machine language)
Thế hệ 2	• Ngôn ngữ cấp thấp (Hợp ngữ - Assembly)
Thế hệ 3	• Ngôn ngữ cấp cao (Pascal, Java, C/C++/C#, …)
Thế hệ 4	• Ngôn ngữ hệ quản trị cơ sở dữ liệu
Thế hệ 5	• Ngôn ngữ trí tuệ nhân tạo

3. Tổng quan về ngôn ngữ lập trình

- Lập trình tuần tự (Assembly)
- Lập trình hướng cấu trúc (PASCAL, C, …)
- Lập trình hướng đối tượng (Java, C#, …)
- Lập trình hàm (R, Matlab, Mathemayica, …)
- Lập trình logic (PROLOG)

C++ ngôn ngữ lai, cho phép lập trình cả theo hướng cấu trúc và hướng đối tượng

4. Giới thiệu ngôn ngữ lập trình C

- Giới thiệu
  - Ngôn ngữ C do Dennis Ritchie sáng chế tại Bell Telephone (AT&T) năm 1972 nhằm mục đích viết hệ điều hành Unix
  - Tiền thân của ngôn ngữ B, KenThompson, cũng tại Bell Telephone.
  - C được viện chuẩn hoá Mỹ (ANSI: American National Standard Institute) làm thành tiêu chuẩn với tên gọi ANSI C năm 1983.
  - Là ngôn ngữ lập trình có cấu trúc và phân biệt chữ HOA - thường (case sensitive)

4. Giới thiệu ngôn ngữ lập trình C

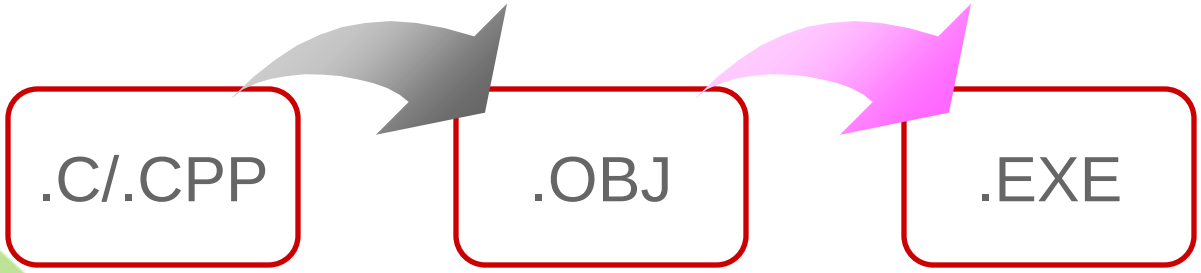
- Ưu điểm của C
  - Rất mạnh và mềm dẻo, có khả năng thể hiện bất cứ ý tưởng nào, dùng viết hệ điều hành, các trình điều khiển, soạn thảo văn bản,…, chương trình dịch
  - Được sử dụng rộng rãi bởi các nhà lập trình chuyên nghiệp. Chương trình viết bởi C rất hiệu quả (có thể đạt 80% tính năng của chương trình đó viết bằng mã máy)
  - Có tính khả chuyển, dễ thích nghi, ít thay đổi trên các hệ thống máy tính khác nhau.
  - C có ít từ khoá.
  - C có cấu trúc modul, sử dụng chương trình con loại hàm, có thể sử dụng nhiều lần

4. Giới thiệu ngôn ngữ lập trình C

- Nhược điểm của C
  - Cú pháp lạ và khó học
  - Một số kí hiệu của C có nhiều nghĩa khác nhau (ví dụ kí hiệu * là toán tử nhân, toán tử không định hướng, thay thế…)
  - C quá mềm dẻo (truy nhập tự do vào dữ liệu, trộn lẫn toán tử…)
- C là ngôn ngữ bậc trung (medium-level language)
  - C kết hợp được các tính năng ngôn ngữ bậc cao với ngôn ngữ bậc thấp
  - C mạnh về xử lí bit, địa chỉ ô nhớ → thích hợp lập trình hệ thống

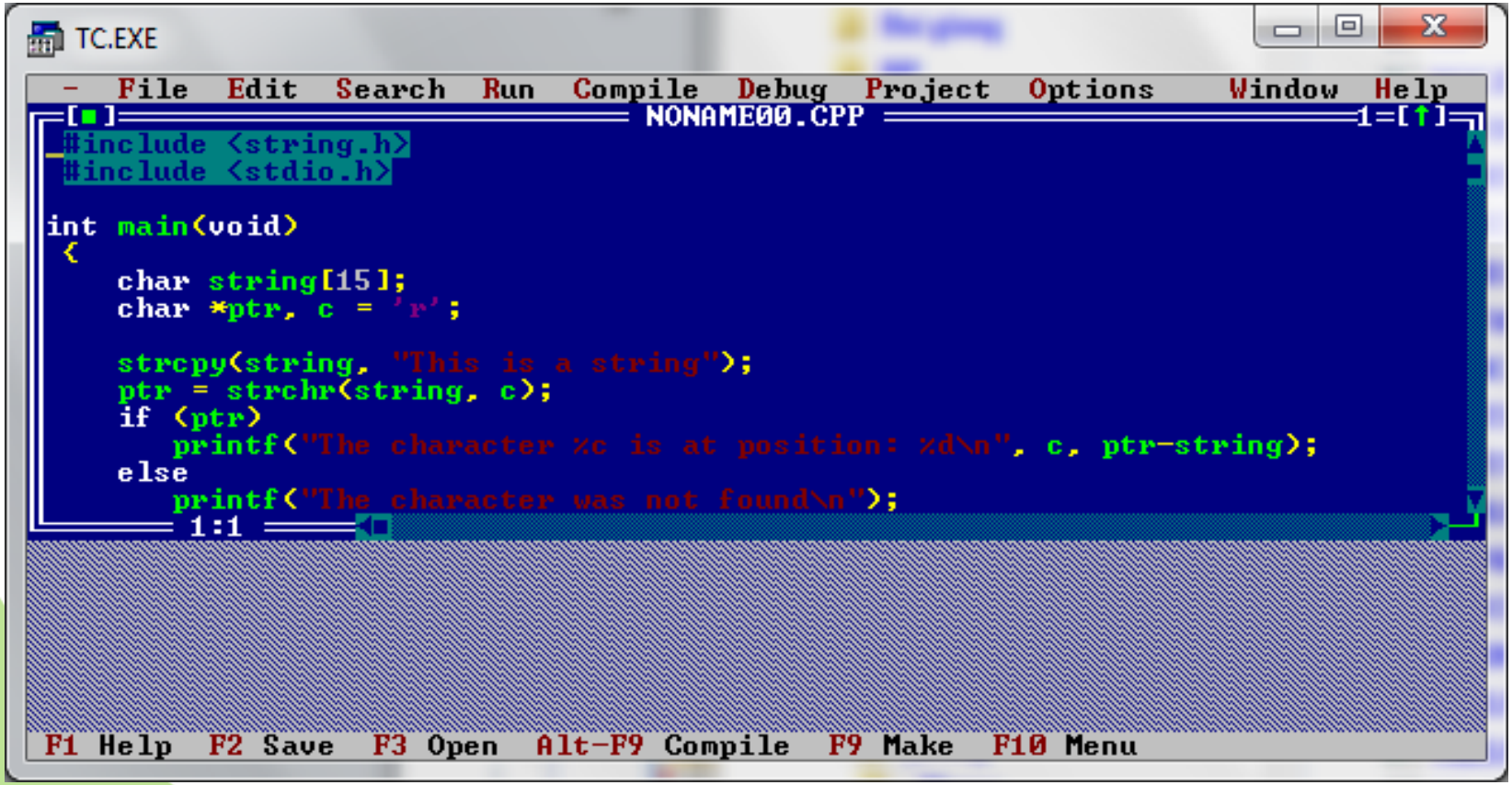
4. Giới thiệu ngôn ngữ lập trình C

- Môi trường phát triển tích hợp IDE (Integrated Development Environment)
  - Biên tập chương trình nguồn (Trình EDIT).
  - Biên dịch chương trình (Trình COMPILE).
  - Chạy chương trình nguồn (Trình RUNTIME).
  - Sửa lỗi chương trình nguồn (Trình DEBUG).



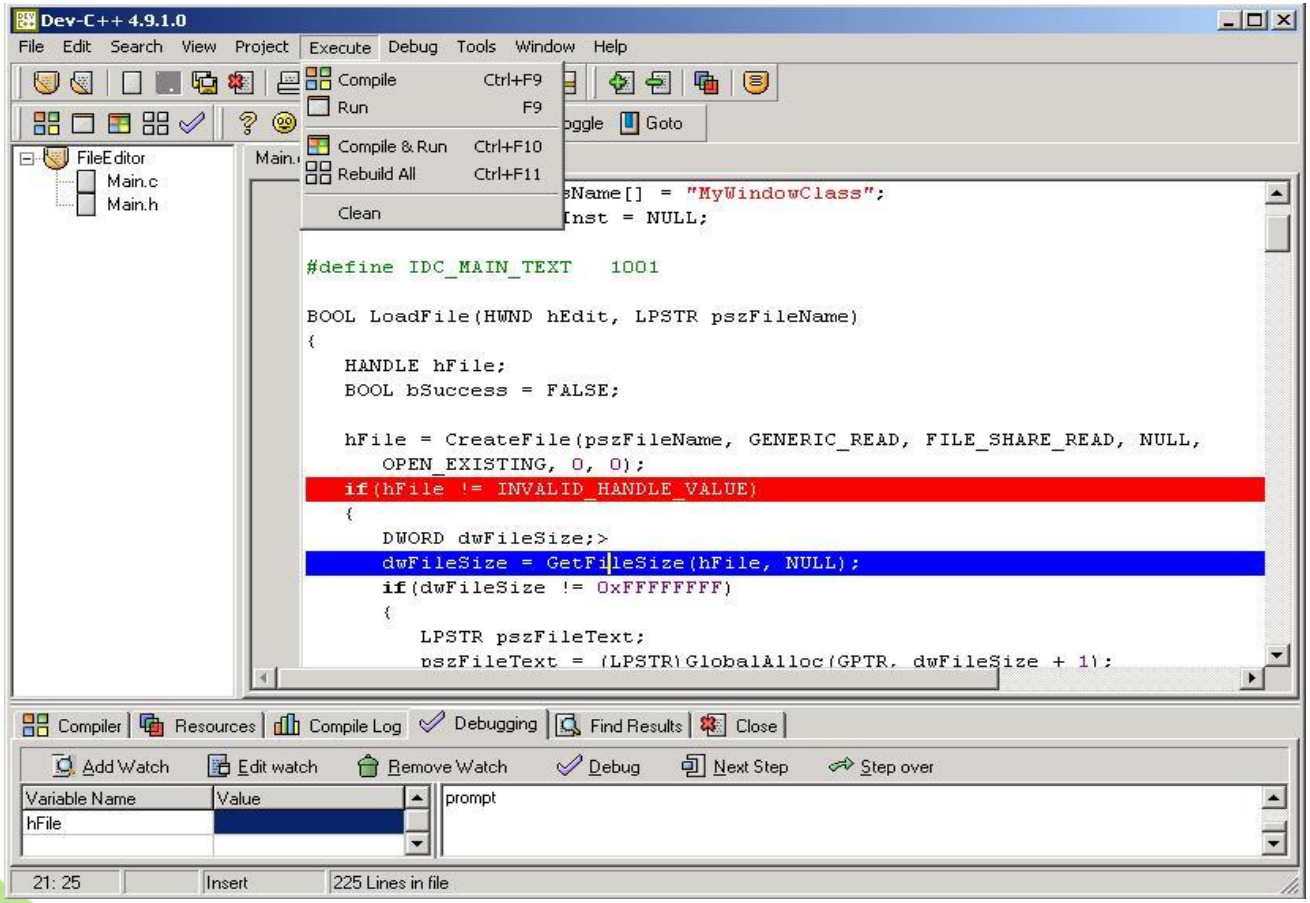
4. Giới thiệu ngôn ngữ lập trình C

- Turbo C++ 3 for DOS.
  - Thực thi file TC\BIN\TC.EXE



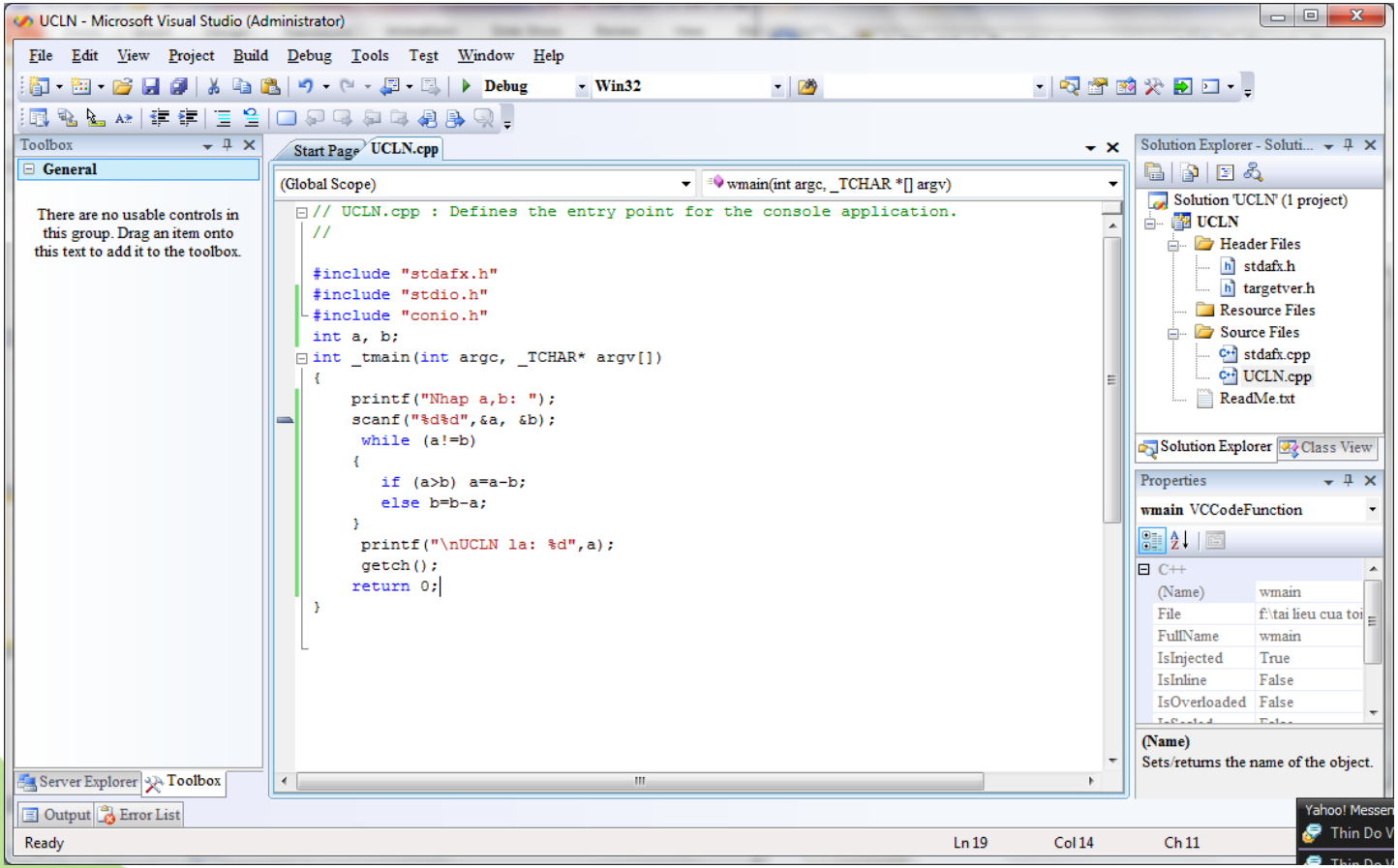
4. Giới thiệu ngôn ngữ lập trình C

- Dev-C
  - ***Dev-C++ 5.0*** (http://www.bloodshed.net/dev/devcpp.html)



4. Giới thiệu ngôn ngữ lập trình C

- Visual Studio
  - VS 6.0, VS2003, VS2005, VS2008, VS2010…



Tóm tắt

- Các khái niệm cơ bản
  - Vấn đề & bài toán
  - Thuật toán & chương trình
- Các bước xây dựng chương trình
  - Ngôn ngữ tự nhiên
  - Lưu đồ (sơ đồ khối)
  - Mã giả
- Tổng quan ngôn ngữ lập trình
  - Lịch sử phát triển
  - Phương pháp tiếp cận
- Giới thiệu ngôn ngữ lập trình C
  - Lịch sử phát triển
  - Ưu, nhược điểm
  - Môi trường phát triển tích hợp

CƠ SỞ LẬP TRÌNH

CÁC PHẦN TỬ CƠ BẢN CỦA NGÔN NGỮ C

Nội dung

- Các thành phần cơ bản
- Cấu trúc chương trình C
- Các kiểu dữ liệu cơ sở
- Câu lệnh - biểu thức
- Thứ tự ưu tiên các phép toán
- Vào - ra dữ liệu trong C

1. Các thành phần cơ bản

- Bộ từ vựng của C
  - Các chữ cái hoa: A, B, C, …, Z
  - Các chữ cái thường: a, b, c, …, z
  - Các chữ số : 0, 1, 2, 4, 5, 6, 7, 8, 9
  - Các ký hiệu toán học : + – * / = < > ( )
  - Các ký tự đặc biệt : . , : ; [ ] % \ # $ ‘ ^ & @
  - Ký tự gạch nối _ và khoảng trắng ‘ ’, dấu tab, xuống dòng

1. Các thành phần cơ bản (tt)

- Từ khóa (keyword)
  - Các từ dành riêng trong ngôn ngữ, mỗi từ có tác dụng và ý nghĩa cụ thể
  - Không thể sử dụng từ khóa để đặt tên cho biến, hàm, tên chương trình con.
  - Một số từ khóa thông dụng:
    - const, enum, signed, struct, typedef, unsigned…
    - char, double, float, int, long, short, void
    - case, default, else, if, switch
    - do, for, while
    - break, continue, goto, return

1. Các thành phần cơ bản (tt)

- Tên/Định danh (Identificater)
  - Tên là dãy kí tự liền nhau gồm các chữ cái a..z, A..Z, các chữ số 0..9, và dấu gạch nối.
  - Mọi tên đều phải khai báo trước khi sử dụng
  - Tên trong C phân biệt chữ HOA, thường
  - Độ dài tối đa mặc định là 32 kí tự
- Quy tắc đặt tên
  - Tên không được trùng với các từ khoá
  - Không được bắt đầu bằng chữ số
  - Không chứa kí tự đặc biệt như dấu cách, dấu chấm
  - Tên phải gợi nhớ về đối tượng được đặt tên
  - Cùng phạm vi không được đặt 2 tên trùng nhau

1. Các thành phần cơ bản (tt)

- Ví dụ Tên/Định danh (Identifier)
  - Các tên hợp lệ: GiaiPhuongTrinh, Bai_Tap1, PI
  - Các tên không hợp lệ:
    - 1A bắt đầu bằng chữ số
    - PI$ chứa kí hiệu $
    - Giai phuong trinh chứa dấu cách
    - char trùng từ khoá char
  - Phân biệt chữ hoa chữ thường, do đó các tên sau đây khác nhau:
    - A, a
    - BaiTap, baitap, BAITAP, bAItaP, …
- Thường dùng chữ HOA đặt tên cho hằng, chữ thường cho các đối tượng khác.

1. Các thành phần cơ bản (tt)

- Dấu chấm phẩy ;
  - Dùng để phân cách các câu lệnh.
  - Ví dụ: printf(“Hello World!”); printf(“\n”);
- Câu chú thích
  - Đặt giữa cặp dấu /* */ hoặc // (C++)
  - Ví dụ: /*Ho & Ten: NVA*/, // MSSV: 0712078
- Hằng ký tự và hằng chuỗi
  - Hằng ký tự: ‘A’, ‘a’, …
  - Hằng chuỗi: “Hello World!”, “Nguyen Van A”
  - Chú ý: ‘A’ khác “A”

2. Cấu trúc chung chương trình C

```
#include <…>      /*Gọi các tệp tiền xử lý */
#define /* Định nghĩa */
typedef /*Định nghĩa kiểu */
int x;            /* Khai báo biến ngoài */
const …           /*Khai báo hằng */
/*Khai báo các hàm, có thể có hoặc không */
Kiểu_dữ_liệu tên_hàm(các tham số);
{     Khai báo các biến, hằng
      Các lệnh của hàm
      return(); /*Trả lại giá trị */
} ...
main()      /* Bắt buộc phải có hàm main */
{     Khai báo các biến, hằng
      Các lệnh của hàm
      return (); /*Có thể có hoặc không */
}
```

Ví dụ chương trình C

- Ví dụ 1: Viết ra màn hình dòng chữ

  CHAO MUNG DEN VOI NGON NGU C

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()
{
        system("cls"); /*Xoá màn hình */
        printf("CHAO MUNG DEN VOI NGON NGU C");
        getch(); /*Dừng màn hình */
        return 0;
}
```

Ví dụ chương trình C

- Ví dụ 2: Tính chu vi và diện tích hình tròn với bán kính r nhập từ bàn phím.

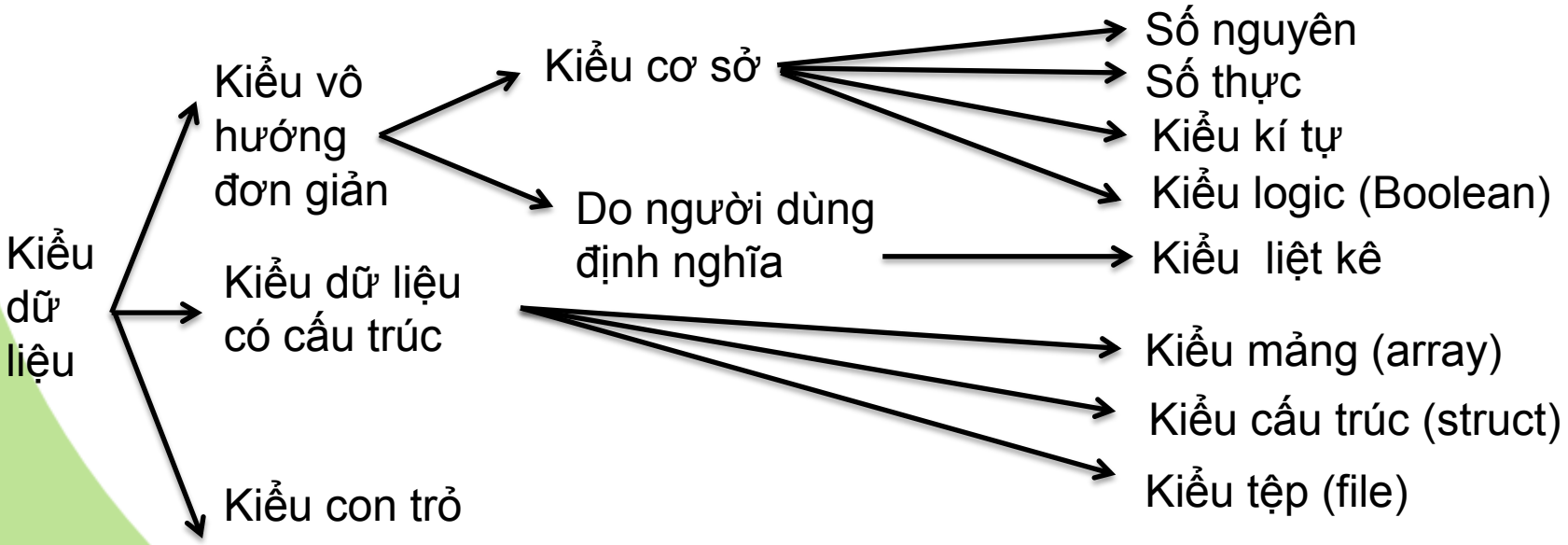
```
#include <stdio.h> /*Thư viện vào ra chuẩn */
#include <conio.h>
#include <stdlib.h>
#include <math.h> /*Thư viện hàm toán học*/
int main()
{       float r,cv,dt; /*Khai báo biến*/
        system("cls"); /*Xoá màn hình */
        printf("Nhap ban kinh: "); scanf("%f",&r);
        cv=2*M_PI*r; dt=M_PI*r*r; /*Tính chu vi, diện tích*/
        printf("Chu vi: %0.2f",cv); printf("Dien tich: %0.2f",dt);
        getch(); /*Dừng màn hình */
        return 0;
}
```

Một số quy tắc khi viết chương trình

- Mỗi câu lệnh có thể viết trên một hay nhiều dòng, nhưng phải kết thúc bằng dấu ;
- Để báo cho C biết một chuỗi kí tự vẫn còn ở dòng dưới, thêm dấu \ trước khi xuống dòng
  - Ví dụ:
```
printf("CHAO MUNG \
        DEN VOI NGON NGU C");
```
- Lời chú thích có thể viết trên 1 hoặc nhiều dòng, đặt giữa cặp dấu /*...*/
- Các lệnh theo cùng nhóm phải thẳng hàng theo chiều dọc

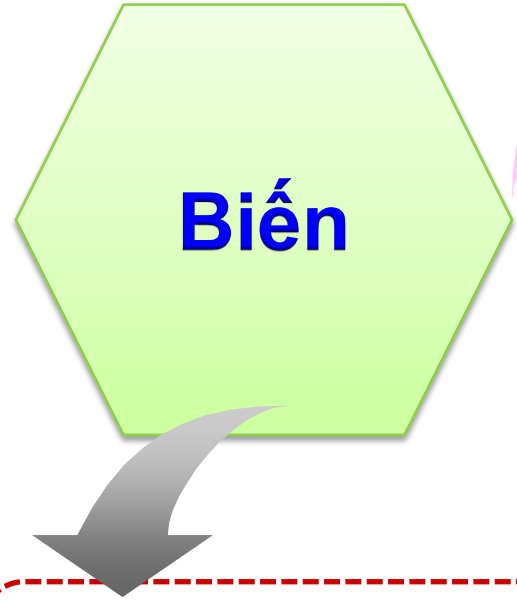
3. Các kiểu dữ liệu cơ sở

- Kiểu dữ liệu (data type) là:
  - Một tập hợp các giá trị mà một biến thuộc kiểu đó có thể nhận được,
  - Trên đó xác định một số phép toán



Biến

- Là đại lượng có thể thay đổi được giá trị



Ví dụ:

```
int i;
int j, k;
unsigned char dem;
float ketqua, delta;
```

Cú pháp

<kiểu dữ liệu> <danh sách các biến>;

- Trong C, giá trị `i` được chứa trong ô nhớ có địa chỉ `&i`

Hằng

- Là đại lượng có giá trị không đổi

Hằng thường

Ví dụ

```
const int A = 1506;
const int B = 01506;
const int C = 0x1506;
const float D = 15.06e-3;
const char RC='\r'
```

Hằng tượng trưng

Ví dụ

