TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# TIN HỌC ĐẠI CƯƠNG

## Phần 2. Lập trình

## Bài 8. Tệp dữ liệu

# Nội dung

8.1. Khái niệm và phân loại tệp

8.2. Các thao tác cơ bản với tệp

8.3. Tệp văn bản

8.4. Tệp nhị phân

# 8.1. Khái niệm tệp

- Tệp dữ liệu (File) là một tập hợp các dữ liệu có liên quan với nhau và có cùng kiểu dữ liệu.
- Tệp được lưu trữ trên các ***thiết bị nhớ ngoài*** (đĩa cứng, USB, thẻ nhớ …) với một tên nào đó để phân biệt với nhau.
- Vai trò của tệp: Tệp là phương tiện để cất giữ ***dữ liệu lâu dài***, dữ liệu lưu trữ trong tệp ***không bị mất đi*** khi chương trình kết thúc hay khi tắt máy

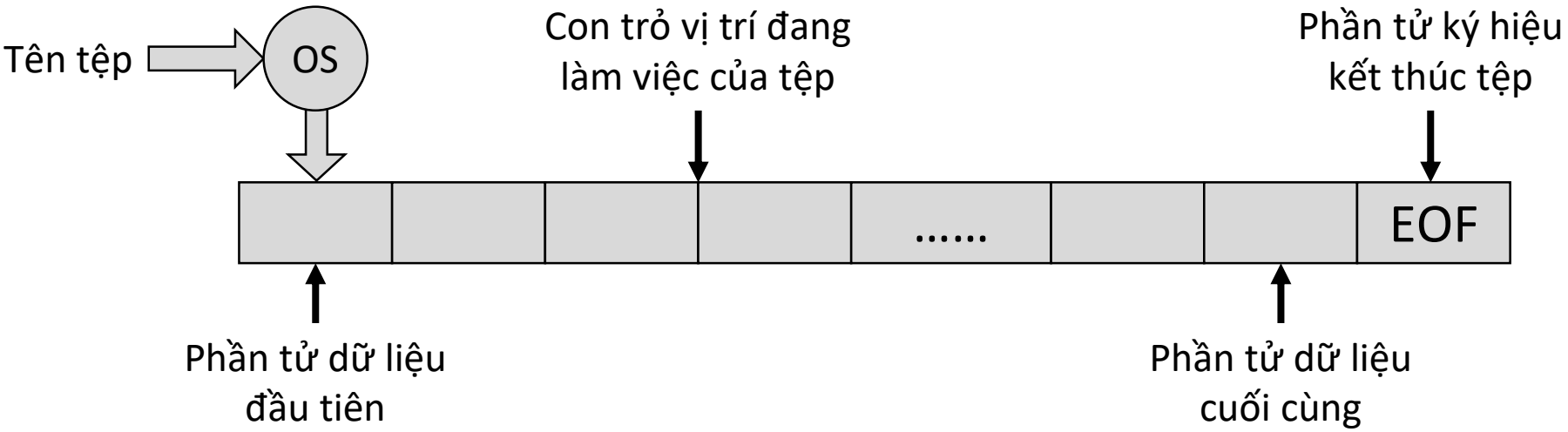
# Phân loại tệp: tệp văn bản và tệp nhị phân

- Tệp văn bản (text file): chỉ chứa các ký tự như chữ cái, chữ số, các dấu câu, dấu cách …
- Tệp nhị phân (binary file): thông tin được mã hóa nhị phân bao gồm số nguyên, số thực, ...

# Phân biệt giữa tệp và mảng

- Tệp lưu trữ ở bộ nhớ ngoài, do vậy dữ liệu trên tệp tồn tại lâu dài.
- Mảng lưu trữ ở bộ nhớ trong nên dữ liệu trong mảng sẽ không còn khi chương trình kết thúc hoặc khi tắt máy.
- Kích thước của tệp có thể rất lớn do bộ nhớ ngoài có dung lượng lớn. Còn kích thước của mảng bị giới hạn bởi kích thước bộ nhớ trong.

# Cấu trúc của tệp, Con trỏ tệp




- Các phần tử của tệp tạo thành một dãy, tại một thời điểm chỉ có thể truy cập được vào một phần tử của tệp (thông qua biến gọi là con trỏ tệp).

