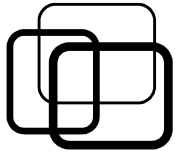
Xử lý tập tin

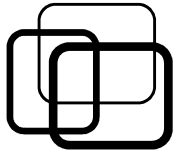
GV. Nguyễn Minh Huy

Nội dung



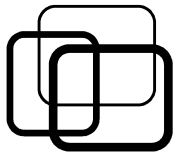
- Thiết bị và stream.
- File stream.
- Các thao tác trên tập tin.

Nội dung



- **Thiết bị và stream.**
- File stream.
- Các thao tác trên tập tin.

Thiết bị và stream

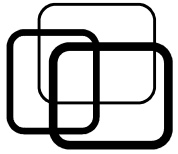


- Khái niệm thiết bị:
  - Dữ liệu → Chương trình → Kết quả.
  - Chương trình lấy dữ liệu từ đâu?
  - Chương trình xuất kết quả ra đâu?

    ➔ Thiết bị (device).
  - Phân loại thiết bị:
    - Thiết bị nhập: bàn phím, con chuột, tập tin, …
    - Thiết bị xuất: màn hình, máy in, tập tin, …

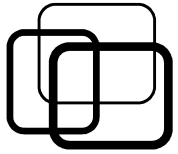
    ➔ Tập tin là thiết bị vừa nhập vừa xuất.

Thiết bị và stream



- Khái niệm stream:
  - Chương trình đọc/ghi dữ liệu từ thiết bị thế nào?
  - ➔ Thông qua những “dòng chảy” dữ liệu.
  - ➔ Stream.
  - Phân loại stream:
    - Stream nhập: “dòng chảy” từ thiết bị nhập.
    - Stream xuất: “dòng chảy” đến thiết bị xuất.

Thiết bị và stream



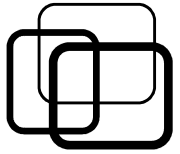
- Khái niệm stream:
  - Các stream định nghĩa sẵn trong C:

Stream	Ý nghĩa	Thiết bị kết nối
stdin	Stream nhập chuẩn.	Bàn phím
stdout	Stream xuất chuẩn.	Màn hình
stderr	Stream lỗi chuẩn.	Màn hình
stdprn	Stream in chuẩn	Máy in

  - Lệnh nhập xuất tổng quát:
    - **fscanf(**<Stream>, "<Định dạng kiểu>", &<Biến 1>, …**)**;
    - **fprintf(**<Stream>, "<Định dạng xuất>", <Biến 1>, …**)**;

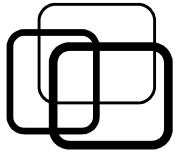
```
fprintf( stdout, "Hello World" );        // Xuất ra màn hình.
fprintf( stdprn, "Hello World" );        // Xuất ra máy in.
```

Nội dung



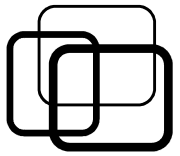
- Thiết bị và stream.
- **File stream.**
- Các thao tác trên tập tin.

File stream



- Nhập xuất bằng tập tin:
  - Ưu điểm:
    - Chứa được nhiều dữ liệu.
    - Không cần thông qua người dùng.
    - Lưu trữ được lâu dài.
  - Khuyết điểm:
    - Tốc độ xử lý chậm.

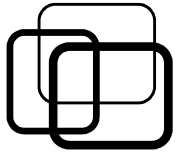
File stream



- Nhập xuất tập tin trong C:
  - Dùng file stream.
    - "Dòng chảy" kết nối chương trình và tập tin.
    - Khai báo: **FILE** *<Tên stream>.

      FILE *f1;
      FILE *f2;
  - Các bước xử lý tập tin:
    - Bước 1: mở tập tin.
    - Bước 2: thao tác trên tập tin.
    - Bước 3: đóng tập tin.