```
#define MAX 100
#define PI 3.14
#define TRUE 1
#define FALSE 0
```

Các kiểu dữ liệu cơ sở (tự đọc)

- C có 4 kiểu cơ sở
  - Kiểu số nguyên: giá trị của nó là các số nguyên như 2912, -1706, …
  - Kiểu số thực: giá trị của nó là các số thực như 3.1415, 29.12, -17.06, …
  - Kiểu ký tự: 256 ký tự trong bảng mã ASCII.
  - Kiểu boolean: giá trị đúng hoặc sai.

Kiểu số nguyên

- Các kiểu số nguyên (có dấu)
  - n bit có dấu: $-2^{n-1} \ldots +2^{n-1} - 1$

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
char	1	–128 … +127
int	2	–32.768 … +32.767
short	2	–32.768 … +32.767
long	4	–2.147.483.648 … +2.147.483.647

Kiểu số nguyên (tt)

- Các kiểu số nguyên (không dấu)
  - n bit không dấu: 0 … $2^n - 1$

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
unsigned char	1	0 … 255
unsigned int	2	0 … 65.535
unsigned short	2	0 … 65.535
unsigned long	4	0 … 4.294.967.295

Kiểu số nguyên (tt)

- Các phép tính số học với số nguyên

Phép toán	Kí hiệu	Ví dụ	Ví dụ bằng số
Cộng	+	x+y	
Trừ	-	x-y	
Nhân	*	x*y	
Chia lấy phần nguyên	/	x/y	3/2=1 chứ không phải là 1.5
Chia lấy số dư	%	x%y	5%3 = 2

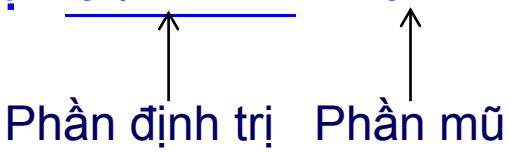
- Chú ý:
  - Chia 2 số nguyên là số nguyên, muốn là số thực phải viết `(float)x/y`
  - Thận trọng tránh hiện tượng tràn số

Kiểu số nguyên (tt)

- Biểu diễn số nguyên dạng hệ đếm 16 (Hexa)
  - Bắt đầu bằng kí tự `0x` hoặc `0X`
  - Ví dụ: `65` được viết là `0x41` hoặc `0X41`
    `15` được viết là `0xF` hoặc `0XF`
- Biểu diễn số nguyên dạng hệ đếm 8 (Octa)
  - Bắt đầu bằng kí tự `0`
  - Ví dụ: `65` được viết là `0101`
    `15` được viết là `017`
- Hằng số nguyên định trước kiểu
  - Thêm một kí tự cuối vào số: **L** (long), **U** (unsigned integer, **UL** (unsigned long)
  - Ví dụ: `50000U, 012345L, 0x50000U`

Kiểu số thực

- Dạng viết bình thường
  - Ví dụ: `3.14  3.0  -24.12345  -0.453`
- Dạng viết khoa học
  - Gồm: phần định trị và phần mũ viết sau chữ **E** (hoặc **e**), giữa chúng không có khoảng cách
  - Ví dụ: `6.2144E+02`



  - Chú ý:
    - Nếu không có phần mũ thì phần định trị bắt buộc phải có dấu **.**
    - Có thể không cần số 0 ở đầu (vd: **.1212**)
    - **KHÔNG** tồn tại phép % cho số thực

Kiểu số thực

Các kiểu số thực

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
float	4	1.24E-38 … 3.4E38 Độ chính xác khoảng 7 chữ số
double	8	2.2E-308 … 1.8E308 Độ chính xác khoảng 15 chữ số
long double	10	3.4E4932…3.4E4932 Độ chính xác khoảng 19 chữ số

Hằng số thực định trước kiểu

- Thêm kí tự cuối: F (float), L (long double)
- Ví dụ: `0.1234567-20L` `4E12F`

Kiểu ký tự

- Đặc điểm
  - Tên kiểu: char
  - Miền giá trị: 256 ký tự trong bảng mã ASCII.
  - Chính là kiểu số nguyên do:
    - Không lưu trực tiếp ký tự mà chỉ lưu mã ASCII của ký tự đó.
- Ví dụ
  - Lưu số 65 tương đương với ký tự ‘A’
  - Lưu số 97 tương đương với ký tự ‘a’
- Hằng kí tự
  - Đặt giữa hai dấu phẩy trên
  - Ví dụ: `'a'  'A'  'z'`

Kiểu ký tự (tt)

- Biểu diễn một kí tự trong bảng mã ASCII
  - \xHHH (HHH là giá trị số Hexa của kí tự)
  - \DDD (DDD là giá trị số Octa của kí tự)
  - Ví dụ: 'A' được viết dưới dạng `\x41` hoặc `\101`

Kí tự	Dãy mã	Giá trị trong bảng ASCII
Xoá trái	\b	x08
Nhảy cách ngang	\t	x09
Xuống dòng	\n	x0A
Dấu "	\"	x22
Dấu '	\'	x27
Dấu \	\\	x5C
Mã null	\0	x00

Kiểu ký tự (tt)

- Các hàm xử lí kí tự
  - `toASCII(c)`: chuyển c thành giá trị mã ASCII
  - `tolower(c)`: chuyển thành chữ thường
  - `toupper(c)`: chuyển thành chữ hoa
- Hằng xâu kí tự
  - Hằng xâu kí tự được viết trong cặp nháy kép “”
  - Xâu kí tự được lưu trữ trong một mảng ô nhớ liền nhau và có ô cuối cùng chứa mã số 0 (null)
    - Ví dụ: Xâu “`Viet nam`” được lưu là:

V	i	e	t		n	a	m	\0

  - Hằng xâu kí tự không được viết trong biểu thức số học

Kiểu Boolean

- Đặc điểm
  - C ngầm định một cách không tường minh:
    - false (sai): giá trị 0.
    - true (đúng): giá trị khác 0, thường là 1.
  - C++: bool
- Ví dụ
  - 0 (false), 1 (true), 2 (true), 2.5 (true)
  - 1 > 2 (0, false), 1 < 2 (1, true)

4. Câu lệnh – Biểu thức

- Biểu thức
  - Tạo thành từ các toán tử (Operator) và các toán hạng (Operand).
  - Toán tử tác động lên các giá trị của toán hạng và cho giá trị có kiểu nhất định.
  - Toán tử: +, –, *, /, %….
  - Toán hạng: hằng, biến, lời gọi hàm...
- Ví dụ
  - 2 + 3, a / 5, (a + b) * 5, …

Câu lệnh

- Khái niệm
  - Là một chỉ thị trực tiếp, hoàn chỉnh nhằm ra lệnh cho máy tính thực hiện một số tác vụ nhất định nào đó.
  - Các câu lệnh cách nhau bằng dấu chấm phẩy ;
  - Trình biên dịch bỏ qua các khoảng trắng (hay tab hoặc xuống dòng) chen giữa lệnh.
- Ví dụ

```
a=2912;
a = 2912;
a
=
2912;
```

Câu lệnh (tt)

- Phân loại
  - Câu lệnh đơn: chỉ gồm một câu lệnh.
  - Câu lệnh phức (khối lệnh): gồm nhiều câu lệnh đơn được bao bởi { và }
- Ví dụ

```
a = 2912;                    // Câu lệnh đơn

{                            // Câu lệnh phức/khối lệnh
      a = 2912;
      b = 1706;
}
```

Phép gán

- Phép gán giá trị đơn giản

  <Tên biến> = <Biểu thức>

  Ví dụ: `i=3;`

  `i=i+4;`

  - Giá trị của biểu thức bên phải dấu gán = được đặt vào ô nhớ của biến nằm bên trái dấu gán.
  - Vế trái chỉ có thể là tên của một biến hoặc là giá trị dạng một địa chỉ ô nhớ.

- Phép gán kép

  `a=b=c=3;` Gán giá trị 3 cho cả 3 biến a,b,c

  `a=b+(c=3);` Gán 3 cho c, sau đó cộng với b và gán kết quả nhận được cho a

Sự hiệu chỉnh dữ liệu khi tính toán

- Máy tự đông chuyển kiểu đơn giản lên kiểu cao hơn để quy đổi kiểu kết quả, theo thứ tự:

  int → long → float → double → long double

- Chuyển đổi cho kiểu kí tự char
  - Chuyển đổi qua lại giữa char và int
  - Ví dụ: 'A' + 1 = 66
- Cố ý chuyển đổi kiểu giá trị (typecast)
  - Cú pháp: `kiểu(biến)` hoặc `(kiểu)biến`
  - Ví dụ: f = float(1) / 2; g = float(1 / 2);

5. Thứ tự ưu tiên các phép toán

Toán tử toán học

- Toán tử 1 ngôi
  - Chỉ có một toán hạng trong biểu thức.
  - ++ (tăng 1 đơn vị), -- (giảm 1 đơn vị)
  - Đặt trước toán hạng
    - Ví dụ ++x hay --x: thực hiện tăng/giảm trước.
  - Đặt sau toán hạng
    - Ví dụ x++ hay x--: thực hiện tăng/giảm sau.
- Ví dụ
  - x = 10; y = x++; // y = 10 và x = 11
  - x = 10; y = ++x; // x = 11 và y = 11

Các toán tử toán học

- Toán tử 2 ngôi
  - Có hai toán hạng trong biểu thức.
  - +, –, *, /, % (chia lấy phần dư)
- Ví dụ
  - a = 1 + 2; b = 1 – 2; c = 1 * 2; d = 1 / 2;
  - e = 1*1.0 / 2;
  - h = 1 % 2;
  - x = x * (2 + 3*5); ⇔ x *= 2 + 3*5;

Phép gán mở rộng

- C cho phép viết lệnh gán mở rộng theo quy định:

```
x+=y  ⇔  x= x+y
x-=y  ⇔  x= x-y
x*=y  ⇔  x= x*y
x/=y  ⇔  x= x/y
x%=y  ⇔  x= x%y
x>>=y ⇔  x= x>>y
x<<=y ⇔  x= x<<y
x&=y  ⇔  x= x&y
x|=y  ⇔  x= x|y
x^=y  ⇔  x= x^y
```

Các toán tử trên bit

Các toán tử trên bit

- Tác động lên các bit của toán hạng (nguyên).
- & (and), | (or), ^ (xor), ~ (not hay lấy số bù 1)
- >> (shift right), << (shift left)
- Toán tử gộp: &=, |=, ^=, ~=, >>=, <<=

&	0	1
0	0	0
1	0	**1**

|	0	1
0	0	**1**
1	**1**	**1**

^	0	1
0	0	**1**
1	**1**	0

~	0	1
	1	0

>>	**a>>n = a/2^n**
<<	**a<<n = a*2^n**

Các toán tử trên bit

- Ví dụ

```
void main()
{
        int a = 5;  // 0000 0000 0000 0101
        int b = 6;  // 0000 0000 0000 0110

        int z1, z2, z3, z4, z5, z6;
        z1 = a & b; // 0000 0000 0000 0100
        z2 = a | b; // 0000 0000 0000 0111
        z3 = a ^ b; // 0000 0000 0000 0011
        z4 = ~a;    // 1111 1111 1111 1010
        z5 = a >> 2;// 0000 0000 0000 0001
        z6 = a << 2;// 0000 0000 0001 0100
}
```

Các toán tử quan hệ

- Các toán tử quan hệ
  - So sánh 2 biểu thức với nhau
  - Cho ra kết quả 0 (hay false nếu sai) hoặc 1 (hay true nếu đúng)
  - ==, >, <, >=, <, <=, !=
- Ví dụ
  - s1 = (1 == 2);s2 = (1 != 2);
  - s3 = (1 > 2); s4 = (1 >= 2);
  - s5 = (1 < 2); s6 = (1 <= 2);

Các toán tử logic

Các toán tử logic

- Tổ hợp nhiều biểu thức quan hệ với nhau.
- && (and), || (or), ! (not)

&&	0	1
0	0	0
1	0	**1**

||	0	1
0	0	**1**
1	**1**	**1**

- Ví dụ
  - s1 = (1 > 2) && (3 > 4);
  - s2 = (1 > 2) || (3 > 4);
  - s3 = !(1 > 2);

Toán tử điều kiện

- Toán tử điều kiện
  - Đây là toán tử 3 ngôi (gồm có 3 toán hạng)
  - <biểu thức 1> ? <biểu thức 2> : <biểu thức 3>
    - <biểu thức 1> đúng thì giá trị là <biểu thức 2>.
    - <biểu thức 1> sai thì giá trị là <biểu thức 3>.
- Ví dụ
  - s1 = (1 > 2) ? 2912 : 1706;
  - int s2 = 0;
  - 1 < 2 ? s2 = 2912 : s2 = 1706;

Toán tử phẩy

- Toán tử phẩy
  - Các biểu thức đặt cách nhau bằng dấu ,
  - Các biểu thức con lần lượt được tính từ trái sang phải.
  - Biểu thức mới nhận được là giá trị của biểu thức bên phải cùng.
- Ví dụ
  - x = (a++, b = b + 2);
  - ⇔ a++; b = b + 2; x = b;

Độ ưu tiên của các toán tử

Toán tử	Độ ưu tiên
() [] ->	→
! ++ -- - ~ sizeof() (toán tử 1 ngôi)	←
* / %	→
+ -	→
<< >>	→
< <= > >=	→
== !=	→
&	→
|	→
^	→
&&	→
||	→
?:	←
= += -= *= /= %= &= ^= |= <<= >>=	←

Độ ưu tiên của các toán tử

- Quy tắc thực hiện
  - Thực hiện biểu thức trong ( ) sâu nhất trước.
  - Thực hiện theo thứ tự ưu tiên các toán tử.

  => Tự chủ động thêm ( )
- Ví dụ
  - n = 2 + 3 * 5;

    => n = 2 + (3 * 5);
  - a > 1 && b < 2

    => (a > 1) && (b < 2)

Viết biểu thức cho các mệnh đề

- x lớn hơn hay bằng 3

  x >= 3

- a và b cùng dấu

  ((a>0) && (b>0)) || ((a<0) && (b<0))

  (a>0 && b>0) || (a<0 && b<0)

- p bằng q bằng r

  (p == q) && (q == r) hoặc (p == q && q == r)

- –5 < x < 5

  (x > –5) && (x < 5) hoặc (x > –5 && x < 5)

6. Vào – Ra dữ liệu trong C

Xuất dữ liệu

- Thư viện
  - #include <stdio.h> (standard input/output)
- Cú pháp
  - printf(“dãy mã quy cách”, dãy các biểu thức)
  - dãy mã quy cách là dãy các định dạng được đặt trong cặp nháy kép “ ”.
    - Văn bản thường (literal text)
    - Ký tự điều khiển (escape sequence)
    - Đặc tả (conversion specifier)

Mã quy cách

- Văn bản thường (literal text)
  - Được xuất y hệt như lúc gõ trong chuỗi định dạng.
- Ví dụ
  - Xuất chuỗi Hello World
    - ➔ printf(“Hello ”); printf(“World”);
    - ➔ printf(“Hello World”);
  - Xuất chuỗi a + b
    - ➔ printf(“a + b”);

Mã quy cách (tt)

- Ký tự điều khiển (escape sequence)
  - Gồm dấu \ và một ký tự như trong bảng sau:

Ký tự điều khiển	Ý nghĩa
\a	Tiếng chuông
\b	Lùi lại một bước
\n	Xuống dòng
\t	Dấu tab
\\	In dấu \
\?	In dấu ?
\”	In dấu “

- Ví dụ
  - printf(“\t”); printf(“\n”);
  - printf(“\t\n”);

Mã quy cách (tt)

- Đặc tả (conversion specifier)
  - Gồm dấu % và một ký tự.
  - Xác định kiểu của biến/giá trị muốn xuất.
  - Các đối số chính là các biến/giá trị muốn xuất, được liệt kê theo thứ tự cách nhau dấu phẩy.

Đặc tả	Ý nghĩa	
%c	Ký tự	char
%d, %ld	Số nguyên có dấu	char, int, short, long
%f, %lf	Số thực	float, double
%s	Chuỗi ký tự	char[], char*
%u %lu	Số nguyên không dấu	unsigned int/short/long
%x, %X	Số nguyên dạng Hexa	
%o	Số nguyên dạng Octa	
%e, %E	Số thực dạng mũ	

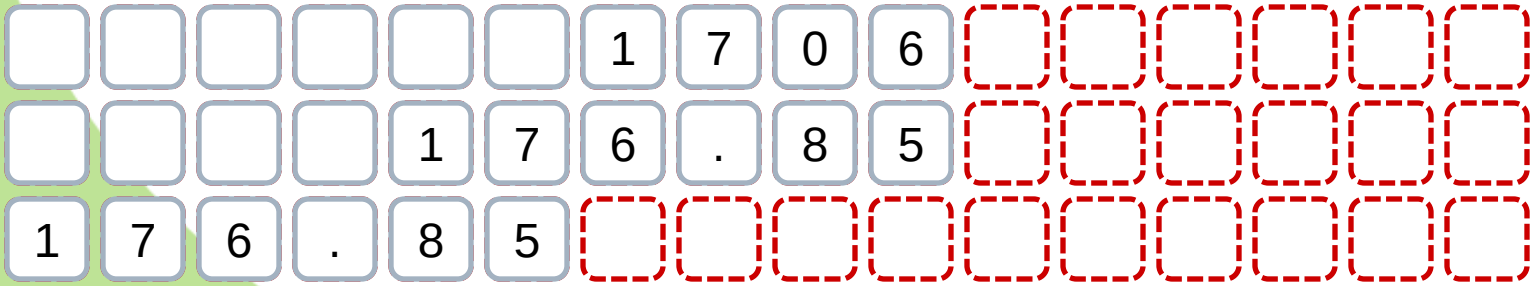
Mã quy cách (tt)

- Ví dụ
  - int a = 10, b = 20;
  - printf("%d", a); ➔ Xuất ra 10
  - printf("%d", b); ➔ Xuất ra 20
  - printf("%d %d", a, b); ➔ Xuất ra 10 20
  - float x = 15.06;
  - printf("%f", x); ➔ Xuất ra 15.060000
  - printf("%f", 1.0/3); ➔ Xuất ra 0.333333

Định dạng xuất

- Cú pháp
  - Định dạng xuất số nguyên: %nd
  - Định dạng xuất số thực: %n.kf

```
int a = 1706;
float x = 176.85;
printf("%10d", a);printf("\n");
printf("%10.2f", x);printf("\n");
printf("%.2f", x);printf("\n");
```



Mã quy cách (tt)

- Phối hợp các thành phần
  - int a = 1, b = 2;
  - Xuất 1 cong 2 bang 3 và xuống dòng.
    - printf("%d", a); // Xuất giá trị của biến a
    - printf(" cong "); // Xuất chuỗi " cong "
    - printf("%d", b); // Xuất giá trị của biến b
    - printf(" bang "); // Xuất chuỗi " bang "
    - printf("%d", a + b); // Xuất giá trị của a + b
    - printf("\n"); // Xuất điều khiển xuống dòng \n

    ➔ printf("%d cong %d bang %d\n", a, b, a+b);

Nhập dữ liệu

- Thư viện
  - #include <stdio.h> (standard input/output)
- Hàm scanf()
  - scanf("dãy mã quy cách", dãy các địa chỉ các biến);
  - Danh sách các biến: là tên các biến sẽ chứa giá trị nhập và có dấu & đặt trước

Mã quy cách	Ý nghĩa
%c	Ký tự char
%d	Số nguyên int
%u	Số nguyên unsigned int
%hd, %hu	Số nguyên short int/ unsigned int
%ld %lu	Số nguyên long int, unsiged long
%f, %e	Số thực
%lf hoặc %lu	Số thực double
%s	Xâu không chứa dấu cách, dùng với địa chỉ xâu

Nhập dữ liệu (tt)

- Nguyên tắc đọc
  - Số: máy nhảy qua các kí tự là các dấu chấm câu cho đến khi gặp kí tự là chữ số, đọc đến khi gặp kí tự không là chữ số.
  - Xâu kí tự: đọc đúng số kí tự mà ta yêu cầu
- Vai trò của dấu cách trong mã định dạng
  - Gặp dấu cách trong mã định dạng, máy nhảy qua các dấu cách để đọc số/kí tự
- Khuôn đọc
  - Đọc một số trong phạm vi m chữ số gõ vào
  - Ví dụ: `scanf("%3d%3d",&n,&p)`
- Xoá bộ nhớ đệm: `fflush(stdin);`

Nhập dữ liệu (tt)

- Hàm getchar()
  - Đọc một kí tự từ bàn phím
- Hàm getch()
  - Đọc kí tự từ bàn phím ngay khi gõ vào không đợi ấn phím Enter và không hiển thị ra màn hình.
  - Đọc thẳng từ bàn phím, không qua bộ nhớ đệm
  - Dùng để dừng chương trình xem kết quả
- Hàm getche()
  - Giống getch(), nhưng hiển thị kí tự lên màn hình
- Hàm gets()
  - Đọc một xâu kí tự cho đến khi gõ Enter

Bài tập lý thuyết

1. Trình bày các kiểu dữ liệu cơ sở trong C và cho ví dụ.
2. Trình bày khái niệm về biến và cách sử dụng lệnh gán.
3. Phân biệt hằng thường và hằng ký hiệu. Cho ví dụ minh họa.
4. Trình bày khái niệm về biểu thức.

   Tại sao nên sử dụng cặp ngoặc đơn.
5. Trình bày cách định dạng xuất.

Bài tập thực hành

6. Nhập năm sinh của một người và tính tuổi của người đó.
7. Nhập 2 số a và b. Tính tổng, hiệu, tính và thương của hai số đó.
8. Nhập tên sản phẩm, số lượng và đơn giá. Tính tiền và thuế giá trị gia tăng phải trả, biết:
   a. tiền = số lượng * đơn giá
   b. thuế giá trị gia tăng = 10% tiền

Bài tập thực hành

9. Nhập điểm thi và hệ số 3 môn Toán, Lý, Hóa của một sinh viên. Tính điểm trung bình của sinh viên đó.
10. Nhập bán kính của đường tròn. Tính chu vi và diện tích của hình tròn đó.
11. Nhập vào số xe (gồm 4 chữ số) của bạn. Cho biết số xe của bạn được mấy nước?

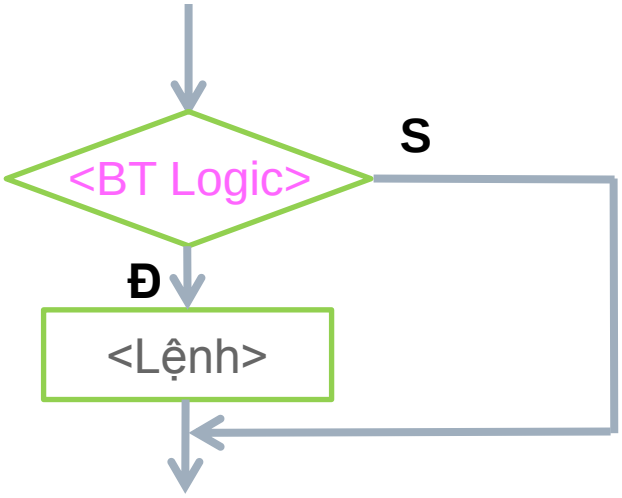
CƠ SỞ LẬP TRÌNH

CÁC CẤU TRÚC ĐIỀU KHIỂN TRONG NGÔN NGỮ C

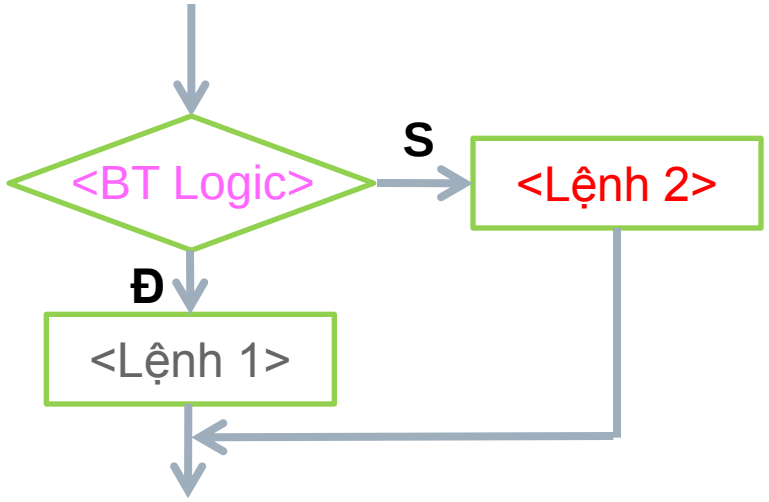
Nội dung

- Cấu trúc rẽ nhánh
  - Câu lệnh điều kiện if
  - Câu lệnh rẽ nhánh switch
  - Toán tử goto
- Cấu trúc lặp
  - Vòng lặp xác định for
  - Vòng lặp không xác định while
  - Vòng lặp không xác định do … while

Cấu trúc rẽ nhánh: Câu lệnh if … else




```
if (<BT Logic>)
        <Lệnh>;
```




```
if (<BT Logic>)
        <Lệnh 1>;
else
        <Lệnh 2>;
```

Cấu trúc rẽ nhánh: Câu lệnh if … else

```
void main()
{
        if (a == 0)
                printf("a bang 0");
        else
                printf("a khac 0");

        if (a == 0)
        {
                printf("a bang 0");
                a = 2912;
        }
        else
                printf("a khac 0");
}
```

Cấu trúc rẽ nhánh: Câu lệnh if … else

- Câu lệnh if và câu lệnh if… else là một câu lệnh đơn

```
{
        if (a == 0)
                printf("a bang 0");
}
{
        if (a == 0)
        {
                printf("a bang 0");
                a = 2912;
        }
        else
                printf("a khac 0");
}
```

Cấu trúc rẽ nhánh: Câu lệnh if … else

- Câu lệnh if có thể lồng vào nhau và else sẽ tương ứng với if gần nó nhất.

```
if (a != 0)
        if (b > 0)
                printf("a != 0 va b > 0");
        else
        printf("a != 0 va b <= 0");

if (a !=0)
{
        if (b > 0)
                printf("a != 0 va b > 0");
        else
                printf("a != 0 va b <= 0");
}
```

Cấu trúc rẽ nhánh: Câu lệnh if … else

- Nên dùng else để loại trừ trường hợp.

