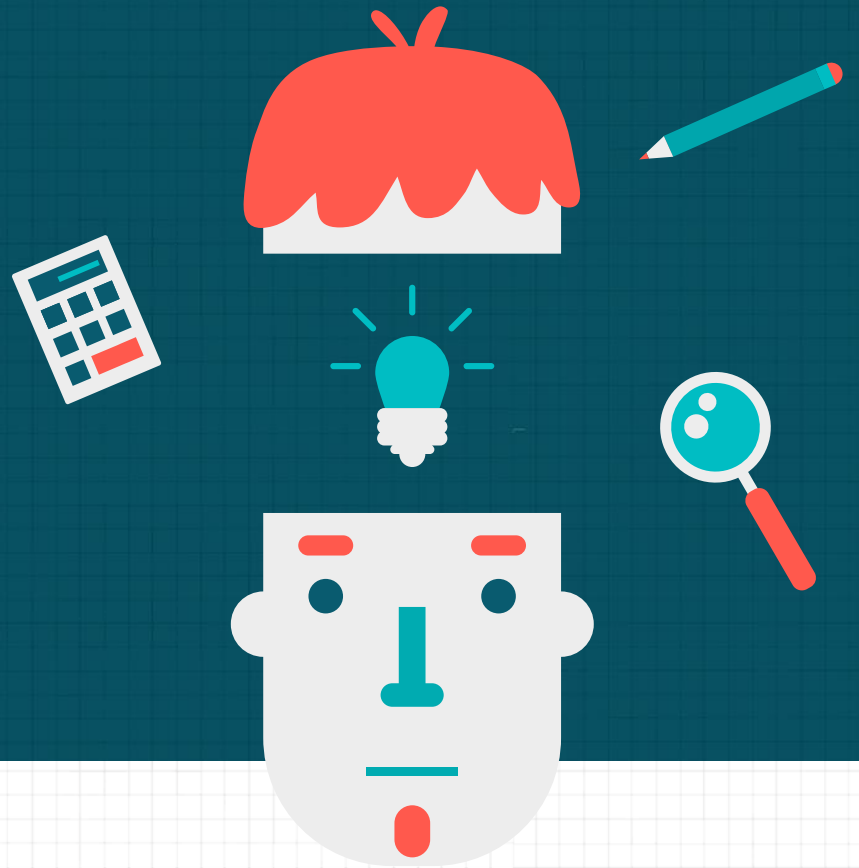
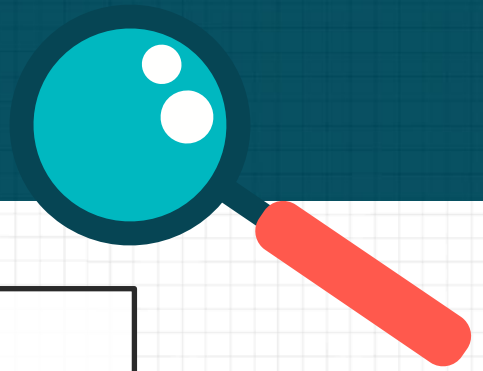
# Bài 3
# Tập tin