# Con trỏ tệp

- Khi mở tệp, con trỏ tệp sẽ luôn trỏ đến phần tử đầu tiên của tệp.
- Sau mỗi thao tác đọc/ghi trên tệp, con trỏ tệp sẽ tự động dịch chuyển về phía cuối tệp.
- Khoảng cách dịch chuyển (tính theo byte) sẽ bằng số byte đã được đọc từ tệp hoặc ghi lên tệp.

# 8.2. Quy trình làm việc với tệp

- Khai báo tệp
- Mở tệp để làm việc
- Truy nhập tệp
- Đóng tệp

# 8.2. Quy trình làm việc với tệp

- Khai báo con trỏ tệp:
  - Cú pháp: `FILE *tên_con_trỏ_tệp;`
  - Ví dụ:

```
FILE *f1, *f2;
```

# 8.2. Quy trình làm việc với tệp

- Mở tệp:
  - Cú pháp:

```
tên_con_trỏ_tệp = fopen(tên_tệp, chế_độ_mở_tệp);
```

  - **tên_tệp**: là đường dẫn đầy đủ đến tệp trên đĩa, nếu chỉ có tên tệp chương trình sẽ tìm trong thư mục hiện tại.
  - **chế_độ_mở_tệp**: “r”, “w”, “a”, “rb”, “wb”, “ab”, “r+b”, “w+b”, “a+b”, “rt”, “wt”, “at”, “r+t”, “w+t”, “a+t”.
  - “b”: tệp nhị phân, “t”: tệp văn bản.
  - Nếu mở thành công, hàm trả về con trỏ tệp tương ứng với tệp được mở, nếu không thì trả về NULL.

# Chế độ mở tệp

| Kí hiệu | Mục đích sử dụng tệp |
|---|---|
| “r” | Mở tệp đã có để đọc, không được ghi. Nếu tệp không tồn tại, hàm fopen() sẽ trả lại trạng thái lỗi. |
| “w” | Mở tệp mới để ghi. Nếu tệp đã tồn tại, nội dung của nó sẽ bị xóa hết. |
| “a” | Mở tệp để ghi thêm dữ liệu vào cuối tệp. Nếu tệp chưa tồn tại, nó sẽ được tạo mới. |
| “r+” | Mở tệp để vừa đọc vừa ghi. Nếu tệp chưa tồn tại thì sẽ báo lỗi. |
| “w+” | Mở tệp để vừa đọc vừa ghi. Nếu tệp đã tồn tại, nội dung của nó sẽ bị xóa hết. |
| “a+” | Mở tệp để ghi thêm dữ liệu vào cuối tệp. Tệp mới sẽ được tạo nếu nó chưa tồn tại. |

# Lưu ý

- Khi mở tệp, nếu không chỉ rõ bản chất dữ liệu của tệp thì C sẽ ngầm hiểu đó là tệp văn bản.
- Nếu muốn chỉ rõ tệp nhị phân thì sử dụng thêm kí hiệu “b” đi kèm khi mở tệp bằng câu lệnh **fopen**
- Ví dụ:
  - mở tệp văn bản để đọc:

```
f1 = fopen(“c:\\abc.txt”, “r”);
```

  - mở tệp văn bản để vừa đọc và ghi:

```
f2 = fopen(“c:\\abc.txt”, “r+”);
```

  - mở tệp nhị phân để ghi:

```
f3 = fopen(“c:\\ho_so.dat”, “wb”);
```

# Kiểm tra lỗi phát sinh khi mở tệp

```
// Trường hợp mở tệp có lỗi
if (con_trỏ_tệp = fopen(tên_tệp, chế_độ_mở_tệp)
== NULL) {
    // Xử lý cho trường hợp mở tệp không thành
công
} else {
    // Xử lý khi mở tệp thành công
}
```

# 8.3. Thao tác với tệp văn bản

- Đọc dữ liệu từ tệp văn bản: **fscanf()**, **fgets()**, **getc()**
- Ghi dữ liệu vào tệp văn bản: **fprintf()**, **fputs()**, **putc()**

