TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
MÔN: **NHẬP MÔN LẬP TRÌNH**

HƯỚNG DẪN THỰC HÀNH

TUẦN 09

XỬ LÝ TẬP TIN VĂN BẢN

TP.HCM, ngày 09 tháng 09 năm 2017

MỤC LỤC

1 Khái niệm

Trong phần này ta sẽ tìm hiểu các vấn đề liên quan tới xử lý tập tin văn bản. Nhìn chung khi làm việc với tập tin ta có các bước cơ bản sau:

- Bước 1: Mở tập tin, ta cần cung cấp đường dẫn và tên tập tin chính xác.
- Bước 2: Sử dụng tập tin (Sau khi mở thành công)
  - Bước 2.1: Đọc dữ liệu tập tin đưa vào biến bộ nhớ
  - Bước 2.2: Ghi dữ liệu từ biến bộ nhớ trong chương trình lên tập tin sau khi đã xử lý
- Bước 3: Đóng tập tin

Bên dưới là bảng chứa các hàm mà ngôn ngữ C cung cấp để hỗ trợ việc xử lý tập tin

Thao tác	**Hàm hỗ trợ**
Mở tập tin	fopen(): Trả về biến kiểu FILE*
Đọc dữ liệu	fscanf(), fgets(), fgetc()
Ghi dữ liệu	fprintf(), fputs(), fputc()
Kiểm tra tới cuối tập tin	feof(): trả về 1 nếu đến cuối và 0 nếu chưa tới
Đóng tập tin	fclose()

2 Thao tác trên tập tin văn bản chứa kí tự

Trong phần này ta sẽ lần lượt xem qua các thao tác trên văn bản chứa chủ yếu là kí tự, không cần có các xử lý số học. Các ví dụ đều có liên hệ với nhau nên ta cần xem xét theo thứ tự từ trên xuống.

2.1 Đọc nội dung tập tin in ra màn hình

Trong ví dụ này ta sẽ xây dựng hàm TransFile nhằm đọc dữ liệu từ tập tin nằm trên ổ cứng sau đó xuất ra màn hình (Ngôn ngữ C dùng biến stdout đại diện cho màn hình)

Dòng	Mô tả
1	#include <stdio.h>
2	void TransFile(FILE* inFile, FILE* outFile){
3	int ch;
4	while(1){
5	ch = fgetc(inFile);
6	if(!feof(inFile))
7	fputc(ch, outFile);
8	else
9	break;
10	}

```
11  }
12
13  void main(){
14    FILE* fpIn;
15    char* fname = “Data.txt”;
16    fpIn = fopen(fname, “rt”);
17    if(fpIn == NULL){
18      printf(“File %s not found\n”, fname);
19      return;
20    }
21    TransFile(fpIn, stdout);
22    fclose(fpIn);
23  }
```

Trong ví dụ trên ta thấy hàm TransFile có hai tham số đầu vào, tham số thứ nhất là tên tập tin đầu vào (dùng kiểu char* để giữ chuỗi kí tự) và tham số thứ hai cũng có kiểu là FILE* (vị trí này có thể là stdout (in ra màn hình) hay cũng có thể là một tập tin thông thường khác (trường hợp sao chép tập tin)). Lưu ý: Hàm này mặc định đã chứa tập tin đầu vào được mở thành công. Trong hàm này ta dùng vòng lặp while để đọc từng kí tự từ tập tin vào biến ‘ch’, sau đó ghi biến ‘ch’ vào lại một thiết bị khác mà biến outFile lưu giữ (màn hình hay tập tin thông thường).

Trong hàm main ta cần mở tập tin và kiểm tra xem có thành công hay không, sau khi đã ổn ta chỉ cần gọi hàm TransFile và in toàn bộ nội dung tập tin văn bản ra màn hình.

2.2 Đọc nội dung tập tin in ra một tập tin khác (sao chép)

Cũng hoàn toàn tương tự như ví dụ trên, nhưng thay vì in ra màn hình, giờ ta in ra một tập tin khác, và đây chính là thao tác copy/paste trong windows mà ta thường dùng. Như vậy hàm ‘TransFile’ ta tái sử dụng lại hoàn toàn và chỉ cần chỉnh sửa lại hàm main cho phù hợp với nhu cầu mới.

```
Dòng  Mô tả
1   #include <stdio.h>
2   //…
3   void main(){
4     FILE* fpIn, *fpOut;
5     char* fname = “Data.txt”, *fcopy = “DataCopy.txt”;
6     fpIn = fopen(fname, “rt”);
7     if(fpIn == NULL){
8       printf(“File %s not found\n”, fname);
9       return;
10    }
```

11	fpOut = fopen(fcopy, “wt”);
12	if(fpOut == NULL){
13	printf(“Cannot open file %s\n”, fcopy);
14	fclose(fpIn);
15	return;
16	}
17	TransFile(fpIn, fpOut);
18	fclose(fpIn);
19	fclose(fpOut);
20	}

Ta thấy đoạn mã hàm main bổ sung thêm fpOut, các thao tác mở tập tin hoàn toàn giống nhau với lưu ý nhỏ ở dòng 11, thay vì chế độ “rt” để đọc thì ta thay bằng “wt” để ghi. Nếu không mở được tập tin để ghi thì nhớ đóng tập tin để đọc đã mở thành công trước đó (dòng mã 14). Sau cùng gọi TransFile ở dòng 17 và dùng fpOut thay cho vị trí của stdout lúc trước.

2.3 Chỉnh sửa trực tiếp trên một tập tin

Trong phần này ta sẽ được hướng dẫn chỉnh sửa nội dung trực tiếp trên một tập tin. Giả sử ta có tập tin văn bản “Data.txt”, giờ ta muốn chuyển tất cả các kí tự trong tập tin này thành in hoa (Lưu ý: ta cũng có thể làm điều này trong quá trình sao chép ở ví dụ trước). Để thực hiện xử lý trên cùng một tập tin ta cần điều khiển con trỏ File tới vị trí cần chỉnh sửa. Qui trình theo các bước sau:

- Bước 1: Đọc một kí tự từ tập tin vào biến ‘ch’
- Bước 2: Ghi nhận vị trí con trỏ hiện hành vào biến pos
- Bước 3: Chuyển kí tự ch thành in hoa (nếu là chữ thường)
- Bước 4: Dời vị trí con trỏ về trước một vị trí (pos - 1) và sau đó ghi ‘ch’ vào vị trí này
- Bước 5: Dời vị trí con trỏ về chỗ cũ tại pos
- Bước 6: Kiểm tra xem tới cuối tập tin hay chưa, nếu chưa quay lại bước 1

Tiếp theo ta sẽ xem đoạn mã minh họa

Dòng	Mô tả
1	#include <stdio.h>
2	#include <ctype.h>
3	void UpcaseFile(FILE* f){
4	char ch; long pos;
5	while(!feof(f)){
6	ch = fgetc(f);

```
7     if(feof(f))
8       break;
9     pos = ftell(f);
10    fseek(f, pos – 1, SEEK_SET); // SEEK_SET cờ lấy vị trí đầu tập tin làm mốc
11    ch = toupper(ch);
12    fputc(ch, f);
13    fseek(f, pos, SEEK_SET);
14   }
15  }
16
17  void main(){
18   char* fname = “Data.txt”;
19   FILE* f = fopen(fname, “rt”);
20   if(f == NULL){
21    printf(“File %s not found!\n”, fname);
22    return;
23   }
24   UpcaseFile(f);
25   fclose(f);
26  }
```

Ta lưu ý cờ SEEK_SET ý là tính từ đầu tập tin, đây là mốc để trình biên dịch sẽ thêm (pos – 1) vị trí theo sau.

