

CONFIDENTIAL

C Programming Basic – week 1

For HEDSPI Project

Lecturer :

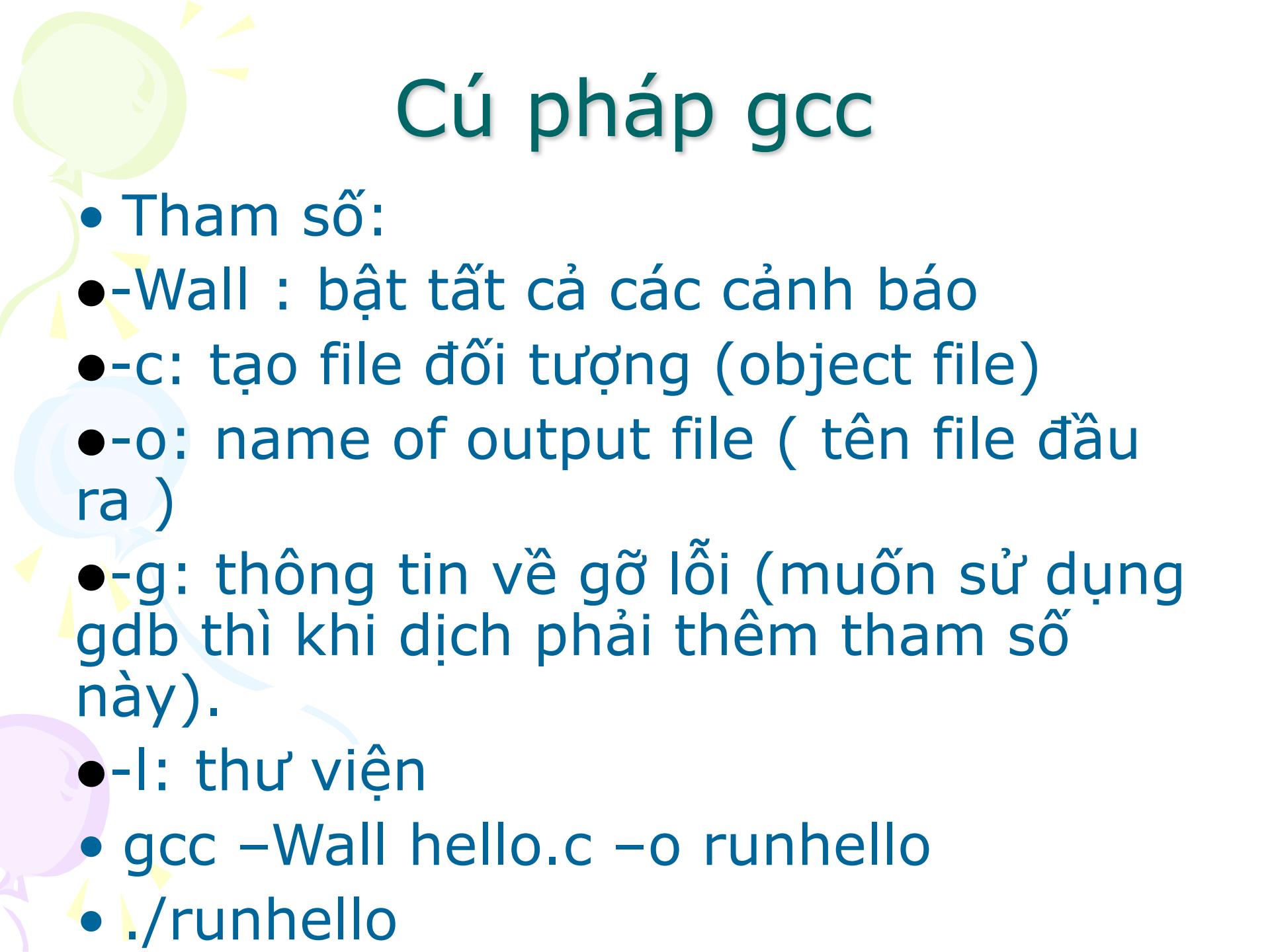
Do Quoc Huy

**Dept of Computer Sience
Hanoi University of Technology**



Giới thiệu

- Thực hành lập trình C trên môi trường UNIX.
- Chủ đề thực hành liên quan đến cấu trúc dữ liệu và giải thuật.
- trình biên dịch: gcc
- Trình soạn thảo: Emacs, K-Developer.

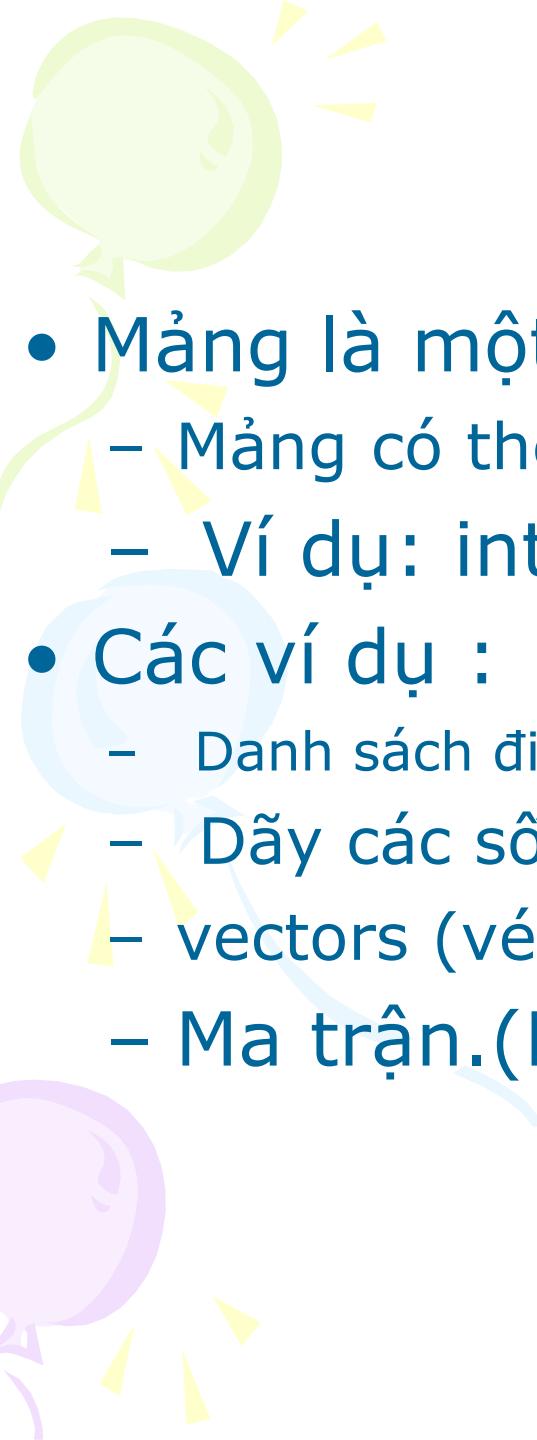


Cú pháp gcc

- Tham số:
 - -Wall : bật tất cả các cảnh báo
 - -c: tạo file đối tượng (object file)
 - -o: name of output file (tên file đầu ra)
 - -g: thông tin về gỡ lỗi (muốn sử dụng gdb thì khi dịch phải thêm tham số này).
 - -l: thư viện
- gcc -Wall hello.c -o runhello
- ./runhello

Tuần này: Các cấu trúc dữ liệu cơ sở và giải thuật.