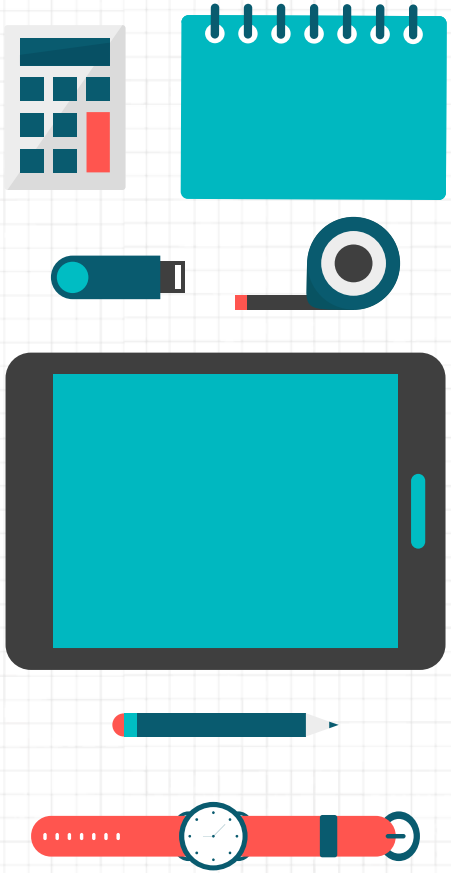


Ths. Phạm Minh Hoàng

# Nội dung



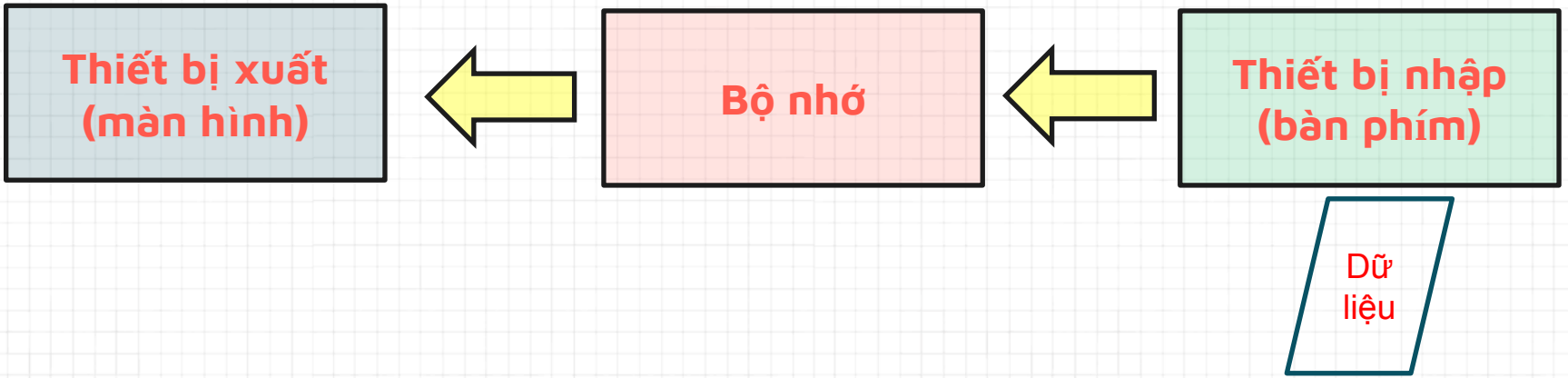
- Các khái niệm cơ bản
- Xử lý trên tập tin
- Thiết kế tập tin
- Tham số dòng lệnh



# 01

## Các khái niệm cơ bản

# Tập tin (file)




- Toàn bộ dữ liệu lưu trên bộ nhớ sẽ mất khi chương trình kết thúc
- Tốn thời gian khi nhập dữ liệu từ bàn phím. Khi cần sửa chữa phải nhập lại từ đầu
- Cần thiết bị lưu trữ cho dữ liệu sau khi kết thúc chương trình, dùng được nhiều lần, và kích thước gần như không hạn chế

# Tập tin (file)

- Khái niệm
  - File là đối tượng lưu trữ dữ liệu được tổ chức theo một dạng nào đó với tên gọi xác định
  - File được lưu trên thiết bị lưu ngoài: ổ cứng, đĩa mềm, USB,…
    - File vẫn tồn tại khi chương trình kết thúc
    - Kích thước file không hạn chế
    - Giao tiếp với các chương trình khác
    - Có thể mang sang máy khác
  - Cho phép đọc và ghi dữ liệu trên file

# Tập tin vs Bộ nhớ

| | Tập tin | Bộ nhớ |
|---|---|---|
| **Tốc độ** | Chậm | Nhanh |
| **Truy xuất** | Tuần tự | Ngẫu nhiên |
| **Chi phí** | Rẻ hơn | Đắt hơn |
| **Dung lượng** | Lớn | Nhỏ |
| **Lưu trữ** | Lâu dài | Nhất thời |

# Phân loại Tập tin (file)

- Theo người dùng: Phân loại theo phần mở rộng của file
  - File chương trình: *.exe
  - File hình ảnh: *.jpg, *bmp, *.png,…
  - Các loại file khác: *doc, *xls, *ppt, …
- Theo người lập trình
  - File văn bản
    - Gồm tập hợp các ký tự tổ chức thành nhiều dòng, mỗi dòng tối đa 255 ký tự
    - Mỗi dòng kết thúc bằng ‘\0’ hoặc ‘\n’
  - File nhị phân
    - Gồm dãy các byte liên tục nhau

# Luồng (stream)

- Chương trình giao tiếp với thiết bị thông qua luồng (stream)
- Luồng (stream) là môi trường trung gian để giao tiếp (nhận/gửi dữ liệu) giữa chương trình và thiết bị
- Stream đảm bảo tính chất độc lập thiết bị. Để gửi/nhận dữ liệu từ 1 thiết bị ta chỉ gửi/nhận dữ liệu từ stream nối giữa chương trình và thiết bị đó
- Stream gồm dãy byte dữ liệu
  - Đưa dữ liệu vào chương trình gọi là stream nhập
  - Gửi dữ liệu ra khỏi chương trình gọi là stream xuất

# Stream nhập xuất

- Stream 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 |

# Stream nhập xuất

- Lệnh nhập/xuất chuẩn trong C: fscanf/fprintf
- Include thư viện stdio.h

```
fscanf(<stream>, "<định dạng kiểu>", &<biến>,…);
fprintf(<stream>, "<định dạng kiểu>", <biến>,…);
```

- Ví dụ

```
#include <stdio.h>
Void main(){
    int n;
    fscanf(stdin, "%d", &n);
    fprintf(stdout, "n = %d", n);
}
```

