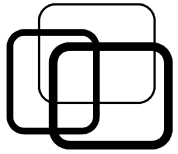
Tập tin nhị phân

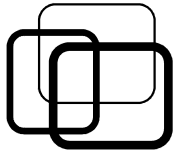
GV. Nguyễn Minh Huy

Nội dung



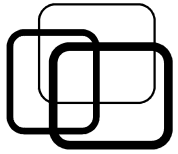
- Stream nhập xuất.
- Tập tin nhị phân.
- Tham số hàm main.

Nội dung



- **Stream nhập xuất.**
- Tập tin nhị phân.
- Tham số hàm main.

Stream nhập xuất



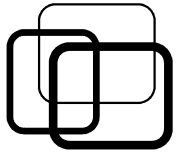
- Thiết bị nhập xuất:
  - Chương trình là "cỗ máy" làm việc.

    ➔ Dữ liệu → Chương trình → Kết quả.
  - Chương trình lấy dữ liệu và xuất kết quả thế nào?

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

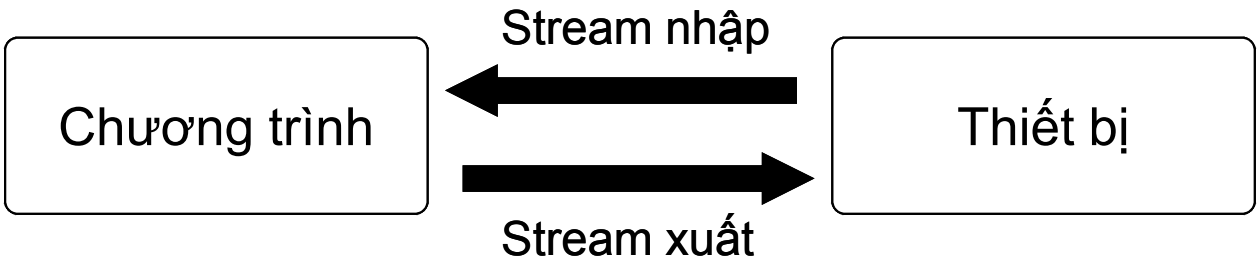
    ➔ Tập tin là thiết bị vừa nhập vừa xuất.

Stream nhập xuất

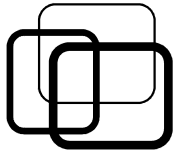


- Khái niệm stream:
  - Chương trình giao tiếp với thiết bị bằng cách nào?

  ➔ Thông qua "dòng chảy" dữ liệu.
  - Stream: "dòng chảy" kết nối chương trình và thiết bị.
  - Phân loại stream:
    - Stream nhập: "dòng chảy" kết nối thiết bị nhập.
    - Stream xuất: "dòng chảy" kết nối thiết bị xuất.

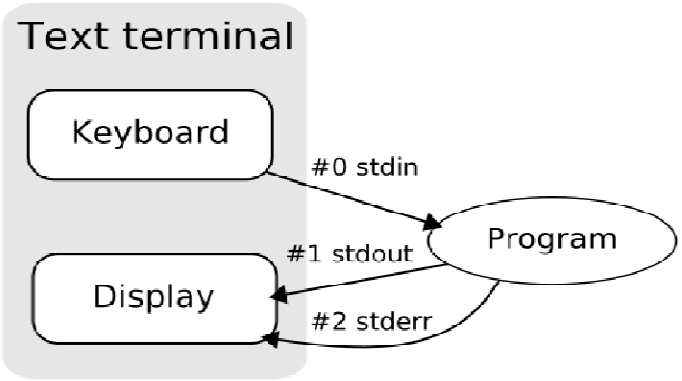


Stream nhập xuất



- Stream trong C:
  - Các stream khai báo sẵn trong C:

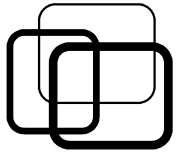
Stream	Ý nghĩa	Thiết bị kết nối
stdin	Stream nhập chuẩn.	Bàn phím
stdout	Stream xuất chuẩn.	Màn hình
stderr	Stream lỗi chuẩn.	Màn hình