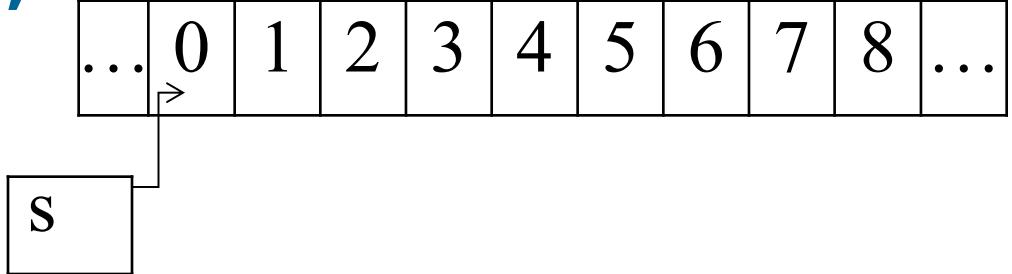
- Chủ đề:
 - Array, String, Pointer Review
 - Các phép toán về File dựa trên kí tự trong UNIX.
 - Các bài tập lập trình.



Array

- Mảng là một khối các biến có cùng kiểu dữ liệu
 - Mảng có thể được khai báo với mọi kiểu dữ liệu.
 - Ví dụ: int A[10] là 1 mảng 10 số nguyên.
- Các ví dụ :
 - Danh sách điểm của SV (list of students' marks)
 - Dãy các số được nhập vào bởi người sử dụng.
 - vectors (véc tơ)
 - Ma trận.(Matrices)

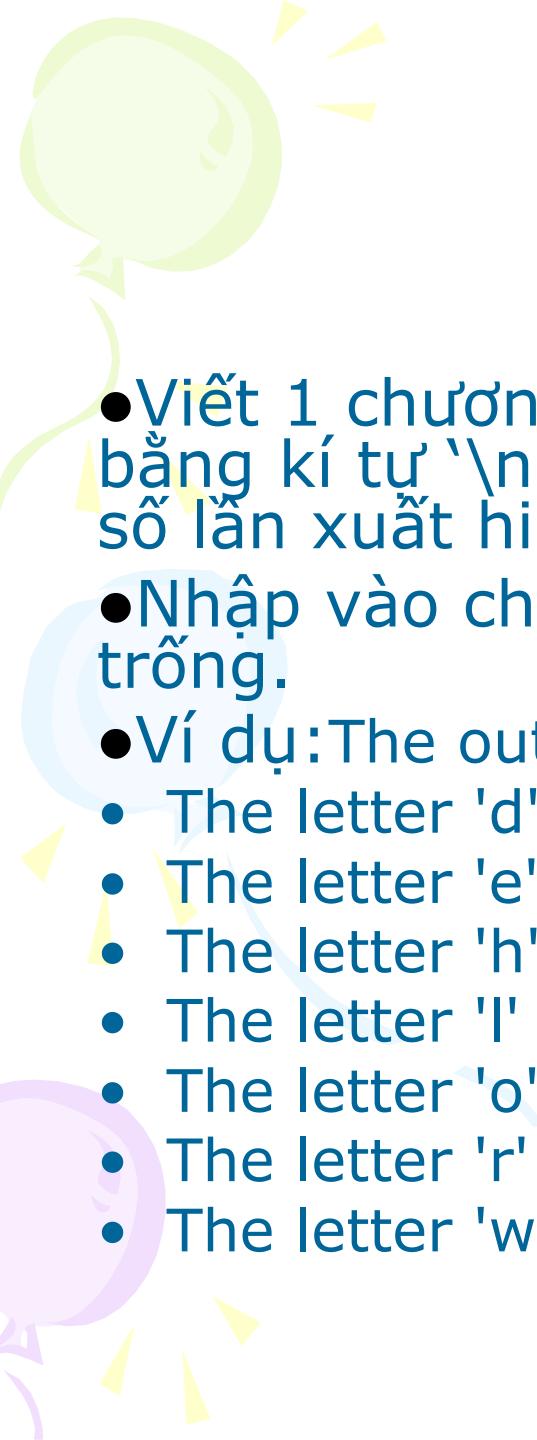
Mảng trong bộ nhớ

- Là dãy các biến có kiểu dữ liệu đã được định trước.
 - Bản thân mảng đã lưu giữ địa chỉ đầu tiên của dãy trong bộ nhớ.
 - Ví dụ: **double S[10];**
- 
- 1 mảng k phần tử A được khai báo là A [k-1] (**0-based**)



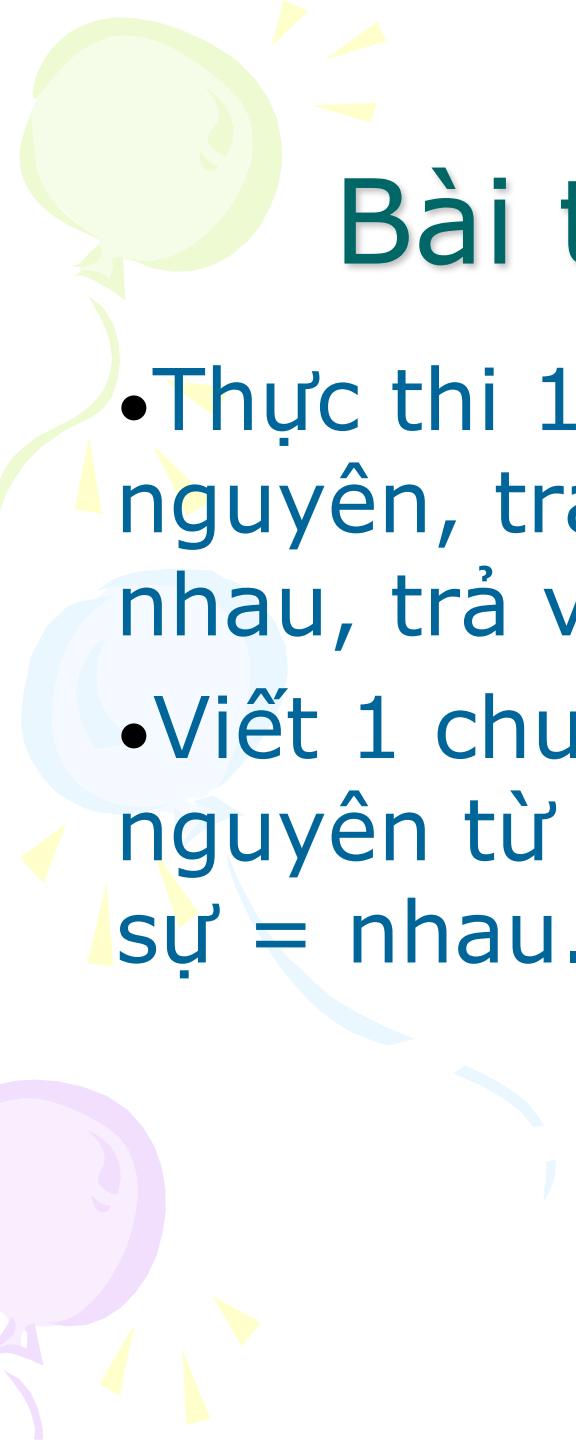
Ví dụ - Đảo ngược

```
#include <stdio.h>
int main(void)
{
    int i, A[10];
    printf("please enter 10 numbers:\n");
    for(i=0; i<10; i++)
        scanf("%d", &A[i]);
    printf("numbers in reversed order:\n");
    for(i=9; i>=0; i--)
        printf("%d\n", A[i]);
    return 0;
}
```