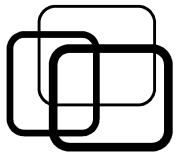
File stream



- Lệnh mở tập tin:
  - Thiết lập kết nối đến tập tin.
    - Cú pháp: **fopen(**“<Đường dẫn tập tin>”, “<Chế độ mở>”**)**;
    - Trả về: file stream (thành công), NULL (thất bại).
    - Bảng các chế độ mở tập tin:

Chế độ mở	Ý nghĩa
r	Read-only, mở để đọc dữ liệu (kiểu text). Trả về NULL nếu không tìm thấy tập tin.
w	Write-only, mở để ghi dữ liệu (kiểu text). Tập tin sẽ được tạo nếu chưa có, ngược lại sẽ ghi đè.
a	Append-only, mở để ghi thêm dữ liệu (kiểu text). Tập tin sẽ được tạo nếu chưa có.
[Chế độ mở]+	Kết hợp đọc, ghi cùng lúc.
[Chế độ mở]b	Đọc, ghi kiểu nhị phân (binary).

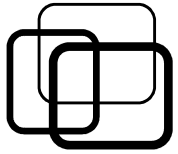
File stream



- Lệnh đóng tập tin:
  - Ngắt kết nối đến tập tin.
    - Cú pháp: **fclose(**<File stream>**)**;
    - Chỉ đóng được tập tin đang mở.
    - Luôn đóng tập tin sau khi kết thúc công việc.

```
FILE  *f = fopen(“C:\\BaiTap.txt”, “r”);
if ( f == NULL )
{
    printf(“Khong mo duoc tap tin!\n”);
    fclose( f );          // Sai.
}
else
{
    printf(“Da mo duoc tap tin!\n”);
    fclose( f );          // Đúng.
}
```

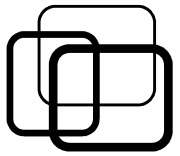
File stream



- Vị trí nhập xuất hiện hành:
  - Điểm kết nối của File stream vào tập tin.
  - Nhập xuất luôn bắt đầu từ vị trí hiện hành.
  - Vị trí hiện hành thay đổi sau mỗi thao tác nhập xuất.

Chế độ mở	Vị trí nhập xuất hiện hành
r	Đầu tập tin.
w	Đầu tập tin.
a	Cuối tập tin.
[Chế độ mở]+	Tùy thuộc chế độ mở.
[Chế độ mở]b	Tùy thuộc chế độ mở.

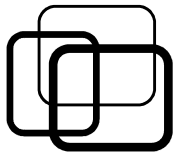
File stream



- Lệnh thay đổi vị trí hiện hành:
  - Thay đổi điểm kết nối File stream vào tập tin.
    - Cú pháp: **fseek(**<File stream>, <Độ dời>, <Vị trí gốc>**)**;
    - <Vị trí gốc>:
      - SEEK_SET (đầu tập tin).
      - SEEK_CUR (vị trí hiện hành).
      - SEEK_END (cuối tập tin).
    - Chỉ thay đổi vị trí hiện hành với tập tin đang mở.

```
FILE  *f = fopen(“C:\\BaiTap.txt”, “r”);
if ( f != NULL )
{
     fseek( f, 2, SEEK_CUR );          // Tiến tới 2 bytes.
     fclose( f );
}
```

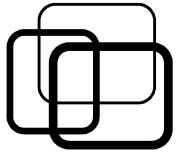
File stream



- Lệnh thông báo vị trí hiện hành:
  - Thông báo vị trí hiện hành so với đầu tập tin.
    - Cú pháp: **ftell(**<File stream>**)**;
    - Chỉ thông báo vị trí hiện hành với tập tin đang mở.

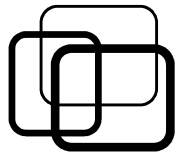
```
FILE  *f = fopen(“C:\\BaiTap.txt”, “r”);
if ( f != NULL )
{
    fseek( f, 0, SEEK_END );
    long  filesize = ftell( f );          // Lấy kích thước tập tin.
    fclose( f );
}
```

Nội dung



- Thiết bị và stream.
- File stream.
- **Các thao tác trên tập tin.**

Các thao tác trên tập tin



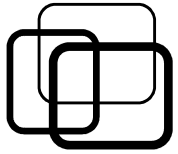
- Các thao tác cơ bản:
  - Đọc dữ liệu: theo định dạng.
    - Cú pháp:

      fscanf(<File stream>, "<Định dạng kiểu>", &<Biến 1>, …**)**;
    - Trả về: số biến đọc được (thành công), EOF (thất bại).

```
int     a, b;
FILE  *f = fopen("C:\\BaiTap.txt", "r");

if ( f == NULL )
    printf("Khong mo duoc tap tin!\n");
else
{
    fscanf( f, "%d %d", &a, &b );
    fclose( f );
}
```