```
if (delta < 0)
        printf("PT vo nghiem");
if (delta == 0)
        printf("PT co nghiem kep");
if (delta > 0)
        printf("PT co 2 nghiem");

if (delta < 0)
        printf("PT vo nghiem");
else    // delta >= 0
        if (delta == 0)
                printf("PT co nghiem kep");
        else
                printf("PT co 2 nghiem");
```

Cấu trúc rẽ nhánh: Câu lệnh if … else

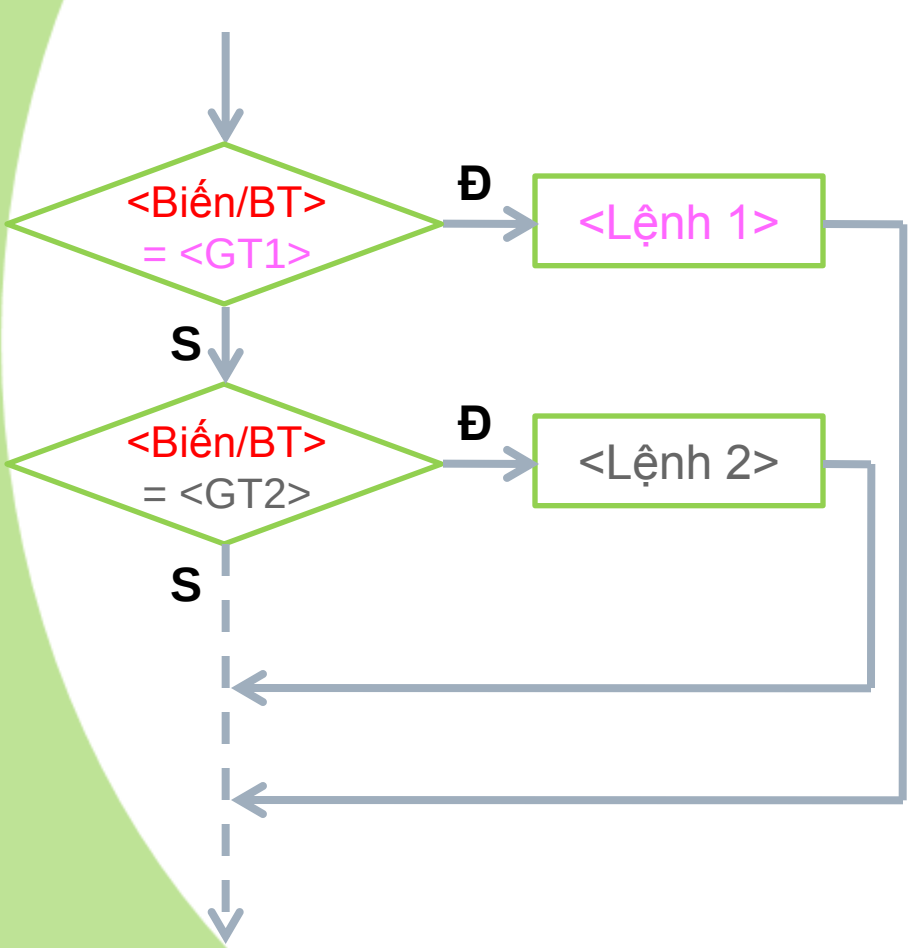
- Không được thêm ; sau điều kiện của if.

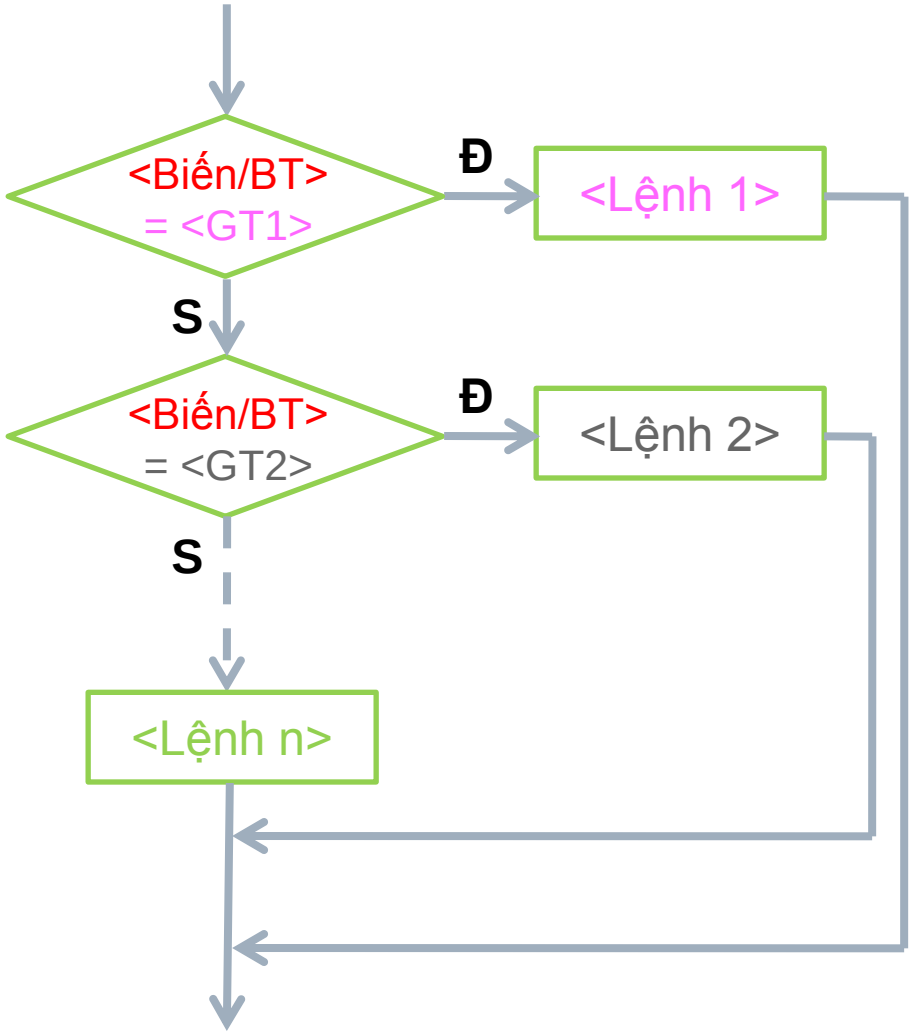
```
void main()
{
        int a = 0;
        if (a != 0)
                printf("a khac 0.");

        if (a != 0);
                printf("a khac 0.");

        if (a != 0)
        {
        };
        printf("a khac 0.");
}
```

Cấu trúc rẽ nhánh: Câu lệnh switch





Cấu trúc rẽ nhánh: Câu lệnh switch

```
void main()
{
        int a;
        printf("Nhap a: ");
        scanf("%d", &a);

        switch (a)
        {
                case 1 : printf("Mot"); break;
                case 2 : printf("Hai"); break;
                case 3 : printf("Ba"); break;
        }
}
```

Cấu trúc rẽ nhánh: Câu lệnh switch

```
void main()
{
        int a;
        printf("Nhap a: ");
        scanf("%d", &a);

        switch (a)
        {
                case 1 : printf("Mot"); break;
                case 2 : printf("Hai"); break;
                case 3 : printf("Ba"); break;
                default : printf("Ko biet doc");
        }
}
```

Cấu trúc rẽ nhánh: Câu lệnh switch

- Câu lệnh switch là một câu lệnh đơn và có thể lồng nhau.

```
{
        switch (a)
        {
                case 1 : printf("Mot"); break;
                case 2 : switch (b)
                         {
                                 case 1 : printf("A"); break;
                                 case 2 : printf("B"); break;
                         } break;
                case 3 : printf("Ba"); break;
                default : printf("Khong biet doc");
        }
}
```

Cấu trúc rẽ nhánh: Câu lệnh switch

- Các giá trị trong mỗi trường hợp phải khác nhau.

```
switch (a)
{
        case 1 : printf(“Mot”); break;
        case 1 : printf(“MOT”); break;
        case 2 : printf(“Hai”); break;
        case 3 : printf(“Ba”); break;
        case 1 : printf(“1”); break;
        case 1 : printf(“mot”); break;
        default : printf(“Khong biet doc”);
}
```

Cấu trúc rẽ nhánh: Câu lệnh switch

- switch sẽ nhảy đến case tương ứng và thực hiện đến khi nào gặp break hoặc cuối switch sẽ kết thúc.

```
switch (a)
{
        case 1 : printf("Mot"); break;
        case 2 : printf("Hai"); break;
        case 3 : printf("Ba"); break;
}
```

Cấu trúc rẽ nhánh: Câu lệnh switch

- switch nhảy đến case tương ứng và thực hiện đến khi nào gặp break hoặc cuối switch sẽ kết thúc.

```
switch (a)
{
        case 1 : printf(“Mot”); break;
        case 2 : printf(“Hai”); break;
        case 3 : printf(“Ba”); break;
}
switch (a)
{
        case 1 : printf(“Mot”); break;
        case 2 : printf(“Hai”); break;
        case 3 : printf(“Ba”); break;
}
```

Cấu trúc rẽ nhánh: Câu lệnh switch

- Tận dụng tính chất khi bỏ break;

```
switch (a)
{
        case 1 : printf("So le"); break;
        case 2 : printf("So chan"); break;
        case 3 : printf("So le"); break;
        case 4 : printf("So chan"); break;
}
switch (a)
{
        case 1 :
        case 3 : printf("So le"); break;
        case 2 :
        case 4 : printf("So chan"); break;
}
```

Cấu trúc rẽ nhánh: Toán tử goto

- Nhãn được viết như tên biến và có thêm dấu: (hai chấm) đứng sau, nhãn có thể được gán cho bất kì câu lệnh nào trong chương trình
- Lệnh nhảy goto có dạng:

  goto nhan;

  - Khi gặp lệnh này, máy nhảy đến nhãn viết sau từ khoá goto
- Ví dụ:

```
main()
{       int i;
        vaosl: printf("Nhap i: "); scanf("%d",&i);
        if (n<10) goto vaosl;
}
```

Cấu trúc rẽ nhánh: Bài tập thực hành

1. Nhập một số bất kỳ. Hãy đọc giá trị của số nguyên đó nếu nó có giá trị từ 1 đến 9, ngược lại thông báo không đọc được.
2. Nhập một chữ cái. Nếu là chữ thường thì đổi sang chữ hoa, ngược lại đổi sang chữ thường.
3. Giải phương trình bậc nhất $ax + b = 0$.
4. Giải phương trình bậc hai $ax^2 + bx + c = 0$.

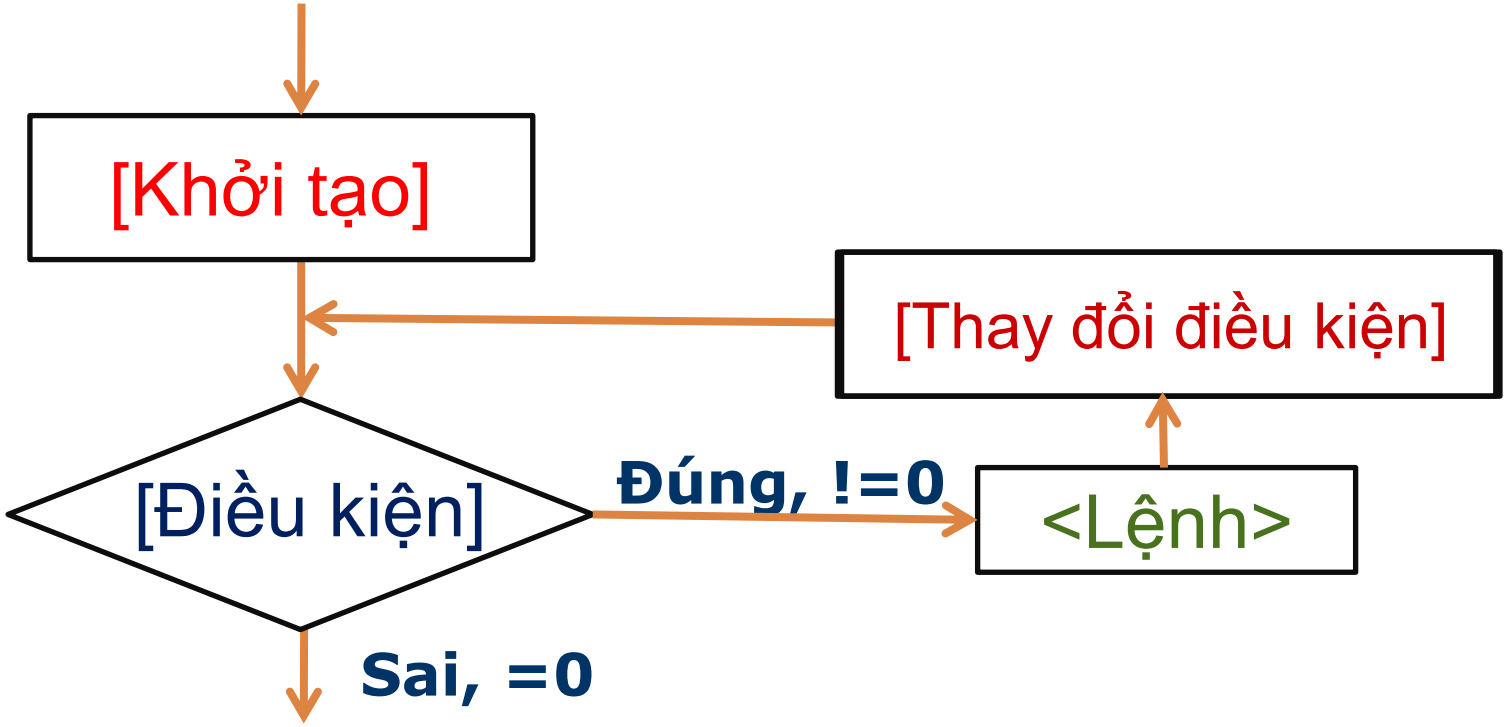
Cấu trúc rẽ nhánh: Bài tập thực hành

5. Nhập 4 số nguyên a, b, c và d. Tìm số có giá trị nhỏ nhất (min).
6. Nhập 4 số nguyên a, b, c và d. Hãy sắp xếp giá trị của 4 số nguyên này theo thứ tự tăng dần.
7. Tính tiền đi taxi từ số km nhập vào. Biết:
   a. 1 km đầu giá 15000đ
   b. Từ km thứ 2 đến km thứ 5 giá 13500đ
   c. Từ km thứ 6 trở đi giá 11000đ
   d. Nếu trên 120km được giảm 10% tổng tiền.

Cấu trúc rẽ nhánh: Bài tập thực hành

8. Nhập vào tháng và năm. Cho biết tháng đó có bao nhiêu ngày.
9. Nhập độ dài 3 cạnh 1 tam giác. Kiểm tra đó có phải là tam giác không và là tam giác gì?

Cấu trúc lặp: Câu lệnh for



for ([Khởi tạo]**;** [Điều kiện]**;** [Thay đổi điều kiện]**)**

<Lệnh>**;**

Cấu trúc lặp: Câu lệnh for

BÀI TOÁN: Tính tổng n số tự nhiên đầu tiên
INPUT: Số tự nhiên n nhập từ bàn phím
OUTPUT: Tổng n số tự nhiên đầu tiên S=1+2+…+n

THUẬT TOÁN:

c 1. Nhập số tự nhiên *n*

Bước 2. Gán *Tong* = 0; *Dem* = *1*;

c 3. Nếu *Dem* <= *n* là sai,đưa ra *Tong* và kết thúc

c 4. Gán *Tong* = *Tong* + *Dem*

Bước 5. Tăng số đếm: *Dem* = *Dem* + 1

Bước 6. Quay lại Bước 3

Cấu trúc lặp: Câu lệnh for

BÀI TOÁN: Tính tổng n số tự nhiên đầu tiên
INPUT: Số tự nhiên n nhập từ bàn phím
OUTPUT: Tổng n số tự nhiên đầu tiên S=1+2+…+n

```
#include <stdio.h>
#include <conio.h>
int main()
{
  int n, tong, dem;
  printf("Nhap so tu nhien n= "); scanf("%d",&n);
  tong = 0;
  for (dem = 0; dem <=n; dem++)
    {
      tong = tong + dem;
    }
  printf("Tong %d so tu nhien dau tien la: %d",n , tong);
  getch();
}
```

Cấu trúc lặp: Câu lệnh for

- Trong câu lệnh for, có thể sẽ không có phần [Khởi tạo]
- Ví dụ: Viết ra màn hình các số từ 1 đến 10

Cách viết 1:

```
int i;
for (i = 1; i <= 10; i++)
      printf("%d ", i);
```

Cách viết 2:

```
int i = 1;
for (; i <= 10; i++)
      printf("%d", i);
```

Cấu trúc lặp: Câu lệnh for

- Trong câu lệnh for, có thể sẽ không có phần [Điều kiện]
  - Khi đó [Điều kiện] là luôn luôn đúng
- Ví dụ: Viết ra màn hình các số từ 1 đến 10

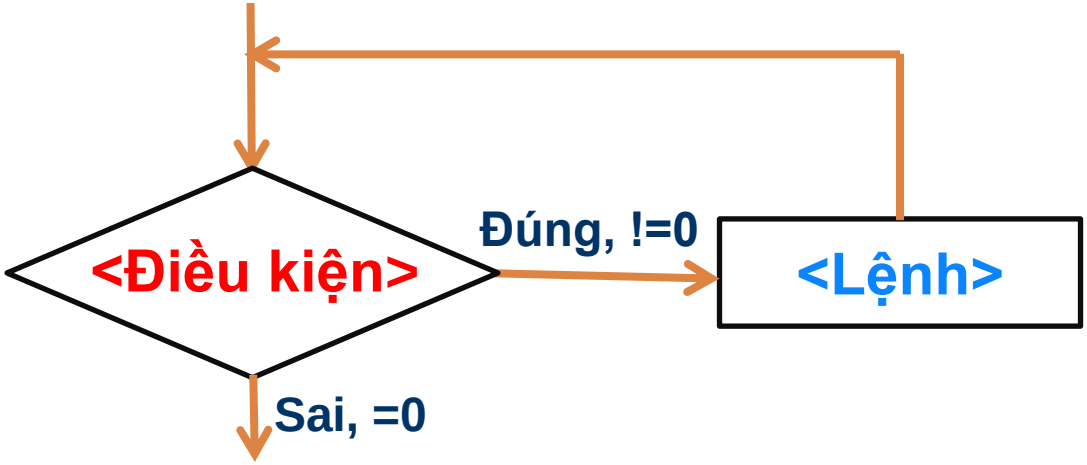
Cách viết 1:

```
int i;
for (i = 1; i <= 10; i++)
      printf("%d ", i);
```

Cách viết 2:

```
for (i = 1; ; i++)
{
      if (i > 10) break;
      printf("%d", i);
}
```

Cấu trúc lặp: Câu lệnh while



while (<Điều kiện>**)**

<Lệnh>**;**

Cấu trúc lặp: Câu lệnh while

BÀI TOÁN: Nhập một dãy số tự nhiên từ bàn phím cho đến khi số 0 được nhập và đếm số lượng số vừa nhập
INPUT: Nhập các số nguyên từ bàn phím
OUTPUT: Số lượng số vừa nhập

THUẬT TOÁN:

Bước 1. Gán *Dem = 0*;

c 2. Nhập một số nguyên *k*;

c 3. Nếu *k = 0* là đúng,đưa ra *Dem* và kết thúc

Bước 4. Tăng số đếm: *Dem = Dem + 1*

c 5. Nhập một số nguyên *k*;

Bước 6. Quay lại Bước 3

Cấu trúc lặp: Câu lệnh while

BÀI TOÁN: Nhập một dãy số tự nhiên từ bàn phím cho đến khi số 0 được nhập và đếm số lượng số vừa nhập
INPUT: Nhập các số nguyên từ bàn phím
OUTPUT: Số lượng số vừa nhập

```c
#include <stdio.h>
#include <conio.h>
int main()
{
  int k, dem = 0;
  scanf("%d",&k);
  while (k!=0)
  {
    dem++;
    scanf("%d",&k);
  }
  printf("So luong cac so nguyen da nhap la: %d",dem);
  getch();
}
```

Cấu trúc lặp: Câu lệnh while

- Câu lệnh while có thể bị lặp vô tận (loop)

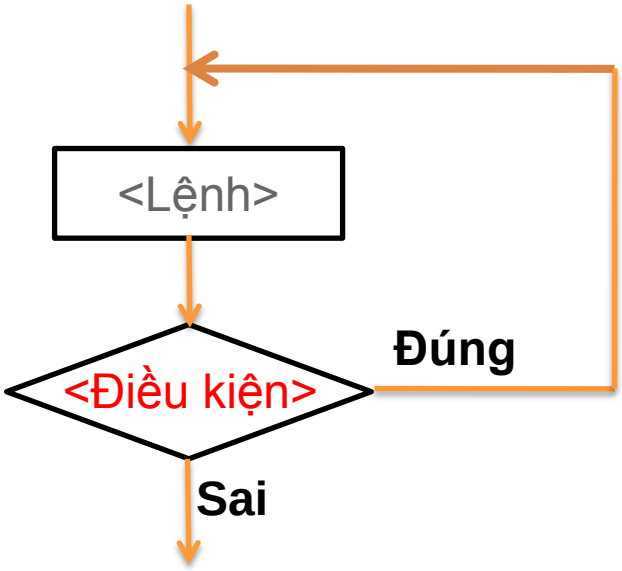
```
void main()
{
        int n;
        n = 1;
        while (n < 10) printf("%d", n);
}
```

```
void main()
{
        int n = 1;
        while (1)
                printf("%d", n);
}
```

  - Lặp vô hạn chỉ kết thúc được nhờ câu lệnh break

➔ Trong thân vòng lặp, cần có lệnh thay đổi giá trị <Điều kiện>

Cấu trúc lặp: Câu lệnh do … while



do

<Lệnh>**;**

while (<Điều kiện>**)**

Cấu trúc lặp: Câu lệnh do … while

BÀI TOÁN: Nhập một dãy số tự nhiên từ bàn phím cho đến khi số 0 được nhập và đếm số lượng số vừa nhập
INPUT: Nhập các số nguyên từ bàn phím
OUTPUT: Số lượng số vừa nhập

```
#include <stdio.h>
#include <conio.h>
int main()
{
  int k, dem = 0;
  printf("Nhap day cac so nguyen cho den khi nhap so 0\n");
  do
  {
    scanf("%d",&k);
    dem++;
  } while (k!=0);
  printf("So luong cac so nguyen da nhap la: %d",dem-1);
  getch();
}
```

Cấu trúc lặp: Câu lệnh do … while

- Câu lệnh do… while có thể bị lặp vô tận (loop)

```
        int n = 1;
        do
                printf("%d", n);
        while (n < 10);
```

```
void main()
{
        int n = 1;
        do
                printf("%d", n);
        while (1)
}
```

Một số lưu ý

- Cho phép ra khỏi for, while, do…while và switch
- Khi có nhiều chu trình lồng nhau, break đưa máy ra khỏi chu trình bên trong nhất chứa nó.
- Ví dụ: Thuật toán kiểm tra tính nguyên tố của n

```
int i,n, ng_to = 1;
for (i=2;i<=sqrt(n);i++)
   if (n%i==0)
      { ng_to=0;
        break;
      }
if (ng_to)
      printf("\n%d la so nguyen to",n);
else printf("\n%d la hop so",n);
```

Một số lưu ý (tt)

- continue dùng để bắt đầu một vòng mới của chu trình bên trong nhất chứa nó:
  - Trong thân toán tử for: máy sẽ chuyển tới bước khởi đầu lại
  - Trong thân của while hoặc do…while: máy chuyển tới xác định giá trị biểu thức (viết sau while), sau đó tiến hành kiểm tra điều kiện kết thúc chu trình
- Lưu ý: continue không áp dụng cho switch

Ví dụ câu lệnh continue

```
#include <stdio.h>
void main()
{ int i;
  for (i=1;i<=5;i++)
      { printf("\nBat dau %d",i);
        if (i<4) continue;
       printf("\nChao ban");
      }
}
```

- Kết quả:

```
Bat dau 1
Bat dau 2
Bat dau 3
Bat dau 4
Chao ban
Bat dau 5
Chao ban
```

Cấu trúc lặp: Bài tập thực hành

1. Nhập một số nguyên dương n (n > 0).

   Hãy cho biết:

   a. Có phải là số đối xứng? Ví dụ: 121, 12321, …
   b. Có phải là số chính phương? Ví dụ: 4, 9, 16, …
   c. Có phải là số nguyên tố? Ví dụ: 2, 3, 5, 7, …
   d. Chữ số lớn nhất và nhỏ nhất?
   e. Các chữ số có tăng dần hay giảm dần không?

Cấu trúc lặp: Bài tập thực hành

2. Nhập một số nguyên dương n. Tính:
   a. $S = 1 + 2 + \ldots + n$
   b. $S = 1^2 + 2^2 + \ldots + n^2$
   c. $S = 1 + 1/2 + \ldots + 1/n$
   d. $S = 1*2*\ldots*n = n!$
   e. $S = 1! + 2! + \ldots + n!$
3. Nhập 3 số nguyên a, b và n với a, b < n. Tính tổng các số nguyên dương nhỏ hơn n chia hết cho a nhưng không chia hết cho b.
4. Tính tổng các số nguyên tố nhỏ hơn n
   (0 < n < 50000)

Cấu trúc lặp: Bài tập thực hành

5. Nhập một số nguyên dương n. Xuất ra số ngược lại. Ví dụ: Nhập 1706 ➔ Xuất 6071.
6. Tìm và in lên màn hình tất cả các số nguyên trong phạm vi từ 10 đến 99 sao cho tích của 2 chữ số bằng 2 lần tổng của 2 chữ số đó.
7. Tìm bội số chung lớn nhất của 2 số nguyên dương a và b nhập từ bàn phím.
8. Nhập n. In n số đầu tiên trong dãy Fibonacy biết

$$F_0=F_1=1$$

$$F_n = F_{n-1} + F_{n-2}$$

CƠ SỞ LẬP TRÌNH

CHƯƠNG TRÌNH CON

Nội dung

- Khái niệm
- Cách xây dựng hàm
- Tầm tác dụng của biến
- Truyền tham số cho hàm
- Hàm đệ quy
- Một số hàm thông dụng

4.1 Khái niệm

- Chương trình con là:
  - Một đoạn chương trình có tên, đầu vào và đầu ra.
  - Có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính.
  - Được gọi nhiều lần với các tham số khác nhau.
  - Được sử dụng khi có nhu cầu:
    - Tái sử dụng: có một số chương trình được thực hiện ở nhiều nơi, bản chất không đổi nhưng giá trị các tham số cung cấp khác nhau.
    - Chia để trị: chia chương trình lớn thành các chương trình nhỏ rồi ghép lại.

  ➔ Giúp chương trình trong sáng, dễ hiểu, dễ phát hiện lỗi và cải tiến.

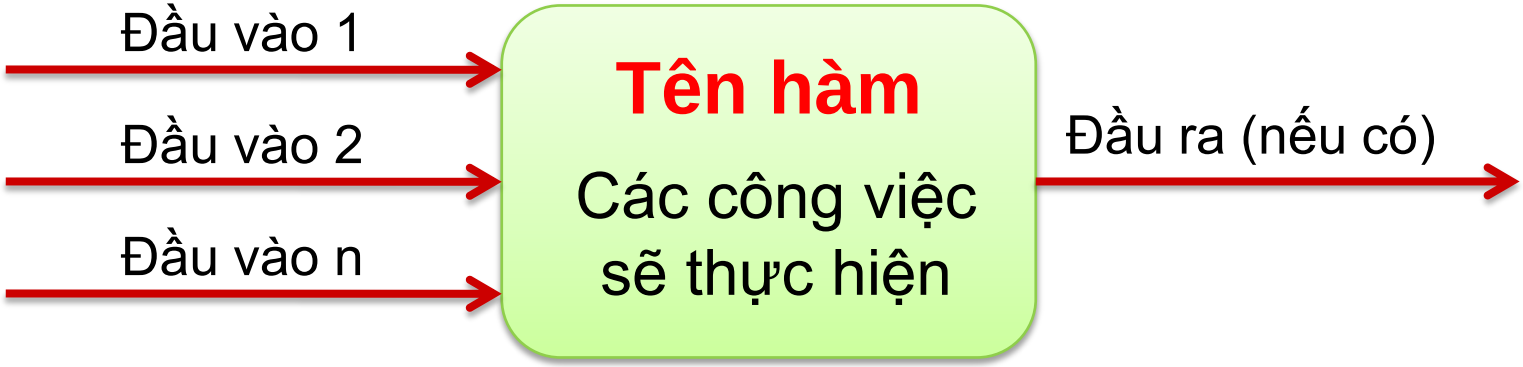
4.1 Khái niệm (tt)

- Trong các ngôn ngữ khác, có 2 loại chương trình con:
  - Hàm (function): trả về giá trị thông qua tên hàm, sử dụng trong các biểu thức và không được gọi như một lệnh.
  - Thủ tục: không có giá trị trả về, có thể tồn tại độc lập và được gọi như là một câu lệnh.
- Trong C: **chỉ tồn tại chương trình con dưới dạng hàm**, **không có thủ tục.**
  - Giá trị hàm có thể không cần dùng đến
  - Có thể không có giá trị nào gán vào tên hàm (void)
  - Cung cấp các giá trị không phải là vô hướng

4.2 Cách xây dựng hàm

Cú pháp

```
<kiểu trả về> <tên hàm>([danh sách tham số])
{
        <các câu lệnh>
        [return <giá trị>;]
}
```



Một số quy tắc

- Tham số thực sự và tham số hình thức
  - Tham số hình thức: tham số dùng khi khai báo
  - Tham số thực sự: tham số được cung cấp cho hàm khi được sử dụng
  - Tham số thực sự có thể là một biểu thức còn tham số hình thức thì không thể là một biểu thức
- Lệnh return
  - Tương đương lệnh `<Tên hàm> = <Giá trị>`
  - return có thể trả lại giá trị cả một biểu thức

    Ví dụ: `return x*x + b*x + c`
  - return có thể xuất hiện nhiều lần trong hàm

```
Ví dụ: if (s>0) return (s);
       else return (-s);
```

Một số quy tắc (tt)

- Hàm không trả lại giá trị
  - Dùng từ khoá **`void`** để khai báo (Ví dụ 1)
- Hàm không có tham số
  - Khai báo: `Tên_hàm(void)`
  - Ví dụ: Nhập số nguyên, trả về giá trị số nhập vào

```
int Nhap()
{
        int n;
        printf("Nhap mot so nguyen: ");
        scanf("%d", &n);
        return n;
}
```

Một số quy tắc (tt)

- Hàm phải được khai báo và định nghĩa trước khi sử dụng và thường đặt ở trên hàm chính (hàm main).
- Ví dụ:

```
int Tong(int a, int b)
{
return a + b;
}
void main()
{
int a = 2912, b = 1706;
int sum = Tong(a, b);    /* Loi goi ham */
```

Một số quy tắc (tt)

- Thông thường, trước hàm main ta chỉ xác định tên hàm, các tham số và giá trị trả về, phần định nghĩa sẽ được đưa xuống dưới cùng. Phần này được gọi là nguyên mẫu hàm (function prototype).

```
int Tong(int a, int b); // prototype ham Tong
void main()
{
int a = 2912, b = 1706;
int sum = Tong(a, b);    /* Loi goi ham */
}
int Tong(int a, int b)   /* Mo ta ham tong */
{
return a + b; }
```

Một số ví dụ

- Ví dụ 1: Chuyển chữ thường thành chữ hoa

```c
#include <stdio.h>
#include <conio.h>
char to_UpperCase(char ch)
{
    if (ch>='a' && ch <='z')
        return (ch + 'A'-'a');
    else return ch;
}

int main()
{   char thuong,hoa;
    printf("Nhap vao mot ki tu: ");
    scanf("%c", &thuong);
    hoa = to_UpperCase(thuong);
    printf("\nChu hoa tuong ung la: %c\n",hoa);
    getch();
}
```

4.3 Tầm tác dụng của biến

- Các loại biến
  - Toàn cục: khai báo trong ngoài tất cả các hàm và có tác dụng lên toàn bộ chương trình.
  - Cục bộ: khai báo trong hàm hoặc khối { } và chỉ có tác dụng trong bản thân hàm hoặc khối đó (kể cả khối con nó). Biến cục bộ sẽ bị xóa khỏi bộ nhớ khi kết thúc khối khai báo nó.

```
#include <stdio.h>
int i; /*Bien toan cuc */
main()
{ … }
void thi_du()
{
        int m=3; /*Bien cuc bo */
}
```

Chú ý

- Cấp phát bộ nhớ tĩnh cho biến cục bộ:

  static <tên kiểu> <tên biến>;

- Khai báo kiểu bố trí ô nhớ cho biến int nào đó được sử dụng rất nhiều là kiểu bộ nhớ thanh ghi **register** để tăng tốc độ xử lý. Biến thanh ghi thường là các biến đếm trong một vòng lặp nào đó.
  - ***Ví dụ:***

```
register int t;
for (t=0; t<1000; t++)
printf("Lan goi thu %d",t);
```

4.4 Truyền tham số cho hàm

- Trong C thực hiện truyền tham số theo một kiểu duy nhất: truyền giá trị

```
#include <stdio.h>
void hoan_vi(int a, int b); /* prototype */
main()
{       int n = 10, p=20;
        printf("Truoc khi goi ham: %d  %d\n",n,p);
        hoan_vi(n,p);
        printf("Sau khi goi ham: %d  %d\n",n,p);
}
void hoan_vi(int a, int b)
{       int t;
        printf("Truoc khi hoan vi: %d  %d\n",a,b);
        t=a; a=b; b=t;
        printf("Sau khi hoan vi: %d  %d\n",a,b);
}
```

Truyền tham số cho hàm

- Truyền địa chỉ cho hàm
  - ***a** là giá trị được lưu trữ trong bộ nhớ có địa chỉ **a**
  - &a là địa chỉ bộ nhớ chứa giá trị **a**

```
void hoan_vi(int *a, int *b);
main()
{       int n=10, p=20;
        printf("Truoc khi goi ham: %d  %d\n",n,p);
        hoan_vi(&n,&p);
        printf("Sau khi goi ham: %d %d",n,p);
}
void hoan_vi(int *a, int *b)
{       int t;
        printf("Truoc khi hoan vi: %d  %d\n",*a,*b);
        t=*a;    *a=*b;    *b=t;
        printf("Sau khi hoan vi: %d  %d\n",*a,*b);
}
```

Lưu ý khi truyền đối số

- Lưu ý
  - Trong một hàm, các tham số có thể truyền theo nhiều cách.

```
void HonHop(int x, int *y)
{
        ...
}
```

  - **KHÔNG** được truyền giá trị cho tham số *a và *b trong ví dụ trên, ví dụ, không viết: `hoan_vi(1,5)`
  - Truyền giá trị được sử dụng khi không có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm
  - Truyền địa chỉ hoặc truyền tham biến được sử dụng khi có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm

Lời gọi hàm

- Cách thực hiện
  - Gọi tên của hàm đồng thời truyền các đối số (hằng, biến, biểu thức) cho các tham số theo đúng thứ tự đã được khai báo trong hàm.
  - Các biến hoặc trị này cách nhau bằng dấu ,
  - Các đối số này được được đặt trong cặp dấu ngoặc đơn ( )

```
<tên hàm> (<đối số 1>,… , <đối số n>);
```

4.6 Một số hàm thông dụng

- Các hàm toán học (trong **stdlib.h**)
  - `int abs(int x);` giá trị tuyệt đối của số nguyên x
  - `long int labs(long int x);` giá trị tuyệt đối của số nguyên dài x
  - `double fabs(double x);`
  - `int rand(void);` cho giá trị ngẫu nhiên từ 0 đến 32767
  - `int random(int n);` cho một giá trị ngẫu nhiên từ 0 đến n-1
  - `void srand(unsigned seed);` khởi đầu bộ số ngẫu nhiên bằng giá trị seed
  - `void randomize(void);` tạo điểm xuất phát ngẫu nhiên cho các hàm rand() và random() ở trên

Một số hàm thông dụng (tt)

- Các hàm toán học
  - ***Các hàm lượng giác***: `sin, asin, sinh, cos, cosh, acos, tan, atan, tanh`
  - `double log(double x);` tính logarit tự nhiên của x
  - `double log10(double x);` tính logarit cơ số 10 của x
  - `double pow(double x, double y);` tính x^y
  - `double ceil(double x);` hàm làm tròn lên, trả về số nguyên nhỏ nhất lớn hơn x

  Ví dụ: `ceil(123.54) = 124`
  - `double floor(double x);` hàm làm tròn xuống, cắt phần lẻ.

  Ví dụ: `floor(123.54) = 123`

Một số hàm thông dụng (tt)

- Các hàm thời gian
  - **`void getime(struct time *t);`** Trả về giờ hệ thống và đặt vào các thành phần của một biến cấu trúc kiểu time do con trỏ **`t`** trỏ tới.
  - Kiểu cấu trúc **`time`** trong dos.h được định nghĩa:

```
struct time
{  unsigned ti_hour;  /*giờ */
   unsigned ti_min; /*phút*/
   unsigned ti_sec; /*giây */
   unsigned ti_hund; /*phần trăm*/
}
```

  - **`void settime(struct time *t);`** đặt lại giờ hệ thống theo giá trị các thành phần của một cấu trúc kiểu time do con trỏ **`t`** trỏ tới.

Một số hàm thông dụng (tt)

- Các hàm ngày tháng
  - **`getdate(struct date *d);`** Hàm này nhận ngày hệ thống và đặt vào các thành phần của một biến cấu trúc kiểu **`date`** do con trỏ **`d`** trỏ tới.
  - Kiểu cấu trúc **`date`** được định nghĩa trong dos.h

```
struct date
{
        int da_year;
        char da_mon;
        char da_day;
}
```

  - **`void setdate(struct date *d);`** Đặt lại ngày hệ thống theo giá trị các thành phần của một biến cấu trúc kiểu date do con trỏ d trỏ tới.

Một số hàm thông dụng (tt)

Hàm chuyển đổi xâu kí tự

- `char *itoa (int x, char *s, int cs);` chuyển đổi số nguyên `x` trong hệ đếm cơ số `cs` sang chuỗi và lưu vào vùng nhớ `s`, trả về địa chỉ vùng `s`.
- `char *ltoa (long x, char *s, int cs);` chuyển đổi số nguyên dài kiểu `long x` trong hệ đếm cơ số `cs` sang chuỗi và lưu vào vùng nhớ `s`, trả về địa chỉ vùng `s`.
- `char *ultoa (unsigned long x, char *s, int cs);` chuyển số kiểu `unsigned long x` trong hệ đếm cơ số `cs` sang chuỗi và lưu vào vùng nhớ `s`, trả về địa chỉ của vùng `s`.
- `double atof (const char *s);` chuyển xâu `str` thành số `float`
- `int atoi (const char*s);` chuyển xâu `str` thành số `int`
- `long atol (cont char *s);` chuyển xâu `str` thành số `long`

Một số hàm thông dụng (tt)

Các hàm cấp phát động

- `unsigned coreleft (void);` cho biết số bộ nhớ khả dụng trong vùng cấp phát động đối với mô hình **tiny**, **small** và **medium**
- `unsigned long coreleft (void);` cho biết số bộ nhớ khả dụng trong vùng cấp phát động đối với mô hình **compact large** và **huge**
- `void *calloc (size_t n, size_t size);` cấp phát vùng nhớ cho n đối tượng cỡ **size byte**
- `void *malloc (size_t size);` cấp phát vùng nhớ cho **size byte**
- `void *realloc (void *block, size_t size);` cấp phát lại bộ nhớ
- `void free (void *block);` giải phóng vùng nhớ đã cấp

CƠ SỞ LẬP TRÌNH

KIỂU DỮ LIỆU MẢNG

Nội dung

- Mảng một chiều
  - Khái niệm
  - Khai báo
  - Thao tác trên mảng
  - Hàm có mảng một chiều là tham số
- Mảng nhiều chiều
  - Khái niệm
  - Khai báo
  - Thao tác trên mảng
  - Hàm có mảng nhiều chiều là tham số

Đặt vấn đề

- Ví dụ
  - Chương trình cần lưu trữ 3 số nguyên?
    => Khai báo 3 biến int a1, a2, a3;
  - Chương trình cần lưu trữ 100 số nguyên?
    => Khai báo 100 biến kiểu số nguyên!
  - Người dùng muốn nhập n số nguyên?
    => Không thực hiện được!
- Giải pháp
  - Kiểu dữ liệu mới cho phép lưu trữ một dãy các số nguyên và dễ dàng truy xuất.

Mảng một chiều

- Khái niệm
- Khai báo
- Thao tác trên mảng
- Hàm có mảng một chiều là tham số

5.1.1 Khái niệm

Khái niệm

- Là một kiểu dữ liệu có cấu trúc do người lập trình định nghĩa.
- Biểu diễn một dãy các biến có cùng kiểu. Ví dụ: dãy các số nguyên, dãy các ký tự…
- Kích thước được xác định ngay khi khai báo và không bao giờ thay đổi.
- C luôn chỉ định một khối nhớ liên tục cho một biến kiểu mảng.

5.1.2 Khai báo

Khai báo tường minh

```
<kiểu cơ sở> <tên biến mảng>[<số phần tử>];
<kiểu cơ sở> <tên biến mảng>[<N1>][<N2>]…[<Nn>];
```

- <N1>, …, <Nn> : số lượng phần tử của mỗi chiều.

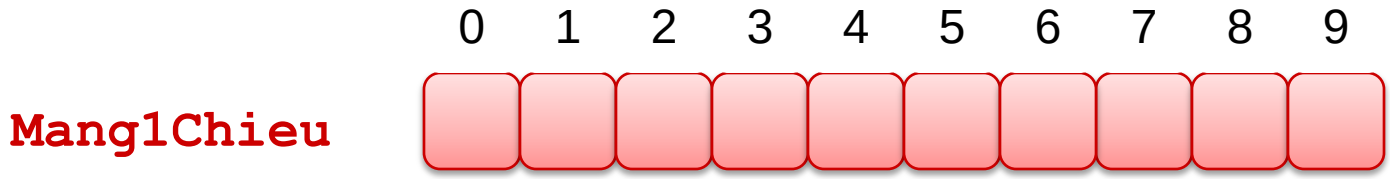
Lưu ý

- Phải xác định <số phần tử> cụ thể (hằng) khi khai báo.
- Mảng nhiều chiều: <tổng số phần tử> = N1*N2*…*Nn
- Bộ nhớ sử dụng = <tổng số phần tử>*sizeof(<kiểu cơ sở>)
- Một dãy liên tục có chỉ số từ 0 đến <tổng số phần tử>-1

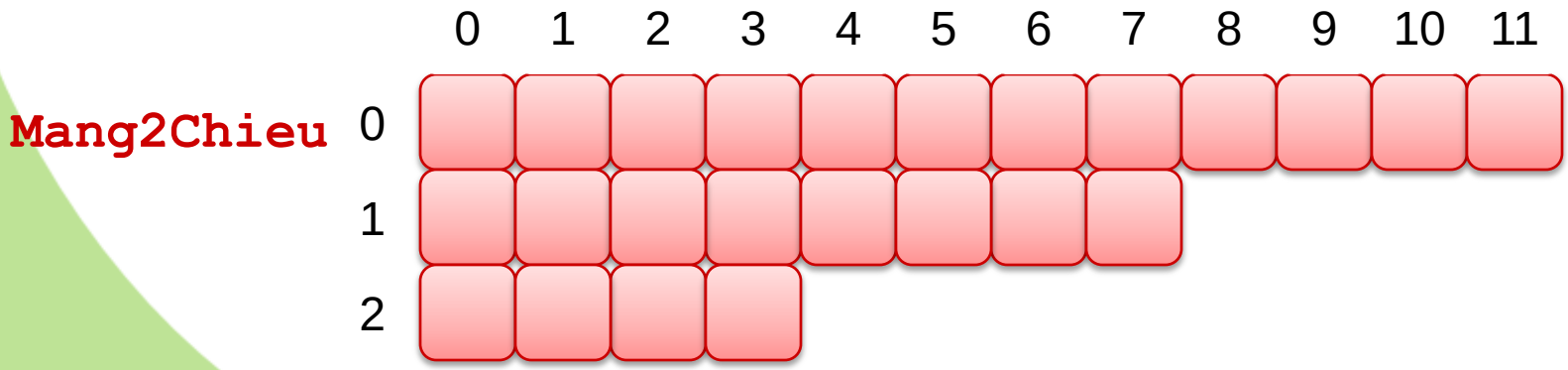
Khai báo tường minh (tt)

Ví dụ

```
int Mang1Chieu[10];
```




```
int Mang2Chieu[3][4];
```



Khai báo không tường minh

Cú pháp

- Không tường minh (thông qua khai báo kiểu)

```
typedef <kiểu cơ sở> <tên kiểu mảng>[<số phần tử>];
typedef <kiểu cơ sở> <tên kiểu mảng>[<N1>]…[<Nn>];
<tên kiểu mảng> <tên biến mảng>;
```

Ví dụ

```
typedef int Mang1Chieu[10];
typedef int Mang2Chieu[3][4];

Mang1Chieu m1, m2, m3;
Mang2Chieu m4, m5;
```

Số phần tử của mảng

- Phải xác định cụ thể số phần tử ngay lúc khai báo, không được sử dụng biến hoặc hằng thường

```
int n1 = 10; int a[n1];
const int n2 = 20; int b[n2];
```

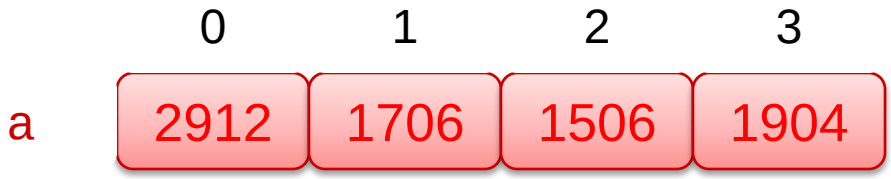
- Nên sử dụng chỉ thị tiền xử lý #define để định nghĩa số phần tử mảng

```
#define n1 10
#define n2 20
int a[n1];          // ⇔ int a[10];
int b[n1][n2];      // ⇔ int b[10][20];
```

Khởi tạo giá trị cho mảng lúc khai báo

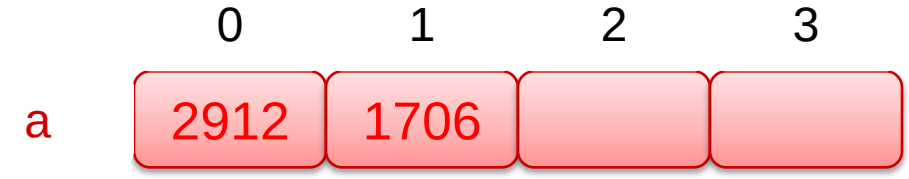
- Gồm các cách sau
  - Khởi tạo giá trị cho mọi phần tử của mảng

```
int a[4] = {2912, 1706, 1506, 1904};
```



  - Khởi tạo giá trị cho một số phần tử đầu mảng

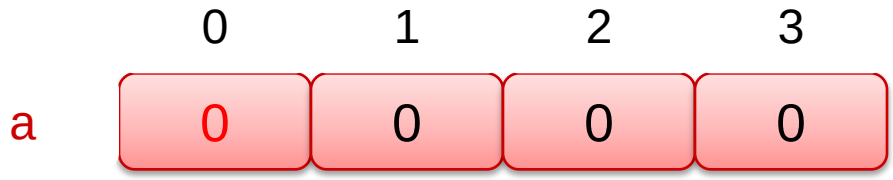
```
int a[4] = {2912, 1706};
```



Khởi tạo giá trị cho mảng lúc khai báo

- Gồm các cách sau
  - Khởi tạo giá trị 0 cho mọi phần tử của mảng

```
int a[4] = {0};
```



  - Tự động xác định số lượng phần tử

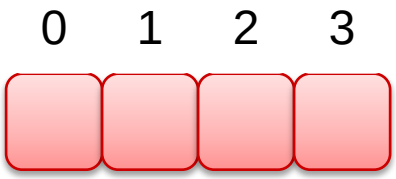
```
int a[] = {2912, 1706, 1506, 1904};
```

5.1.3 Truy xuất đến một phần tử

- Thông qua chỉ số

```
<tên biến mảng>[<chỉ số>]
```

- Ví dụ



  - Cho mảng như sau

```
int a[4];
```

  - Các truy xuất
    - Hợp lệ: a[0], a[1], a[2], a[3]
    - Không hợp lệ: a[-1], a[4], a[5], …

      => Cho kết thường không như mong muốn!

Gán dữ liệu kiểu mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử tương ứng

```
<biến mảng đích> = <biến mảng nguồn>; //sai
<biến mảng đích>[<chỉ số thứ i>] = <giá trị>;
```

- Ví dụ

```
#define MAX 3
typedef int MangSo[MAX];
MangSo a = {1, 2, 3}, b;

b = a;        // Sai
for (int i = 0; i < 3; i++) b[i] = a[i];
```

Một số lỗi thường gặp

- Khai báo không chỉ rõ số lượng phần tử
  - int a[]; → int a[100];
- Số lượng phần tử liên quan đến biến hoặc hằng
  - int n1 = 10; int a[n1];
- Khởi tạo cách biệt với khai báo
  - int a[4]; a = {2912, 1706, 1506, 1904};
    → int a[4] = {2912, 1706, 1506, 1904};
- Chỉ số mảng không hợp lệ
  - int a[4];
  - a[-1] = 1; a[10] = 0;

5.1.4 Một số bài toán trên mảng 1 chiều

- Nhập mảng
- Xuất mảng
- Tìm kiếm một phần tử trong mảng
- *Tìm giá trị nhỏ nhất/lớn nhất của mảng*
- *Sắp xếp mảng giảm dần/tăng dần*
- *Thêm/Xóa/Sửa một phần tử vào mảng*

Nhập mảng

- Đoạn chương trình nhập vào từ bàn phím một mảng 1 chiều a gồm n phần tử

```
#define MAXN 100
int a[MAXN];
...
printf("Nhap so luong phan tu n: ");
scanf("%d", &n);
for (int i = 0; i < n; i++)
      {
            printf("Nhap phan tu thu %d: ", i);
            scanf("%d", &a[i]);
      }
```

Xuất mảng

- Đoạn chương trình in ra màn hình một mảng 1 chiều a gồm n phần tử

```
printf("Noi dung cua mang la: ");
for (int i = 0; i < n; i++)
      printf("%d ", a[i]);
printf("\n");  /*Đưa con trỏ màn hình xuống dòng
                  tiếp theo */
```

Hàm có mảng là tham số

- Khai báo
  - Tham số kiểu mảng trong khai báo hàm giống như khai báo biến mảng

```
void NhapMang(int a[]);
```

  - Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng
    - Có thể bỏ số lượng phần tử hoặc sử dụng con trỏ.
    - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

```
void NhapMang(int a[]);
void NhapMang(int *a);
```

  - Số lượng phần tử thực sự truyền qua biến khác

```
void NhapMang(int a[100], int n);
void NhapMang(int a[], int n);
void NhapMang(int *a, int n);
```

Hàm nhập mảng

```
void NhapMang(int a[], int &n)
{
        printf("Nhap so luong phan tu n: ");
        scanf("%d", &n);

        for (int i = 0; i < n; i++)
        {
                printf("Nhap phan tu thu %d: ", i);
                scanf("%d", &a[i]);
        }
}
```

Hàm xuất mảng

```
void XuatMang(int a[], int n)
{
        printf("Noi dung cua mang la: ");

        for (int i = 0; i < n; i++)
              printf("%d ", a[i]);

        printf("\n");
}
```

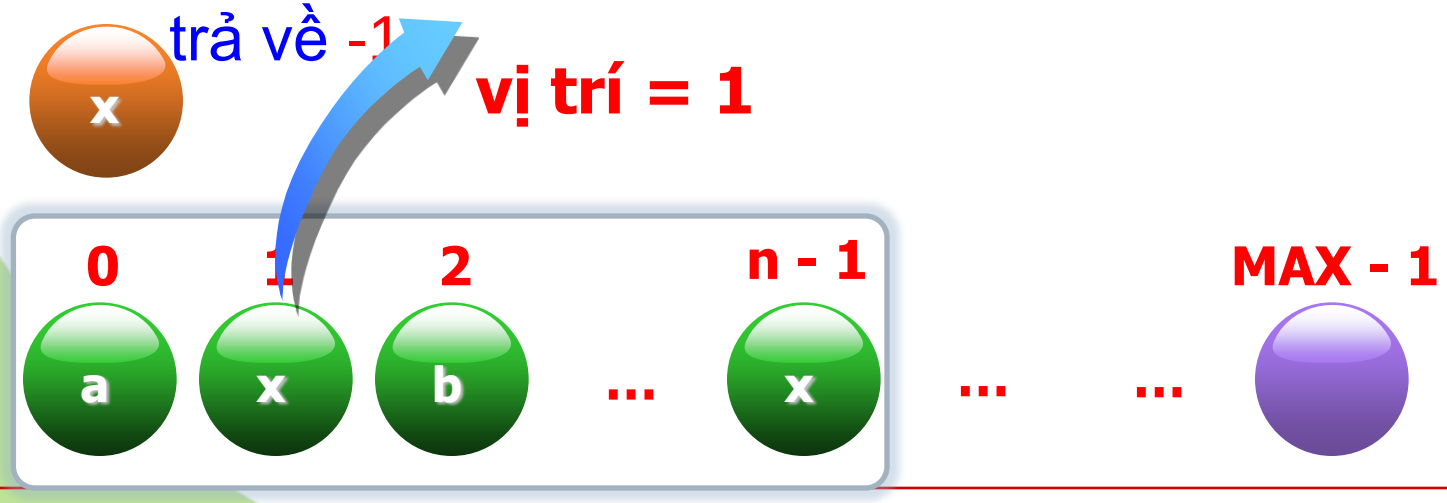
Truyền tham số mảng cho hàm (sử dụng)

- Lời gọi hàm

```
void NhapMang(int a[], int &n);
void XuatMang(int a[], int n);
int main()
{
	int a[100], n;
	NhapMang(a, n);
	XuatMang(a, n);
}
```

Tìm kiếm một phần tử trong mảng

- Yêu cầu
  - Tìm xem phần tử x có nằm trong mảng a kích thước n hay không? Nếu có thì nó nằm ở vị trí đầu tiên nào.
- Ý tưởng (tìm kiếm tuần tự)
  - Xét từng phần của mảng a. Nếu phần tử đang xét bằng x thì trả về vị trí đó. Nếu không tìm được thì trả về -1



Hàm tìm kiếm

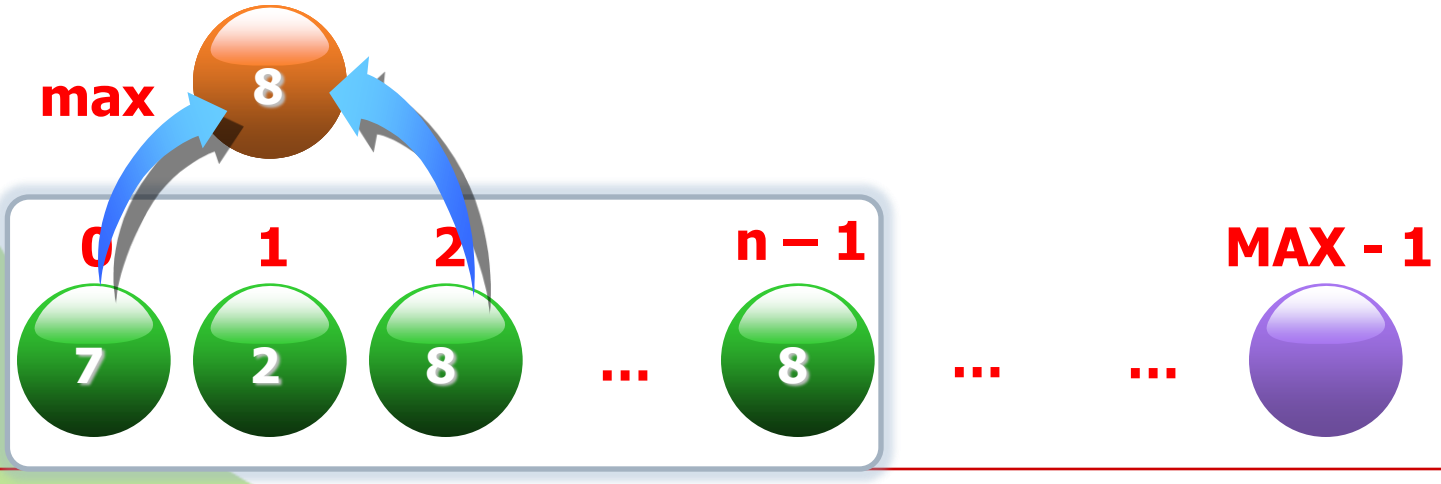
```
int TimKiem(int a[], int n, int x)
{
        for (int vt = 0; vt < n; vt++)
                if (a[vt] == x)
                        return vt;

        return -1;
}
```

- **Bài tập**:
  - Viết lại hàm tìm kiếm sử dụng câu lệnh while

Tìm giá trị lớn nhất của mảng

- Yêu cầu
  - Cho trước mảng a có n phần tử. Tìm giá trị lớn nhất trong a
- Ý tưởng
  - Giả sử giá trị max hiện tại là giá trị phần tử đầu tiên a[0]
  - Với mọi i từ 1 đến n-1, nếu a[i] > max thì max = a[i]



Hàm tìm giá trị lớn nhất của mảng

```
int TimMax(int a[], int n)
{
        int max = a[0];

        for (int i = 1; i < n; i++)
              if (a[i] > max)
                    max = a[i];

        return max;
}
```

Thêm một phần tử vào mảng

- Yêu cầu
  - Thêm phần tử x vào mảng a kích thước n tại vị trí vt.
- Ý tưởng
  - "Đẩy" các phần tử bắt đầu tại vị trí vt sang phải 1 vị trí.
  - Đưa x vào vị trí vt trong mảng.
  - Tăng n lên 1 đơn vị.

Hàm thêm một phần tử vào mảng

```
void Them(int a[], int &n, int vt, int x)
{
        if (vt >= 0 && vt <= n)
        {
                for (int i = n; i > vt; i--)
                        a[i] = a[i - 1];

                a[vt] = x;
                n++;
        }
}
```

Xóa một phần tử trong mảng

- Yêu cầu
  - Xóa một phần tử trong mảng a kích thước n tại vị trí vt
- Ý tưởng
  - "Kéo" các phần tử bên phải vị trí vt sang trái 1 vị trí.
  - Giảm n xuống 1 đơn vị.

Hàm xóa một phần tử trong mảng