Bài tập 1.1

- Viết 1 chương trình nhận vào 1 dòng kí tự (kết thúc bằng kí tự '\n') từ người sử dụng, và đưa ra màn hình số lần xuất hiện của mỗi chữ cái trong dòng đó.
- Nhập vào chỉ gồm các chữ cái thường và dấu khoảng trắng.
- Ví dụ: The output for the input line: "hello, world!"
 - The letter 'd' appears 1 time(s).
 - The letter 'e' appears 1 time(s).
 - The letter 'h' appears 1 time(s).
 - The letter 'l' appears 3 time(s).
 - The letter 'o' appears 2 time(s).
 - The letter 'r' appears 1 time(s).
 - The letter 'w' appears 1 time(s).

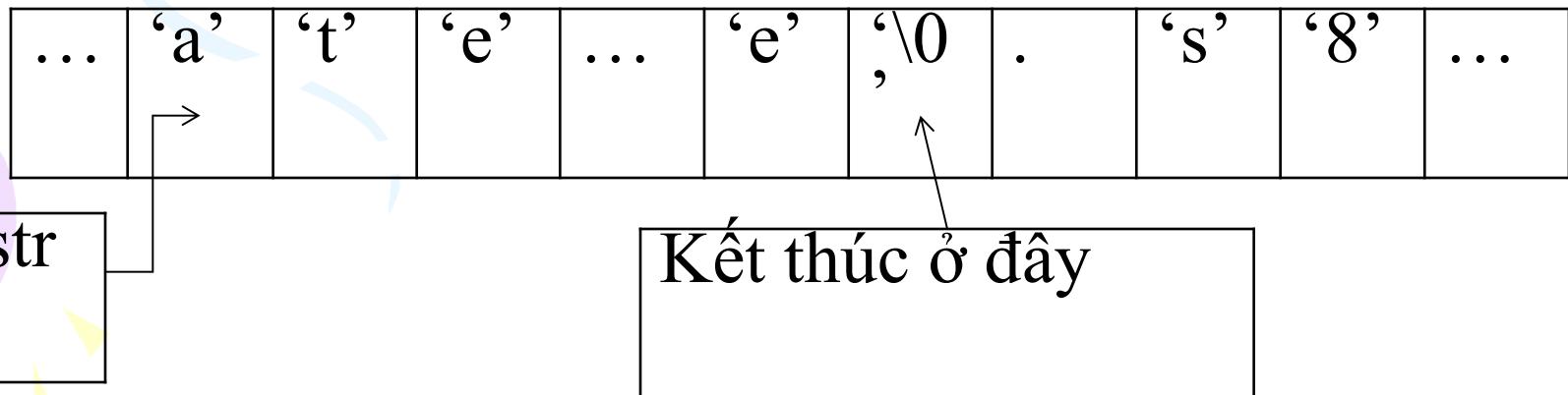


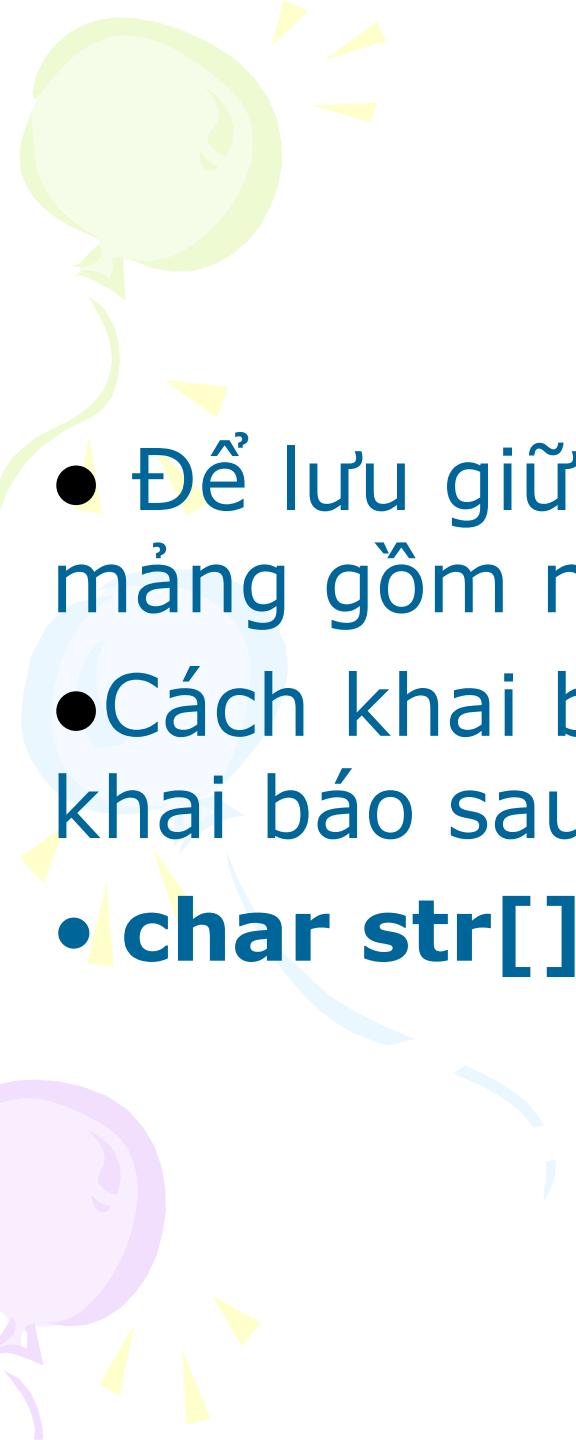
Bài tập 1.2 (20 phút)

- Thực thi 1 hàm nhận vào 2 mảng số nguyên, trả về 1 nếu 2 mảng giống nhau, trả về 0 nếu không.
- Viết 1 chương trình nhận 2 mảng số nguyên từ người sử dụng và kiểm tra $\text{sự} = \text{nhau}$.

Strings

- 1 mảng các kí tự.
- Sử dụng để lưu trữ văn bản.
- 1 cách khác để khởi tạo string:
 - **char str[] = "Text";**





String

- Để lưu giữ 1 xâu gồm n kí tự ta cần 1 mảng gồm n+1 phần tử.
- Cách khai báo trên tương đương với cách khai báo sau:
- **char str[] = {'b', 'I', 'a', 'b', 'I','\0'} ;**

Các hàm liên quan đến xâu và kí tự

- **getchar()**
 - c = getchar(); // đọc 1 kí tự và lưu vào c
- **scanf**
 - scanf("%s", str);
- **gets()**
 - gets(str); // chú ý trước khi dùng gets thường sử dụng lệnh while(getchar() != '\n'); để tránh lỗi.