# Đọc dữ liệu từ tệp văn bản

- Hàm **fscanf()**:
- Cú pháp:

```
int fscanf(FILE* con_trỏ_tệp, xâu_định_dạng,
[danh_sách_địa_chỉ]);
```

- Giống hàm scanf() nhưng nguồn dữ liệu đến từ tệp
- Hàm trả về số giá được gán thành công vào các địa chỉ
- Ví dụ:

```
ret = fscanf(fptr, “%d %c”, &a, &c);
```

# Đọc dữ liệu từ tệp văn bản

- Hàm **fflush()**:
- Cú pháp:

```
int fflush(FILE* con_trỏ_tệp);
```

- Ghi toàn bộ dữ liệu chứa trong vùng đệm của tệp (nằm ở bộ nhớ trong) tương ứng với con trỏ tệp ra vùng nhớ của tệp trên bộ nhớ ngoài.
- Giá trị trả về:
  - thực hiện thành công: trả về 0
  - không thành công: trả về EOF

# Đọc dữ liệu từ tệp văn bản

- Hàm **fgets():**
- Cú pháp:

```
char* fgets(char* xâu_kí_tự, int n, FILE*
con_trỏ_tệp);
```

- Đọc từ tệp một xâu kí tự (cho phép chứa dấu cách) và lưu vào biến xâu_kí_tự.
- Việc đọc tệp sẽ dừng lại khi fgets() đọc đủ n-1 kí tự hoặc khi gặp kí tự xuống dòng.
- Giá trị trả về:
  - Nếu đọc thành công: trả về xâu kí tự trỏ bởi xâu_kí_tự
  - Nếu có lỗi: trả về con trỏ NULL

# Đọc dữ liệu từ tệp văn bản

- Hàm **getc():**
- Cú pháp:

```
int getc(FILE* con_trỏ_tệp);
```

- Đọc từ tệp một kí tự (tức là một byte dữ liệu), trả về giá trị số nguyên tương ứng (mã ASCII)
- Giá trị trả về:
  - Thành công: trả về kí tự đọc được sau khi đã chuyển sang dạng int.
  - Nếu không thành công: trả về EOF

# Ghi dữ liệu lên tệp văn bản

- Hàm **fprintf()**
- Cú pháp:

```
int fprintf(FILE* con_trỏ_tệp,
xâu_định_dạng, [danh_sách_tham_số]);
```

- Nếu thành công, hàm fprintf() trả về một giá trị số nguyên là số byte dữ liệu đã ghi lên tệp.
- Nếu không thành công, hàm fprintf() trả về giá trị EOF.

# Ghi dữ liệu lên tệp văn bản

- Hàm **fputs()**
- Cú pháp:

```
int fputs(char* xâu_kí_tự, FILE* con_trỏ_tệp);
```

- Ghi nội dung của xâu_kí_tự lên tệp tương ứng với con_trỏ_tệp
- Không tự động ghi thêm kí tự xuống dòng lên tệp.
- Giá trị trả về:
  - Nếu thành công: trả về kí tự cuối cùng mà đã ghi được lên tệp.
  - Nếu không thành công: trả về EOF.

# Ghi dữ liệu lên tệp văn bản

- Hàm **putc()**
- Cú pháp:

```
int putc(int ch, FILE* con_trỏ_tệp);
```

- Ghi nội dung của kí tự chứa trong biến int ch (kí tự được chứa trong byte thấp của biến ch) lên tệp tương ứng với con_trỏ_tệp.
- Giá trị trả về:
  - Nếu thành công: trả về số nguyên (kiểu int) là số thứ tự trong bảng mã ASCII của kí tự đã ghi lên tệp.
  - Nếu không thành công: trả về giá trị EOF

# Hàm **feof()**

- Cú pháp:

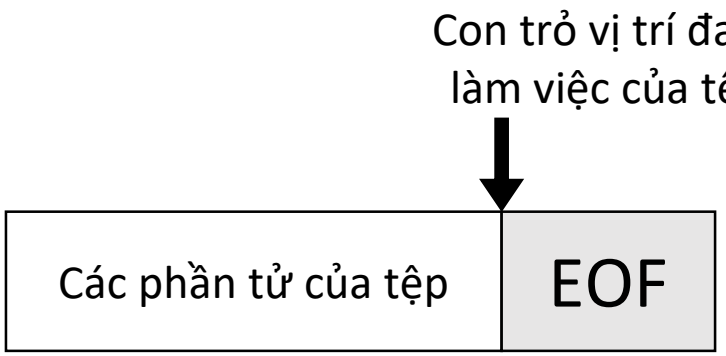
```
int feof(FILE* con_trỏ_tệp);
```

- Hàm feof() dùng để kiểm tra xem đã duyệt đến vị trí cuối tệp hay chưa.
- Kiểm tra xem trong khối dữ liệu được đọc vào ở lần thực hiện gần nhất có phần tử EOF hay không (tức là phần tử EOF có được đọc trong lần đọc gần đây nhất hay không),
- Nếu có thì hàm feof() trả về một giá trị khác 0 (TRUE)
- Nếu chưa thì trả về giá trị 0 (FALSE)

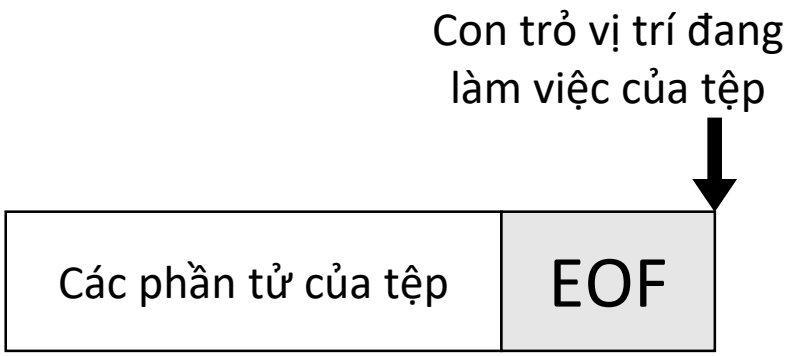
# Hàm **feof()**

Thường lầm tưởng hàm feof() kiểm tra việc đã duyệt đến phần tử cuối cùng của tệp hay chưa bằng cách kiểm tra xem con trỏ tệp đã trỏ đến phần tử cuối cùng hay chưa.

Tuy nhiên khi con trỏ vừa đọc xong phần tử cuối cùng, thì phần tử EOF vẫn chưa được đọc.




Chưa đọc phần tử EOF
feof() = 0 => FALSE




Đã đọc phần tử EOF
feof() != 0 => TRUE

# 8.4. Truy cập tệp nhị phân

- Đọc dữ liệu : hàm **fread()**
- Cú pháp

```
int fread(void *địa_chỉ_biến, int
kích_thước_phần_tử, int số_phần_tử, FILE*
con_trỏ_tệp);
```

- Hàm fread() đọc từ tệp một khối dữ liệu kích thước **số_phần_tử x kích_thước_phần_tử** byte, sau đó ghi khối dữ liệu đó lên vùng nhớ có địa chỉ là **địa_chỉ_biến**.
- Kết quả trả về:
  - Nếu đọc thành công, hàm fread() trả về một giá trị nguyên là số mục (không phải số byte) đọc được từ tệp.
  - Nếu không thành công: hàm trả về giá trị 0.

# 8.4. Truy cập tệp nhị phân

- Ghi dữ liệu : hàm **fwrite()**
- Cú pháp:

```
fwrite(void *địa_chỉ_biến, int
kích_thước_phần_tử, int số_phần_tử, FILE*
con_trỏ_tệp);
```

- Ghi khối dữ liệu có kích thước **số_phần_tử x kích_thước_phần_tử** byte từ vùng nhớ có địa chỉ là **địa_chỉ_biến** vào tệp.
- Kết quả trả về:
  - Nếu ghi dữ liệu lên tệp thực hiện thành công: hàm trả về một giá trị số nguyên là số mục (không phải số byte) đã ghi lên tệp.
  - Nếu thực hiện không thành công: hàm trả về giá trị 0.

# Nhận xét

Các hàm trong các cặp hàm fread() - fwrite(), fscanf() - fprintf(), fputs() - fgets(), getc() - putc() có chức năng đối ngẫu nhau.

# Hàm **fseek()**

- Cú pháp

```
int fseek(FILE* con_trỏ_tệp, long int n, int vị_trí_ban_đầu);
```

- Dùng để dịch chuyển con trỏ tệp từ **vị_trí_ban_đầu** đi một khoảng cách có độ dài n byte.
- Giá trị trả về:
  - trả về giá trị 0 nếu việc dịch chuyển thành công,
  - trả về giá trị khác 0 nếu việc dịch chuyển không thành công.
- Giá trị của biến n có thể lớn hơn, nhỏ hơn hoặc bằng 0.
  - > 0: hướng của sự dịch chuyển là về phía cuối tệp
  - < 0: dịch chuyển về phía đầu tệp
  - = 0: không dịch chuyển

# Tham số **vị_trí_ban_đầu**

| Tên hằng | Giá trị | Ý nghĩa |
|---|---|---|
| SEEK_SET | 0 | Vị trí ban đầu là đầu tệp |
| SEEK_CUR | 1 | Vị trí ban đầu là vị trí hiện thời của con trỏ vị trí làm việc của tệp |
| SEEK_END | 2 | Vị trí ban đầu là cuối tệp |

## Ví dụ:

```
fseek(file_ptr, 0, SEEK_SET); // di chuyển về đầu tệp
fseek(file_ptr, 0, SEEK_END); // di chuyển về cuối tệp
fseek(file_ptr, 50, SEEK_CUR);
fseek(file_ptr, -40, 2);
```

# Hàm **rewind()**

- Cú pháp:

```
void rewind(FILE* con_trỏ_tệp);
```

- Đưa con trỏ tệp về đầu tệp.
- Với biến file_ptr là một biến con trỏ tệp, hàm rewind(file_ptr) tương đương với

```
fseek(file_ptr, 0, SEEK_SET);
```

- Hàm này không có giá trị trả về.

# Hàm **fclose()**

- Cú pháp:

```
int fclose(FILE* con_trỏ_tệp);
```

- Đóng tệp là đảm bảo những thay đổi dữ liệu được lưu lại trên tệp.
- Hàm trả về giá trị 0 nếu đóng thành công
- Nếu không đóng tệp thành công, hàm trả về giá trị EOF.

# Ví dụ 1: Thao tác với tệp văn bản

Nhập thông tin điểm thi toán, lý, hóa từ bàn phím

Ghi các điểm này vào tệp văn bản có tên "my_score.txt".

Đọc từ tệp trên, tính điểm trung bình và in kết quả ra màn hình.

# Ví dụ 1: Ghi dữ liệu vào file

```c
#include <stdio.h>
void main()
{
    FILE *f;
    float dToan, dLy, dHoa;

    // Nhap diem tu ban phim
    printf("Nhap diem thi Toan, Ly Hoa: ");
    scanf("%f%f%f", &dToan, &dLy, &dHoa);

    // Luu vao file van ban
    f = fopen("C:\\Test\\my_score.txt", "w");
    fprintf(f, "%f\n%f\n%f\n", dToan, dLy, dHoa);
    fclose(f);
}
```

# Ví dụ 1: Đọc dữ liệu từ file

```c
#include <stdio.h>
void main()
{
    FILE *f;
    float dToan, dLy, dHoa, dTb;

    // Doc du lieu tu file van ban
    f = fopen("C:\\Test\\my_score.txt", "r");
    fscanf(f, "%f%f%f", &dToan, &dLy, &dHoa);
    fclose(f);

    // Tinh diem trung binh
    dTb = (dToan + dLy + dHoa) / 3;
    printf("Diem trung binh: %f", dTb);
}
```

# Ví dụ 2: Thao tác với tệp nhị phân

- Tạo tệp nhị phân my_float.dat đặt tại ổ đĩa C để ghi 100 số thực.
- Mở lại tệp này và ghi nội dung sang một tệp khác, tệp này có tên được nhập từ bàn phím.
- Hiển thị giá trị số thực đầu tiên trong tệp my_float.dat
- Cho phép người dùng nhập một số thứ tự từ bàn phím và hiển thị số thực ở vị trí này trong tệp my_float.dat

# Ví dụ 2: Ghi số thực vào file nhị phân

```
#include <stdio.h>
void main() {
    FILE *f;
    int i;
    float a[100];

    // Tao mang chua 100 so thuc
    for (i = 0; i < 100; i++)
        a[i] = i * 2 + 1;

    // Luu vao file nhi phan
    f = fopen("C:\\Test\\my_float.dat", "wb");
    fwrite(a, sizeof(float), 100, f);
    fclose(f);
}
```