3 Thao tác trên tập tin văn bản chứa số

Tiếp theo ta sẽ tìm hiểu các tập tin văn bản chủ yếu được xử lý số học. Giả sử ta có một tập tin chứa một dãy sao nguyên. Ta cần đọc dãy số này và sau đó tính tổng, kết quả tính tổng sẽ in ra màn hình. Bên dưới là đoạn mã minh họa

```
Dòng
1   #include <stdio.h>
2
3   int TinhTong(FILE*);
4
5   int TinhTong(FILE* f){
6    int sum = 0, i;
7    while(fscanf(f, “%d”, &i) > 0){
8     sum+=i;
9    }
10   return sum;
11  }
```

12	
13	void main(){
14	char* fname = “NumData.txt”;
15	FILE* f = fopen(fname, “rt”);
16	if(f != NULL){
17	int sum = TinhTong(f);
18	printf(“Tong la %d\n”, sum);
19	fclose(f);
20	}
21	else{
22	printf(“File %s not found!\n”, fname);
23	}
24	}

Ta dùng hàm fscanf để thực hiện đọc từng kí số (theo định dạng “%d”). Hàm sẽ trả ra 0 nếu thất bại. Vì vậy ta dùng đặc điểm này để biết được việc đọc có thành công hay không.

4 Xử lý tập tin văn bản lưu trữ dạng cấu trúc

Ta đã được hướng dẫn về kiểu dữ liệu có cấu trúc. Như vậy cũng cần có các kĩ thuật để xử lý loại dữ liệu này. Giả sử trên ổ cứng ta có tập tin tên là “Student.txt” với nội dung:

Student.txt	
Dòng	
1	0807621
2	Nguyen Minh Khang
3	10/02/1992
4	7.5 10.0 9.5
5	0800682
6	Luong Huy Minh Tam
7	18/06/1990
8	8.9 8.3 7.0
9	0800455
10	Van Thi Ha My
11	13/04/1991
12	9.1 6.5 8.0
13	0801516
14	To My Hien
15	10/09/1989
16	6.8 9.0 8.5

Nội dung tập tin chủ yếu lưu danh sách thông tin một loạt các sinh viên. Ta cần một chương trình để thực hiện điều này. Trước tiên ta cần một hàm 'fgetline' đọc **trọn vẹn nội dung một dòng** trong một tập tin.

Dòng	
1	#include <stdio.h>
2	#include <string.h>
3	#define MAXCHARS 1024
4	
5	char* fgetline(FILE* fp, char* str, int maxSize){
6	int ch, len;
7	if(feof(fp)) return NULL;
8	str[0] = 0; len = 0;
9	do{
10	ch = fgetc(fp);
11	if(ch == '\n' || ch = EOF) break;
12	if(len < maxSize - 1){
13	str[len] = ch;
14	len++;
15	}
16	}while(1);
17	str[len] = 0;
18	return str;
19	}

Hàm trên được xem là hàm phụ trợ cho các tác vụ sau. Khi đã có được hàm này, ta sẽ xây dựng hàm 'getStudent' để đọc trọn vẹn nội dung MỘT student vào trong một biến kiểu STUDENT (dĩ nhiên ta phải tạo thêm struct). Xem tiếp đoạn mã bên dưới (Đây là đoạn mã tiếp theo của ví dụ trên).

