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

LẬP TRÌNH C CƠ BẢN

Chương 1. Khởi động&nhập xuất dữ liệu với tệp-P2

Soạn bởi: TS. Nguyễn Bá Ngọc
(Dựa trên bài giảng sử dụng chung)

2021

Tệp văn bản và tệp nhị phân

- Tệp văn bản:
  - Các bytes biểu diễn các ký tự.
- Tệp nhị phân
  - Các bytes có thể biểu diễn đối tượng bất kỳ, ví dụ một nhóm bytes có thể biểu diễn một giá trị kiểu int, một giá trị kiểu double, một đối tượng cấu trúc v.v..
- Ví dụ: int x = 64; // 4 bytes
  - Tệp văn bản: num.txt - 64
  - Tệp nhị phân (big edian): num.dat - 00000000 00000000 00000000 01000000
- Tệp văn bản có thể được chia thành các dòng. Có 2 cách chính biểu diễn dấu hiệu kết thúc dòng:
  - Môi trường Windows sử dụng cặp ký tự CR (\r) và LF (\n)
  - Môi trường Unix/Linux sử dụng 1 ký tự LF (\n)
  - char *s1 = “Hanoi\r\n”, *s2 = “Hanoi\n”;
  - Tuy chuỗi s1 và s2 khác nhau nhưng printf(“%s", s1) và printf(“%s", s2) có hiệu ứng như nhau.

Ví dụ 1.3. Tính tổng dãy số

vd1-3.c

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
  if (argc != 2) {
    printf("Usage: ./prog input.txt\n");
    return 1;
  }
  FILE *inp = fopen(argv[1], "r");
  int n;
  fscanf(inp, "%d", &n);
  int sum = 0, x;
  for (int i = 0; i < n; ++i) {
    fscanf(inp, "%d", &x);
    sum += x;
  }
  printf("Tổng các số là: %d\n", sum);
  fclose(inp);
  return 0;
}

```

Mở tệp

Xử lý, đọc/ghi

Đóng tệp, kết thúc

```
bangoc: ~/git/c-basic
File  Edit  View  Search  Terminal  Help
bangoc:$gcc -o prog vd1-3.c
bangoc:$cat vd1-3-inp.txt
5
1 2 3 4 5
bangoc:$./prog
Usage: ./prog input.txt
bangoc:$./prog vd1-3-inp.txt
Tổng các số là: 15
bangoc:$
```

Nhập/xuất dữ liệu

- Trong C luồng có thể là bất kỳ nguồn dữ liệu nhập hoặc bất kỳ nơi xuất dữ liệu nào.
- Truy cập luồng được thực hiện thông qua con trỏ tệp, kiểu FILE *
- Các con trỏ tệp được định nghĩa sẵn cho các luồng nhập/xuất tiêu chuẩn: stdin (nhập từ bàn phím), stdout (xuất thông tin bình thường ra màn hình), stderr (xuất thông tin lỗi ra màn hình).
- Một số cặp hàm đọc/ghi tệp thông dụng trong thư viện **stdio.h**:
  - fgetc/fputc - Đọc/ghi theo ký tự
  - fgets/fputs - Đọc/ghi theo dòng
  - fscanf/fprintf - Đọc/ghi theo định dạng
    - Các hàm đa chức năng
    - Có cú pháp tương tự scanf/printf (nhưng khái quát hơn).
    - fscanf với stdin = scanf, fprintf với stdout = printf
    - Ngoài ra còn có sscanf và sprintf với cú pháp tương tự để đọc/ghi với chuỗi ký tự (ví dụ sử dụng: Chuyển đổi từ số sang chuỗi và ngược lại).

Đọc và ghi tệp văn bản theo từng ký tự với **fgetc/fputc**

Bài tập 1.12. Sao chép nội dung tệp văn bản

- Tạo một tệp văn bản với tên lab1.txt với nội dung bất kỳ, lưu trong thư mục cùng với chương trình.
- Viết chương trình sao chép nội dung tệp lab1.txt sang lab1-copy.txt sử dụng các hàm fgetc và fputc

Gợi ý:

```
int c;
while ((c = fgetc(inp)) != EOF) {...}
```

Lưu ý: Trong trường hợp có quy ước sử dụng một ký tự làm dấu hiệu kết thúc tệp văn bản, sao chép tệp ở chế độ văn bản có thể sai vì bỏ quả phần nội dung có thể tồn tại sau dấu hiệu kết thúc tệp văn bản. Sao chép tệp thường được thực hiện ở chế độ nhị phân.

Bài tập 1.13. Đảo ngược hoa và thường

- Viết chương trình sử dụng fgetc và fputc để đọc nội dung từ một tệp văn bản, mỗi lần đọc một ký tự, sau đó chuyển chữ cái hoa thành chữ cái thường và ngược lại rồi ghi vào một tệp văn bản khác.
- Các ký tự không phải chữ cái được giữ nguyên khi sao chép sang tệp mới.

Gợi ý:

```
ctype.h: toupper, tolower, isalpha
```

Bài tập 1.14. Tham số dòng lệnh

- Điều chỉnh bài tập 1.12 sử dụng các tham số dòng lệnh cho tệp nguồn và tệp đích:

```
./mycp <tập_nguồn> <tập_đích>
```

- Chương trình phải kiểm tra cú pháp dòng lệnh, thông báo lỗi và hiển thị hướng dẫn khi cần.

Ví dụ 1.4. Một lời giải bài tập 1.14

Trong trường hợp không mở được tệp hàm fopen trả về NULL (NULL về bản chất là 0)

fgetc *trả về giá trị kiểu* int, fputc *nhận tham số kiểu int,* EOF *là macro (-1).*

vd1-4.c

```c
#include <stdio.h>
int main(int argc, char *argv[]) {
  if (argc != 3) {
    printf("Usage: ./prog input.txt output.txt\n");
    return 1;
  }
  FILE *inp = fopen(argv[1], "r");
  if (!inp) {
    printf("Lỗi mở tệp %s\n", argv[1]);
    return 1;
  }
  FILE *out = fopen(argv[2], "w");
  if (!out) {
    printf("Lỗi mở tệp %s\n", argv[2]);
    return 1;
  }
  int c;  // trong khoảng [0..255] hoặc -1 (EOF)
  while ((c = fgetc(inp)) != EOF) {
    fputc(c, out);
  }
  fclose(out);
  fclose(inp);
  return 0;
}
```

Bài tập 1.15. Nối tệp

- Viết chương trình nhận tên hai tệp văn bản qua đối số dòng lệnh, sau đó tiến hành ghép nội dung của tệp thứ hai vào cuối tệp thứ nhất. Giả sử cả hai tập tin đều tồn tại.
- Cú pháp sử dụng:

```
./apd <tệp 1> <tệp 2>
```

- *Ví dụ*

```
tep1.txt        tep2.txt
Wellcome to     C programming

Nội dung tep1.txt sau khi thực hiện lệnh
    ./apd tep1.txt tep2.txt
Wellcome to
C programming
```

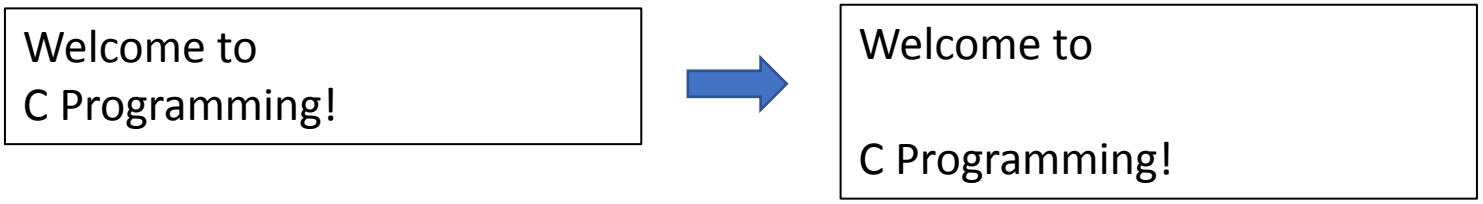
Gợi ý: Mở tệp 1 ở chế độ “a”

Bài tập 1.16. Đổi sang chữ hoa

- Viết chương trình có tên ucv có chức năng chuyển đổi tất cả các chữ cái trong nội dung một tệp (tên tệp được cung cấp qua tham số dòng lệnh) thành chữ hoa và ghi lại nội dung mới vào một tệp khác.
- Cú pháp: `./ucv inp.txt out.txt`
- Ví dụ :
  - Tệp inp.txt: hello world!
  - Tệp out.txt: HELLO WORD!

Bài tập 1.17. Thêm dòng trống

- Viết chương trình với tên dbl nhận một tệp đầu vào và biến đổi nội dung của nó như sau: Chương trình chèn thêm 1 dòng trống giữa các dòng trong văn bản. Kết quả được ghi vào một tệp đầu ra. Các tên tệp được cung cấp qua tham số dòng lệnh:
  ./double_line <File 1> <File 2>
- Ví dụ minh họa nội dung hai file khi chạy chương trình



Gợi ý: Viết thêm 1 ký tự \n nếu đọc được ký tự \n và vẫn tồn tại ký tự tiếp theo.

Bài tập 1.18. Mã Caesar

- Viết một chương trình có thể sử dụng cùng một lúc hai chức năng mã hóa và giải mã một tập tin văn bản sử dụng mật mã Caesar (mã hóa cộng) như sau. Chương trình nhận ba đối số:
  - <tập tin nguồn> <độ dịch chuyển> < tập tin đích>
- Khi cần mã hóa, chạy chương trình với độ dịch chuyển (offset) n là một số nguyên dương. Chương trình sẽ thay thế mỗi ký tự trong tập tin bởi một ký tự đứng sau nó n vị trí trong bảng mã ASCII, dịch chuyển được thực hiện theo quy tắc xoay vòng. Ví dụ với offset = 3 thì A□D, B□ E
- Khi giải mã, chạy chương trình với đầu vào là tập tin mã hóa và giá trị độ dịch chuyển là số âm tương ứng (VD offset = -3)

Đọc và ghi tệp văn bản theo chuỗi ký tự **fgets/fputs**

Bài tập 1.19

- Thực hiện lại bài tập 1.12, tuy nhiên thay vì sử dụng cặp hàm fgetc/fputc để sao chép tệp – chúng ta sử dụng cặp hàm fgets/fputs cho bài tập này.

Gợi ý:

```
char buff[N];
while (fgets(buff, N, inp)) {...}
```

Bài tập 1.20. Đếm dòng & hiển thị nội dung

- Điều chỉnh bài tập 1.19 để chương trình hiển thị số lượng dòng có trong văn bản lên màn hình.

Gợi ý: số lượng dòng = #ký tự \n + 1

Bài tập 1.21. Hiển thị số thứ tự dòng trong tệp văn bản

- Viết chương trình đọc một tệp văn bản và hiển thị ra màn hình số thứ tự từng dòng trước nội dung của dòng đó. Tên tệp được cung cấp thông qua tham số dòng lệnh.
- Ví dụ với tập tin có nội dung

  Wellcome to

  C programming!
- Kết quả trên màn hình là:

  1 Wellcome to

  2 C programming!

Gợi ý: Tăng chỉ số dòng sau khi đọc được chuỗi ký tự có '\n' là ký tự cuối cùng và in ra chỉ số dòng nếu còn dữ liệu sau đó.

Bài tập 1.22. Trộn hai tệp theo dòng

- Viết chương trình có đối số dòng lệnh như sau
  - ./merge <tệp 1> <tệp 2> <tệp 3>
- Chương trình ghi vào tệp 3 bằng cách đọc và đan xen từng dòng từ tệp 1 và tệp 2. Giống như chương trình đọc một dòng từ tệp 1 rồi ghi vào tệp 3 sau đó đọc một dòng từ tệp 2 rồi ghi vào tệp 3.
- Tệp 1 và tệp 2 có thể có số dòng khác nhau, khi đọc hết nội dung một tệp, chương trình sao chép các dòng tiếp theo của tệp còn lại vào tệp 3.

Gợi ý: Đọc một dòng có thể có kích thước lớn hơn kích thước bộ nhớ đệm:

```
while (fgets(buff, N, inp)) {
  … if (buff[strlen(buff) - 1] == '\n') …
}
```

Bài tập 1.23. Hiển thị theo từng trang

- Viết chương trình mycat đọc và hiển thị nội dung một tệp văn bản trên màn hình. Chương trình hỗ trợ hai cú pháp sử dụng như sau:
  `mycat <filename>` : Hiển thị một lần toàn bộ nội dung

  `mycat <filename> -p` : Hiển thị theo từng trang, mỗi trang 10 dòng. Người dùng nhấn Enter để xem trang tiếp theo.

Gợi ý:

`string.h`

```
if (strcmp(argv[2], “-p") == 0) {...}
```

`getchar()`

Để giữ tính khả chuyển chúng ta không sử dụng getch() và cả conio.h.

Bài tập 1.24. Ký tự đầu tiên của các dòng

- Viết chương trình nhận đối số dòng lệnh là đường dẫn đến một tệp văn bản chứa không quá 80 dòng và không chứa dòng trắng. Chương trình thêm một dòng mới vào cuối tệp nói trên với nội dung chứa các ký tự đầu tiên của các dòng trong tệp ban đầu.

Gợi ý: Sử dụng một mảng ký tự (chuỗi ký tự) để lưu ký tự đầu tiên của mỗi dòng trong tệp ban đầu.

Bài tập 1.25. Bổ xung kích thước dòng

- Viết chương trình đọc từng dòng từ một tệp văn bản, sau đó tính số lượng bytes trong biểu diễn dòng và ghi ra một tập tin mới theo định dạng: <số lượng bytes> <Nội dung dòng>
  Trong phạm vi bài tập này mỗi dòng có kích thước tối đa không vượt quá 1024 bytes.
- Ví dụ với một dòng trong tệp ban đầu:

  The quick brown fox jumps over the lazy dog The.

  Dòng tương ứng trong tệp đầu ra là:

  48 The quick brown fox jumps over the lazy dog The.

Đọc và ghi tệp văn bản sử dụng định dạng **fscanf/fprintf**

Bài tập 1.26. Thứ tự ngược của dãy số

- Viết chương trình đọc từ bàn phím một số nguyên dương n và dãy n số nguyên, sau đó ghi các số đã nhập ra tệp “out.txt” theo thứ tự ngược lại và tổng các số vào cuối tệp. Được biết n không lớn hơn 100.
- Ví dụ nhập: 3 12 -45 56
  - Nội dung tệp “out.txt”: 56 -45 12 23

Gợi ý:

```
fprintf(out, “%d ", a[i]);
```

Bài tập 1.27. Danh sách sản phẩm

- Tạo một tệp văn bản có tên **product.txt**, mỗi dòng trong đó chứa thông tin về một sản phẩm: ID (kiểu int), Product Name (xâu ký tự không chứa ký tự trắng), Price (kiểu double). Các trường dữ liệu trên được phân tách với nhau bởi một dấu cách.
- Ví dụ

```
product.txt
1   Samsung_Television_4K 20000000
2   Apple_MacBook_2020 18560000
```

- Viết chương trình đọc tệp **product.txt** vào một mảng cấu trúc và sau đó in danh sách sản phẩm ra màn hình theo định dạng:

```
No   Product Name            Price
1    Samsung_Television_4K   20000000
2    Apple_MacBook_2020      18560000
```

Gợi ý: `fscanf(inp, "%d%s%lf", ...) != 3`

Bài tập 1.28. Danh sách sinh viên

- Tạo một tệp văn bản với tên sinhvien.txt có nội dung là danh sách lớp gồm ít nhất 6 sinh viên. Mỗi dòng gồm 4 trường sau: STT(int), Mã số sinh viên (chuỗi liền/không chứa ký tự trắng và không quá 10 ký tự) Họ và tên (chuỗi liền không quá 64 ký tự) Số điện thoại (chuỗi liền không quá 16 ký tự). Ví dụ:

```
1 20181110 Bui_Van 0903112234
2 20182111 Joshua_Kim 0912123232
```

- Viết chương trình đọc tệp sinhvien.txt vào một mảng cấu trúc, sau đó chương trình hỏi nhập điểm cho mỗi sinh viên rồi ghi lại tất cả các thông tin về sinh viên cùng với điểm vào tệp bangdiem.txt.

Gợi ý: Lô-gic đọc tệp tương tự bài tập 1.27.

So sánh các hàm nhập xuất

- Nhập dữ liệu từ tệp văn bản:
  - fscanf có thể nhập dữ liệu theo nhiều định dạng
  - fscanf(..., “%c", …) và fgetc(...) đều cho phép nhập một ký tự, nhưng có sự khác biệt về phong cách - kết quả đọc của fscanf được lưu trong biến, còn fgetc cung cấp thông qua giá trị được trả về.
  - fscanf(..., “%[^\n]", ...), fscanf(..., “%N[^\n]", ...) - (N là số nguyên dương) và fgets(...) đều cho phép nhập chuỗi ký tự có chứa khoảng trắng nhưng fgets và fscanf(..., “%N[^\n]", ...) an toàn hơn (vì có giới hạn số lượng ký tự giúp tránh được lỗi tràn mảng), tuy nhiên cú pháp của fgets có thể đơn giản nhất.
- Xuất dữ liệu vào tệp văn bản:
  - fprintf(..., “%c", …) tương đương với fputc(...)
  - fprintf(..., “%s", ...) tương đương với fputs(...)

Đọc và ghi tệp nhị phân
fread/fwrite

Ví dụ 1.5. Tệp dữ liệu sinh viên ở dạng nhị phân

vd1-5-r.c

```c
#include <stdio.h>
struct sinhvien {
  int mssv;
  char hoten[64];
};
int main() {
  struct sinhvien a[2];
  FILE *inp = fopen("sv.dat", "rb");
  int n;
  fread(&n, sizeof(n), 1, inp);
  fread(a, sizeof(struct sinhvien), n, inp);
  printf("%8s%32s\n", "STT", "Ho ten");
  for (int i = 0; i < n; ++i) {
    printf("%8d%32s\n", a[i].mssv, a[i].hoten
  }
  fclose(inp);
  return 0;
}
```

vd1-5-w.c

```c
#include <stdio.h>
struct sinhvien {
  int mssv;
  char hoten[64];
};
int main() {
  struct sinhvien a[2] = {
    {1, "Nguyen Van A"},
    {2, "Tran Thi B"}
  };
  FILE *out = fopen("sv.dat", "wb");
  int n = sizeof(a)/sizeof(struct sinhvien);
  fwrite(&n, sizeof(n), 1, out);
  fwrite(a, sizeof(struct sinhvien), n, out);
  return 0;
}
```

fread *và* ***fwrite*** *đọc và ghi khối dữ liệu được biểu diễn tương tự như mảng. Các tham số cho biết địa chỉ bắt đầu khối nhớ, kích thước 1 phần tử, số lượng phần tử và cuối cùng là con trỏ tệp (nhị phân).*