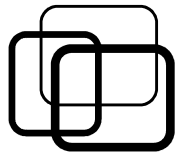
Các thao tác trên tập tin



- Các thao tác cơ bản:
  - Đọc chuỗi:
    - Cú pháp: **fgets(**<Biến chuỗi>, <Số ký tự>, <File stream>**)**;
    - Ý nghĩa: đọc chuỗi đến khi gặp ký tự \n hoặc đủ <Số ký tự>.
    - Trả về: độ dài chuỗi (thành công), EOF (thất bại).

```
char   s[100];
FILE  *f = fopen("C:\\BaiTap.txt", "r");
if ( f != NULL )
{
     fgets( s, 99, f );
     fclose( f );
}
```

Các thao tác trên tập tin

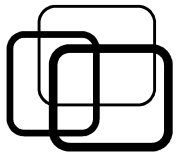


- Các thao tác cơ bản:
  - Ghi dữ liệu: theo định dạng.
    - Cú pháp:

      fprintf(<File stream>, "<Định dạng kiểu>", <Biến 1>, …**)**;
    - Trả về: số byte ghi được (thành công), EOF (thất bại).

```
int     a = 12;
float   b = 4.165;
char    c = 'A';
FILE  *f = fopen("C:\\BaiTap.txt", "w");
if ( f == NULL )
    printf("Khong tao duoc tap tin!\n");
else
{
    fprintf( f, "Gia tri = %d %f %c", a, b, c );
    fclose( f );
}
```

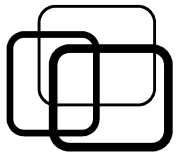
Các thao tác trên tập tin



- Một số lệnh quản lý tập tin:
  - Xóa tập tin:
    - Cú pháp: **remove(**“<Đường dẫn tập tin>”**)**;
    - Trả về: 0 (thành công), -1 (thất bại).
    - Không cần phải mở và đóng tập tin.

```
if (remove(“C:\\BaiTap1.txt”) == 0)
      printf(“Da xoa thanh cong!\n”);
else
      printf(“Khong xoa duoc!\n”);
```

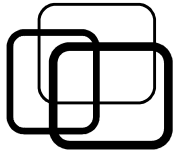
Các thao tác trên tập tin



- Một số lệnh quản lý tập tin:
  - Đổi tên tập tin:
    - Cú pháp: **rename(**“<Đường dẫn cũ”, “<Đường dẫn mới>”**)**;
    - Hai đường dẫn phải cùng một ổ đĩa.
    - Trả về: 0 (thành công), -1 (thất bại).
    - Có thể dùng để di chuyển tập tin.
    - Không cần phải mở và đóng tập tin.

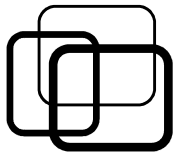
```
if (rename(“C:/BaiTap1.txt”, “C:/BaiTap\\BaiTap2.txt”) == 0)
      printf(“Da doi ten thanh cong!\n”);
else
      printf(“Khong doi ten duoc!\n”);
```

Tóm tắt



- Thiết bị và stream:
  - Chương trình trao đổi dữ liệu với thiết bị.
  - Stream kết nối chương trình và thiết bị.
- File stream:
  - Kết nối chương trình và tập tin.
  - Các bước: mở tập tin, thao tác tập tin, đóng tập tin.
- Các thao tác trên tập tin:
  - Đọc: fscanf, fgets.
  - Ghi: fprintf.
  - Xóa: remove
  - Đổi tên/Di chuyển: rename.

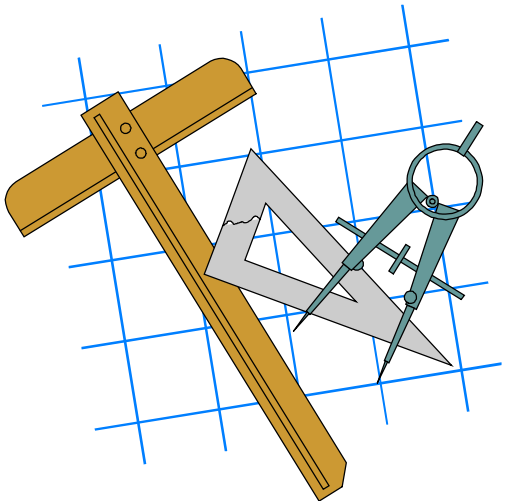
Bài tập



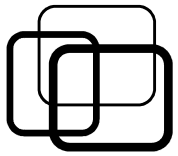
- **Bài tập 9.1:**

  Viết chương trình C tìm số nguyên tố như sau:

  - Đọc vào 2 số nguyên dương M, N từ file input.txt.
  - Xuất các số nguyên tố trong đoạn [M, N] vào file output.txt.