# Stream nhập xuất

- Stream trong C++

| Stream | Ý nghĩa | Thiết bị kết nối |
|---|---|---|
| cin | Stream nhập chuẩn | Bàn phím |
| cout | Stream xuất chuẩn | Màn hình |
| cerr | Stream lỗi chuẩn | Màn hình |

# Stream nhập xuất

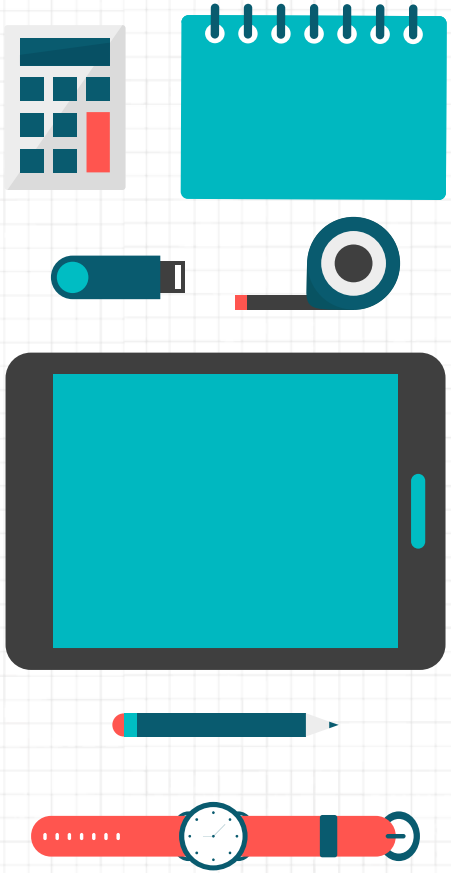
- Stream trong C++
  - include iostream
  - using namspace std
  - Toán tử nhập >>
    - Cú pháp

```
<stream> >> <biến>;
```

  - Toán tử xuất <<
    - Cú pháp

```
<stream> << <biến>;
```

```
#include <iostream>
using namespace std;
void main(){
    int n;
    cout << "Nhập n;";
    cin >> n;
}
```

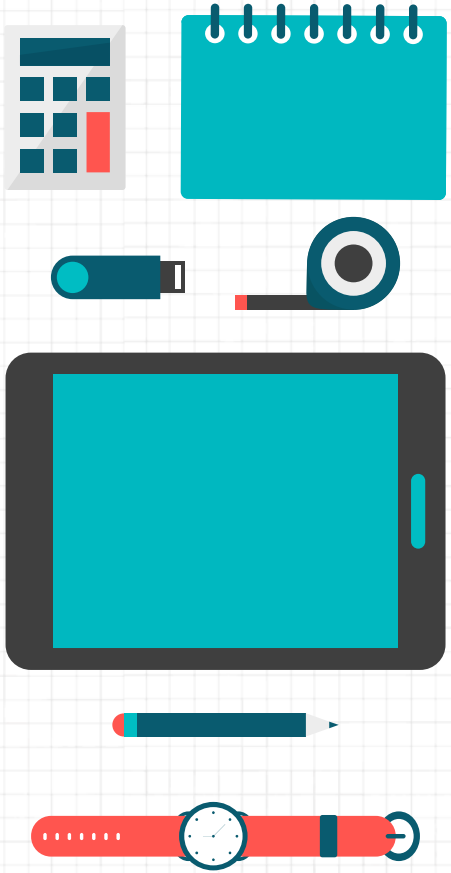


# 02

# Xử lý tập tin

# Các bước xử lý tập tin

- Bước 1: Mở/tạo tập tin: mở file (trong trường hợp file đã có sẵn) hoặc tạo mới file (trong trường hợp file chưa có)
- Bước 2: Thao tác trên tập tin
  - Đọc dữ liệu: đưa dữ liệu lưu trên file vào chương trình
  - Ghi dữ liệu: đưa dữ liệu từ chương trình vào file
- Bước 3: Đóng tập tin: sau khi đã thao tác xong



# 2a

## Mở tập tin (file)

# Mở file trong C

- Trong C, dùng câu lệnh mở file fopen
  - Include thư viện stdio.h
  - Cú pháp

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

  - Ý nghĩa: mở file theo đường dẫn filename, với kiểu mở mode
  - Trả về:
    - Nếu thành công: trả về con trỏ đến cấu trúc FILE
    - Nếu thất bại: trả về NULL

# Mở file trong C

- Mở file trong secure mode
  - Include thư viện stdio.h
  - Cú pháp

```
errno_t fopen_s(FILE** file, <file name>, <mode>);
```

  - Ý nghĩa: mở file theo đường dẫn filename, với kiểu mở mode
  - Trả về:
    - Nếu thành công: trả về 0 và con trỏ đến cấu trúc FILE
    - Nếu thất bại: trả về số khác 0