```
void Xoa(int a[], int &n, int vt)
{
        if (vt >= 0 && vt < n)
        {
                for (int i = vt; i < n - 1; i++)
                        a[i] = a[i + 1];

                n--;
        }
}
```

Bài tập thực hành

1. Các thao tác nhập xuất
   a. Nhập mảng gồm n số nguyên từ bàn phím
   b. Xuất mảng vừa nhập ra màn hình
2. Các thao tác kiểm tra

Nhập vào một mảng gồm n số nguyên

a. Mảng vừa nhập có phải là mảng toàn số chẵn (số lẻ) hay không?
b. Mảng có phải là mảng toàn số nguyên tố? Số chính phương?
c. Mảng có phải là mảng tăng dần hay giảm dần hay không?

Bài tập thực hành

3. Các thao tác tính toán

Nhập vào mảng gồm n số nguyên, và 2 số m và k

a. Mảng có bao nhiêu số chia hết cho m nhưng không chia hết cho k?
b. Tổng các số nguyên tố, chính phương có trong mảng?

4. Các thao tác tìm kiếm

Nhập vào mảng gồm n số nguyên và số nguyên x

a. Tìm vị trí cuối cùng của phần tử x trong mảng
b. Tìm vị trí số nguyên tố đầu tiên trong mảng nếu có
c. Tìm số nhỏ nhất trong mảng
d. Tìm số dương nhỏ nhất, số âm lớn nhất

Bài tập thực hành

5. Các thao tác xử lý

Nhập mảng a gồm n số nguyên

a. Xây dựng mảng b gồm các số nguyên tố có trong mảng a
b. Xây dựng mảng b (chứa các số nguyên dương) và mảng c (chứa các số còn lại)
c. Sắp xếp mảng theo chiều giảm dần
d. Sắp xếp mảng sao cho các số dương đứng đầu mảng giảm dần, kế đến là các số âm tăng dần, cuối cùng là các số 0.

Bài tập thực hành

6. Các thao tác thêm/xóa/sửa
   a. Sửa các số nguyên tố có trong mảng thành số 0, tất cả các số chính phương thành số -1
   b. Chèn số 0 đằng sau mỗi số nguyên tố trong mảng, chèn số -1 phía trước mỗi số chính phương trong mảng.
   c. Xóa tất cả số nguyên tố trong mảng

7* . Nhập vào từ bàn phím một số nguyên dương n (n<10). Viết chương trình con in ra tất cả các hoán vị của dãy số 1 2 3…n

Ví dụ: n = 3 thì in ra:

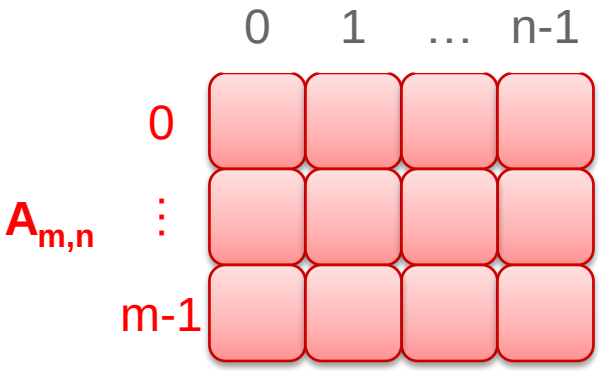
```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

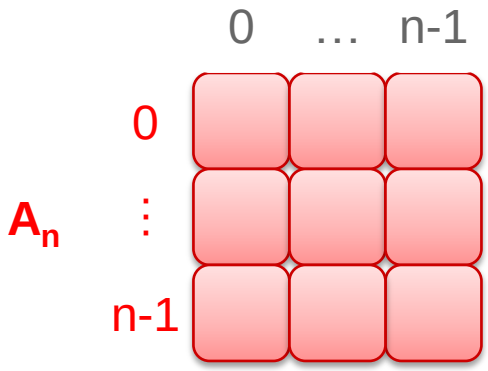
Mảng nhiều chiều

- Khái niệm
- Khai báo
- Thao tác trên mảng
- Hàm có mảng nhiều chiều là tham số

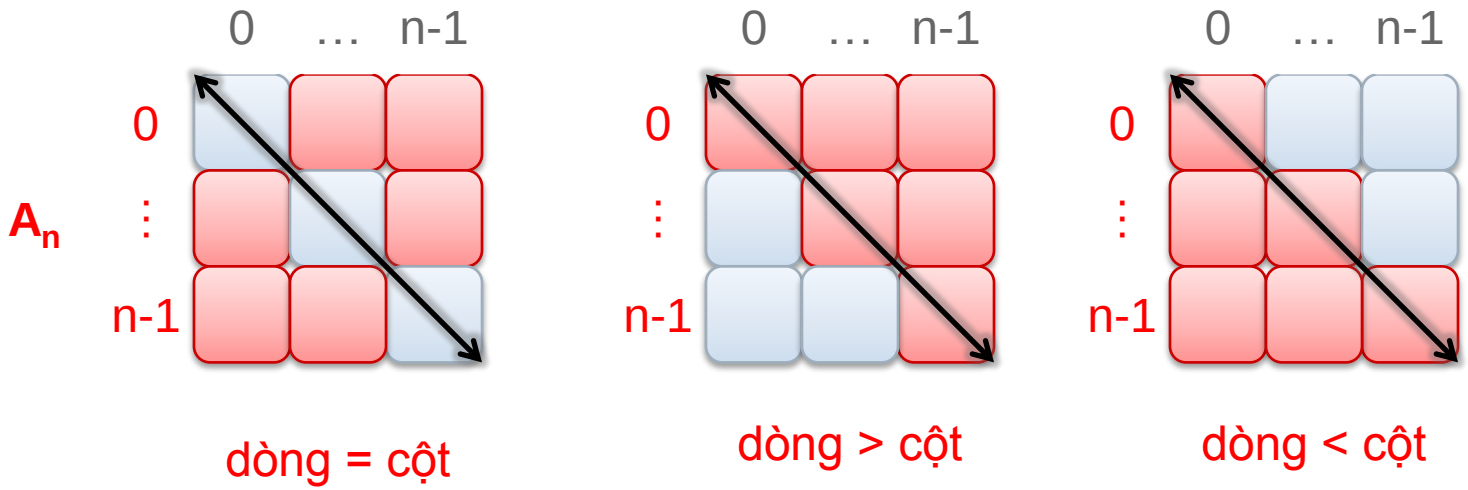
5.2.1 Khái niệm

- Ma trận





Ma trận vuông



5.2.2 Khai báo

Khai báo mảng 2 chiều

- Tường minh

```
<kiểu cơ sở> <tên biến>[<N1>][<N2>];
```

- Không tường minh (thông qua kiểu)

```
typedef <kiểu cơ sở> <tên kiểu>[<N1>][<N2>];

<tên kiểu> <tên biến>;
<tên kiểu> <tên biến 1>, <tên biến 2>;
```

  - N1, N2: số lượng phần tử mỗi chiều

Chú ý:

Ví dụ khai báo mảng 3 chiều: `int m[3][5][8]`

Khai báo mảng 2 chiều (tt)

- Ví dụ
  - Tường minh

```
int a[10][20], b[10][20];
int c[5][10];
int d[10][20];
```

  - Không tường minh (thông qua kiểu)

```
typedef int MaTran10x20[10][20];
typedef int MaTran5x10[5][10];

MaTran10x20 a, b;
MaTran11x11 c;
MaTran10x20 d;
```

Khởi tạo giá trị cho mảng nhiều chiều

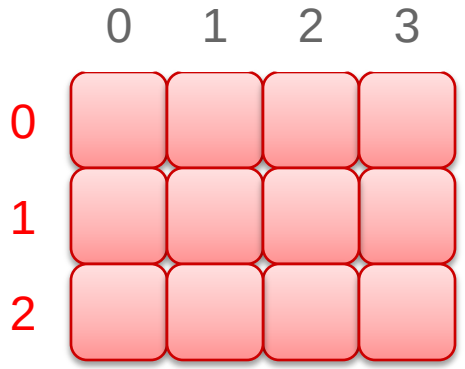
- Tương tự như khởi tạo giá trị ban đầu cho mảng 1 chiều
- Ví dụ:
  - int x[3][2]={{1,2},{3,4},{5,6}};
  - int y[3][2]={1,2,3,4,5,6};
  - int z[5][4]={0}

5.2.3 Truy xuất đến một phần tử

- Thông qua chỉ số

```
<tên biến mảng>[<chỉ số 1>][<chỉ số 2>]
```

- Ví dụ
  - Cho mảng 2 chiều như sau:




```
int a[3][4];
```

  - Các truy xuất
    - Hợp lệ: a[0][0], a[0][1], …, a[2][2], a[2][3]
    - Không hợp lệ: a[-1][0], a[2][4], a[3][3]

Gán giá trị cho mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử

```
<biến mảng đích> = <biến mảng nguồn>; //sai
<biến mảng đích>[<chỉ số 1>][chỉ số 2]=<giá trị>;
```

- Ví dụ

```
int a[5][10], b[5][10];

b = a;          // Sai
int i, j;
for (i = 0; i < 5; i++)
      for (j = 0; j < 10; j++)
            b[i][j] = a[i][j];
```

5.2.4 Một số bài toán cơ bản

- Nhập mảng
- Xuất mảng
- Tìm kiếm một phần tử trong mảng
- Kiểm tra tính chất của mảng
- Tìm giá trị nhỏ nhất/lớn nhất của mảng
- …

Nhập mảng 2 chiều

- Yêu cầu
  - Nhập vào từ bàn phím một mảng a gồm m dòng, n cột
- Ý tưởng
  - Khai báo một mảng 2 chiều có dòng tối đa là MAXD, số cột tối đa là MAXC. (dùng #define để định nghĩa)
  - Nhập số dòng m và số cột n thực sự của mảng
  - Nhập từng phần tử từ [0][0] đến [m-1][n-1].

Nhập mảng 2 chiều

- Đoạn chương trình nhập vào từ bàn phím một mảng 2 chiều a gồm m dòng, n cột

```
#define MAXD 100
#define MAXC 100
int a[MAXD][MAXC];
...
int main()
{ printf("Nhap so dong, so cot cua ma tran: ");
  scanf("%d%d", &m, &n);
  int i, j;
  for (i=0; i<m; i++)
       for (j=0; j<n; j++)
              {printf("Nhap a[%d][%d]: ", i, j);
               scanf("%d", &a[i][j]); }
}
```

Xuất mảng 2 chiều

- Yêu cầu
  - In ra màn hình mảng a gồm m dòng, n cột
- Ý tưởng
  - In giá trị từng phần tử của mảng 2 chiều từ dòng có 0 đến dòng m-1;
  - Mỗi dòng giá trị của cột 0 đến cột n-1 trên dòng đó;
  - Kết thúc mỗi dòng chèn thêm dấu xuống dòng "\n"

Xuất mảng 2 chiều (tt)

- Đoạn chương trình in ra màn hình một mảng 2 chiều a gồm m dòng, n cột

```
int main()
{
        /*Các thao tác nhập, xử lý mảng...*/
        /*in mảng ra màn hình */
        int i, j;
        for (i=0; i<m; i++)
        {
                for (j=0; j<n; j++)
                        printf("%d  ", a[i][j]);
                printf("\n");
        }
}
```

Hàm có mảng là tham số

- Tham số kiểu mảng trong khai báo hàm giống như khai báo biến mảng

```
void NhapMaTran(int a[50][100]);
```

- Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng
  - Có thể bỏ số lượng phần tử chiều thứ 2 hoặc con trỏ.
  - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

```
void NhapMaTran(int a[][100]);
void NhapMaTran(int (*a)[100]);
```

- Số lượng phần tử thực sự truyền qua biến khác

```
void XuatMaTran(int a[50][100], int m, int n);
void XuatMaTran(int a[][100], int m, int n);
void XuatMaTran(int (*a)[100], int m, int n);
```

Hàm nhập mảng 2 chiều

```
void NhapMaTran(int a[][MAXC], int &m, int &n)
{
        printf("Nhap so dong, so cot cua ma tran: ");
        scanf("%d%d", &m, &n);

        int i, j;
        for (i=0; i<m; i++)
                for (j=0; j<n; j++)
                {
                        printf("Nhap a[%d][%d]: ", i, j);
                        scanf("%d", &a[i][j]);
                }
}
```

Hàm xuất mảng 2 chiều

```
void XuatMaTran(int a[][MAXC], int m, int n)
{
        int i, j;
        for (i=0; i<m; i++)
        {
                for (j=0; j<n; j++)
                        printf("%d ", a[i][j]);

                printf("\n");
        }
}
```

Truyền mảng cho hàm (sử dụng hàm)

- Lời gọi hàm

```
void NhapMaTran(int a[][100], int &m, int &n);
void XuatMaTran(int a[][100], int m, int n);
void main()
{
        int a[50][100], m, n;
        NhapMaTran(a, m, n);
        XuatMaTran(a, m, n);
}
```

Ví dụ

- Viết chương trình nhập 2 ma trận a và b gồm m dòng và n cột, tính ma trận c=a+b theo công thức `c[i][j]=a[i][j]+b[i][j]`.

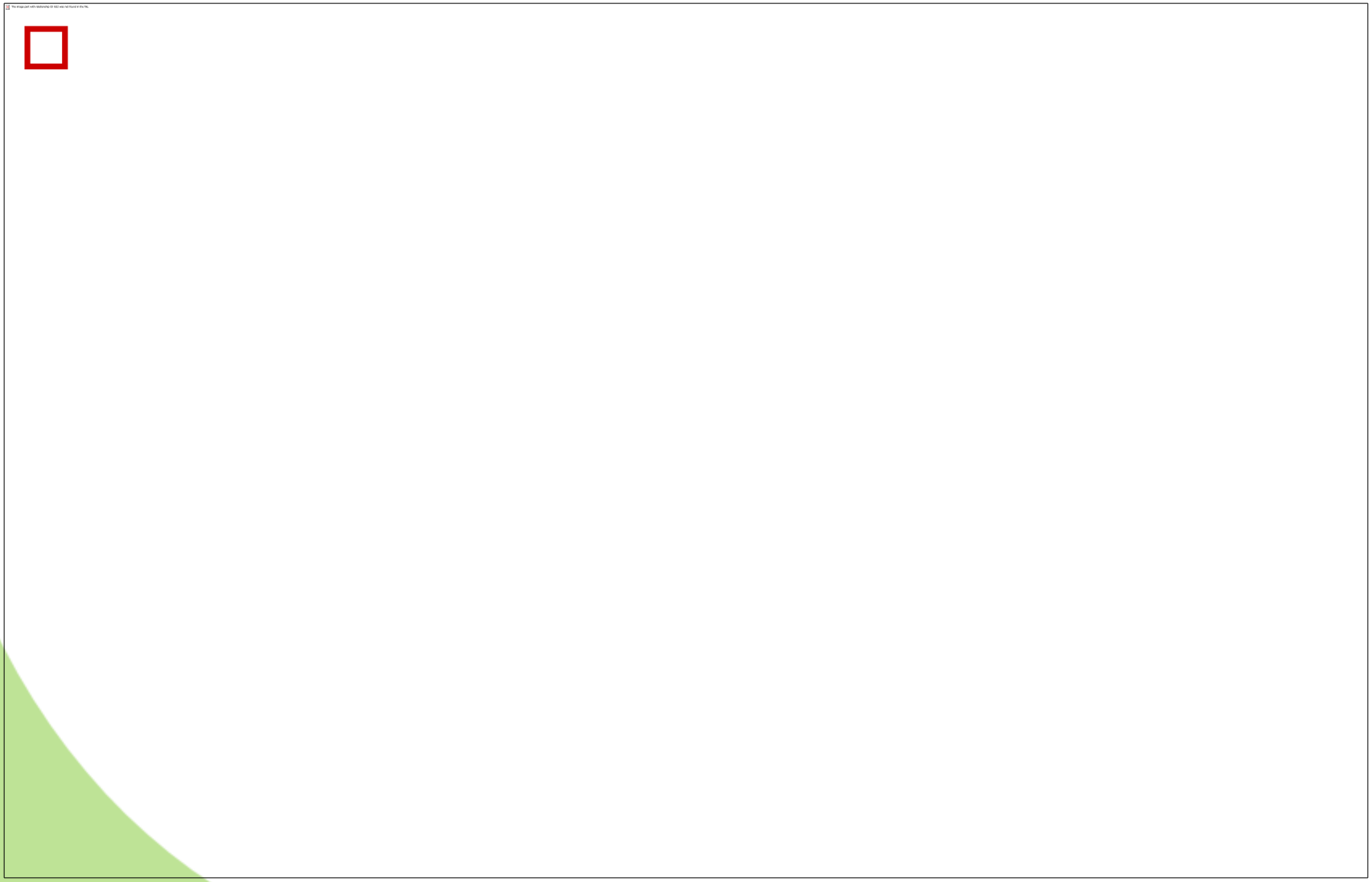
  In c ra màn hình.
- Ý tưởng
  - Viết các hàm: nhập, cộng, in ma trận
  - Nhập số dòng/số cột cho ma trận
  - Nhập giá trị cho a và b
  - Thực hiện cộng
  - In kết quả ra màn hình

Ví dụ (tt)

Hàm cộng ma trận

```
/* Cong 2 ma tran A & B ket qua la ma tran C*/
void  CongMaTran(int  a[][MAXC],int  b[][MAXC],int  M,int
N,int c[][MAXC])
{
        int i,j;
        for(i=0;i<M;i++)
                for(j=0; j<N; j++)
                        c[i][j]=a[i][j]+b[i][j];
}
int main() //Chương trình chính
{
        /*Nhập m,n */
        NhapMaTran(a, m, n); NhapMaTran(b, m, n);
        CongMaTran(a,b,m,n,c); XuatMaTran(c,m,n);
}
```

Bài tập thực hành



Bài tập thực hành

3. Nhập ma trận vuông a cấp n

   a. Tính tổng các phần tử dương trên đường chéo chính.

   b. Tính tổng các phần tử là số nguyên tố trong ma trận tam giác trên.

   c. Đếm số phần tử là số chính phương trong ma trận tam giác dưới.

4. Nhập vào ma trận a (m dòng, n cột), đưa ra các phần tử yên ngựa trong a.

   Phần tử yên ngựa là phần tử nhỏ nhất trên dòng nhưng lớn nhất trên cột hoặc nhỏ nhất trên cột nhưng lớn nhất trên dòng.

CƠ SỞ LẬP TRÌNH

KIỂU CON TRỎ

Nội dung

- Con trỏ và địa chỉ
- Khai báo con trỏ
- Con trỏ và mảng một chiều
- Con trỏ và mảng nhiều chiều
- Mảng các con trỏ
- Con trỏ hàm
- Cấp phát bộ nhớ động

1. Con trỏ và địa chỉ

- Ví dụ:

```
float a=10.12;
```

  - Xác định một biến có tên a có kiểu float và có giá trị 10.12.
  - Máy cấp phát cho x một vùng nhớ gồm 4 byte liên tiếp.
  - Địa chỉ của biến là số thứ tự của byte đầu tiên
- Có nhiều kiểu địa chỉ khác nhau tương ứng với các kiểu biến khác nhau.

Con trỏ và địa chỉ

- Con trỏ là một biến dùng để chứa địa chỉ. Có nhiều kiểu con trỏ tương ứng với nhiều loại địa chỉ.
  - Ví dụ:
    - Con trỏ kiểu int chứa địa chỉ các biến kiểu int…
- ***a** là giá trị được lưu trong bộ nhớ có địa chỉ **a**
- &a là địa chỉ bộ nhớ chứa giá trị **a**

2. Khai báo con trỏ

- Khai báo trực tiếp

  <kiểu dữ liệu> *<tên biến con trỏ>;

  Trong đó: * là toán tử con trỏ

  - Ví dụ:

```
int *p1,m,n;
p1=&n;
*p1=10; /* ô nhớ do con trỏ p1 trỏ đến được
           gán giá trị 10 */
```

- **Chú ý**: Khi gán địa chỉ của 1 biến cho 1 biến con trỏ, mọi sự thay đổi trên nội dung ô nhớ con trỏ chỉ tới sẽ làm giá trị của biến thay đổi theo.
  - Ví dụ: `int *p2,a=10; p2=&a; *p2=*p2+3;`

    Khi đó a sẽ có giá trị 13.

Khai báo con trỏ (tt)

- Khai báo gián tiếp

```
typedef <kiểu dữ liệu> *<tên kiểu con trỏ>;

<tên kiểu con trỏ> <tên biến con trỏ>;
```

  - Ví dụ

```
typedef int *pint;
int *p1;
pint p2, p3;
```

- Kích thước của con trỏ
  - Con trỏ chỉ lưu địa chỉ nên kích thước của mọi con trỏ là như nhau:
    - Môi trường MD-DOS (16 bit): 2 bytes
    - Môi trường Windows (32 bit): 4 bytes

Con trỏ NULL

- Khái niệm
  - Con trỏ NULL là con trỏ không trỏ vào đâu cả.
  - Khác với con trỏ chưa được khởi tạo.

```
int n;
int *p1 = &n;
int *p2;     // unreferenced local varialbe
int *p3 = NULL;
```

NULL

Khởi tạo kiểu con trỏ

- Khởi tạo

```
<tên biến con trỏ> = &<tên biến>;
```

  - Khi mới khai báo, biến con trỏ được đặt ở địa chỉ nào đó (không biết trước).
    - ➔ chứa giá trị không xác định
    - ➔ trỏ đến vùng nhớ không biết trước.
  - Đặt địa chỉ của biến vào con trỏ (toán tử &)

- Ví dụ

```
int a, b;
int *pa = &a, *pb;
pb = &b;
```

Sử dụng con trỏ

- Truy xuất đến ô nhớ mà con trỏ trỏ đến
  - Con trỏ chứa một số nguyên chỉ địa chỉ.
  - Sử dụng toán tử *.
- Ví dụ

```
int n=10;
int *p;
printf("\nDia chi cua n: %p",&n);
printf("\nGia tri cua n: %d",n);
p=&n; //Con tro p tro toi n
printf("\nDia chi cua con tro: %p",&p);
printf("\nGia tri cua con tro: %p",p);
printf("\nGia tri duoc tro toi la: %d",*p);
```

Sử dụng con trỏ (tt)

- Sử dụng tên con trỏ
  - Giá trị của con trỏ (địa chỉ của biến) được sử dụng trong biểu thức
  - Nếu tên con trỏ ở bên trái toán tử gán thì giá trị của biểu thức bên phải phải là địa chỉ.
  - Ví dụ: `float a,*p,*q;  p=&a;  q=p;`
    - ➔ Kết quả con trỏ q chứa địa chỉ của biến a
- Sử dụng dạng khai báo
  - Ví dụ:

```
float x,*px;
px=&x; //px trỏ tới x
```

  - Nếu con trỏ px trỏ tới biến x thì cách viết x và *px là tương đương trong mọi ngữ cảnh.

Các cách truyền đối số

- Truyền giá trị (tham trị)

```
#include <stdio.h>

void hoanvi(int x, int y);

main()
{
        int a = 5, b = 6;
        hoanvi(a, b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int x, int y)
{
        int t = x; x = y; y = t;
}
```

Các cách truyền đối số

- Truyền địa chỉ (con trỏ)

```
#include <stdio.h>

void hoanvi(int *x, int *y);

main()
{
        int a = 2912, b = 1706;
        hoanvi(&a, &b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int *x, int *y)
{
        int t = *x; *x = *y; *y = t;
}
```

Các cách truyền đối số

- Truyền tham chiếu (C++)

```
#include <stdio.h>

void hoanvi(int &x, int &y);

main()
{
        int a = 2912, b = 1706;
        hoanvi(a, b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int &x, int &y)
{
        int t = x; x = y; y = t;
}
```

Một số lưu ý

Một số lưu ý

- Nắm rõ quy tắc sau, ví dụ int a, *pa = &a;
  - *pa và a đều chỉ nội dung của biến a.
  - pa và &a đều chỉ địa chỉ của biến a.
- Không nên sử dụng con trỏ khi chưa được khởi tạo. Kết quả sẽ không lường trước được.
- Ví dụ:

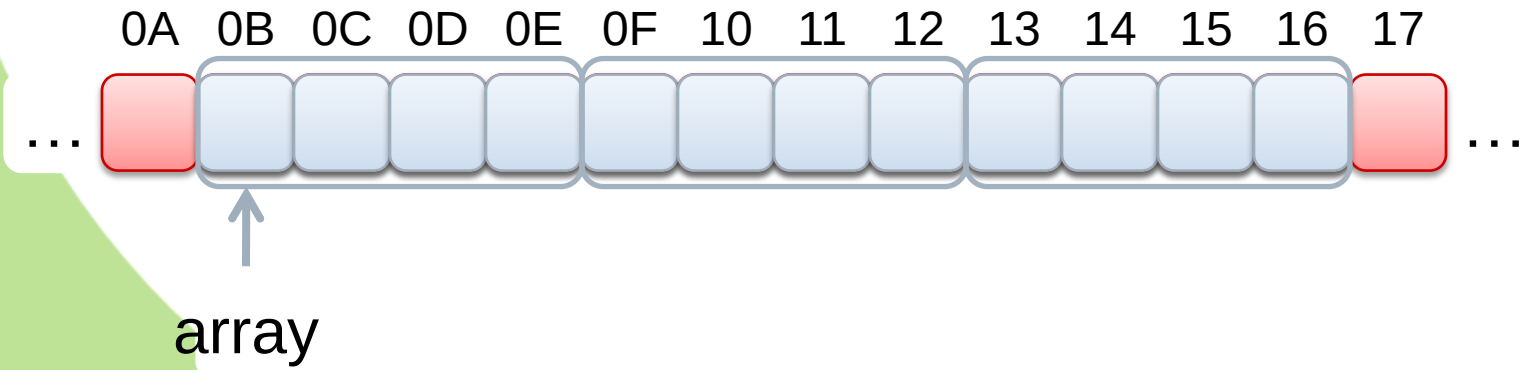
```
int *pa; *pa = 1904;
```

3. Con trỏ và mảng một chiều

Mảng một chiều

```
int array[3];
```

- Tên mảng array là một hằng con trỏ
  ➔ không thể thay đổi giá trị của hằng này.
- array là địa chỉ đầu tiên của mảng
  ➔array == &array[0]