Các hàm liên quan đến xâu và kí tự

- **strlen(const char s[])**
return độ dài xâu s
- **strcmp(const char s1[],const char s2[])**
trả về 0 nếu 2 xâu = nhau;
 1 nếu xâu s1 lớn hơn s2;
 -1 nếu s1 nhỏ hơn s2;
- **strcpy(char s1[],const char s2[])**
copy nội dung của xâu s2 vào xâu s1.

Bài tập 1.3

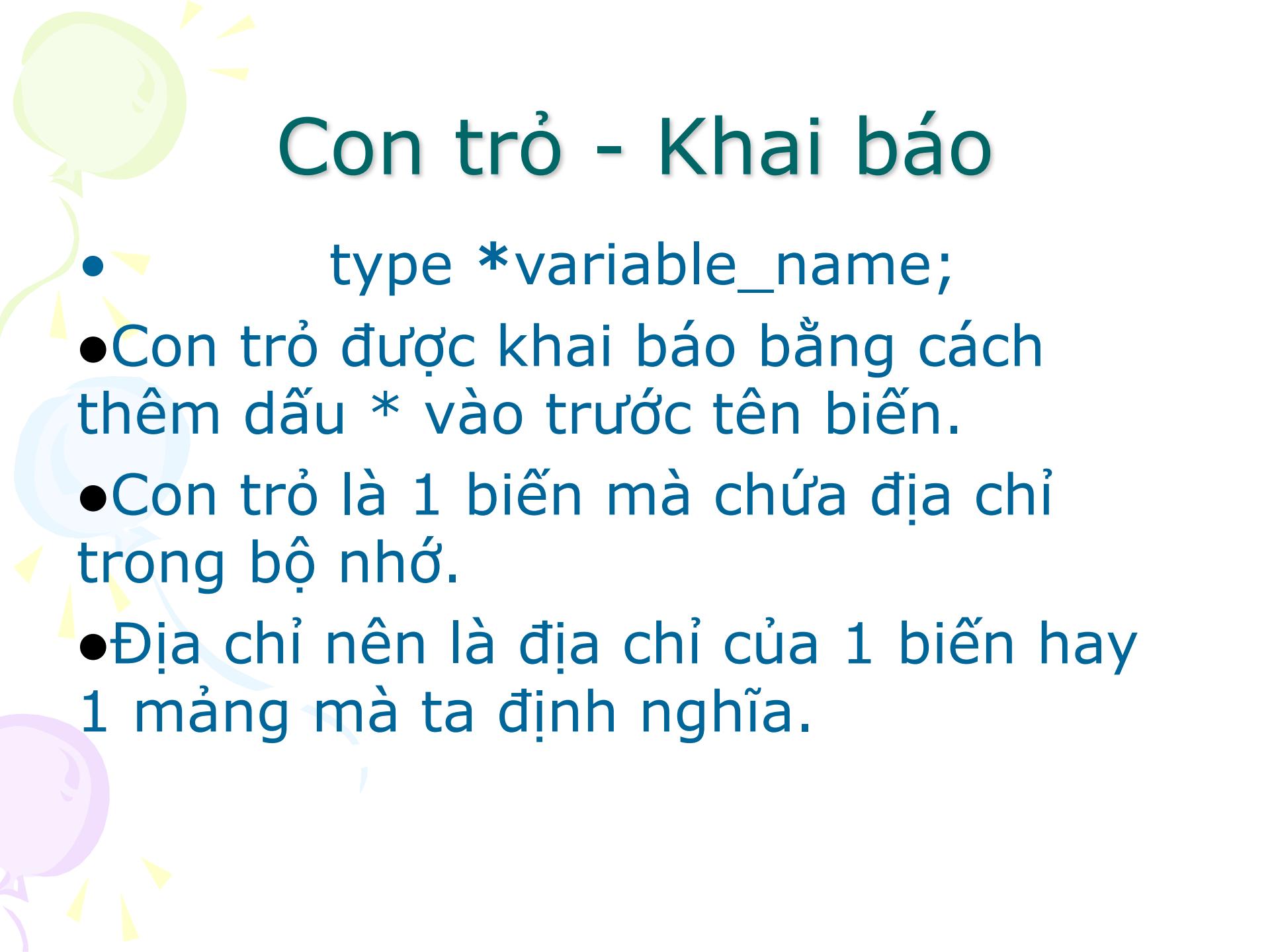
- **Viết 1 hàm:**

- Nhận 1 xâu và 2 kí tự.
- Hàm quét toàn bộ xâu và thay kí tự thứ nhất mỗi lần nó xuất hiện trong xâu bởi kí tự thứ 2.

- Viết 1 chương trình để test hàm trên. Đầu vào do người sử dụng nhập, xâu kí tự không có dấu khoảng trắng. In ra xâu kết quả.

- **Example:**

- input: “papa”, ‘p’, ‘m’
- output: “mama”



Con trỏ - Khai báo

type *variable_name;

- Con trỏ được khai báo bằng cách thêm dấu * vào trước tên biến.
- Con trỏ là 1 biến mà chứa địa chỉ trong bộ nhớ.
- Địa chỉ nên là địa chỉ của 1 biến hay 1 mảng mà ta định nghĩa.

Tham chiếu và khử tham chiếu

```
int n;
```

```
int *iptr; /* khai báo p là con trỏ kiểu int*/
```

```
n = 7;
```

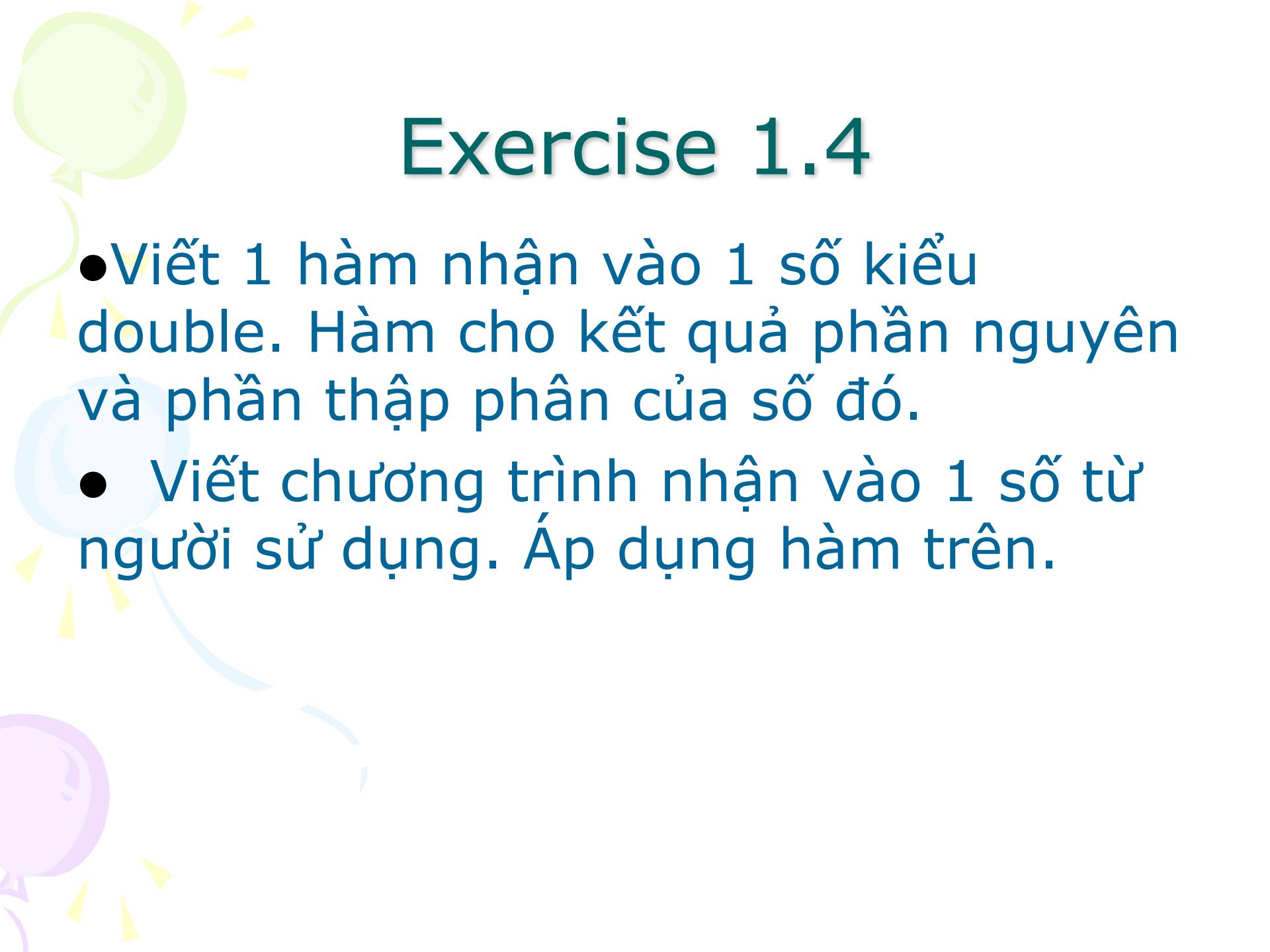
```
iptr = &n;
```

```
printf("%d", *iptr); /* Prints out '7'*/
```

```
*iptr = 177;
```