Bài tập



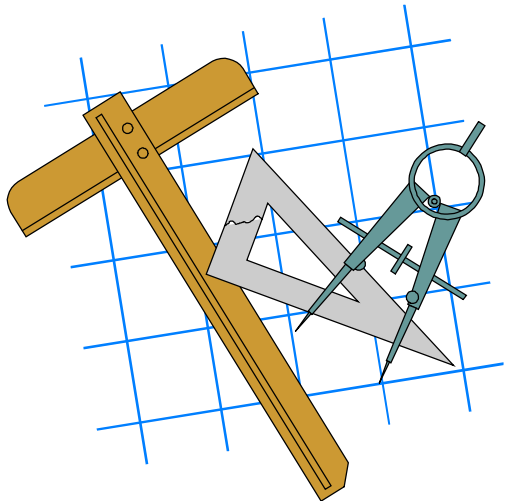
Bài tập 9.2:

Ma trận M x N được lưu trong file matrix.txt như sau:

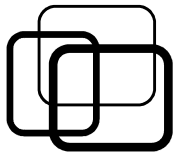
- Dòng đầu tiên chứa: M N (kích thước ma trận).
- M dòng tiếp theo, dòng i (0 <= i <= M – 1) chứa: a_{i1} a_{i2} … a_{iN} (N phần tử thuộc dòng i của ma trận).

Viết chương trình C thao tác ma trận như sau:

- Nhập vào ma trận M x N từ file matrix.txt.
- Xuất phần tử có giá trị bằng tổng các phần tử còn lại trong ma trận ra file result.txt.



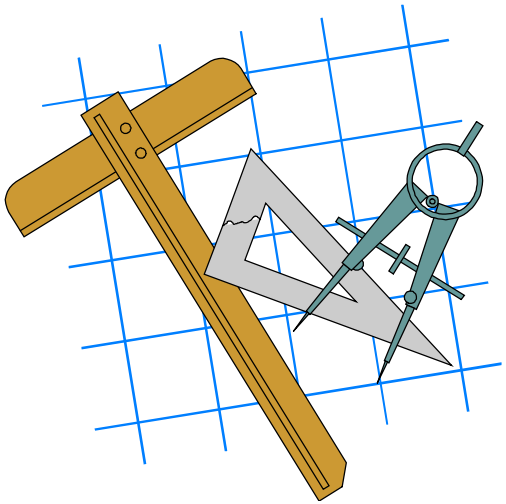
Bài tập



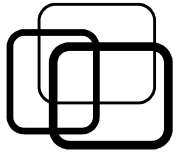
- **Bài tập 9.3:**

  Viết chương trình C nối 2 files như sau:

  - Nhập vào đường dẫn file 1.
  - Nhập vào đường dẫn file 2.
  - Thực hiện nối nội dung file 2 vào cuối file 1.



Bài tập



Bài tập 9.4:

Hoàng đế Ceasar thường dùng mật lệnh để trao đổi với các tướng lĩnh ở chiến trường xa. Để bảo vệ mật lệnh không bị lộ, Ceasar sử dụng cách thức mã hóa như sau:

- Thống nhất với các tướng lĩnh một số nguyên K để làm khóa.
- Khi viết mật lệnh, mỗi chữ cái gốc (A – Z) sẽ được dịch chuyển sang phải một khoảng K để trở thành chữ cái mã hóa.
- Để giải mã, mỗi chữ cái mã hóa trong mật lệnh sẽ được dịch chuyển ngược lại sang trái một khoảng K để trở về chữ cái gốc.

Hãy viết chương trình C mã hóa và giải mã như sau:

a) Tạo file mật lệnh mã hóa từ một file văn bản.

b) Giải mã một file mật lệnh thành file văn bản.