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

```
fscanf( stdin, "%d", &x);          // Nhập từ bàn phím.
fprintf( stdout, "Hello World" );  // Xuất ra màn hình.
```

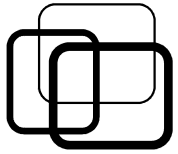
Stream nhập xuất



- Stream trong C++:
  - Nâng cấp stream của C:
    - Tương thích với stream của C.
    - Sử dụng tiếp cận hướng đối tượng.
    - Dễ sử dụng hơn.
  - Các stream khai báo sẵn 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++:
  - Toán tử nhập >>:
    - Nhập dữ liệu từ stream.
    - Không cần định dạng kiểu.
    - Cú pháp:

      <stream> >> <biến>;
  - Toán tử xuất <<:
    - Xuất dữ liệu ra stream.
    - Không cần định dạng kiểu.
    - Cú pháp:

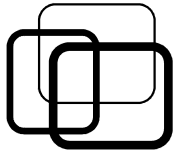
      <stream> << <biến/trị>;

```
void main()
{
    int     n;
    float  a[10];

    cout << "Nhap n = ";
    cin >> n;

    for (int i = 0; i < n; i++)
    {
       cout << "Nhap a[" << i << "] = ";
       cin >> a[ i ];
    }
}
```

Stream nhập xuất

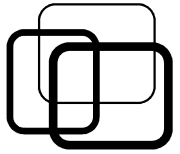


- Tập tin vs. Bộ nhớ:

Tiêu chí	Tập tin (Bộ nhớ ngoài)	Bộ nhớ
Tốc độ xử lý	Chậm	Nhanh
Khả năng truy xuất	Tuần tự	Ngẫu nhiên
Chi phí	Rẻ tiền	Đắc tiền
Dung lượng	Lớn	Nhỏ
Thời gian	Lưu trữ lâu dài	Lưu trữ nhất thời

- Tập tin vs. Bàn phím và Màn hình:
  - Không cần thông qua người dùng.
  - Đọc/ghi tự động, nhiều lần.
  - Giao tiếp được với chương trình khác.

Stream nhập xuất

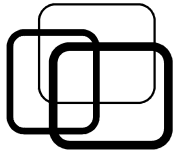


- File stream:
  - "Dòng chảy" kết nối chương trình và tập tin.
  - Khái báo: **FILE** *<tên stream>;

    ➔ Con trỏ tập tin.

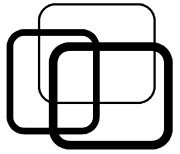
    FILE *f1;
  - Các bước xử lý tập tin:
    - Bước 1: mở tập tin.
    - Bước 2: thao tác trên tập tin.
    - Bước 3: đóng tập tin.

Stream nhập xuất



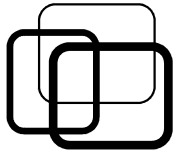
- Các lệnh cơ bản:
  - Mở tập tin: fopen, freopen.
  - Đóng tập tin: fclose.
  - Đọc/ghi tập tin: fscanf, fgets, fprintf.

Nội dung



- Stream nhập xuất.
- **Tập tin nhị phân.**
- Tham số hàm main.

Tập tin nhị phân

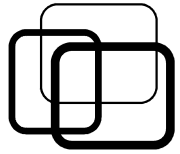


- Chế độ mở nhị phân:
  - Bảng chế độ mở tập tin:

Chế độ mở	Ý nghĩa
r	**R**ead-only, mở để đọc dữ liệu (kiểu text).
w	**W**rite-only, mở để ghi dữ liệu (kiểu text).
a	**A**ppend-only, mở để ghi thêm dữ liệu (kiểu text).
[Chế độ mở]+	Kết hợp đọc, ghi cùng lúc.
[Chế độ mở]b	Mở kiểu nhị phân (binary).

  - Cho phép đọc/ghi thô từng byte trên tập tin.
  - Ánh xạ từng byte tập tin vào từng byte bộ nhớ.

Tập tin nhị phân



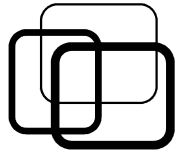
- Lệnh fread:
  - Đọc block bytes từ tập tin.
    - Cú pháp: **fread(**<Địa chỉ vùng nhớ>, <Kích thước block> , <Số block cần đọc>, <Con trỏ tập tin>**)**;
    - Trả về: số block đọc được.

    ➔ Kết thúc tập tin khi số block đọc được < số block cần đọc.

```
int     x;
char   *p = new char[ 100 ];
FILE  *f = fopen("C:\\BaiTap.txt", "rb");

if ( f != NULL )
{
    fread( &x, sizeof(int), 1, f );     // Đọc 4 bytes vào số nguyên x.
    fread( p, sizeof(char), 100, f );// Đọc 100 bytes vào vùng nhớ p.
    fclose( f );
}
```

Tập tin nhị phân

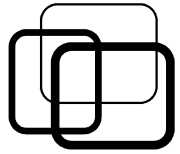


- Lệnh fwrite:
  - Ghi block bytes vào tập tin.
    - Cú pháp: **fwrite(**<Địa chỉ vùng nhớ>, <Kích thước block> , <Số block cần đọc>, <Con trỏ tập tin>**)**;
    - Trả về: số block ghi được.

```
int      x = 123456;
char   s[ ] = “Hello World”;
FILE  *f = fopen(“C:\\BaiTap.txt”, “wb”);

if ( f != NULL )
{
     fwrite( &x, sizeof(int), 1, f );            // Ghi 4 bytes số nguyên x.
     fwrite( s, sizeof(char), strlen(s), f );  // Ghi 11 bytes chuỗi s.
     fclose( f );
}
```

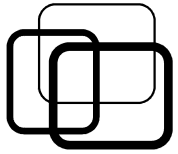
Tập tin nhị phân



- Lệnh fseek:
  - Thay đổi vị trí con trỏ tập tin.
    - Cú pháp: **fseek(**<Con trỏ tập tin>, <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 làm việc với tập tin đang mở.

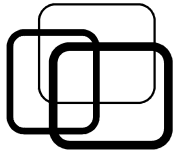
```
FILE  *f = fopen(“C:\\BaiTap.txt”, “r”);
if ( f != NULL )
{
     fseek( f, 2, SEEK_CUR );  // Chuyển con trỏ tới 2 bytes.
     fclose( f );
}
```

Nội dung



- Stream nhập xuất.
- Tập tin nhị phân.
- **Tham số hàm main.**

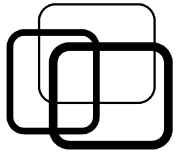
Tham số hàm main



- Khái niệm tham số dòng lệnh:
  - Chương trình là một hàm khổng lồ!!
  - Truyền dữ liệu đầu vào cho chương trình thế nào?
  - Tham số dòng lệnh:
    - Tham số truyền khi gọi thực hiện chương trình.
    - Hàm main có thể nhận các tham số này để xử lý.
  - Cách truyền tham số:
    - Gọi chương trình ở chế độ command line của hệ điều hành.
    - Cú pháp: <chương trình> **<tham số 1> <tham số 2> …**

      C:\>BaiTap\baitap1.exe **hello 5 /abc**
      C:\>copy **C:\BaiTap\baitap1.exe D:\Files\baitap1.exe**

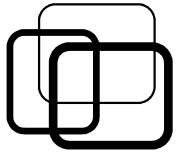
Tham số hàm main



- Sử dụng tham số hàm main:
  - Khai báo nhận tham số:
    - Cú pháp: void **main**(int **argc**, char ****argv**);
    - argc: số lượng tham số.
    - argv: danh sách các tham số.
    - Các tham số ở dạng chuỗi ký tự.
    - Tham số đầu tiên là tên chương trình.

```
void main(int  argc, char  **argv)
{
        cout << “So luong tham so = “ << argc;
        cout << “Danh sach tham so:” << endl;
        for (int i = 0; i < argc; i++)
              cout << argv[ i ] << endl;
}
```

Tham số hàm main



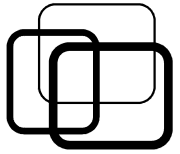
- Sử dụng tham số hàm main:

```
void main(int  argc, char  **argv)
{
    if (argc == 1)
        cout << “Error: no input argument.”;
    else
    {
        if (strcmp(argv[ 1 ], “/?”)
            showHelp();
        else if (argc == 3)
            copyFile(argv[ 1 ], argv[ 2 ]);
    }
}
```

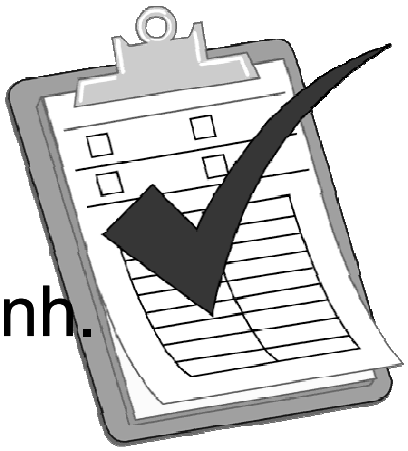
```
void showHelp()
{
        // …
}

void copyFile(char *file1, char *file2)
{
        // …
}
```

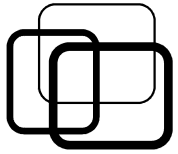
Tóm tắt



- Stream nhập xuất:
  - "Dòng chảy" kết nối chương trình và thiết bị.
  - Nhập xuất trong C: printf, scanf.
  - Nhập xuất trong C++: cin >>, cout <<.
  - File stream: con trỏ tập tin, fprintf, fscanf.
- Tập tin nhị phân:
  - Đọc/ghi thô từng byte.
  - Lệnh thao tác: fread, fwrite, fseek.
- Tham số hàm main:
  - Tham số dòng lệnh truyền vào chương trình.
  - Cú pháp: void main(int argc, char **argv);



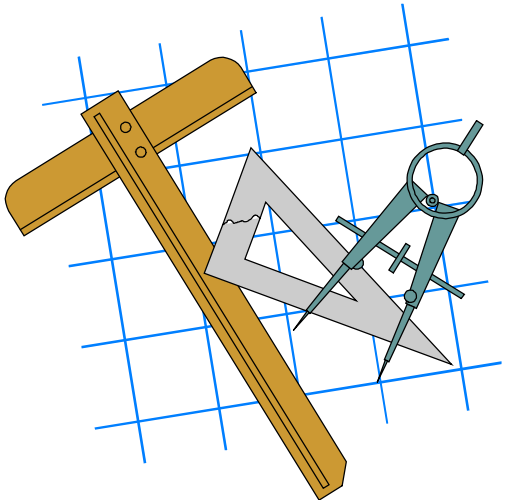
Bài tập



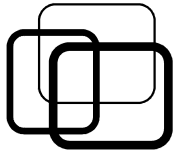
- Bài tập 4.1:

  Viết chương trình C như sau:

  - Nhập từ bàn phím tên file nguồn.
  - Nhập từ bàn phím tên file đích.
  - Thực hiện sao chép file nguồn sang file đích.



Bài tập



Bài tập 4.2:

Viết chương trình C mô phỏng lệnh COPY trong Command Line của Windows (thực hiện sao chép tập tin bằng tham số dòng lệnh).

Cú pháp lệnh COPY:

- Sao chép đặt tên mới:

  COPY **<file nguồn> <file đích>**

- Sao chép giữ tên cũ:

  COPY **<file nguồn> <folder đích>**

- Nối tập tin:

  COPY **<file 1> + <file 2> <file đích>**

- Xem hướng dẫn:

  COPY **/?**