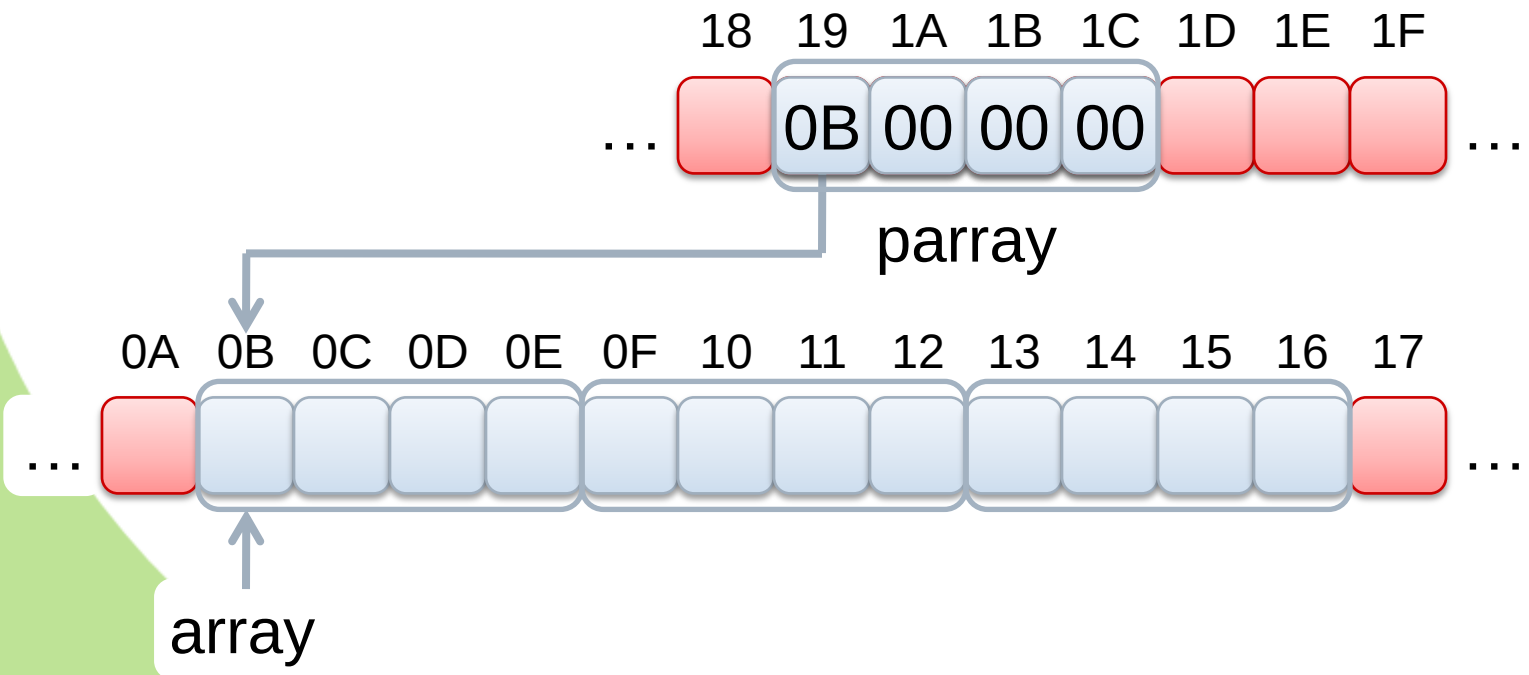


Con trỏ và mảng một chiều

Con trỏ đến mảng một chiều

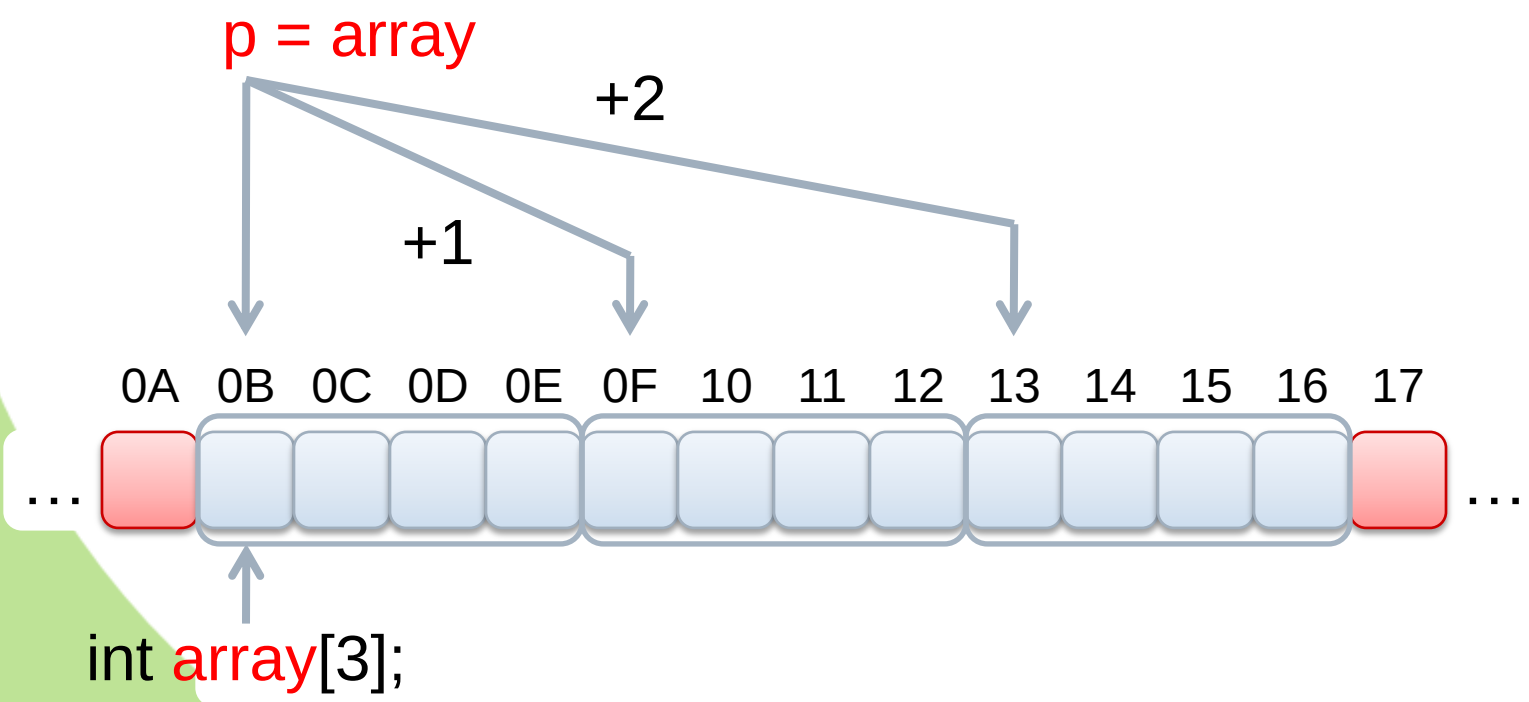
```
int array[3], *parray;

parray = array;          // Cách 1
parray = &array[0];      // Cách 2
```



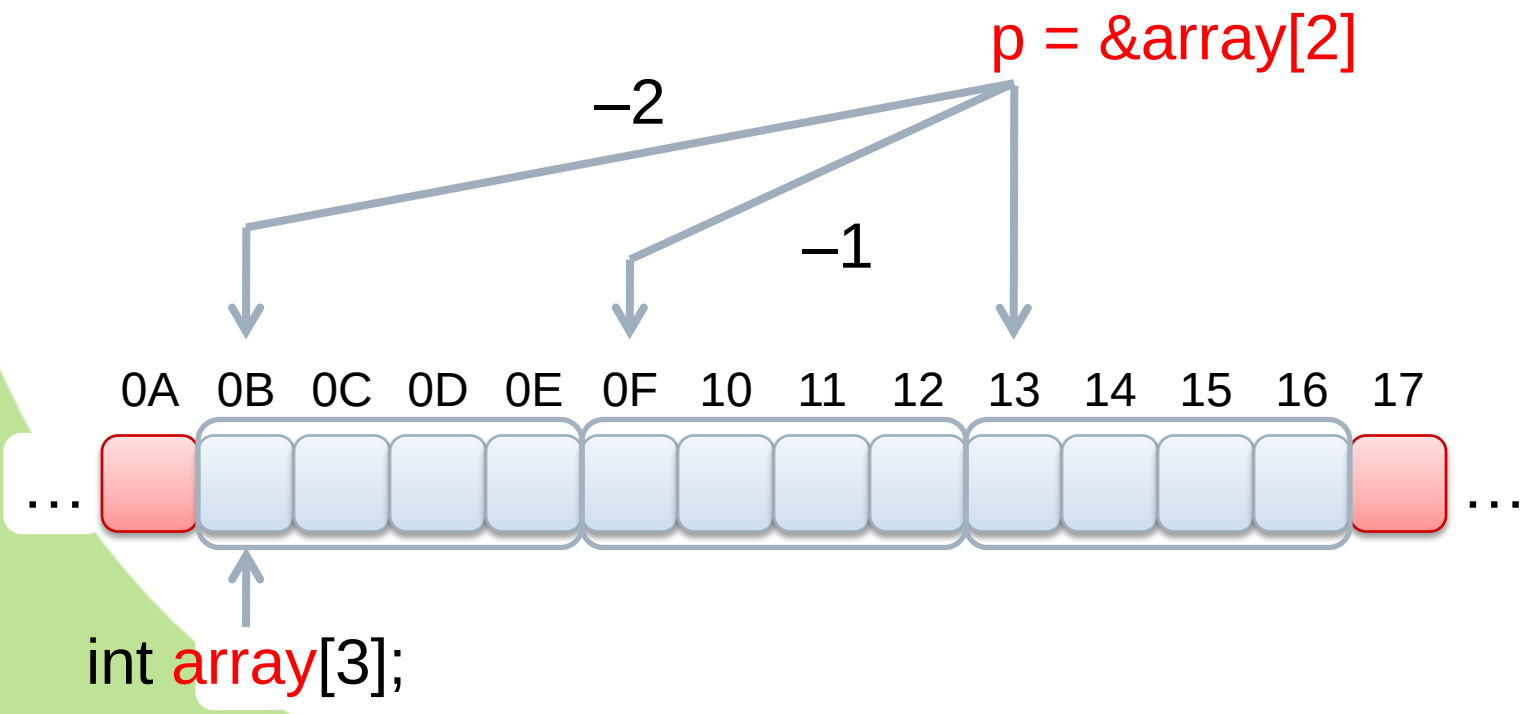
Phép toán số học trên con trỏ

- Phép cộng (tăng)
  - + n ⇔ + n * sizeof(<kiểu dữ liệu>)
  - Có thể sử dụng toán tử gộp += hoặc ++



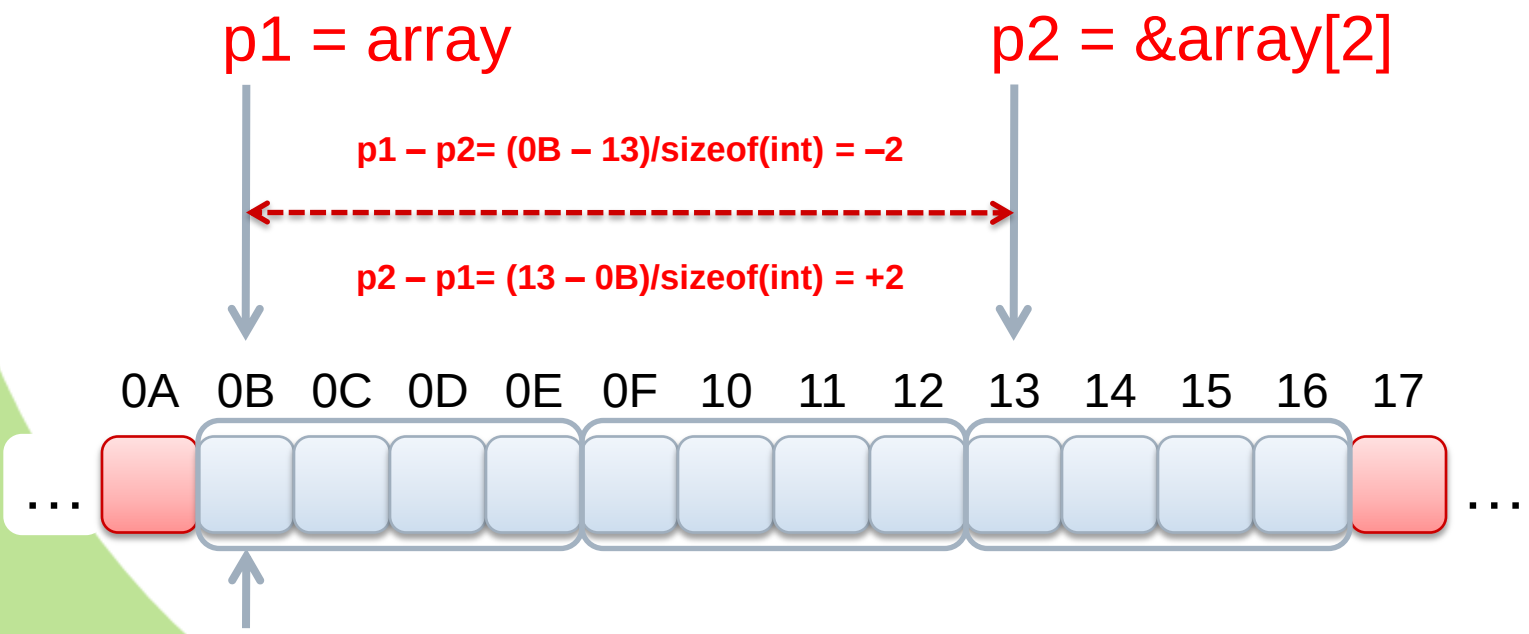
Phép toán số học trên con trỏ

- Phép trừ (giảm)
  - – n ⇔ – n * sizeof(<kiểu dữ liệu>)
  - Có thể sử dụng toán tử gộp –= hoặc – –



Phép toán số học trên con trỏ

- Phép toán tính khoảng cách giữa 2 con trỏ
  - <kiểu dữ liệu> *p1, *p2;
  - p1 – p2 cho ta khoảng cách (theo số phần tử) giữa hai con trỏ (cùng kiểu)

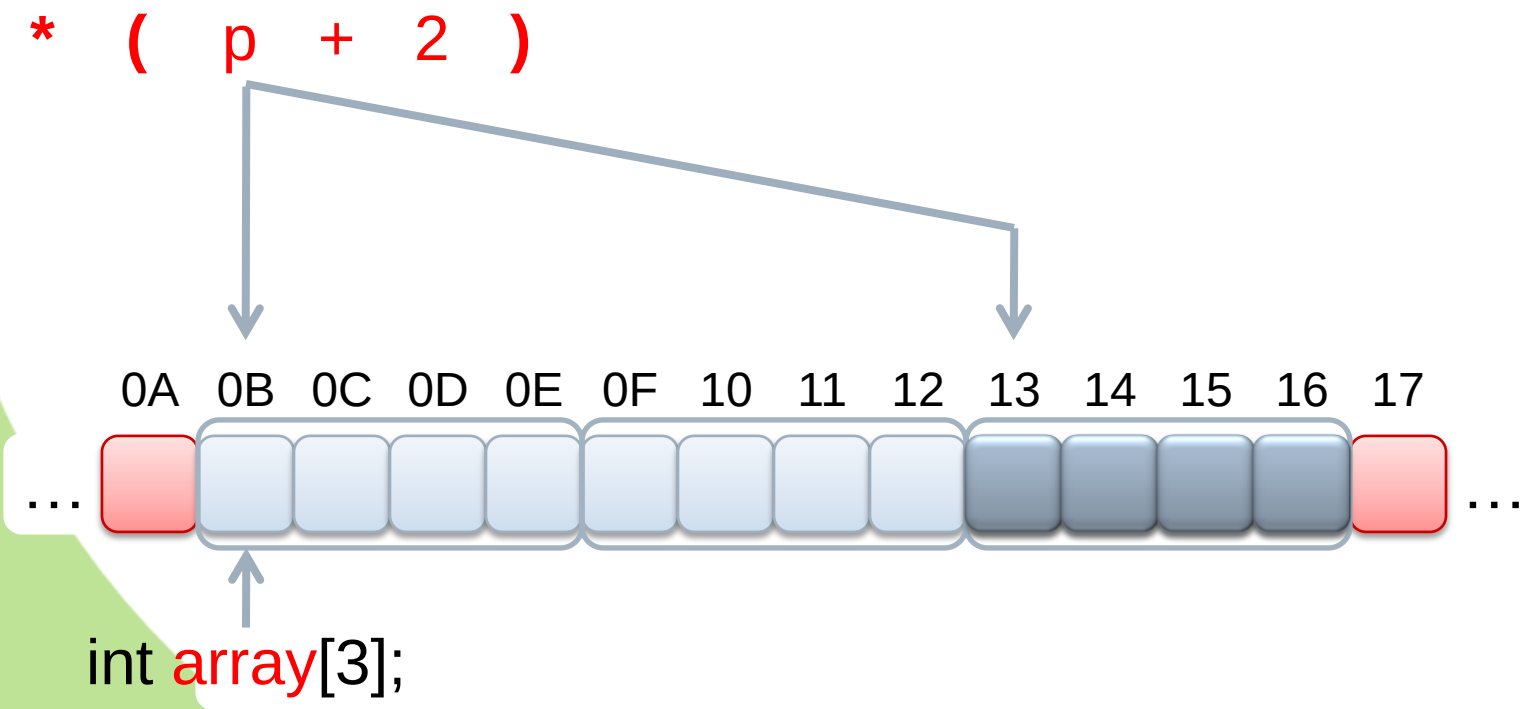


Phép toán số học trên con trỏ

- Các phép toán khác
  - Phép so sánh: So sánh địa chỉ giữa hai con trỏ (thứ tự ô nhớ)
    - == !=
    - > >=
    - < <=
  - Không thể thực hiện các phép toán: * / %

Con trỏ và mảng một chiều

- Truy xuất đến phần tử thứ n của mảng (không sử dụng biến mảng)
  - array[n] == p[n] == *(p + n)



Con trỏ và mảng một chiều (tt)

- Ví dụ: Nhập vào một dãy các số nguyên, tính tổng các số dương trong dãy.

```
void nhapmang(int *a,int &n)
{ int i;
  printf("Nhap so phan tu cua mang: ");
  scanf("%d",&n);
  for(i=0;i<n;i++)
    {    printf("a[%d]= ",i);
         scanf("%d",a+i);    }
}
void xuatmang(int *a,int n)
{    int i;
     printf("\nMang vua nhap la: ");
     for(i=0;i<n;i++)
     printf("%d ",*(a+i));
}
```

Con trỏ và mảng một chiều (tt)

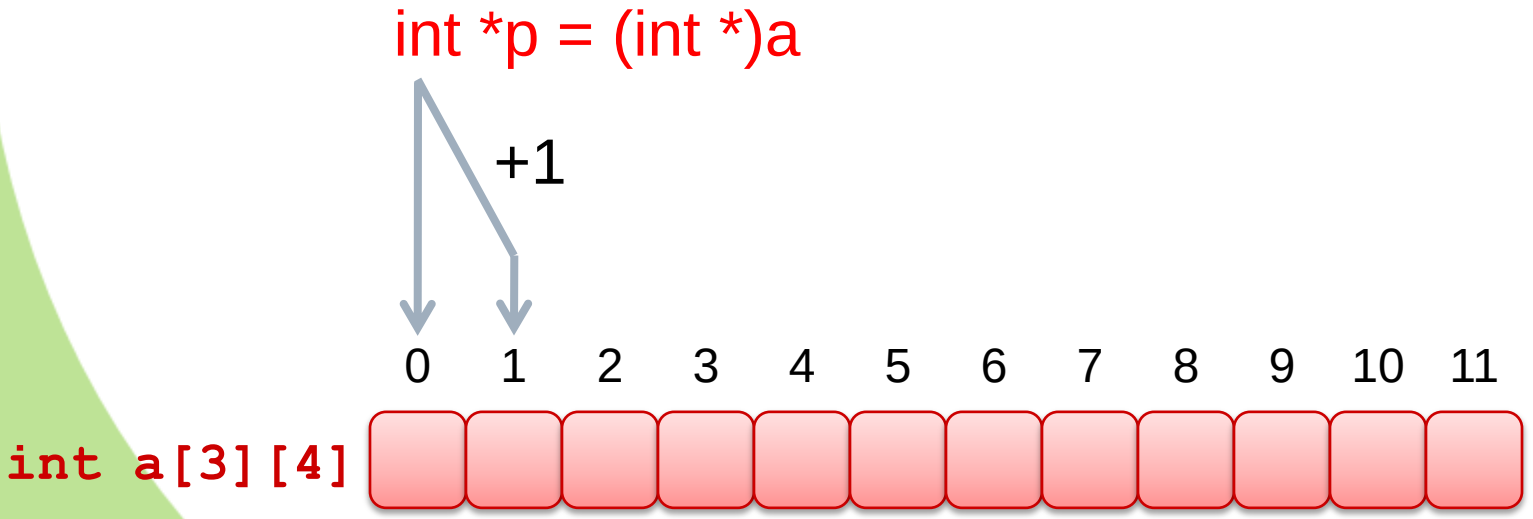
```
long tong(int *a,int n)
{
     int i;
     long t; t=0;
     for(i=0;i<n;i++)
       if (*(a+i)>0) t=t+*(a+i);
     return t;
}
int main()
{
       int a[20],n,i;
       nhapmang(a,n);
       xuatmang(a,n);
       printf("\nTong cac phan tu duong la: %ld",
                      tong(a,n));
       getch();
}
```


Con trỏ và mảng một chiều (tt)

- Lưu ý
  - Không thực hiện các phép toán nhân, chia, lấy phần dư.
  - Tăng/giảm con trỏ n đơn vị có nghĩa là tăng/giảm giá trị của nó n*sizeof(<kiểu dữ liệu mà nó trỏ đến>)
  - Không thể tăng/giảm biến mảng. Hãy gán một con trỏ đến địa chỉ đầu của mảng và tăng/giảm nó.
  - Đối số mảng một chiều truyền cho hàm là địa chỉ phần tử đầu tiên của mảng.

4. Con trỏ và mảng 2 chiều

- Hướng tiếp cận
  - Các phần tử tạo thành mảng 1 chiều
  - Sử dụng con trỏ int * để duyệt mảng 1 chiều



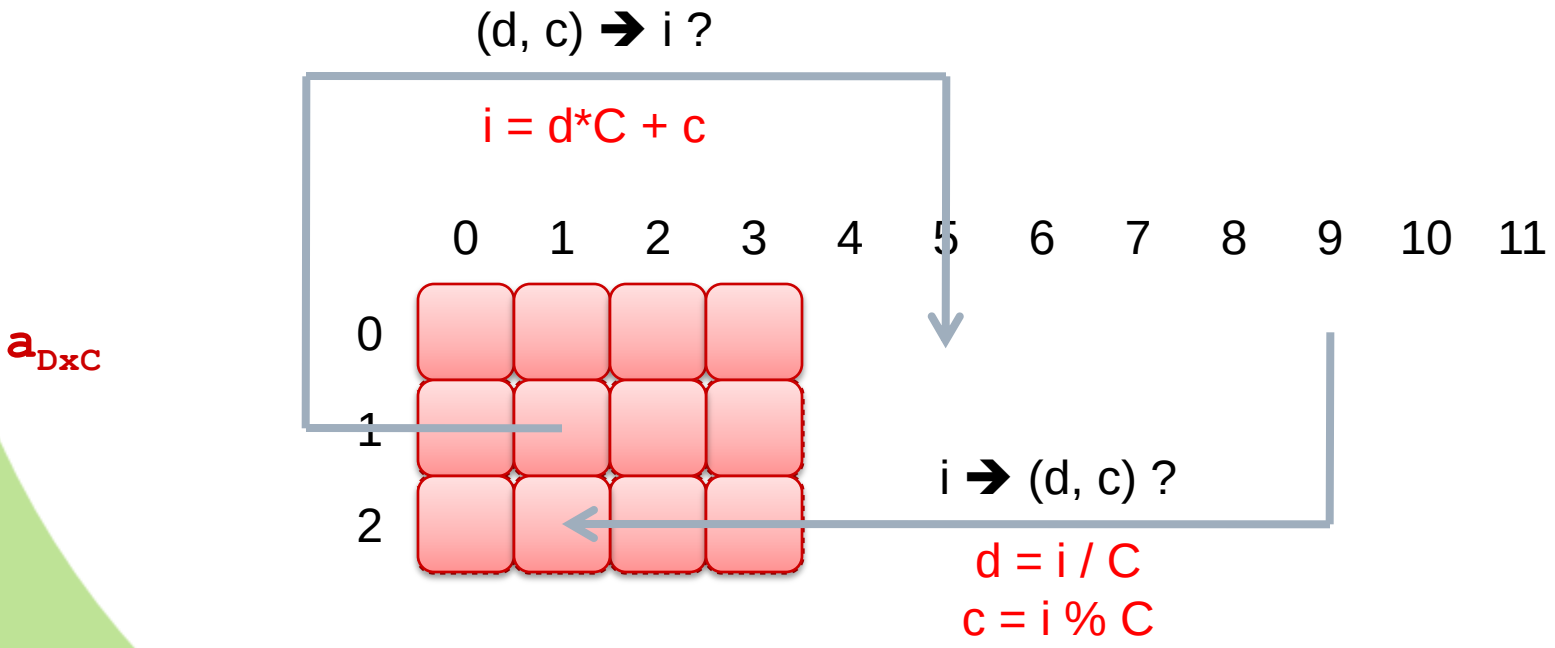
Con trỏ và mảng 2 chiều (tt)

- Nhập / Xuất theo chỉ số mảng 1 chiều

```
main()
{
        int a[D][C], i,m,n;
        int *p = (int *)a;
        printf("Nhap so dong, so cot: ");
        scanf("%d%d",&m,&n);
        for (i = 0; i < m*n; i++)
        {       printf("Nhap phan tu thu %d: ", i);
                scanf("%d", p + i);
        }
        for (i = 0; i < m*n; i++)
                {  printf("%5d ", *(p + i));
                   if ((i+1)%n==0) printf("\n");
                }
        getch();
}
```

Con trỏ và mảng 2 chiều (tt)

- Liên hệ giữa chỉ số mảng 1 chiều và chỉ số mảng 2 chiều?



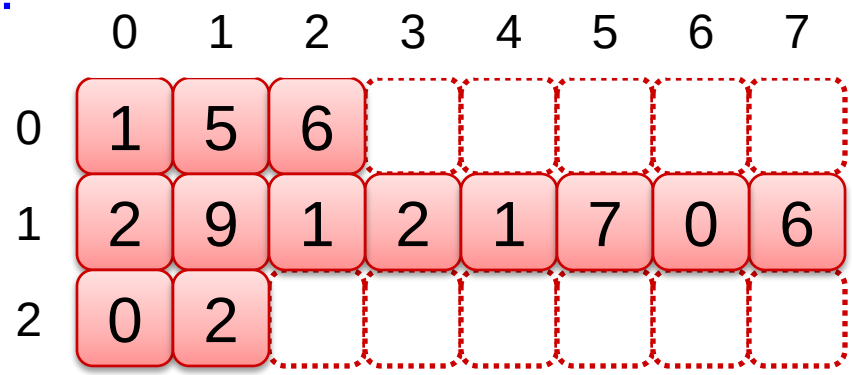
Con trỏ và mảng 2 chiều (tt)

- Nhập / Xuất theo chỉ số mảng 2 chiều

```
#define D 100
#define C 100
int a[D][C], i, d, c,m,n;
int *p = (int *)a;
main()
{
 printf("Nhap so dong, so cot: "); scanf("%d%d",&m,&n);
 for (i = 0; i < m*n; i++)
 {    printf("Nhap a[%d][%d]: ", i / n, i % n);
      scanf("%d", p + i);  }
 for (d = 0; d < m; d++)
 {    for (c = 0; c < n; c++)
      printf("%5d", *(p + d * n + c));// hoac *p++
      printf("\n");
 }
}
```

5. Mảng con trỏ

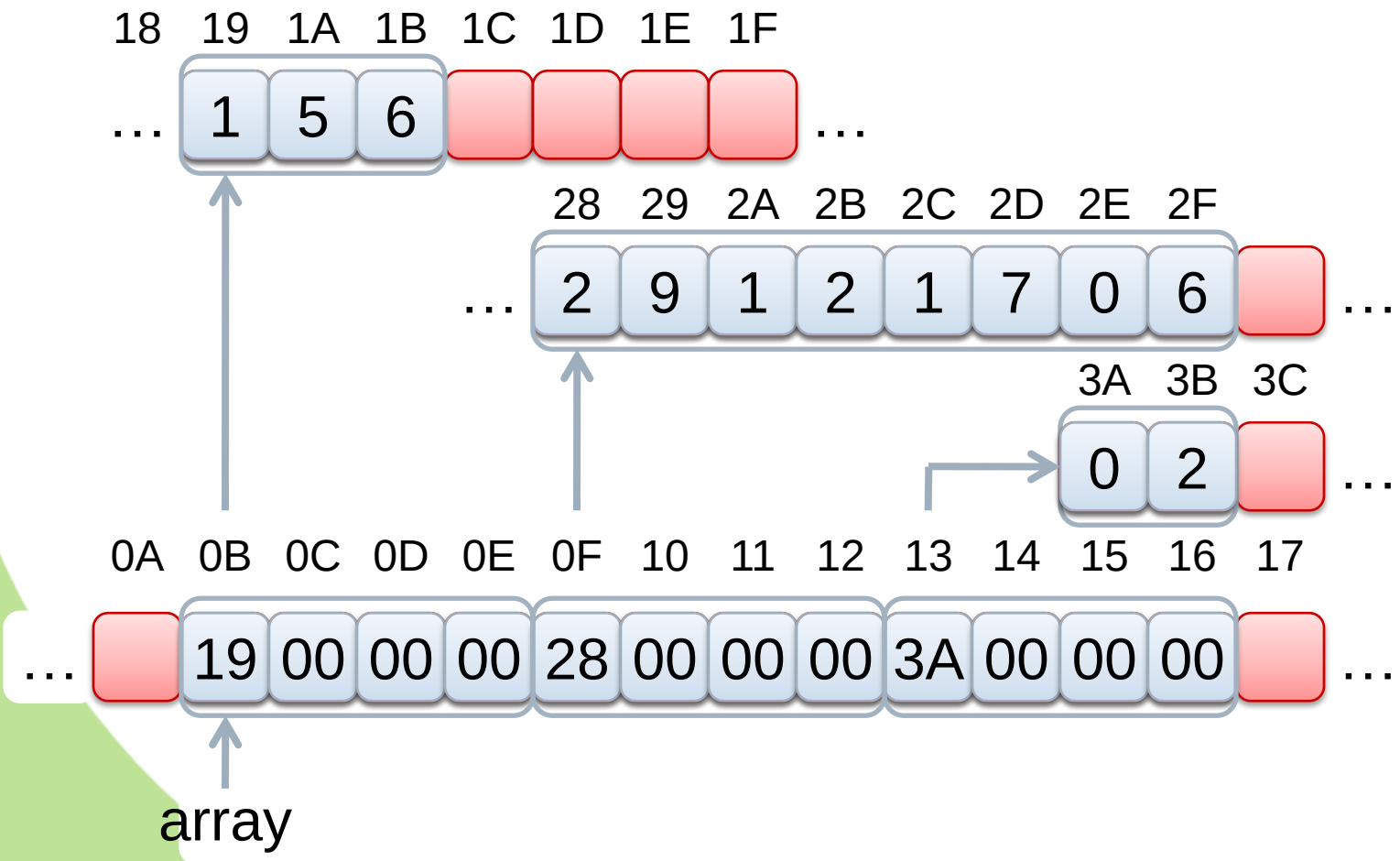
- Bài toán
  - Sử dụng cấu trúc dữ liệu nào để lưu trữ thông tin sau?