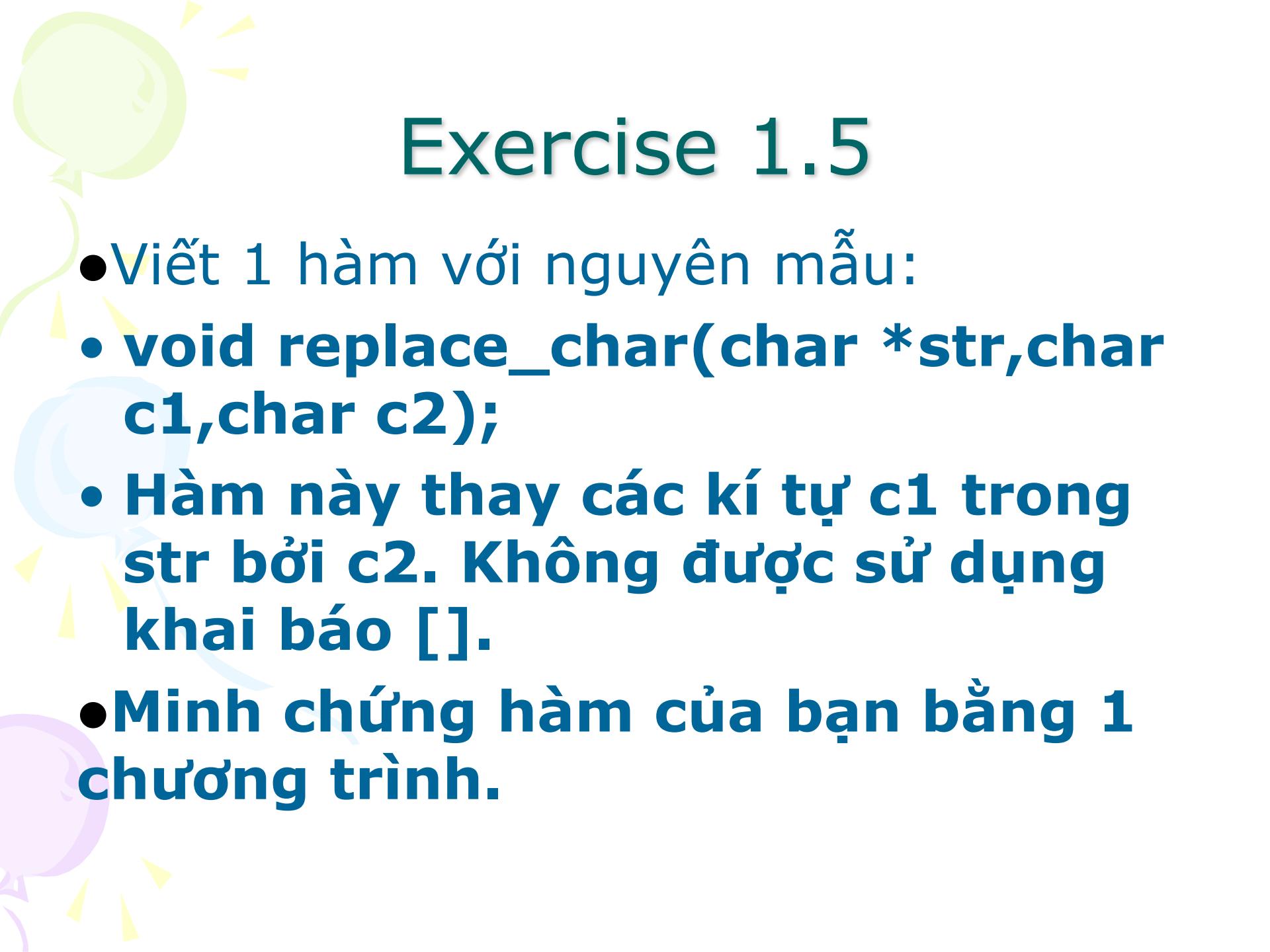
```
printf("%d", n); /* Prints out '177' */
```

```
iptr = 177; /* không dùng cách này */
```



Exercise 1.4

- Viết 1 hàm nhận vào 1 số kiểu double. Hàm cho kết quả phần nguyên và phần thập phân của số đó.
- Viết chương trình nhận vào 1 số từ người sử dụng. Áp dụng hàm trên.