# Ví dụ 2: Sao chép dữ liệu giữa 2 file

```
#include <stdio.h>
void main()
{
    FILE *f1, *f2;
    char filename[64];
    float f;

    // Nhap ten file moi
    printf("Nhap ten file moi: ");
    gets(filename);

    // Mo file f1 de doc, f2 de ghi
    f1 = fopen("C:\\Test\\my_float.dat", "rb");
    f2 = fopen(filename, "wb");
```

# Ví dụ 2: Sao chép dữ liệu giữa 2 file (tiếp)

```
    // Doc tung so float tu file 1 luu vao file 2
    while (!feof(f1))
    {
        fread(&f, sizeof(float), 1, f1);
        fwrite(&f, sizeof(float), 1, f2);
    }

    fclose(f1);
    fclose(f2);
}
```

# Ví dụ 2: Đọc số thực biết trước vị trí

```c
#include <stdio.h>
void main()
{
    FILE *f;
    float x, y;
    int n;

    // Mo file de doc
    f = fopen("C:\\Test\\my_float.dat", "rb");

    // Doc so thuc dau tien trong file
    fread(&x, sizeof(float), 1, f);
    printf("So thuc dau tien trong file: %f\n", x);
```

# Ví dụ 2: Đọc số thực biết trước vị trí (tiếp)

```
    // Nhap vi tri va doc so thuc tai vi tri do
    printf("Nhap vi tri can doc: ");
    scanf("%d", &n);

    // Di chuyen ve dau file
    fseek(f, (n - 1) * sizeof(float), SEEK_SET);
    fread(&y, sizeof(float), 1, f);
    printf("So thuc thu %d trong file: %f\n", n, y);

    fclose(f);
}
```

# Bài tập

1. Viết chương trình nhập từ bàn phím N số thực lưu vào một mảng (N < 100 và N được nhập từ bàn phím). Sau đó ghi ra một file văn bản có tên là “float.dat” theo quy cách: dòng đầu tiên lưu số lượng các số thực, các dòng tiếp theo lưu các số thực, mỗi số lưu trên một dòng. Đọc lại tệp văn bản đó và lưu các số thực đọc được vào một mảng. Sắp xếp các số thực trong mảng theo thứ tự tăng dần và ghi ra một tệp văn bản khác có tên là “float_sx.dat” theo quy cách giống như tệp “float.dat”.
=> Áp dụng với mảng các đối tượng cấu trúc sinh viên gồm các trường: MSSV, Hoten, Diemthi

# Bài tập

2. Viết chương trình so sánh nội dung hai file:

a. Nhập từ bàn phím hai xâu kí tự là đường dẫn tới hai file cần so sánh.

b. Hiển thị ra màn hình dòng thông báo:

Hai file ten_1 và ten_2 giống nhau (nếu hai file có nội dung giống nhau, hai file có cùng kích thước và các byte cùng vị trí thì có giá trị giống nhau).

Hai file ten_1 và ten_2 không giống nhau (nếu hai file có nội dung khác nhau. Ở đây ten_1 và ten_2 là đường dẫn của hai file cần so sánh).

# Bài tập

3. Viết chương trình sao chép nội dung tệp mã nguồn chương trình C có tên là file_1.c sang tệp có tên là file_2.c

4. Viết chương trình ghép nối nội dung hai file:

a. Nhập vào từ bàn phím hai xâu kí tự là đường dẫn của file nguồn và file đích.

b. Ghép nội dung của file nguồn vào cuối file đích.

5. Mở một tệp văn bản, đếm xem trong văn bản đó có bao nhiêu kí tự, bao nhiêu từ, bao nhiêu câu.

# Bài tập

6. Đọc kích thước file ảnh bitmap (BMP).
Chiều rộng và chiều cao của file ảnh BMP được lưu trữ trong phần header có địa chỉ offset lần lượt là 0x12 và 0x14 dưới dạng số nguyên 16 bit.
Hãy đọc các giá trị này từ 1 file ảnh BMP và hiển thị trên màn hình.