- Giải pháp?
  - Cách 1: Mảng 2 chiều 3x8 (tốn bộ nhớ)

Mảng con trỏ (tt)

- Cách 2: Mảng 1 chiều các con trỏ



Mảng con trỏ (tt)

- Cú pháp

```
<Kiểu dữ liệu> *Tên mảng con trỏ [số phần tử];
```

- Ví dụ: Chương trình giải bài toán mở đầu

```
int a[200],n,m;
int sl[20]; //Chua so luong phan tu cua moi mang
int *p[20]; //Mang cac con tro
void nhap()
{ int i,j=0;
  printf("Nhap so mang: "); scanf("%d",&m);
  for (i=0;i<m;i++)
  {  printf("Nhap so phan tu cua mang: "); scanf("%d",&n);
     p[i]=a+j;
     for(int k=0;k<n;k++)
         { printf("a[%d]= ",k);
           scanf("%d",a+j);    j++; }
     sl[i]=n;
  }
}
```

Mảng con trỏ (tt)

```
void xuat()
{     int i,j;
      for(i=0;i<m;i++)
          {
              for (j=0;j<sl[i];j++)
                  printf("%d ",*((p[i])+j));
              printf("\n");
          }
}
main()
{
       nhap();
       xuat();
       getch();
}
```

6. Con trỏ hàm (tự đọc)

- Khai báo tường minh

```
<kiểu trả về> (* <tên biến con trỏ>)(ds tham số);
```

- Ví dụ

```
// Con trỏ đến hàm nhận đối số int, trả về int
int (*ptof1)(int x);

// Con trỏ đến hàm nhận 2 đối số double, không trả về
void (*ptof2)(double x, double y);

// Con trỏ đến hàm nhận đối số mảng, trả về char
char (*ptof3)(char *p[]);

// Con trỏ đến không nhận đối số và không trả về
void (*ptof4)();
```

Con trỏ hàm (tt)

- Khai báo không tường minh (thông qua kiểu)

```
typedef <kiểu trả về> (* <tên kiểu>)(ds tham số);

<tên kiểu> <tên biến con trỏ>;
```

- Ví dụ

```
int (*pt1)(int, int);    // Tường minh

typedef int (*PhepToan)(int, int);

PhepToan pt2, pt3;       // Không tường minh
```

Con trỏ hàm (tt)

- Gán giá trị cho con trỏ hàm

```
<biến con trỏ hàm> = <tên hàm>;

<biến con trỏ hàm> = &<tên hàm>;
```

  - Hàm được gán phải cùng dạng (vào, ra)

- Ví dụ

```
int Cong(int x, int y);                 // Hàm
int Tru(int x, int y);                  // Hàm
int (*tinhtoan)(int x, int y);          // Con trỏ hàm

tinhtoan = Cong;  // Dạng ngắn gọn
tinhtoan = &Tru;  // Dạng sử dụng địa chỉ
tinhtoan = NULL;  // Không trỏ đến đâu cả
```

Con trỏ hàm (tt)

- So sánh con trỏ hàm

```
if  (tinhtoan != NULL)
{
      if (tinhtoan == &Cong)
            printf("Con trỏ đến hàm Cong.");
      else
            if (tinhtoan == &Tru)
                  printf("Con trỏ đến hàm Tru.");
            else
                  printf("Con trỏ đến hàm khác.");
}
else
      printf("Con trỏ chưa được khởi tạo!");
```

Con trỏ hàm (tt)

- Gọi hàm thông qua con trỏ hàm
  - Sử dụng toán tử lấy nội dung "*" (chính quy) nhưng trường hợp này có thể bỏ

```
int Cong(int x, int y);
int Tru(int x, int y);

int (*tinhtoan)(int, int);

tinhtoan = Cong;
int kq1 = (*tinhtoan)(1, 2);   // Chính quy
int kq2 = tinhtoan(1, 2);      // Ngắn gọn
```

Con trỏ hàm (tt)

- Truyền tham số là con trỏ hàm

```
int Cong(int x, int y);
int Tru(int x, int y);
int TinhToan(int x, int y, int (*pheptoan)(int, int))
{
        int kq = (*pheptoan)(x, y);    // Gọi hàm
        return kq;
}

main()
{
        int (*pheptoan)(int, int) = &Cong;
        int kq1 = TinhToan(1, 2, pheptoan);
        int kq2 = TinhToan(1, 2, &Tru);
}
```

Con trỏ hàm (tt)

- Trả về con trỏ hàm

```
int (*LayPhepToan(char code))(int, int)
{
        if (code == '+')
                return &Cong;
        return &Tru;
}

main()
{
        int (*pheptoan)(int, int) = NULL;
        pheptoan = LayPhepToan('+');
        int kq2 = pheptoan(1, 2, &Tru);
}
```

Con trỏ hàm (tt)

- Trả về con trỏ hàm (khai báo kiểu)

```
typedef (*PhepToan)(int, int);
PhepToan LayPhepToan(char code)
{
        if (code == '+')
                return &Cong;
        return &Tru;
}

main()
{
        PhepToan pheptoan = NULL;
        pheptoan = LayPhepToan('+');
        int kq2 = pheptoan(1, 2, &Tru);
}
```

Con trỏ hàm (tt)

- Mảng con trỏ hàm

```
typedef (*PhepToan)(int, int);
main()
{
        int (*array1[2])(int, int);    // tường minh
        PhepToan array2[2];            // kô tường minh

        array1[0] = array2[1] = &Cong;
        array1[1] = array2[0] = &Tru;

        printf("%d\n", (*array1[0])(1, 2));
        printf("%d\n", array1[1](1, 2));
        printf("%d\n", array2[0](1, 2));
        printf("%d\n", array2[1](1, 2));
}
```

Con trỏ hàm (tt)

- Lưu ý
  - Không được quên dấu () khi khai báo con trỏ hàm
    - int (*PhepToan)(int x, int y);
    - int *PhepToan(int x, int y);
  - Có thể bỏ tên biến tham số trong khai báo con trỏ hàm
    - int (*PhepToan)(int x, int y);
    - int (*PhepToan)(int, int);

7. Cấp phát bộ nhớ động

- Biến tĩnh:
  - Ví dụ: int, float, mảng…
  - Là các biến được xác định khi mô tả kiểu, địa chỉ của các biến được xác định ngay khi thực hiện chương trình;
  - Thời gian tồn tại cùng với thời gian tồn tại của khối chương trình chứa khai báo
- Biến động:
  - Được tạo ra lúc chạy chương trình theo yêu cầu
  - Biến động không có tên
  - Truy nhập thông qua các biến con trỏ (là biến tĩnh, chứa địa chỉ của các biến động)

Cấp phát bộ nhớ động

- Cấp phát bộ nhớ (trong thư viện stdlib.h,alloc.h)

void *malloc(Số ô nhớ cần cấp phát)

Cấp phát vùng nhớ có kích thước được chỉ ra.

void *calloc(n,sizeof(object))

Cấp phát vùng nhớ có kích thước là n*sizeof(object)

Dùng khi cấp phát bộ nhớ cho kiểu dữ liệu không phải kiểu cơ sở

- Ví dụ:

```
int a, *pa, *pb;
pa = (int*)malloc(sizeof(int));
pb= (int*)calloc(10, sizeof(int));
```

- **Lưu ý**: Khi sử dụng hàm malloc() hay calloc(), ta phải ép kiểu vì nguyên mẫu các hàm này trả về con trỏ kiểu void.

Cấp phát bộ nhớ động (tt)

Cấp phát lại vùng nhớ cho biến con trỏ

```
void *realloc(void *poiter, kích thước mới)
```

- *Ý nghĩa*:
  - Cấp phát lại 1 vùng nhớ cho con trỏ *pointer* quản lý, vùng nhớ này có kích thước mới là kích thước mới được chỉ ra; khi cấp phát lại thì nội dung vùng nhớ trước đó vẫn tồn tại.
  - Kết quả trả về của hàm là địa chỉ đầu tiên của vùng nhớ mới. Địa chỉ này có thể khác với địa chỉ được chỉ ra khi cấp phát ban đầu.
- *Ví dụ*:

```
int a, *pa;
pa=(int*)malloc(sizeof(int));
pa = realloc(pa, 6); /* Cấp phát lại vùng nhớ có kích thước 6 byte*/
```

Cấp phát bộ nhớ động (tt)

- Giải phóng vùng nhớ đã cấp phát

```
void free(void *pointer)
```

  - *Ý nghĩa*: Giải phóng vùng nhớ được quản lý bởi con trỏ pointer.
  - *Ví dụ*:

Ở ví dụ trên, sau khi thực hiện xong, ta giải phóng vùng nhớ cho 2 biến con trỏ pa và pb:

```
free(pa);
free(pb);
```

Ví dụ cấp phát bộ nhớ động

- Nhập từ bàn phím một số nguyên n, đưa ra màn hình n số nguyên tố đầu tiên

```
int ngto(long a)
{ //Kiem tra so nguyen to
}
int main()
{   long *prime=NULL, x=0;
    int i=0, found=0, n=0,j;
    printf("Ban muon tim bao nhieu so nguyen to? ");
    scanf("%d",&n);
    prime=(long *)malloc(n*sizeof(long));
    if (prime==NULL)
      {
            printf("Khong du bo nho!");
            return 0;
      }
```

Ví dụ cấp phát bộ nhớ động (tt)

```
*prime=2;  //Cac so nguyen to dau tien da biet
*(prime+1)=3;
*(prime+2)=5;
x=5; j=3;
do
      {
          x+=2; //Gia tri tiep theo de kiem tra
          if (ngto(x))
             {          *(prime+j)=x;
                        j++;
             }
      }
 while (j<n);
printf("%d so nguyen to dau tien la: \n",n);
for (i=0;i<n;i++) printf("%ld ",*(prime+i));
getch();
}
```


Bài tập thực hành

- **Bài 1**. Viết chương trình khai báo mảng hai chiều có 12x12 phần tử kiểu char. Gán ký tự 'X' cho mọi phần tử của mảng này. Sử dụng con trỏ đến mảng để in giá trị các phần tử mảng lên màn hình ở dạng lưới.
- **Bài 2**. In giá trị của con trỏ và giá trị của biến mà nó trỏ tới.
- **Bài 3**. Sử dụng con trỏ để làm lại các bài tập về mảng một chiều.
  - Nhập, xuất mảng
  - Tính tổng các phần tử âm, nguyên tố, chính phương…

Bài tập thực hành (tt)

- **Bài 4.** Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 9$) , sắp xếp các chữ số của n theo thứ tự tăng dần.
  Ví dụ:
  - Nhập n = 1536
  - Kết quả sau khi sắp xếp: 1356.
- **Bài 5.** Sử dụng con trỏ để làm lại các bài tập về mảng hai chiều.
  - Nhập, xuất mảng
  - Tính tổng các phần tử trên đường chéo chính…
  - Đưa ra các số nguyên tố, chính phương, số đẹp trong mảng

CƠ SỞ LẬP TRÌNH

KIỂU CHUỖI KÝ TỰ

Nội dung

- Khai báo
- Các thao tác trên chuỗi ký tự
- Mảng chuỗi ký tự
- Một số hàm xử lý chuỗi ký tự

1. Khai báo

- Khái niệm
  - Chuỗi ký tự trong C được xây dựng như mảng một chiều các ký tự
  - Chuỗi ký tự kết thúc bằng ký tự '\0' (ký tự NULL trong bảng mã ASCII)
    - ➔ Độ dài tối đa của chuỗi= kích thước mảng – 1
    - ➔ Khai báo nên dành ra 1 ô nhớ để chứa ký tự '\0'
- Ví dụ

```
char line[80];     // Dài tối đa 79 ký tự
char hoten[30];    // Dài tối đa 29 ký tự
```

Khai báo có khởi tạo giá trị

- Độ dài cụ thể

```
char string[40]="Ngon ngu C";
```

  - Khởi tạo Chuỗi ký tự có độ dài tối đa 39 ký tự với giá trị ban đầu là chuỗi "Ngon ngu C"

- Tự xác định độ dài

```
char str[]="Ngon ngu C";
```

  - Chương trình dịch tự bố trí một mảng để chứa dãy ký tự và 1 ô chứa kí hiệu '\0'

- Chú ý:
  - Khai báo Chuỗi ký tự với con trỏ

```
char *message;
message="Xin chao!";
```

2. Các thao tác trên chuỗi ký tự

- Nhập chuỗi từ bàn phím
- Xuất chuỗi ra màn hình
- Xác định độ dài chuỗi
- Ghép chuỗi
- Sao chép chuỗi
- So sánh chuỗi
- Tìm kiếm ký tự
- …

a) Nhập chuỗi từ bàn phím

- Sử dụng hàm **scanf** với đặc tả "%s"

  ```
  scanf("%s",str);
  ```

  - Chỉ nhận các ký tự từ bàn phím đến khi gặp ký tự dấu cách, tab, ký tự xuống dòng.
  - chuỗi nhận được không bao gồm dấu cách

- Ví dụ:

```
char monhoc[50];
printf("Nhap mot xau ki tu: ");
scanf("%s", monhoc);
printf("Xau nhan duoc la: %s", monhoc);
```

```
Nhap mot chuoi: Ngon ngu lap trinh C
Chuoi nhan duoc la: Ngon_
```

a) Nhập chuỗi từ bàn phím (tt)

- Sử dụng hàm **gets**

```
gets(str);
```

  - Nhận các ký tự từ bàn phím đến khi gặp ký tự xuống dòng.
  - chuỗi nhận được là những gì người dùng nhập (trừ ký tự xuống dòng).

- Ví dụ

```
char monhoc[50];
printf("Nhap mot chuoi: ");
gets(monhoc);
printf("Chuoi nhan duoc la: %s", monhoc);
```

```
Nhap mot chuoi: Ngon ngu lap trinh C
Chuoi nhan duoc la: Ngon ngu lap trinh C_
```

b) Xuất chuỗi ra màn hình

- Sử dụng hàm **printf** với đặc tả "%s"

```
char monhoc[50] = "Ngon ngu C";
printf("%s", monhoc);
```

```
Ngon ngu C_
```

- Sử dụng hàm **puts**

```
char monhoc[50] = "Ngon ngu C";
puts(monhoc);
```

```
Ngon ngu C
_
```

⇔ `printf("%s\n", monhoc);`

c) Xác định độ dài chuỗi

- Tự xác định

  Đếm cho đến khi gặp ký tự '\0'

```
char str[]="Ngon ngu C";
int dem=0;
while (str[dem]!='\0') dem++;
printf("Do dai chuoi la: %d ky tu",dem);
```

- Sử dụng con trỏ để xử lý chuỗi

```
char *message;
message="Ngon ngu C";
int dem=0;
while (*message!='\0'){*message++;dem++;}
printf("Do dai chuoi la: %d ky tu",dem);
```

c) Xác định độ dài chuỗi (tt)

- Dùng hàm

  Hàm **strlen(str)** trong thư viện `string.h`

```
printf("Do dai chuoi
la:%d",strlen(str));
```

- Bài tập:
  1. Nhập từ bàn phím chuỗi st1, viết ra màn hình chuỗi đó theo chiều ngược lại.
  2. Nhập từ bàn phím chuỗi st2, chuyển chuỗi st2 sang chữ hoa và viết ra màn hình chuỗi kết quả ra màn hình.

d) Ghép chuỗi

- Ghép chuỗi st2 vào sau chuỗi st1
  - Hàm strcat(st1,st2): nối chuỗi st2 vào sau chuỗi st1

```
#include <stdio.h>
#include <string.h>
#define maxst 40
main()
{    char st1[maxst]="Chao mung";
     char st2[maxst]="Ngon ngu C";
     if (maxst>strlen(st1)+strlen(st2))
          puts(strcat(st1,st2));
     else printf("Khong du bo nho!");
}
```

  - Chú ý: không viết `st=st1+st2;`

e) Sao chép chuỗi

- C không cho phép gán một chuỗi cho một biến do Chuỗi ký tự là một mảng.
  - Ví dụ: Không viết được `line="Hello";`

  ➔ Dùng hàm sao chép chuỗi hoặc viết vòng lặp sao chép từng ký tự
- Hàm sao chép chuỗi
  - `strcpy(st1,st2)` – gán chuỗi st2 cho chuỗi st1
  - Ví dụ: `strcpy(line,"Hello");`
- Chú ý:
  - Hàm không kiểm tra tính đúng đắn về kích thước ô nhớ của st1 có đủ chứa st2 hay không, do đó, cần lưu ý về kích thước Chuỗi ký tự.

f) So sánh chuỗi ký tự

- Quy tắc so sánh
  - Các ký tự của 2 chuỗi được so sánh từng cặp từ trái qua phải theo giá trị của mã ASCII
- Hàm **`strcmp(st1,st2)`** trong thư viện string.h
  - Trả về 0 nếu st1==st2
  - <0 nếu st1<st2
  - >0 nếu st>st2
- Ví dụ: Nhập các chuỗi cho đến khi nhập chuỗi "done"

```
char st[80];
do
{       gets(st);
        printf("Chuoi vua nhap: %s \n",st);
}   while (strcmp(st,"done"));
```

f) So sánh chuỗi ký tự (tt)

- Một số hàm so sánh khác
  - **`stricmp(st1,st2);`** so sánh chuỗi st1 với st2 nhưng không phân biệt chữ hoa-thường.
  - **`strncmp(st1,st2,n);`** so sánh n ký tự đầu tiên của st1 và st2.
  - **`strnicmp(st1,st2,n);`** so sánh n ký tự đầu tiên của st1 và st2 nhưng không phân biệt chữ hoa-thường.

g) Tìm kiếm ký tự

- Hàm **strchr(str,c)** tìm kiếm ký tự c trong chuỗi **str**
  - Kết quả là con trỏ trỏ tới vị trí của ký tự c
  - Nếu không tìm thấy trả về NULL

```
char str[80];   char c;
printf("Nhap chuoi: ");gets(str);
printf("Nhap ky tu can tim: "); c=getchar();
if (strchr(str,c)) printf("Tim thay %c");
else printf("Khong tim thay!");
```

- Hàm **strstr(str1,str2)** tìm kiếm chuỗi str2 trong chuỗi **str1**
  - Kết quả là con trỏ trỏ tới vị trí của chuỗi str2
  - Nếu không tìm thấy trả về NULL

3. Mảng chuỗi ký tự

- Bài toán:
  - Nhập mảng các chuỗi ký tự, sắp xếp các chuỗi đó theo thứ tự từ điển
  - ➔ Sử dụng mảng 2 chiều để lưu các chuỗi ký tự

```
void sapxep(int n,char x[][80])
{     char temp[80];
      int i,j;
      for (i=0;i<n-1;i++)
          for (j=i+1;j<n;j++)
              if (strcmp(x[i],x[j])>0)
              {strcpy(temp,x[i]);
               strcpy(x[i],x[j]);
               strcpy(x[j],temp);
              }
}
```

Mảng Chuỗi ký tự (tt)


```
int main()
{
    char st[40][80];
    int i,n=0;
    printf("Nhap cac chuoi ki tu, ket thuc bang chu
\"end\"\n");
    do
    {
     printf("Nhap chuoi thu %d: ",n+1);
     gets(st[n]);
    }
    while (strcmp(st[n++],"end"));
    n--;
    sapxep(n,st);
    printf("\Day cac chuoi ki tu sau khi sap xep\n");
    for(i=0;i<n;i++) puts(st[i]);
}
```

4. Một số hàm xử lý chuỗi ký tự

- Các hàm trong thư viện **`string.h`**
  - Hàm **`strlwr(st)`**: chuyển chuỗi st thành chữ thường
  - Hàm **`strupr(st)`**: chuyển chuỗi st thành chữ hoa
  - Hàm **`strrev(st)`**: đảo ngược chuỗi st
- Các hàm xử lý ký tự trong **`ctype.h`**
  - Hàm **`toupper(c)`**: chuyển c thành chữ hoa
  - Hàm **`tolower(c)`**: chuyển c thành chữ thường
  - Hàm **`isalpha(c)`**: đúng (khác 0) nếu c là chữ cái
  - Hàm **`islower(c)`**: đúng nếu c là chữ cái thường
  - Hàm **`isupper(c)`**: đúng nếu c là chữ cái hoa
  - Hàm **`isspace(c)`**: đúng nếu c là dấu cách, dấu \n, dấu về đầu dòng \r, tab \t

Một số hàm xử lý chuỗi ký tự (tt)

- Các hàm chuyển đổi (trong **`stdlib.h`**)
  - Hàm **`atoi(str)`**: chuyển đổi chuỗi str thành số nguyên int
    - Ví dụ: Đọc một số nguyên có thể dùng cặp lệnh

```
gets(str);
n=atoi(str); //tránh đọc xong số mà bộ đệm vẫn còn \n
```

  - Hàm **`atol(str)`**: chuyển đổi chuỗi str thành số nguyên long
  - Hàm **`atof(str)`**: chuyển đổi str thành số thực float

Các hàm này bỏ qua các dấu cách ở đầu, chuyển cho đến khi gặp ký tự không thích hợp, nếu không chuyển được thì kết quả là 0

Bài tập thực hành

- Bài 1. Viết hàm upper(char s[]) đổi toàn bộ các ký tự sang ký tự hoa (giống hàm strupr)
- Bài 2. Viết hàm lower(char s[]) đổi toàn bộ các ký tự sang ký tự thường (giống hàm strlwr)
- Bài 3. Viết hàm proper(char s[]) đổi các ký tự đầu tiên của mỗi từ sang ký tự hoa.
- Bài 4. Đếm xem có bao nhiêu từ trong s ('từ' là tập hợp các ký tự in được không chứa các dấu cách, xuống dòng, tab). In ra màn hình các từ trong s.
- Bài 5. Viết các hàm left, right, mid…

Bài tập thực hành

- Bài 6. Tìm từ có độ dài lớn nhất trong chuỗi s.
- Bài 7. Xóa tất cả các khoảng trắng của s
- Bài 8. Viết hàm standard(char s[]) loại bỏ toàn bộ khoảng trắng đầu chuỗi, cuối chuỗi và giữa 2 từ trong s chỉ còn 1 khoảng trắng.
- Bài 9. Đếm số lượng từ ‘em’ trong Chuỗi ký tự được đọc từ bàn phím.
- Bài 10. Kiểm tra xem từ có phải là palindrome hay không? Từ là palindrome là từ đọc từ phải sang trái cũng giống như đọc từ trái sang phải.
  Ví dụ: civic, madam, level,…

CƠ SỞ LẬP TRÌNH

KIỂU CẤU TRÚC

Nội dung

- Khái niệm
- Các thao tác với cấu trúc
- Mảng cấu trúc
- Con trỏ cấu trúc
- Chuyển tham số cấu trúc cho hàm
- Kiểu Union

1. Khái niệm kiểu cấu trúc

- Kiểu cấu trúc (struct)
  - Là kiểu dữ liệu bao gồm nhiều thành phần có kiểu khác nhau, mỗi thành phần được gọi là một trường (field)
  - Kiểu cấu trúc và mảng:
    - Các phần tử của mảng là cùng kiểu
    - Các phần tử của cấu trúc có thể có kiểu khác nhau
  - Struct được dùng để định nghĩa các kiểu dữ liệu mới

Khai báo cấu trúc

- Khai báo trực tiếp

```
struct <tên kiểu cấu trúc>
{       <kiểu 1> <trường 1>;
        …
        <kiểu n> <trường n>;
} <tên biến 1>, <tên biến 2>;
```

- Ví dụ
  - Khai báo cấu trúc NgayThang gồm 3 trường: ngày, tháng, năm

```
struct NgayThang
{
        unsigned char Ngay;
        unsigned char Thang;
        unsigned int Nam;
} date1,date2;
```

Khai báo cấu trúc (tt)

- Khai báo gián tiếp

```
typedef struct
{       <kiểu 1> <trường 1>;
        …
        <kiểu n> <trường n>;
} <tên kiểu cấu trúc>;
```

- Ví dụ
  - Khai báo kiểu cấu trúc NgayThang gồm 3 trường: ngày, tháng, năm

```
typedef struct
{
        unsigned char Ngay;
        unsigned char Thang;
        unsigned int Nam;
} NgayThang;
```

Khai báo cấu trúc lồng nhau

- Ví dụ:
  - Khai báo cấu trúc SinhVien gồm: mã sinh viên, họ tên, ngày sinh (thuộc kiểu ngaythang ở trên), giới tính, địa chỉ

Khai báo trực tiếp

```
struct SinhVien
{
        char Masv[10];
        char Hoten[40];
        NgayThang NgaySinh;
        int Gioitinh;
        char Diachi[50];

};
```

Khai báo gián tiếp

```
typedef struct
{
        char Masv[10];
        char Hoten[40];
        NgayThang NgaySinh;
        int Gioitinh;
        char Diachi[50];
} SinhVien;
```

Khai báo biến kiểu cấu trúc

- Khai báo biến kiểu cấu trúc
  - Khai báo tương tự như khai báo biến thuộc kiểu dữ liệu chuẩn
  - Với cách khai báo cấu trúc trực tiếp, có thể khai báo biến ngay khi khai báo cấu trúc
  - Ví dụ 1: Khai báo biến A và B

```
struct Diem
{
    float x;
    float y;
} A,B;
```

  - Ví dụ 2: Khai báo biến SV1,SV2 có kiểu SinhVien

```
SinhVien SV1,SV2;
```

2. Các thao tác với cấu trúc

- Khởi tạo cấu trúc
  - Biến cấu trúc có thể được khởi tạo giá trị trong lúc khai báo.
  - Các trường của cấu trúc được đặt giữa cặp dấu `{` và `}`, ngăn cách bằng dấu phẩy (`,`)
  - Ví dụ:

    Khởi tạo biến cấu trúc ngaysinh

```
struct NgayThang NgaySinh={01,08,1991}
```

Truy cập vào phần tử struct

- Đặc điểm
  - Không thể truy xuất trực tiếp
  - Thông qua toán tử thành phần cấu trúc . hay còn gọi là toán tử chấm (dot operation)
- Cú pháp

```
<tên biến cấu trúc>.<tên trường>
```

- Ví dụ:
  - Viết ra toạ độ điểm A trong khai báo trên

```
printf("x = %f, y = %f", A.x, A.y);
```

- Chú ý:
  - Các biến cấu trúc có thể gán cho nhau, vd: `B=A;`
  - KHÔNG thực hiện được các hàm nhập xuất, các phép quan hệ, số học, logic trên biến cấu trúc

Gán dữ liệu kiểu cấu trúc

- Có 2 cách

```
<biến cấu trúc đích> = <biến cấu trúc nguồn>;

<biến cấu trúc đích>.<tên thành phần> = <giá trị>;
```

- Ví dụ

```
struct Diem
{
        int x, y;
} diem1 = {2912, 1706}, diem2;
…
diem2 = diem1;
diem2.x = diem1.x;
diem2.y = diem1.y * 2;
```

Cấu trúc phức tạp

- Cấu trúc đệ quy (tự trỏ)

```
struct PERSON
{
        char hoten[30];
        struct PERSON *father, *mother;
};

struct NODE
{
        int value;
        struct NODE *pNext;
};
```

  - Sử dụng để tạo danh sách liên kết (đơn – LIFO, FIFO, kép, vòng)

Ví dụ - Tính tổng 2 số phức

- Nhập vào 2 số phức, tính tổng và in kết quả

```
typedef struct
{
float Thuc;
float Ao;
} SoPhuc;
void InSoPhuc(SoPhuc p)
{
      printf("%.2f + i%.2f\n",p.Thuc,p.Ao);
}
int main()
{
SoPhuc p1,p2,p;
printf("Nhap so phuc thu nhat:\n");
printf("Phan thuc: ");scanf("%f",&p1.Thuc);
printf("Phan ao: ");scanf("%f",&p1.Ao);
```

Ví dụ - Tính tổng 2 số phức

```
printf("Nhap so phuc thu hai:\n");
printf("Phan thuc: ");scanf("%f",&p2.Thuc);
printf("Phan ao: ");scanf("%f",&p2.Ao);
printf("So phuc thu nhat: ");
InSoPhuc(p1);
printf("So phuc thu hai: ");
InSoPhuc(p2);
p.Thuc = p1.Thuc+p2.Thuc;
p.Ao = p1.Ao + p2.Ao;
printf("Tong 2 so phuc: ");
InSoPhuc(p);
getch();
}
```

3. Mảng cấu trúc

- Mảng cấu trúc
  - Khai báo tương tự như mảng với kiểu dữ liệu cơ sở (char, int, float, …)
- Ví dụ:
  - Khai báo mảng để lưu danh sách sinh viên

```
struct  SinhVien DanhSach[100];
```

  - Mảng các struct cũng được đánh chỉ số từ 0
  - Truy cập đến Hoten của sinh viên thứ `i`

```
DanhSach[i].Hoten
```

- Bài tập

  Nhập vào danh sách n sinh viên, in danh sách vừa nhập ra màn hình

4. Con trỏ cấu trúc

- C cho phép sử dụng con trỏ trỏ tới cấu trúc cũng như các con trỏ trỏ tới các kiểu dữ liệu khác
- Khai báo

```
struct <Tên cấu trúc> * <Tên biến con trỏ>;
```

Khi khai báo, con trỏ chưa trỏ tới địa chỉ cụ thể nào.