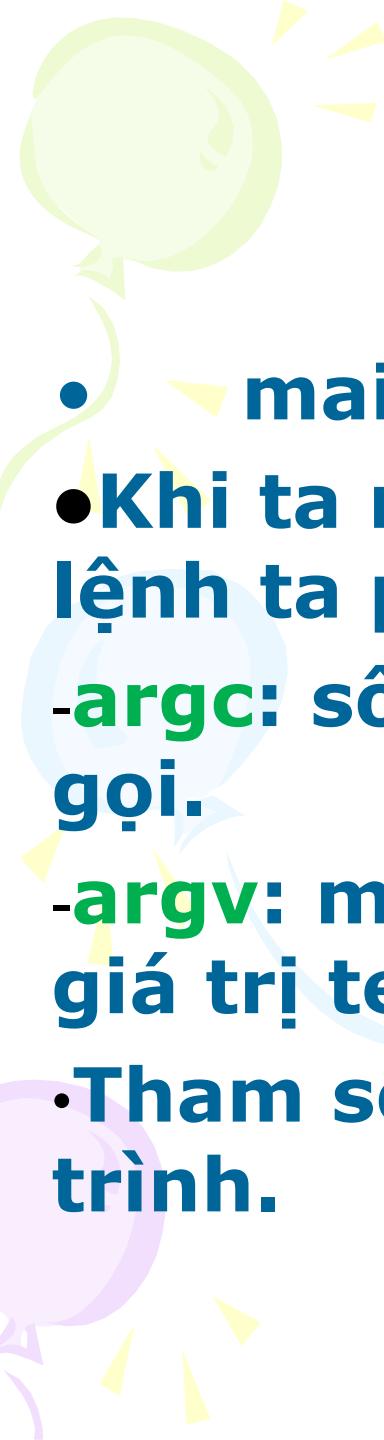


Exercise 1.5

- Viết 1 hàm với nguyên mẫu:
 - **void replace_char(char *str,char c1,char c2);**
 - **Hàm này thay các kí tự c1 trong str bởi c2. Không được sử dụng khai báo [].**
 - **Minh chứng hàm của bạn bằng 1 chương trình.**

Tham số từ dòng lệnh

- Tham số dòng lệnh là tham số cho hàm
- Hàm main là 1 hàm cơ sở.
- Nó có thể nhận đối số như bất kì hàm nào khác.
- Hàm gọi trong trường hợp này là 1 hệ thống tính toán hoặc là 1 chương trình khác.

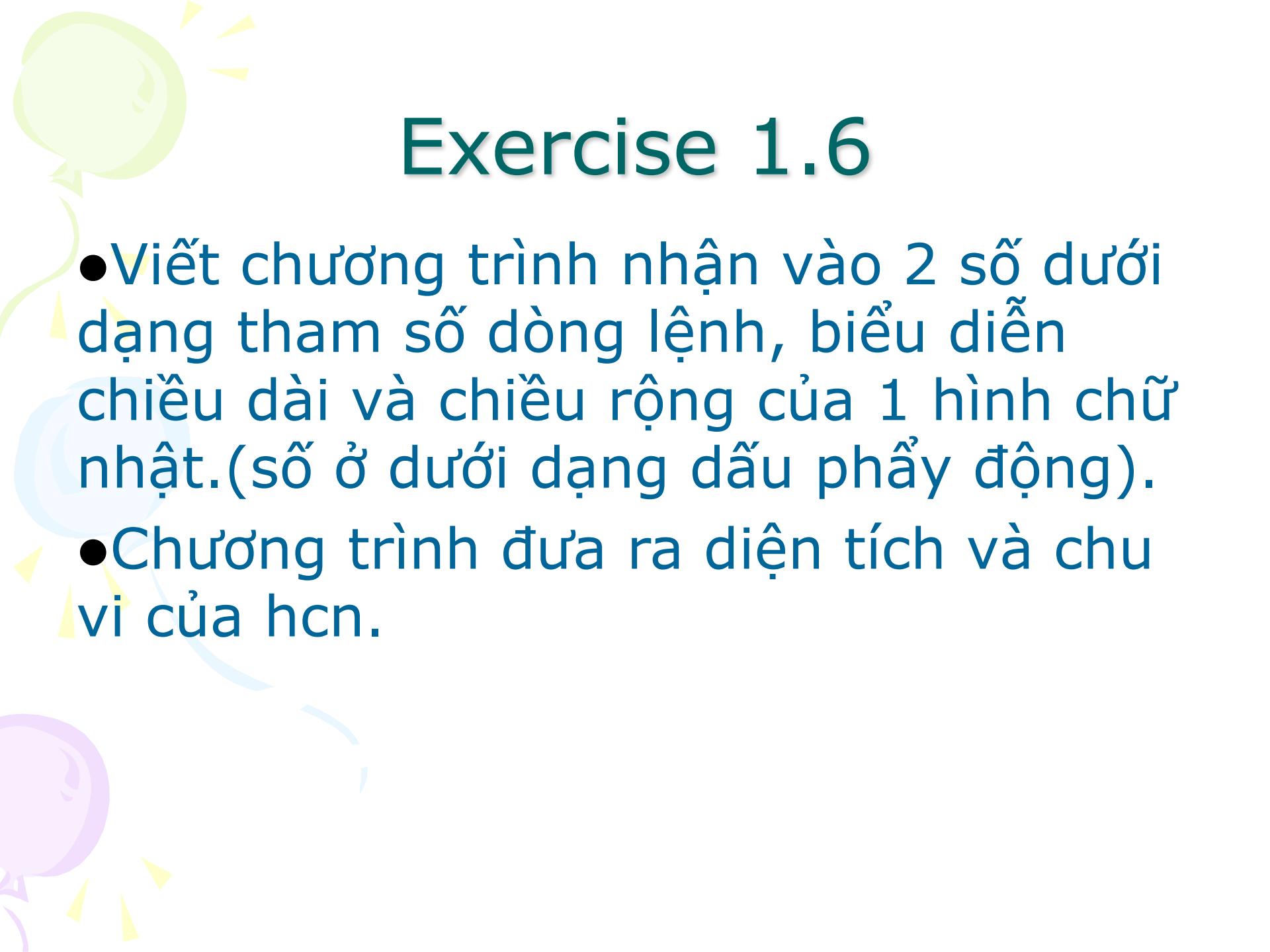


Nguyên mẫu ‘main’

- **main(int argc, char* argv[])**
- **Khi ta muốn hàm main nhận tham số dòng lệnh ta phải định nghĩa như trên.** Trong đó:
 - argc:** số tham số được đưa vào bởi người gọi.
 - argv:** mảng các con trỏ xâu kí tự, lưu giữ giá trị text của tham số.
- **Tham số đầu tiên luôn là tên của chương trình.**

Nguyên mẫu ‘main’

- **int main(int argc, char* argv[])**
- **argc** thường = 3 (Đôi khi cũng có thể là 2 trong 1 số trường hợp, tùy vào dòng lệnh)
- **argv** (dạng mảng các xâu kí tự):
 - Ví dụ:
 - Khi ta chạy dòng lệnh trong terminal:
./abc text.txt 178
 - Thì ta có tương ứng:
 - **argc = 3; argv[0] = “abc”; argv[1] =“text.txt”;**
argv[2]=“178”;
 - Khi đó trong chương trình xâu “text.txt” sẽ được thay cho argv[1], xâu “178” sẽ được thay cho argv[2].
- **Cũng có thể chỉ truyền 2 tham số thôi.** Ví dụ:
./abc text.txt
 - **Thì:argc = 2; argv[0] = “abc”; argv[1] =“text.txt”;**



Exercise 1.6

- Viết chương trình nhận vào 2 số dưới dạng tham số dòng lệnh, biểu diễn chiều dài và chiều rộng của 1 hình chữ nhật.(số ở dưới dạng dấu phẩy động).
- Chương trình đưa ra diện tích và chu vi của hcn.