```
bangoc: ~/git/c-basic
File  Edit  View  Search  Terminal  Help
bangoc:$gcc -o w vd1-5-w.c
bangoc:$./w
bangoc:$gcc -o r vd1-5-r.c
bangoc:$./r
     STT                             Ho ten
       1                      Nguyen Van A
       2                        Tran Thi B
bangoc:$
```

Bài tập 1.29. Sao chép tệp ở chế độ nhị phân

- Thực hiện lại bài tập 1.12 sử dụng các hàm fread, fwrite và mở tệp ở chế độ nhị phân.

Gợi ý:

“rb”, “wb”

```
unsigned char buff[N];
while (fread(...)) { ...fwrite(...)...}
```

Bài tập 1.30. Kích thước tệp

- Điều chỉnh bài tập 1.29 sao cho chương trình in ra kích thước tệp tính bằng Bytes sau khi hoàn thành sao chép.

Gợi ý: cộng dồn giá trị được trả về bởi hàm fread

Bài tập 1.31 *

- Viết chương trình so sánh thời gian sao chép tệp theo các cách:
  1. Đọc/ghi từng ký tự (fgetc/fputc)
  2. Đọc/ghi từng chuỗi (fgets/fputs)
  3. Đọc/ghi từng khối ở chế độ nhị phân (fread/fwrite)

Gợi ý:

```
time.h

double t  = (double)clock()/CLOCKS_PER_SEC;
```

Bài tập 1.32. Danh sách sinh viên

- Viết chương trình đọc thông tin về sinh viên trong tệp văn bản bangdiem.txt (bài tập 1.28) và lưu dữ liệu vào mảng cấu trúc, sau đó ghi dữ liệu ra tập tin nhị phân với tên sinhvien.dat. Cuối cùng chương trình đọc dữ liệu từ tệp bangdiem.txt và xuất ra màn hình để kiểm tra.

Bài tập 1.33. Quản lý danh bạ

- Giả sử bạn cần quản lý một danh bạ điện thoại của mình bằng chương trình. Định nghĩa một cấu trúc biểu diễn danh bạ gồm các trường "name," "telephone number," "e-mail address," và khai báo một mảng chứa tối đa 100 phần tử thuộc kiểu cấu trúc trên.
- Nhập liệu cho khoảng 10 phần tử mảng.
- Chương trình sau đó ghi nội dung mảng với các phần tử nói trên vào tập tin có tên phonebook.dat sử dụng hàm fwrite.
- Đọc lại dữ liệu từ tập tin vào mảng sử dụng hàm fread và in nội dung mảng ra màn hình để kiểm tra.

Bài tập 1.34. Truy cập ngẫu nhiên

- Viết chương trình nhập vào một số nguyên là số thứ tự của bản ghi trong danh bạ được lưu trong tệp phonebook.dat (bài tập 1.33), ví bản ghi thứ 2. Sau đó hỏi người dùng nhập vào giá trị cập nhật cho trường email rồi ghi lại dữ liệu đã cập nhật vào tệp.

Gợi ý:

"rb+"

`fseek`

Bài tập 1.35. Tách tệp

- Sử dụng tệp phonebook.dat (bài tập 1.33) chứa ít nhất 20 số liên lạc. Viết các chương trình sau
- Chương trình **filesplit** nhận hai đối số: tên file nguồn (.dat) và một số nguyên n và tên hai tệp kết quả. Chương trình cần tách tệp nguồn thành 2 tệp, trong đó tệp đầu tiên chứa n số liên lạc đầu tiên và tệp thứ hai chứa các số liên lạc còn lại.

Ví dụ

```
filesplit phone.dat 10 phone1.dat phone2.dat
```

Bài tập 1.35. Ghép tệp

- Viết Chương trình **filemerge** ghép hai tệp lưu 2 danh bạ (bài tập 1.33) khác nhau (không có trùng lặp) thành một tệp chứa tất cả dữ liệu từ 2 danh bạ trong các tệp đầu vào. Các tên tệp được cung cấp thông qua tham số dòng lệnh:

```
filemerge phone1.dat phone2.dat phone.dat
```

- Sau khi hoàn thành ghi tệp, chương trình xuất dữ liệu ra màn hình (để kiểm tra).

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict