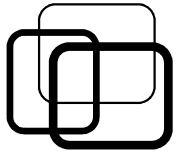
Tập tin nhị phân (tt)

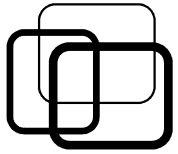
GV. Nguyễn Minh Huy

Nội dung



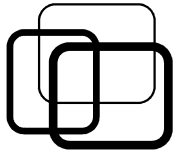
- Chuỗi cấp phát động.
- Bài tập BMP.

Nội dung



- **Chuỗi cấp phát động.**
- Bài tập BMP.

Chuỗi cấp phát động



- Chuỗi ký tự động:
  - Chuỗi = mảng ký tự + ký tự '\0';

char **s1[6]** = { 'H', 'e', 'l', 'l', 'o', '\0' }; **s1**

H	e	l	l	o	\0

char **s2[ ]** = "Hello"; **s2**

H	e	l	l	o	\0

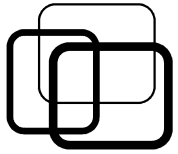
  - Chuỗi cấp phát động:
    - Dùng mảng động → phải cấp phát và thu hồi vùng nhớ.
    - Có thể thay đổi kích thước khi cần.
    - Khai báo: char ***<chuỗi>**;

char ***s3** = new char[6]; **s3** 27 →

?	?	?	?	?	?

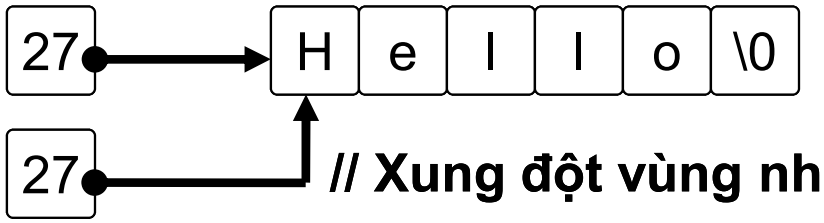
//…
delete [ ]s3;

Chuỗi cấp phát động



- Chuỗi ký tự động:
  - Sao chép chuỗi:
    - Không sao chép bằng toán tử =.

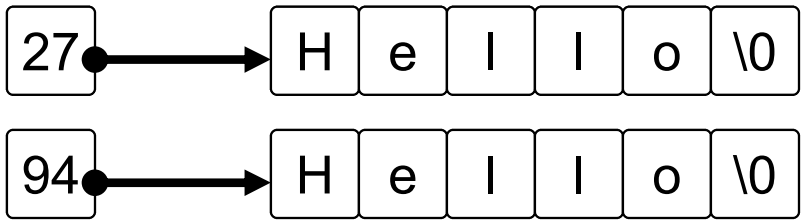
```
char  *s1 = “Hello”;
char  *s2 = s1;
```



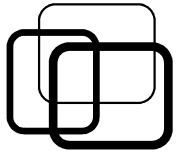
// Xung đột vùng nhớ!

    - Thao tác sao chép chuỗi:
      - B1: khai báo và cấp phát chuỗi mới.
      - B2: dùng strcpy để sao chép.

```
char  *s1 = “Hello”;
char  *s1 = new char[ strlen(s1) + 1 ];
strcpy( s2, s1 );
```

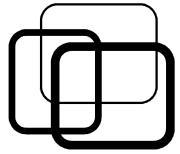


Chuỗi cấp phát động



- Chuỗi ký tự động:
  - Nhập chuỗi động:
    - Khai báo kiểu SinhVien:
      - Mã số: 7 ký tự.
      - Họ tên: 50 ký tự.
      - Điểm trung bình: số thực.
    - Viết hàm nhập thông tin 1 sinh viên.

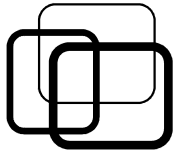
Chuỗi cấp phát động



- Thư viện <string.h>:
  - Lệnh tạo bản sao:
    - Cú pháp: **strdup**(<chuỗi nguồn>);
    - Trả về: chuỗi bản sao, vùng nhớ được tự động cấp phát.
    - Phải thu hồi vùng nhớ chuỗi bản sao khi dùng xong.

```
char  s1 = “Hello”;
char  *s2 = strdup(s1);

// Tương tự….
// char *s2 = new char[ strlen(s1) + 1 ];
// strcpy( s2, s1 );

free(s2);
```

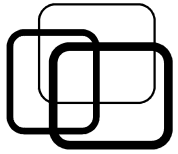
Chuỗi cấp phát động



- Thư viện <string.h>:
  - Lệnh so sánh chuỗi:
    - Cú pháp: **strcmp**(<chuỗi 1>, <chuỗi 2>);
    - Trả về: 0 (bằng), 1 (lớn hơn), -1 (nhỏ hơn).
    - So sánh nội dung 2 chuỗi theo thứ tự từ điển.

```
char  *s1 = “abc”;
char  *s2 = “abaab;
char  s3[10];
strcpy(s3, s1);

int     kq1 = strcmp(s1, s2);   // kq1 = 1.
int     kq2 = strcmp(s1, s3);   // kq2 = 0.
int     kq3 = strcmp(s2, s3);   // kq2 = -1.
```

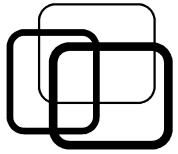
Chuỗi cấp phát động



- Thư viện <string.h>:
  - Lệnh nối chuỗi:
    - Cú pháp: **strcat**(<chuỗi đích>, <chuỗi nguồn>);
    - Nối chuỗi nguồn và cuối chuỗi đích.
    - Chuỗi đích phải đủ vùng nhớ để nối!!

```
char  *s1 = “Hello”;
char  *s2 = “World;
char  *s3 = new char[ strlen(s1) + strlen(s2) + 1 ];

strcat(s3, s1);          // Nối s1 vào s3.
strcat(s3, s2);          // Nối tiếp s2 vào s3.
```

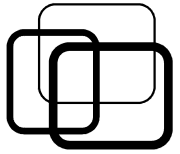
Chuỗi cấp phát động



- Thư viện <string.h>:
  - Lệnh tìm chuỗi con:
    - Cú pháp: **strstr**(<chuỗi nguồn>, <chuỗi con>);
    - Trả về: địa chỉ vị trí đầu tiên xuất hiện chuỗi con (tìm thấy), NULL (thất bại).

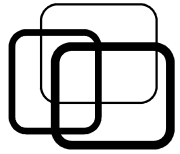
```
char  s1[ ] = “Hello World”;
char  *s2 = “World;
char  *s3 = strstr(s1, s2);

if (s3 == NULL)
    printf(“Không tìm thấy chuỗi con.”);
else
    printf(“Vị trí xuất hiện chuỗi con = %d”, s3 - s1);
```

Nội dung



- Chuỗi cấp phát động.
- **Bài tập BMP.**

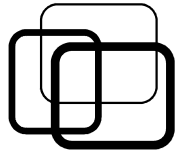
Tập tin nhị phân



- Đọc ghi cấu trúc:
  - Ánh xạ các byte tập tin vào thành phần cấu trúc.
    ➔ Hiệu quả hơn đọc ghi từng thành phần.
  - Thứ tự ánh xạ theo thứ tự khai báo thành phần.

```
struct PhanSo {
        int  tu;
        int  mau;
};

void docPS(FILE *f, PhanSo &p) {          // Đọc 8 bytes vào phân số p.
        fread( &p, sizeof(PhanSo), 1, f );  // 4 bytes đầu vào p.tu.
}                                         // 4 bytes sau vào p.mau.

void ghiPS(FILE *f, PhanSo p) {           // Ghi 8 bytes phân số p ra tập tin.
        fwrite( &p, sizeof(PhanSo), 1, f );  // p.tu ra 4 bytes đầu.
}                                         // p.mau ra 4 bytes sau.
```

Tập tin nhị phân



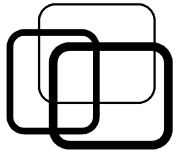
- Thư viện <stdint.h>:
  - Kích thước số nguyên bao nhiêu bytes?

    ➔ Tùy thuộc hệ máy tính.
  - Đọc/ghi nhị phân cần số nguyên kích thước cố định.

    ➔ Thư viện <stdint.h>
  - Kiểu số nguyên xác định:
    - Số nguyên 1 byte: int8_t, uint8_t.
    - Số nguyên 2 bytes: int16_t, uint16_t.
    - Số nguyên 4 bytes: int32_t, uint32_t.
    - Số nguyên 8 bytes: int64_t, uint64_t.

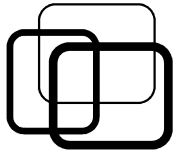
```
#include <stdint.h>

struct PhanSo
{
        int32_t  tu;
        int32_t  mau;
};
```

Tóm tắt



- Chuỗi cấp phát động:
  - Dùng mảng động để biểu diễn chuỗi động.
  - Không sao chép chuỗi bằng toán tử =.
  - Thư viện <string.h>: strdup, strcmp, strstr.

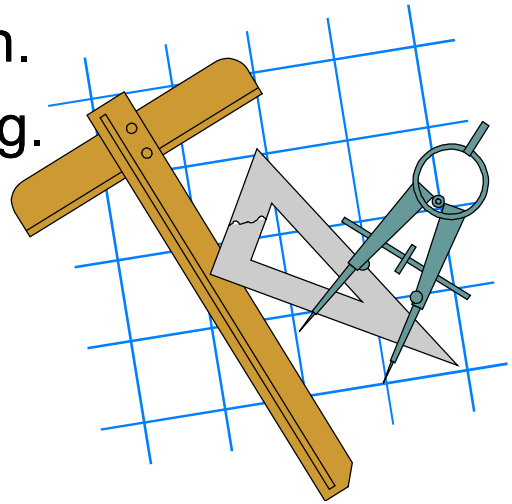
Bài tập



Bài tập 5.1:

Viết chương trình C thực hiện những việc sau:

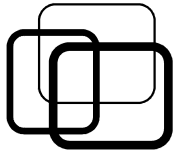
- Nhập vào một đoạn văn dài đến khi kết thúc bằng dấu '.' và xuống hàng.
- Hãy xuất ra những thông tin sau:
  - a) Số từ trong đoạn văn (các từ cách nhau khoảng trắng hoặc dấu câu).
  - b) Đoạn văn đã chuẩn hóa
    - + Bỏ khoảng trắng đầu và cuối đoạn văn.
    - + Các từ cách nhau đúng 1 khoảng trắng.
    - + Viết hoa chữ cái đầu mỗi từ.



Yêu cầu:

- **Dùng chuỗi động.**
- **Cho phép thay đổi thiết bị nhập xuất.**

Bài tập



Bài tập 5.2:

Viết chương trình C thực hiện cắt ảnh BMP thành các phần bằng nhau dùng tham số dòng lệnh. Mỗi phần ảnh cắt được lưu vào một file BMP.

Cú pháp dòng lệnh:

<tên chương trình> <file Bmp> [-h <số phần cắt dọc>] [-w <số phần cắt ngang>]

Ví dụ: chương trình tên cutbmp.exe

- Cắt 3 phần theo chiều cao (lưu vào 3 ảnh BMP):

  cutbmp.exe d:/images/img1.bmp -h 3

- Cắt 2 phần theo chiều cao, 4 phần theo chiều dọc (lưu vào 8 ảnh BMP):

  cutbmp.exe d:/images/img1.bmp -h 2 -w 4