Quản lí file

- C giao tiếp với các file qua sử dụng 1 kiểu dữ liệu gọi là con trỏ file.
- Con trỏ File:
 - Tham chiếu tới 1 file trên đĩa.
 - Được sử dụng để dẫn đường các phép toán của hàm vào/ra.
- FILE *fptr;

4 phép toán quan trọng

- Mở file
- Đọc từ 1 file vào chương trình
- Viết vào 1 file: từ chương trình vào file
- Đóng file

Mở một file

- Hàm fopen()

```
FILE *fopen(const char *filename, const  
char *mode);
```

- Ví dụ:

```
FILE *fptr;  
if ((fptr = fopen("test.txt", "r")) ==NULL)  
{
```

```
    printf("không thể mở test.txt file.\n");  
    exit(1);
```

```
}
```

Mở một file (tiếp..)

- **filename:** tên file.
 - Nó có thể là tên không, ví dụ: “data.txt” (trong trường hợp nằm cùng folder với chương trình C)
 - Hoặc có thể chứa đầy đủ đường dẫn đến file:
“/root/hedspi/CProgrammingBasic/Lab1/data.txt”
 - **Hoặc có thể là 1 mảng kí tự chứa tên file :** **char file_name[] = “junk.txt”;**

Các chế độ cho file text

chế độ	Mô tả
"r"	Mở 1 file text đã có để đọc
“w”	Tạo 1 file text để ghi
“a”	Mở 1 file đã có để thêm dữ liệu vào cuối
“r+”	Mở 1 file text đã có để đọc hoặc ghi
"w+"	Tạo 1 file text để đọc hoặc ghi
"a+"	Tạo mới hoặc mở 1 file đã có để thêm dữ liệu vào cuối

Chế độ cho Binary file (file nhị phân)

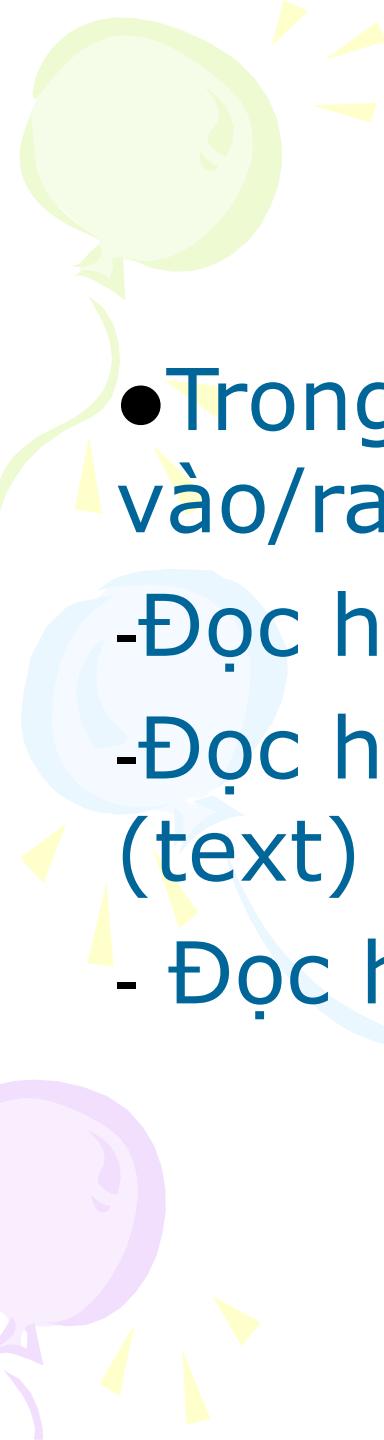
chế độ	Mô tả
"rb"	Mở 1 file binary đã có để đọc
“wb”	Tạo 1 file binary để ghi
“ab”	Mở 1 file đã có để thêm dữ liệu vào cuối
“r+b”	Mở 1 file binary đã có để đọc hoặc ghi
"w+b"	Tạo 1 file binary để đọc hoặc ghi
"a+b"	Tạo mới hoặc mở 1 file đã có để thêm dữ liệu vào cuối

Closing a file

- Lệnh `fclose` dùng để ngắt kết nối của con trỏ file tới file.
- `int fclose(FILE *stream);`

Example: File Open and Close

```
1: /* Opening and closing a file */
2: #include <stdio.h>
3:
4: enum {SUCCESS, FAIL};
5:
6: main(void)
7: {
8:     FILE *fptr;
9:     char filename[] = "haiku.txt";
10:    int reval = SUCCESS;//thành công
11:
12:    if ((fptr = fopen(filename, "r")) == NULL){
13:        printf("Cannot open %s.\n", filename);
14:        reval = FAIL;//thất bại
15:    } else {
16:        printf("giá trị của fptr: 0x%p\n", fptr);//tham số %p :in ra địa chỉ
của con trỏ fptr
17:        printf("Sẵn sàng đóng file");
18:        fclose(fptr);
19:    }
20:
21:    return reval;
22: }
```



Đọc và ghi file

- Trong C có thể thực hiện phép toán vào/ra theo các cách sau:
 - Đọc hoặc ghi từng kí tự một.
 - Đọc hoặc ghi từng dòng văn bản (text) một.
 - Đọc hoặc ghi từng khối kí tự một.
- 

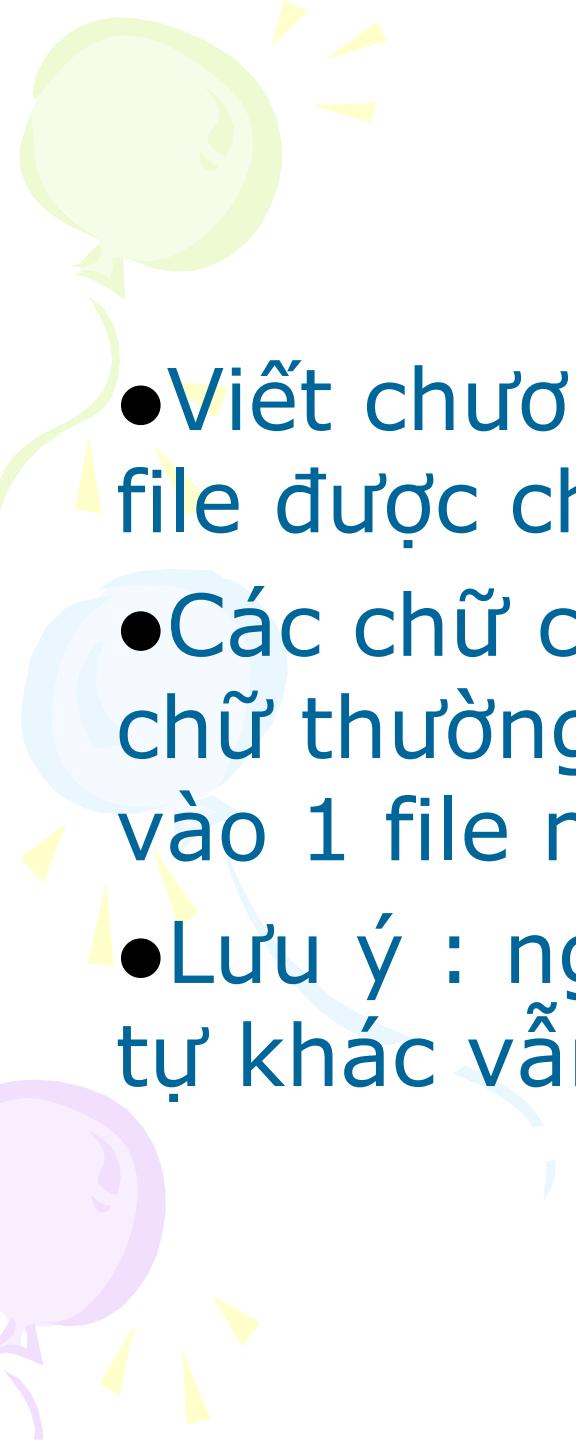
Các phép toán trên kí tự trong UNIX

- Đọc hoặc ghi từng kí tự một:
 - Vào/ra kí tự:
 - fgetc();//đọc từ file 1 kí tự.
`int fgetc(FILE *stream);`
 - fputc();//ghi 1 kí tự vào file.
`int fputc(int c , FILE *stream);`