# Mở file trong C

| Mode | Ý nghĩa |
|---|---|
| b | Mở file kiểu nhị phân (binary) |
| t | Mở file kiểu văn bản (text) (mặc định) |
| r | Mở file chỉ để đọc dữ liệu từ file. Trả về NULL nếu không thấy file |
| w | Mở file chỉ để ghi dữ liệu lên file. File được tạo nếu chưa có, ngược lại dữ liệu bị xóa |
| a | Mở file chỉ để thêm dữ liệu vào cuối file. File sẽ được tạo nếu chưa có |
| r+ | Giống mode r bổ sung thêm tính năng ghi dữ liệu và file được tạo nếu chưa có |
| w+ | Giống mode w bổ sung thêm tính năng đọc |
| a+ | Giống mode a bổ sung thêm tính năng đọc |

# Mở file trong C

- Có thể dùng kết hợp các mode với nhau cho câu lệnh fopen
  - Kết hợp mode read và loại file: “rt”, “rb”, “at”, “ab”, “r+t”, “r+b”, “a+t”, “a+b”
  - Kết hợp mode write và loại file: “wt”, “wb”, “w+t”, “w+b”
- Ví dụ: mở file văn bản để đọc dữ liệu

```
#include <stdio.h>
void main(){
    FILE* f = fopen(”filename.txt”, “rt”);
    if (f == NULL)
        cout << “Cannot open file” << endl;
}
```

# Mở file trong C

- Ví dụ: mở file nhị phân để ghi dữ liệu

```
#include <stdio.h>
void main(){
    FILE* f = NULL;
    errno_t res = fopen_s(&f, ”filename.txt”, “wb”);
    if (res == 0)
        cout << “Cannot open file” << endl;
}
```

# Mở file trong C++

- Trong C++, dùng luồng dành cho file fstream để mở file
  - include <fstream> và using namespace std
  - Các loại luồng trong fstream

| stream | Chức năng |
|---|---|
| ofstream | Tạo stream dùng để ghi dữ liệu lên file |
| ifstream | Tạo stream dùng để đọc dữ liệu từ file |
| fstream | Tạo stram dùng vừa ghi vừa đọc dữ liệu |

# Mở file trong C++

- Cú pháp khai báo stream

```
<stream> <tên stream>;
```

- Mở file bằng stream
  - Dùng câu lệnh open

```
<tên stream>.open(<file name>, <mode>);
```

  - Dùng câu lệnh khai báo stream

```
<stream> <tên stream>(<file name>, <mode>);
```

# Mở file trong C++

- Các mode mở file

| Mode | Chức năng |
|---|---|
| ios::in | Mở file để đọc (mặc định dạng văn bản) |
| ios::out | Mở file để ghi (mặc định dạng văn bản) |
| ios::binary | Mở file dạng nhị phân |
| ios::ate | Mở file và đặt con trỏ vào cuối file |
| ios:app | Mở file và ghi dữ liệu vào cuối file<br>Tạo file mới nếu file không tồn tại |
| ios::trunc | Mở file và xóa nội dung trong file đã mở |

# Mở file trong C++

- Dùng toán tử OR (|) để kết hợp các mode lại với nhau
- Ví dụ: Mở file văn bản để đọc

```
#include <fstream>
using namespace std;
void main(){
    fstream f;
    f.open("example.txt", ios::in); //mở file
    if(f.is_open()){ //kiểm tra mở file thành công
        cout << "Open file sucessfully";
    }
}
```

# Mở file trong C++

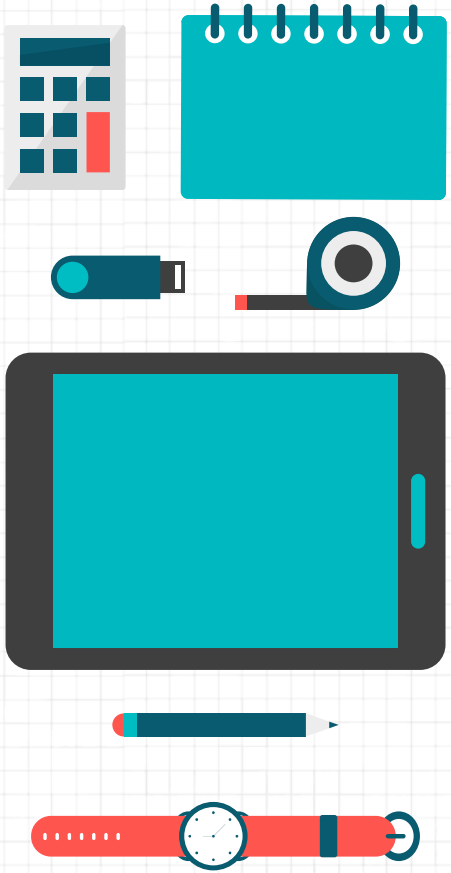
- Ví dụ: Mở file văn bản để ghi

```
#include <fstream>
using namespace std;
void main(){
    ofstream fout("example.txt", ios::out);
    if(fout.is_open()){ //kiểm tra mở file thành công
        cout << "Open file sucessfully";
    }
}
```