- Ví dụ:

Khai báo một con trỏ cấu trúc kiểu NgayThang ở trên

```
struct NgayThang *p;
struct NgayThang date;
p=&date;
```

Khi đó, **p** sẽ chứa địa chỉ của **date**

Con trỏ cấu trúc (tt)

- Truy cập đến các trường của cấu trúc đang được quản lý bởi con trỏ
  - Sử dụng toán tử mũi tên (->)
  - Sử dụng toán tử lấy giá trị (*)

  `p->Ngay` là tương đương `(*p).Ngay`
- Ví dụ: Viết ra ngày-tháng-năm

```
struct NgayThang *p;
    //Sử dụng toán tử mũi tên
printf("%d-%d-%d",p->Ngay,p->Thang,p->Nam);
    //Sử dụng toán tử *
printf("%d-%d-%d",(*p).Ngay,(*p).Thang,(*p).Nam);
```

5. Chuyển tham số struct cho hàm

- Tham số của hàm có thể là
  - Từng trường của cấu trúc
  - Biến cấu trúc (tham số thực sự là giá trị cấu trúc)
  - Con trỏ cấu trúc (tham số thực sự là địa chỉ của biến cấu trúc)
  - Con trỏ cấu trúc hoặc mảng cấu trúc (tham số thực sự là tên mảng cấu trúc)
- Hàm có thể trả về
  - Giá trị cấu trúc
  - Con trỏ cấu trúc

Ví dụ - Số phức

- Khai báo kiểu số phức, viết các hàm
  - SoPhuc cong(SoPhuc u, SoPhuc v); trả về tổng của các giá trị phức u,v.
  - void InSP(SoPhuc u); dùng để in số phức u

```
typedef struct
{
float Thuc;
float Ao;
} SoPhuc;
//Ham tinh tong 2 so phuc u,v
SoPhuc cong(SoPhuc u, SoPhuc v)
{       SoPhuc tong;
        tong.Thuc=u.Thuc+v.Thuc;
        tong.Ao=u.Ao+v.Ao;
        return tong;
}
```

Ví dụ - Số phức (tt)

```
void InSoPhuc(SoPhuc u)
{     printf("%.2f + i%.2f\n",u.Thuc,u.Ao);
}
int main()
{     SoPhuc p1,p2,p;
      printf("Nhap so phuc thu nhat:\n");
      printf("Phan thuc: ");scanf("%f",&p1.Thuc);
      printf("Phan ao: ");scanf("%f",&p1.Ao);
      printf("Nhap so phuc thu hai:\n");
      printf("Phan thuc: ");scanf("%f",&p2.Thuc);
      printf("Phan ao: ");scanf("%f",&p2.Ao);
      printf("Tong 2 so phuc: ");
      InSoPhuc(cong(p1,p2));
      getch();
}
```

6. Union

- Khái niệm
  - Được khai báo và sử dụng như cấu trúc
  - Các thành phần của union có chung địa chỉ đầu (nằm chồng lên nhau trong bộ nhớ)
- Khai báo

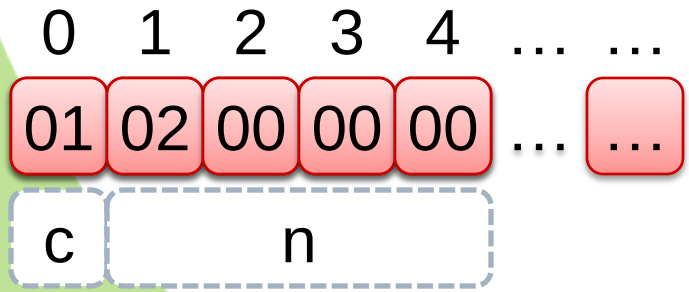
```
union <tên kiểu union>
{
        <kiểu dữ liệu> <tên thành phần 1>;
        …
        <kiểu dữ liệu> <tên thành phần 2>;
};
```

So sánh struct và union

- Ví dụ

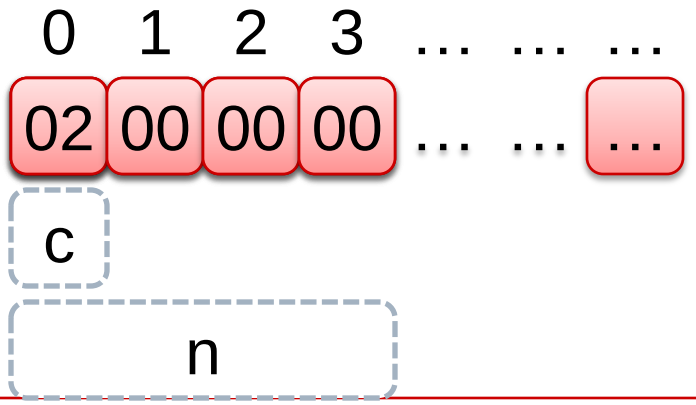
```
struct MYSTRUCT
{
        char c;
        int n;
} s;

s.c = 1; s.n = 2;
```




```
union MYUNION
{
        char c;
        int n;
} u;

u.c = 1; u.n = 2;
```



Bài tập thực hành

1. Phân số
   - Khai báo kiểu dữ liệu phân số (PHANSO)
   - Nhập/Xuất phân số
   - Rút gọn phân số
   - Tính tổng, hiệu, tích, thương hai phân số
   - Kiểm tra phân số tối giản
   - Quy đồng hai phân số
   - So sánh hai phân số

Bài tập thực hành

2. Mảng phân số
   - Nhập/Xuất n phân số
   - Rút gọn mọi phân số
   - Đếm số lượng phân số âm/dương trong mảng
   - Tìm phân số dương đầu tiên trong mảng
   - Tìm phân số nhỏ nhất/lớn nhất trong mảng
   - Sắp xếp mảng tăng dần/giảm dần

Bài tập thực hành

3. Điểm trong mặt phẳng Oxy
   - Khai báo kiểu dữ liệu điểm (DIEM)
   - Nhập/Xuất tọa độ 2 điểm A và B
   - Tính khoảng cách giữa hai điểm A và B
   - Tìm điểm đối xứng qua gốc toạ độ/trục Ox/Oy
4. Tam giác
   - Khai báo kiểu dữ liệu tam giác (TAMGIAC)
   - Nhập/Xuất tam giác
   - Tính chu vi, diện tích tam giác

Bài tập thực hành

5. Mảng điểm
   - Nhập/Xuất n điểm
   - Đếm số lượng điểm có hoành độ dương
   - Tìm điểm có hoành độ lớn nhất/nhỏ nhất
   - Tìm điểm gần gốc tọa độ nhất
6. Quản lý học sinh
   - Khai báo kiểu dữ liệu học sinh (HOCSINH)
   - Nhập n học sinh với các thuộc tính: Họ tên, năm sinh, điểm toán, lý, hoá, tổng điểm. (Tổng điểm = toán + lý + hoá, được tính sau khi nhập các điểm toán, lý, hoá)

Bài tập thực hành

6. Quản lý học sinh
   - Sắp xếp danh sách theo thứ tự giảm của tổng điểm, in kết quả ra màn hình
   - Tìm kiếm theo họ và tên của học sinh, đưa kết quả ra màn hình.
   - In ra màn hình các thí sinh có tổng điểm lớn hơn 15

CƠ SỞ LẬP TRÌNH

KIỂU TẬP TIN

Nội dung

- Khái niệm
- Các thao tác trên tập tin
- Tập tin văn bản
- Tập tin nhị phân
- Các hàm xử lý tập tin
- Truyền tham số là tập tin cho hàm

1. Khái niệm kiểu tập tin

- Tập tin
  - Là tập hợp các dữ liệu có liên quan với nhau và có cùng kiểu được nhóm lại với nhau tạo thành một dãy.
  - Tập được chứa trong thiết bị nhớ ngoài

  → Kích thước và số lượng gần như không hạn chế.
- Phân loại tập tin
  - Theo cách truy cập
    - Tập tin truy cập tuần tự: việc đọc một phần tử bất kỳ của tập tin bắt buộc phải tuần tự đi qua các phần tử trước đó
    - Tập tin truy cập ngẫu nhiên: có thể truy xuất phần tử bất kỳ của tập tin thông qua chỉ số thứ tự phần tử trong tập tin

Khái niệm kiểu tập tin (tt)

- Phân loại tập tin
  - Theo bản chất dữ liệu
    - Tập tin văn bản: chứa các kí tự trong bảng mã ASCII không kể đến các kí tự điều khiển. Dữ liệu được lưu thành các dòng, mỗi dòng được kết thúc bằng ký tự xuống dòng là CR (Carriage Return – về đầu dòng, mã 10) và LF (Line Feed – xuống dòng, mã 13). tập tin văn bản kết thúc bằng kí tự EOF (End Of File) có mã 26 (Ctrl + Z)

      Ví dụ:

      Các tập tin văn bản (text) (*.txt)
    - Tập tin nhị phân: các phần tử là các số nhị phân, và chứa khá nhiều dữ liệu có mã là các kí tự điều khiển.

    Ví dụ:

    Các tập tin chương trình (*.exe, *.com…)

Một số khái niệm

- Biến tập tin
  - Là biến thuộc kiểu tập tin dùng đại diện cho một tập tin.
  - Dữ liệu chứa trong tập tin được truy xuất thông qua các thao tác với biến tập tin.
- Con trỏ tập tin
  - Dùng để xác định vị trí của phần tử hiện tại để đọc hoặc ghi dữ liệu trên tập tin.
  - Khi tập tin được mở để đọc hoặc ghi thì con trỏ tập tin luôn ở vị trí đầu tập tin.
  - Mỗi khi đọc hoặc ghi trên tập tin thì con trỏ tập tin tự động tăng lên một khoảng theo đúng số byte vừa đọc hoặc ghi trên tập tin.

2. Các thao tác trên tập tin

- Các bước cơ bản để xử lý tập tin
  1. Khai báo biến tập tin
  2. Mở tập tin để ghi hoặc đọc
  3. Xử lý dữ liệu trong tập tin
  4. Đóng tập tin
- Các hàm thao tác với tập tin trong thư viện stdio.h

2.1 Khai báo biến tập tin

- Cú pháp

```
FILE *<tên biến TẬP>;
```

  - Trong đó, FILE là từ khoá luôn phải có và viết dạng chữ hoa
  - Các biến tập tin là các biến con trỏ

- Ví dụ:

```
FILE *f1,*f2;
// Khai báo 2 biến tập tin f1 và f2
```

2.2 Mở tập tin

- Cú pháp

```
<Biến TẬP>=fopen(<Tên TẬP>,<Kiểu xử lý TẬP>);
```

  - Trong đó:
    - Tên tập tin: đường dẫn đến tập tin trên đĩa (lưu ý, dấu \ được ghi là \\). Tên tập tin được đặt theo quy tắc đặt tên.
    - Kiểu xử lý tập tin: xác định cách thức mà tập tin được mở
  - fopen trả về một con trỏ tập tin, nếu có lỗi con trỏ trả về NULL
- Ví dụ:

```
f1=fopen("C:\\C-Samples\\VIDU.TXT","w");
```

Mở tập tin VIDU.TXT mới để ghi

```
f2=fopen("C:\\C-Samples\\VIDU.TXT","r");
```

Mở tập tin VIDU.TXT mới để đọc

Các chế độ xử lý tập tin

Chế độ	Ý nghĩa
r	Mở tập tin văn bản để đọc
w	Mở tập tin văn bản để ghi, ghi đè lên tập tin đã có
a	Mở tập tin văn bản và ghi nối vào cuối TẬP, chưa có tạo mới
r+	Mở tập tin văn bản để đọc/ghi
w+	Mở tập tin văn bản để ghi/đọc, ghi đè lên tập tin đã có
a+	Mở tập tin văn bản hoặc tạo mới để đọc và ghi nối vào cuối
rb	Mở tập tin nhị phân để đọc
wb	Mở tập tin nhị phânđể ghi, ghi đè lên tập tin đã có
ab	Ghi nối vào tập tin nhị phân
r+b	Mở ra tập tin nhị phân để đọc/ghi
w+b	Tạo ra tập tin nhị phân để đọc/ghi
a+b	Nối vào hay tạo mới tập tin nhị phân

Ví dụ mở tập tin

- Mở tập tin VIDU.TXT để ghi

```
FILE *f;
f = fopen("VIDU.txt", "w");
if (f!=NULL)
{
        /* Các câu lệnh để thao tác với TẬP*/
        /* Đóng tập tin */
}
else printf("Loi - Khong mo duoc tep!");
```

- Chú ý:
  - Khi mở tập tin để ghi mà tập tin đã tồn tại rồi thì tập tin đó sẽ bị xoá và được thay bằng tập tin khác.
  - Khi mở tập tin để đọc thì tập tin đó phải tồn tại, nếu không sẽ có lỗi.

2.3 Các thao tác khác

- Hàm đóng tập tin

```
fclose(<Tên biến TẬP>);
```

  - Hàm trả về 0 nếu đóng tập tin thành công, trả về EOF nếu có lỗi

- Hàm kiểm tra kết thúc tập tin hay chưa?

```
feof(<Tên biến TẬP>);
```

  - Hàm trả về EOF nếu đã hết tập tin, ngược lại trả về 0

- Hàm di chuyển con trỏ tập tin về đầu

```
rewind(<Tên biến TẬP>);
```

3. Tập tin văn bản

- Ghi dữ liệu lên tập tin văn bản
  - `putc(ch, f);` ghi ký tự ch vào tập tin văn bản `f`, trả về `EOF` nếu gặp lỗi.
  - `fputs(str,f);` ghi chuỗi `str` vào tập tin văn bản `f`, trả về 0 nếu str rỗng, trả về `EOF` nếu gặp lỗi
  - `fprintf(f,chuỗi định dạng, danh sách biểu thức);` ghi vào tập tin văn bản `f` các biểu thức với các định dạng được chỉ ra, tương tự hàm `printf()`

Ghi dữ liệu lên tập tin văn bản

- Ví dụ 1: Ghi một dòng các chữ hoa vào tập tin

```
#include <stdio.h>
#include <ctype.h>
main()
{       FILE *f;     char c;
        f=fopen("C11T001.txt","w");
        do putc(toupper(c=getchar()),f);
        while (c!='\n');
        fclose(f);
}
```

- Ví dụ 2: Ghi chuỗi ký tự vào tập tin

```
main()
{       FILE *f;
        f=fopen("C11T002.txt","w");
        fputs("Ngon ngu lap trinh C",f);
        fclose(f);
}
```

Đọc dữ liệu từ tập tin văn bản

- Các hàm đọc dữ liệu từ tập tin văn bản
  - **`getc(f);`** **`fgetc(f);`** đọc một ký tự từ tập tin văn bản **`f`**. Hàm trả về mã ASCII của ký tự nào đó (kể cả **`EOF`**) trong tập tin **`f`**
  - **`fgets(str,n,f);`** đọc một chuỗi str từ tập tin văn bản **`f`** cho đến khi gặp ký tự xuống dòng **`'\n'`** hoặc ký tự EOF hay đủ n ký tự.
  - **`fscanf(f,chuỗi định dạng, danh sách các biến);`** đọc từ tập tin văn bản **`f`** các biến theo định dạng, tương tự hàm **`scanf()`**

Đọc dữ liệu từ tập tin văn bản (tt)

- Ví dụ 1: Sao chép TẬP
  - Sao chép nội dung tập tin sample.txt sang tập tin sp.txt

```
main()
{       FILE *f1,*f2; int ch;
        f1=fopen("sample.txt","r");
        f2=fopen("sp.txt","w");
        if (f1!=NULL&&f2!=NULL)
        {       ch=fgetc(f1);
                while (!feof(f1))
                {       fputc(ch,f2); //ghi vao f2
                        ch=fgetc(f1); //doc tu f1
                }
                fclose(f1);
                fclose(f2);
        }
}
```

Đọc dữ liệu từ tập tin văn bản (tt)

- Ví dụ 2:

  Đọc từ tập tin songuyen.txt một dãy các số nguyên dương, ghi vào tập tin ketqua.txt các số nguyên tố có trong dãy đó, cuối cùng ghi ra tổng của các số nguyên tố đó.

```
int main()
{FILE *f1,*f2; int n;  long tong=0;
 f1=fopen("songuyen.txt","r");
 f2=fopen("ketqua.txt","w");
 if (f1!=NULL && f2!=NULL)
 {  while (!feof(f1))
     { fscanf(f1,"%d",&n);
       if (ngto(n)) { tong+=n; fprintf(f2,"%d\t",n);}
     } fprintf(f2,"\nTong la: %ld",tong);
    fclose(f1); fclose(f2);
  } }
```

4. Tập tin nhị phân

- Ghi dữ liệu lên tập tin nhị phân
  - Hàm **fwrite(địa chỉ của khối dữ liệu,kích thước mỗi phần tử, số phần tử,f);**
  - Ghi vào trong tập tin **f** khối dữ liệu có địa chỉ, số lượng và kích thước của mỗi phần tử.
  - Giá trị trả về là số phần tử đã được ghi vào tập tin
  - Ví dụ:

```
FILE *f;
int i;
f=fopen("C:\\SN100.txt","wb");
for (i=1;i<=100;i++)
   fwrite(&i,sizeof(int),1,f);
fclose(f);
```

Tập tin nhị phân (tt)

- Đọc dữ liệu từ tập tin nhị phân
  - Hàm **fread(địa chỉ của vùng nhớ nhận dữ liệu,kích thước mỗi phần tử, số phần tử,f);**
  - Đọc từ tập tin **f** số lượng phần tử có kích thước của mỗi phần tử được chỉ ra và lưu vào vùng nhớ nhận dữ liệu
  - Giá trị trả về là số phần tử đã được đọc từ tập tin
  - Ví dụ:

```
g=fopen("C:\\SN100.txt","rb");
do { fread(&i,sizeof(int),1,g);
     if (!feof(g)) printf("%d",i); }
while (!foef(g));
```

Di chuyển con trỏ tập tin

- Hàm fseek

```
fseek(f,No*Kích thước, vị trí)
```

Trong đó:

- f: con trỏ tập tin
- No: số thứ tự phần tử trong tập tin (phần tử đầu tiên đánh số là 0
- Vị trí có thể là:
  - SEEK_SET hoặc 0: di chuyển từ đầu tập tin
  - SEEK_CUR hoặc 1: di chuyển từ vị trí hiện tại
  - SEEK_END hoặc 2: di chuyển từ cuối tập tin

- Hàm trả về 0 nếu di chuyển thành công, trả về khác 0 nếu ngược lại.

Di chuyển con trỏ tập tin (tt)

- Ví dụ: Truy cập trực tiếp để cập nhật dữ liệu

```
main()
{
FILE *f;
int no,number;
f=fopen("C:\\SN100.txt","r+b");
do
{      printf("Vi tri can cap nhat: ");
       scanf("%d",&no);
       printf("Gia tri can cap nhat: ");
       scanf("%d",&number);
       if (no>0)
       {fseek(f,sizeof(int)*(no-1),SEEK_SET);
       fwrite(&number,sizeof(int),1,f);}
} while (no!=0);
fclose(f);
```

Ví dụ - Quản lý sinh viên

- Viết chương trình quản lý sinh viên
  - Mỗi sinh viên cần quản lý ít nhất 2 thông tin: mã sinh viên và họ tên.
  - Viết chương trình cho phép lựa chọn các chức năng:
    - Nhập danh sách sinh viên từ bàn phím rồi ghi lên tập tin SinhVien.dat
    - Đọc dữ liệu từ tập tin SinhVien.dat rồi hiển thị danh sách lên màn hình
    - Tìm kiếm họ tên của một sinh viên nào đó dựa vào mã sinh viên nhập từ bàn phím.

Ví dụ - Quản lý sinh viên (tt)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
typedef struct
{
      char Ma[10];
      char HoTen[40];
} SinhVien;
void WriteFile(char *FileName)
{
      FILE *f;
      int n,i;
      SinhVien sv;
      f=fopen(FileName,"wb");
      printf("Nhap bao nhieu sinh vien?");
      scanf("%d",&n);
      fflush(stdin);
```

Ví dụ - Quản lý sinh viên (tt)

```
for(i=1;i<=n;i++)
        {
                printf("Sinh vien thu %i\n",i);
                printf(" - MSSV: ");gets(sv.Ma);
                printf(" - Ho ten: ");gets(sv.HoTen);
                fwrite(&sv,sizeof(sv),1,f);
                fflush(stdin);
        }
        fclose(f);
        printf("Bam phim bat ky de tiep tuc");
        getch();
}
void ReadFile(char *FileName)
{
        FILE *f;
        SinhVien sv;
        f=fopen(FileName,"rb");
        printf(" MSSV | Ho va ten\n");
        printf("--------------------\n");
```

Ví dụ - Quản lý sinh viên (tt)

```
fread(&sv,sizeof(sv),1,f);
        while (!feof(f))
        {       printf(" %s    | %s\n",sv.Ma,sv.HoTen);
                fread(&sv,sizeof(sv),1,f);
        }
        fclose(f);
        printf("Bam phim bat ky de tiep tuc!!!");
        getch();
}
void Search(char *FileName)
{
        char MSSV[10];
        FILE *f;
        int Found=0;
        SinhVien sv;
        fflush(stdin);
        printf("Ma so sinh vien can tim: ");gets(MSSV);
        f=fopen(FileName,"rb");
```

Ví dụ - Quản lý sinh viên (tt)

```
while (!feof(f) && Found==0)
        {
                fread(&sv,sizeof(sv),1,f);
                if (strcmp(sv.Ma,MSSV)==0) Found=1;
        }
        fclose(f);
        if (Found == 1)
                printf("Tim thay SV co ma %s. Ho ten la:
%s",sv.Ma,sv.HoTen);
        else
                printf("Tim khong thay sinh vien co ma
%s",MSSV);
        printf("\nBam phim bat ky de tiep tuc!!!");
        getch();
}
```

Ví dụ - Quản lý sinh viên (tt)

```
int main()
{       int c;
        for (;;)
        {       printf("\n1. Nhap DSSV\n");
                printf("2. In DSSV\n");
                printf("3. Tim kiem\n");
                printf("4. Thoat\n");
                printf("Ban chon 1, 2, 3, 4: ");
                scanf("%d",&c);
                if(c==1)
                        WriteFile("d:\\SinhVien.Dat");
                else if (c==2)
                                ReadFile("d:\\SinhVien.Dat");
                        else if (c==3)
                                Search("d:\\SinhVien.Dat");
                                else break;
        }
        return 0;
}
```

5. Các hàm xử lý tập tin

- `int ferror(FILE *fp)`
  - Trả về 0 nếu không có lỗi, khác 0 nếu gặp lỗi
- `int remove(char *filename)`
  - Xoá tập tin có tên là filename
  - Trả về 0 nếu thành công
- `int rename(char *old, char *new)`
  - Đổi tên cũ **`old`** thành tên mới **`new`**
  - Trả về 0 nếu đổi thành công
- `int fflush(FILE *fp)`
  - Đổ bộ nhớ đệm
- `long ftell(FILE *fp)`
  - Cho kích thước của tập tin fp

6. Truyền tham số là tập tin cho hàm

- Ví dụ 1. Sao chép 2 tập tin

```
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[])
{
        FILE *in,*out; char ch;
        if (argc!=3)
                {printf("Chua co ten tep.\n"); exit(1);}
        if ((in=fopen(argv[1],"rb"))==NULL)
        {printf("Khong the mo tep nguon!\n");exit(1);}
        if ((out=fopen(argv[2],"wb"))==NULL)
        {printf("Khong the mo tep dich!\n");exit(1);}
        while (!feof(in))
        { ch=getc(in);
          if (!feof(in)) putc(ch,out);
        }
        printf("Da sao chep xong!")
        fclose(in); fclose(out);
}
```

Truyền tham số là tập tin cho hàm (tt)

- Ví dụ 2: Xoá tập tin

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
main(int argc, char *argv[])
{ char ch;
  if (argc!=2)
  {printf("Chua co ten tep.\n"); exit(1);}
  printf("Co xoa tep %s khong? (C/K)?",argv[1]);
  ch=getchar();
  if (toupper(ch)=='C')
      if (remove(argv[1]))
      {printf("Khong the xoa!"); exit(1);}
}
```

Truyền tham số là tập tin cho hàm (tt)

Cách truyền tham số khi thực hiện chương trình

- Ví dụ 1:
  - Dịch ra tập tin mycopy.exe
  - mycopy <tên tập tin nguồn> <tên tập tin đích>

  Chẳng hạn: mycopy sample.txt vd.txt
- Ví dụ 2:
  - Dịch ra tập tin mydel.exe
  - mydel <tên tập tin cần xoá>

  Chẳng hạn: mydel sample.txt

Bài tập thực hành

1. Viết chương trình đếm số chữ của từng loại chữ trong bảng chữ cái được chứa trong một tập tin văn bản (tập tin có bao nhiêu chữ ‘A’, chữ ‘B’…).
2. Viết chương trình đếm trong tập tin văn bản
   - Có bao nhiêu dòng?
   - Có bao nhiêu kí tự?
   - Dòng dài nhất là dòng nào?
   - Từ dài nhất trong văn bản là từ nào (từ là một nhóm các kí tự liền nhau).

Bài tập thực hành

3. Viết chương trình nhập vào từ tập tin văn bản songuyen.txt một dãy các số nguyên, hãy tạo ra các TẬP:
   - prime.txt chứa các số nguyên tố có trong dãy
   - cp.txt chứa các số chính phương
   - dep.txt chứa các số đẹp (vd: 123, 134…)
   - hoanhao.txt chứa các số hoàn hảo
4. Cho 2 tập tin So1.txt và So2.txt chứa các số nguyên.
   - Sắp xếp theo chiều tăng dần các số trong 2 TẬP
   - Tạo tập tin So3.txt gồm các số trong cả 2 tập tin trên và cũng được sắp theo thứ tự tăng dần.