Exercise F1

- Tạo 1 file lab1.txt với nội dung tùy ý.
- Viết 1 chương trình đọc từng kí tự của file trên rồi ghi vào file lab1w.txt



Exercise F2

- Viết chương trình đọc từng kí tự từ 1 file được chỉ định.
- Các chữ cái hoa được chuyển thành chữ thường và ngược lại. Viết kết quả vào 1 file mới.
- Lưu ý : ngoài các chữ cái ra thì các kí tự khác vẫn được giữ nguyên.

Đọc hoặc ghi từng dòng mỗi lần

- 2 hàm fgets() và fputs()
- `char *fgets(char *s, int n, FILE *stream); //đọc n kí tự từ file stream vào xâu s`
 - s: tham chiếu tới 1 mảng kí tự
 - n: số phần tử lớn nhất của mảng
 - Hàm fgets() có thể đọc tới n-1 kí tự, và có thể thêm “kí tự” NULL(là \0) vào sau kí tự cuối cùng được tìm nạp, cho đến tận khi sang dòng mới hoặc kết thúc file.

Đọc hoặc ghi từng dòng mỗi lần

- `int fputs(const char *s,
FILE*stream);`

// ghi xâu s vào file stream.(1 dòng)

- s: array that contains the characters to be written to a file

- trả về giá trị:

- 0 nếu thành công
- #0 nếu ghi thất bại.

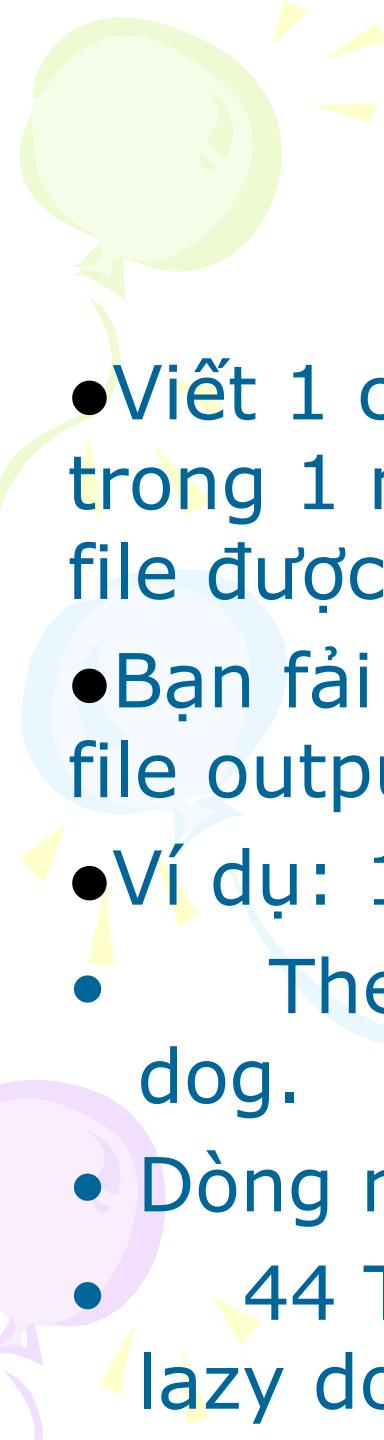


Exercise F3

- Làm lại bài F1 nhưng đổi lại đọc các kí tự theo từng dòng.

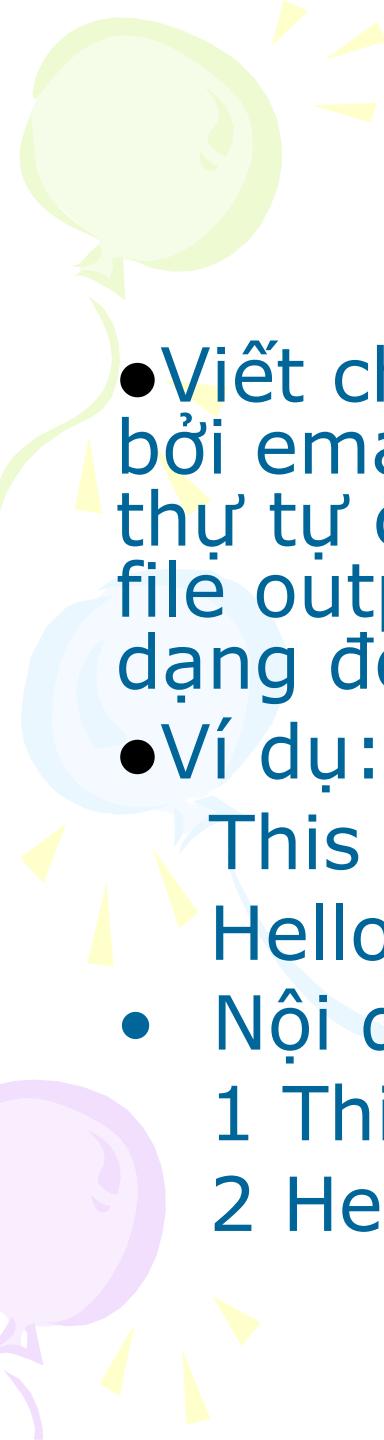
Đọc và ghi văn bản đã được định dạng

- `int fscanf(FILE *stream, const char*format, ...);`
- Vd: `fscanf(ptr,"%d%s",&i,s);`
 - Hàm này làm việc giống như scanf, chỉ khác là nó đọc từ 1 con trỏ file.
- `int fprintf(FILE *stream, const char*format, ...);`
- Vd: `fprintf(ptr,"%d%s",i,s);`
 - Điểm khác biệt duy nhất giữa fprintf và printf là fprintf có thể chuyển hướng đầu ra tới 1 dòng khác biệt.



Exercise F4

- Viết 1 chương trình đọc 2 hoặc nhiều dòng trong 1 mảng các xâu kí tự (từng dòng 1) từ 1 file được chỉ định và tìm ra chiều dài mỗi dòng.
- Bạn phải viết độ dài dòng trước mỗi dòng trong file output.
- Ví dụ: 1 dòng trong file input là:
 - The quick brown fox jumps over the lazy dog.
 - Dòng này khi được ghi vào file output là:
 - 44 The quick brown fox jumps over the lazy dog.



Homework

- Viết chương trình đọc 1 file text được tạo bởi emacs (ví dụ emacs abc.txt), thêm số thứ tự của dòng vào trước mỗi dòng trong file output. Tên file text được nhập vào dưới dạng đối số của hàm main.
- Ví dụ: file input:

This is sample file.

Hello!

- Nội dung file output:
 - 1 This is sample file.
 - 2 Hello!