# Mở file trong C++

- Ví dụ: Mở file nhị phân để đọc

```
#include <fstream>
using namespace std;
void main(){
    ifstream fin("example.txt", ios::in | ios::binary);
    if(fin.is_open()){ //kiểm tra mở file thành công
        cout << "Open file sucessfully";
    }
}
```



# 2b

## Đóng tập tin

# Đóng file trong C

- Trong C, dùng câu lệnh đóng file fclose
  - Include thư viện stdio.h
  - Cú pháp
```
int fclose(FILE* file);
```
  - Ý nghĩa: đóng file đang do con trỏ file trỏ tới
  - Trả về:
    - Nếu thành công: trả về 0 nếu đóng thành công
    - Nếu thất bại: trả về EOF

# Đóng file trong C

- Ví dụ: đóng file văn bản

```
#include <stdio.h>
void main(){
    FILE* f = fopen(”filename.txt”, “rt”);
    if (f == NULL){
        cout << “Cannot open file” << endl;
        return;
    }
    fclose(f);
}
```

# Đóng file trong C++

- Trong C++, dùng câu lệnh đóng file close
  - Include <fstream>
  - Cú pháp

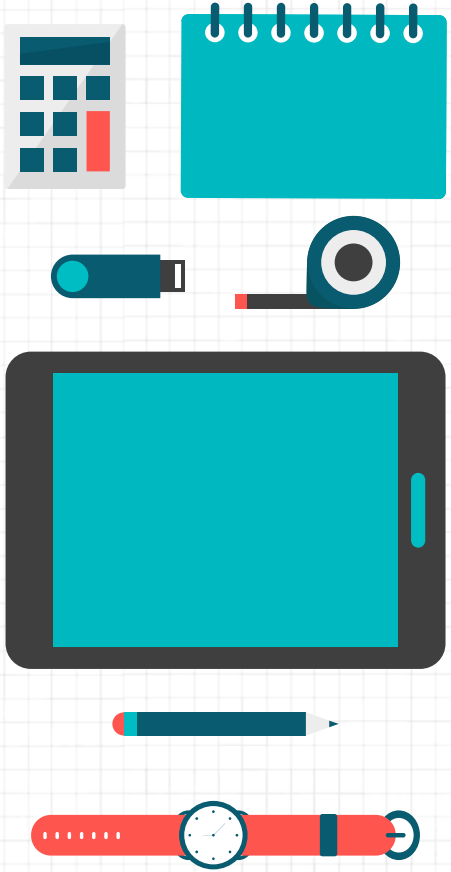
```
void <stream>.close();
```

  - Ý nghĩa: đóng file
  - Trả về: không có

# Đóng file trong C++

- Ví dụ: Mở file nhị phân để đọc

```
#include <fstream>
using namespace std;
void main(){
    ifstream fin("example.txt", ios::in | ios::binary);
    if(fin.is_open()){ //kiểm tra mở file thành công
        cout << "Open file sucessfully";
        return;
    }
    fin.close();
}
```



# 2c

## Đọc ghi file văn bản

# Đọc ghi file văn bản

- Trong C, gồm 3 dạng để đọc ghi dữ liệu trong file văn bản
  - Đọc ghi dữ liệu được định dạng: fscanf/fprintf
  - Đọc ghi ký tự: fgetc/fputc
  - Đọc ghi chuỗi ký tự: fgets/fputs
- Trong C++, đọc ghi file văn bản dùng stream
  - Đọc ghi dữ liệu được định dạng: toán tử >>, <<
  - Đọc ghi ký tự: get/put
  - Đọc ghi chuỗi ký tự: getline/get/toán tử <<

# Đọc ghi dữ liệu định dạng trong C

- Trong C, hàm đọc dữ liệu được định dạng fscanf
  - Cú pháp

```
int fscanf(FILE* fp, <format>, &<var1>, &<var2>,…);
```

  - Ý nghĩa: đọc dữ liệu theo định dạng trong chuỗi format tại vị trí con trỏ file fp đang trỏ tới, và lưu vào các biến var1, var2,… khớp với định dạng. Các biến có thể cách nhau bằng khoảng trắng hoặc ký tư xuống dòng '\n'
  - Trả về:
    - Số lượng biến đã đọc theo chuỗi định dạng
    - 0 nếu đến cuối file hoặc bị lỗi khi đọc file

# Đọc ghi dữ liệu định dạng trong C

- Ví dụ hàm fscanf

```
#include <stdio.h>
void main(){
    FILE* fp = fopen(”filename.txt”, “rt”);
    if (fp == NULL){
        cout << “Cannot open file” << endl;
        return;
    }
    int a = 0, b = 0;
    int n = fscanf(fp, “%d %d”, &a, &b);
    printf(“a = %d, b = %d”, a, b);
    fclose(fp);
}
```

# Đọc ghi dữ liệu định dạng trong C

- Trong C, hàm ghi dữ liệu được định dạng: fprintf
  - Cú pháp

```
int fprintf(FILE* fp, <format>, <var1>, <var2>,…);
```

  - Ý nghĩa: ghi dữ liệu trong các biến var1, var2,… theo định dạng trong chuỗi format tại vị trí con trỏ file fp đang trỏ tới
  - Trả về:
    - Vị trí của con trỏ file fp sau khi đã ghi dữ liệu thành công
    - 0 nếu khi ghi file bị lỗi

# Đọc ghi dữ liệu định dạng trong C

- Ví dụ hàm fprintf

```
#include <stdio.h>
void main(){
    FILE* fp = fopen("filename.txt", "wt");
    if (fp == NULL){
        cout << "Cannot open file" << endl;
        return;
    }
    int a = 9, b = 10;
    fprintf(fp, "%d %d", a, b);
    fclose(fp);
}
```

# Đọc ghi ký tự trong C

- Trong C, hàm đọc 1 ký tự từ file fgetc
  - Cú pháp

```
int fgetc(FILE* fp);
```

  - Ý nghĩa: đọc 1 ký tự từ file tại ví trí con trỏ fp đang trỏ tới
  - Trả về:
    - Ký tự được đọc từ vị trí của con trỏ file fp
    - Vị trí mới của con trỏ fp sẽ bị dịch chuyển
    - EOF nếu ở cuối file hoặc quá trình đọc bị lỗi

# Đọc ghi ký tự trong C

- Ví dụ hàm fgetc

```
#include <stdio.h>
void main(){
   FILE* fp = fopen("filename.txt", "rt");
   if (!fp){
      cout << "Cannot open file" << endl;
      return;
   }
   int n = 0;
   while(fgetc(fp) != EOF){
      n++;
   }
   cout << "Sum of chacracters = " << n;
   fclose(fp);
```

# Đọc ghi ký tự trong C

- Trong C, hàm ghi 1 ký tự vào file fputc
  - Cú pháp

```
int fputc(int c, FILE* fp);
```

  - Ý nghĩa: ghi 1 ký tự c vào file tại ví trí con trỏ fp đang trỏ tới
  - Trả về:
    - Vị trí của con trỏ file fp sau khi ghi dữ liệu
    - EOF nếu quá trình đọc bị lỗi

# Đọc ghi ký tự trong C

- Ví dụ hàm fputc

```
#include <stdio.h>
void main(){
   FILE* fp = fopen(”filename.txt”, “wt”);
   if (!fp){
      cout << “Cannot open file” << endl;
      return;
   }
   int n = 0;
   for(char c = ‘A’; c <= ‘Z’; c++{
      fputc(c, fp);
   }
   fclose(fp);
}
```

# Đọc ghi chuỗi ký tự trong C

- Trong C, hàm đọc chuỗi ký tự từ file fgets
  - Cú pháp

```
char* fgets(char* str, int num, FILE* fp);
```

  - Ý nghĩa: đọc chuỗi ký tự từ file tại ví trí con trỏ fp đang trỏ tới. Việc đọc kết thúc khi đủ num-1 ký tự, gặp ký tự '\n', hay gặp ký tự kết thúc file EOF.
  - Trả về:
    - Nếu thành công, trả về chuỗi str đọc được, và vị trí mới của con trỏ file fp
    - EOF nếu ở cuối file
    - NULL nếu quá trình đọc bị lỗi

# Đọc ghi chuỗi ký tự trong C

- Ví dụ hàm fgets

```
#include <stdio.h>
void main(){
   FILE* fp = fopen(”filename.txt”, “rt”);
   if (!fp){
      cout << “Cannot open file” << endl;
      return;
   }
   char str[100];
   if(fgets(str, 100, fp) != NULL)
      puts(str);
   fclose(fp);
}
```

# Đọc ghi chuỗi ký tự trong C

- Trong C, hàm ghi chuỗi ký tự vào file fputs
  - Cú pháp

```
int fputs(const char* str, FILE* fp);
```

  - Ý nghĩa: ghi chuỗi ký tự str vào file tại ví trí con trỏ fp đang trỏ tới
  - Trả về:
    - Vị trí của con trỏ file fp sau khi ghi dữ liệu
    - EOF nếu quá trình đọc bị lỗi

# Đọc ghi chuỗi ký tự trong C

- Ví dụ hàm fputs

```
#include <stdio.h>
void main(){
    FILE* fp = fopen(”filename.txt”, “wt”);
    if (!fp){
        cout << “Cannot open file” << endl;
        return;
    }
    fputs(“Hello world”, fp);
    fclose(fp);
}
```

# Đọc ghi dữ liệu định dạng trong C++

- Trong C++, để đọc dữ liệu được định dạng toán tử >>
  - Include <fstream>
  - Cú pháp

```
<stream> >> <var1> >> <var2> >> <var3> … ;
```

  - Ý nghĩa: đọc dữ liệu theo định tại vị trí stream, và lưu vào các biến var1, var2,… Các biến có thể cách nhau bằng khoảng trắng hoặc ký tư xuống dòng '\n'
  - Trả về: Vị trí stream sau khi đọc dữ liệu

# Đọc ghi dữ liệu định dạng trong C++

- Ví dụ toán tử >>

```
#include <fstream>
using namespace std;
void main(){
   ifstream fin("filename.txt", ios::in);
   if (!fin.is_open()){
      cout << "Cannot open file" << endl;
      return;
   }
   int a = 0, b = 0;
   int n = fin >> a >> b;
   cout << "a = " << a << "b = " << b;
   fin.close();
}
```

# Đọc ghi dữ liệu định dạng trong C++

- Trong C++, để ghi dữ liệu được định dạng toán tử <<
  - Include <fstream>
  - Cú pháp

```
<stream> << <var1> << <var2> << <var3> … ;
```

  - Ý nghĩa: ghi giá trị của các biến var1, var2, var3,… xuống file tại vị trí stream
  - Trả về: Vị trí stream sau khi đọc dữ liệu

# Đọc ghi dữ liệu định dạng trong C++

- Ví dụ toán tử <<

```
#include <fstream>
using namespace std;
void main(){
   ofstream fout("filename.txt", ios::out);
   if (!fout.is_open()){
      cout << "Cannot open file" << endl;
      return;
   }
   int a = 0, b = 0;
   int n = fin << a << " " << b << endl;
   fout.close();
}
```

# Đọc ghi ký tự trong C++

- Trong C++, hàm đọc ký tự/chuỗi ký tự từ file get
  - Cú pháp

```
ifstream& get(char &c);
```

  - Ý nghĩa: đọc ký tự c từ file
  - Trả về:
    - Ký tự/chuỗi ký tự đọc được
    - Vị trí của con trỏ file fp sau khi ghi dữ liệu
    - EOF nếu quá trình đọc bị lỗi hoặc kết thúc file

# Đọc ghi ký tự trong C++

- Ví dụ get

```
#include <fstream>
using namespace std;
void main(){
    ifstream fin("filename.txt", ios::in);
    if (!fin.is_open()){
        cout << "Cannot open file" << endl;
        return;
    }
    char c;
    while(fin.get(c))
        cout << c;
    fin.close();
}
```

# Đọc ghi ký tự trong C++

- Trong C++, hàm ghi ký tự xuống file put
  - Cú pháp

```
ofstream& put(char &c);
```

  - Ý nghĩa: đọc ký tự c /chuỗi ký tự s từ file
  - Lưu ý: hàm put chỉ ghi được ký tự, không ghi chuỗi
  - Trả về: Vị trí của con trỏ file sau khi ghi dữ liệu

# Đọc ghi ký tự trong C++

- Ví dụ put

```
#include <fstream>
using namespace std;
void main(){
   ofstream fout("filename.txt", ios::out);
   if (!fin.is_open()){
      cout << "Cannot open file" << endl;
      return;
   }
   for(char c='A'; c < 'Z'; c++)
      fout.put(c);
   fout.close();
}
```

# Đọc ghi chuỗi ký tự trong C++

- Trong C++, để đọc ghi chuỗi ký tự có thể dùng toán tử >>, <<
  - Đọc chuỗi std::string, dùng toán tử >>

```
string str1, str2;
fin >> str1 >> str2 ;
```

  - Ghi chuỗi std::string hoặc char*, dùng toán tử <<

```
string str1 = "Hello world";
fout << str1 << endl ;
char str2[100] = "C++ programming";
fout << str2 << endl ;
```

# Đọc ghi chuỗi ký tự trong C++

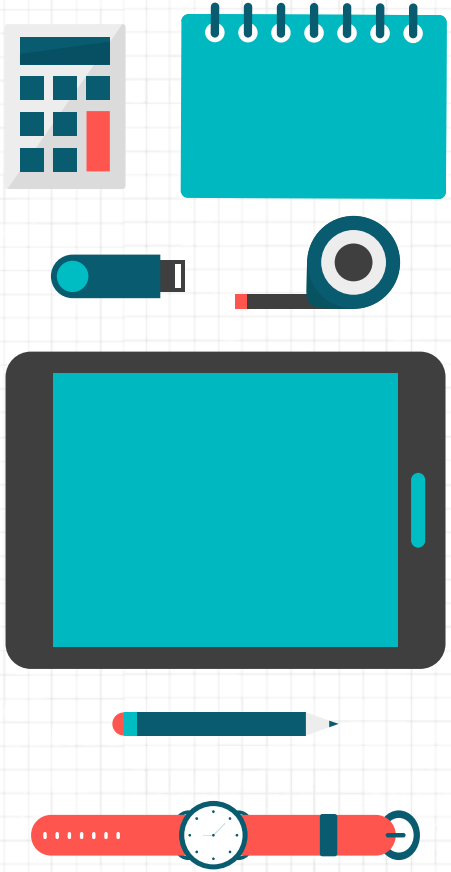
- Ngoài ra, có đọc chuỗi ký tự dạng char* từ file bằng get/getline
  - Cú pháp

```
ifstream& get(char* s, streamsize n);
ifstream& get(char* s, streamsize n, char d);
ifstream& getline(char* s, streamsize n);
ifstream& getline(char* s, streamsize n, char d);
```

  - Ý nghĩa: đọc chuỗi ký tự dạng char* s từ file. Việc đọc sẽ kết thúc khi đủ (n-1) ký tự hoặc gặp ký tự kết thúc d, hoặc gặp ký tự xuống dòng '\n', hay ký tự kết thúc file EOF
  - Trả về:
    - Chuỗi ký tự dạng char* đọc được
    - Vị trí của con trỏ file sau khi ghi dữ liệu
    - EOF nếu quá trình đọc bị lỗi hoặc kết thúc file

# Bài tập

- Viết hàm đọc mảng 2 chiều gồm M dòng, N cột từ file có cấu trúc như sau
  - Dòng đầu tiên lưu M và N cách nhau bởi khoảng trắng
  - M dòng tiếp theo, mỗi dòng lưu N phần tử cách nhau bởi khoảng trắng
  - Sắp xếp lại các dòng tăng dần theo tổng phần tử trên dòng
  - Lưu mảng sau sắp xếp xuống file theo cấu trúc trên



# 2d

## Đọc ghi file nhị phân

# Đọc ghi file nhị phân trong C

- Đọc file nhị phân bằng fread
  - Include <stdio.h>
  - Cú pháp

```
size_t fread(const void* ptr, size_t size, size_t count, FILE* fp);
```

  - Ý nghĩa: đọc 1 mảng gồm count phần tử, mỗi phần tử có kích thước size byte từ vị trí con trỏ file fp vào vùng nhớ mà con trỏ ptr đang trỏ đến
  - Trả về thành công: số lượng phần tử (không phải số byte) đã đọc được

# Đọc ghi file nhị phân trong C

- Ví dụ

```
int main() {
  char buffer[20]; // Buffer to store data
  FILE* stream;
  stream = fopen("file.bin", "rb");
  int count = fread(&buffer, sizeof(char), 20, stream);
  fclose(stream);
  printf("Data read from file: %s \n", buffer);
  printf("Elements read: %d", count);
  return 0;
}
```

# Đọc ghi file nhị phân trong C

- Ghi file nhị phân bằng fwrite
  - Include <stdio.h>
  - Cú pháp

```
size_t fwrite(const void* ptr, size_t size, size_t count, FILE* fp);
```

  - Ý nghĩa: ghi 1 mảng gồm count phần tử, mỗi phần tử có kích thước size byte từ vùng nhớ mà con trỏ ptr đang trỏ đến vào vị trí của con trỏ file fp
  - Trả về thành công: số lượng phần tử (không phải số byte) đã ghi được

# Đọc ghi file nhị phân trong C

- Ví dụ

```
int main () {
   FILE* pFile;
   char buffer[] = { 'x' , 'y' , 'z' };
   pFile = fopen ("myfile.bin", "wb");
   fwrite(buffer, sizeof(char), sizeof(buffer), pFile);
   fclose(pFile); return 0;
}
```

# Đọc ghi file nhị phân trong C++

- Đọc file nhị phân bằng read
  - Include <fstream>
  - Cú pháp

```
ifstream& read(const char* ptr, streamsize n);
```

  - Ý nghĩa: đọc 1 mảng gồm n byte từ vị trí stream vào vùng nhớ mà con trỏ ptr đang trỏ đến
  - Trả về thành công: vị trí mới của stream sau khi đọc

# Đọc ghi file nhị phân trong C++

- Ví dụ

```
int main () {
   ifstream is("test.txt", ios::in | ios::binary);
   if (!is)
      return 0;
   char buffer[100];
   is.read(buffer, 100);
   if (is)
      cout << "all characters read successfully.";
   return 1;
}
```

# Đọc ghi file nhị phân trong C++

- Ghi file nhị phân bằng write
  - Include <fstream>
  - Cú pháp

```
ofstream& write(const char* ptr, streamsize n);
```

  - Ý nghĩa: ghi 1 mảng gồm n byte từ vùng nhớ mà con trỏ ptr đang trỏ đến vào file tại vị trí của stream
  - Trả về thành công: vị trí mới của stream sau khi ghi

# Đọc ghi file nhị phân trong C++

- Ví dụ

```
int main () {
   ofstream os ("test.txt", ios::out | ios::binary);
   if (!os)
      return 0;
   char buffer[100] = “Hello world!”;
   is.write (buffer, 100);
   return 1;
}
```

# Bài tập

- Viết hàm nhập mảng N phân số
- Sắp xếp mảng tăng dần
- Ghi N phân số đã sắp xếp xuống file nhị phân

# Bài tập

- Viết hàm đọc file văn bản input, sau đó xuất file văn bản output với mỗi dòng trong file output ghi số lượng từ xuất hiện trên dòng tương ứng của file input.