Dòng	
1	#include <stdio.h>
2	#include <string.h>
3	//…
4	#define szCode 8
5	#define szName 25
6	#define szDate 11
7	
8	struct student{
9	char Code[szCode];
10	char Name[szName];
11	char BirthDate[szDate];

```
12    double Grade1, Grade2, Grade3;
13    double GPA;
14  };
15  typedef struct student STUDENT;
16
17  int getStudent(FILE* fp, STUDENT& sv){
18    double s1, s2, s3; int ch;
19    fgetline(fp, sv.Code, szCode);
20    fgetline(fp, sv.Name, szName);
21    fgetline(fp, sv.BirthDate, szDate);
22    if(fscanf(fp, "%lf %lf %lf", &s1, &s2, &s3) <= 0)return 0;
23    ch = fgetc(fp);//Lay ki tu '\n'
24    sv.Grade1 = s1; sv.Grade2 = s2; sv.Grade3 = s3; sv.GPA = (s1 + s2 + s3)/3;
25    return 1;
26  }
```

Sau khi đã có hàm 'getStudent' để lấy trọn vẹn một sinh viên. Tiếp theo ta xây dựng hàm đọc toàn bộ nội dung tập tin "Student.txt" vào **một mảng mà mỗi phần tử có kiểu là STUDENT**.

```
Dòng
1   #include <stdio.h>
2   #include <string.h>
3   //…
4   #define maxStudent 100
5
6   void getAllStudents(FILE* fp, STUDENT svList[], int &n){
7     STUDENT st; n = 0;
8     while(!feof(fp)){
9       if(n < maxStudent && getStudent(fp, sv))
10        svList[n] = sv;
11        n++;
12      }
13      else
14        break;
15    }
16  }
17
18  void main(){
19    STUDENT lst[maxStudent]; int n;
20    FILE* fp = fopen("Student.txt", "rt");
21    if(fp == NULL) return;
```

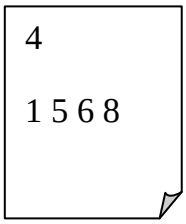
22	getAllStudents(fp, lst, n);
23	fclose(fp);
24	}

Trong đoạn mã trên ta thấy hàm getAllStudents lần lượt sẽ đọc toàn bộ nội dung của tập tin 'Student.txt' vào mảng một chiều các STUDENT. Lưu ý dòng mã số 10, việc gán nội dung một biến kiểu STUDENT cho một kiểu biến cùng kiểu STUDENT đã được ngôn ngữ C hỗ trợ (vì vậy ta không cần định nghĩa toán tử operator '=' cho kiểu mới ta tự tạo). Tuy nhiên trường hợp trong kiểu STUDENT có trường 'con trỏ' thì ta cần định nghĩa lại toán tử '='.

5 Bài tập

Sinh viên thực hiện các bài tập sau đây:

5.1 Cho tập tin chứa số ví dụ có dạng như sau. Trong số đầu tiên là số lượng chữ số mà tập tin chứa. Dòng còn lại chứa các số cách nhau bởi khoảng trắng.



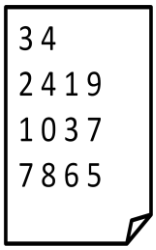
5.1.1 Viết hàm đọc nội dung tập tin vào trong mảng một chiều

5.1.2 Viết hàm ghi nội dung ra một tập tin khác với điều kiện chỉ ghi số nguyên tố

5.1.3 Viết hàm ghi nội dung ra một tập tin khác với điều kiện chỉ ghi số đối xứng

5.2 Viết bổ sung **hàm sắp xếp** các sinh viên trong ví dụ trên theo thứ tự tăng dần của trường GPA. Sau đó **ghi lại vào một tập tin khác** có tên là "SortedStudent.txt".

5.3 Cho tập tin có nội dung như sau. Trong đó 3 và 4 đại diện cho số dòng và cột của ma trận. Các số còn lại là giá trị các phần tử trong ma trận.



5.3.1 Viết hàm đọc nội dung tập tin vào một ma trận

5.3.2 Viết hàm sắp xếp tăng dần ma trận sau đó ghi ra một tập tin mới

5.3.3 Viết hàm tính tổng các phần tử chẵn trong ma trận sau đó ghi thêm vào **tập tin gốc** ở cuối một dòng chữ có nội dung như sau: ‘Tong cac phan tu chan trong ma tran la: …’