3. NHẬP/XUẤT TRÊN TẬP TIN VĂN BẢN

3.1. Nhập

Để đọc một file văn bản, cần thực hiện các bước sau

B1:

```
#include <stdio.h>
```

```
// mo file de doc
FILE* fi = fopen("input.txt", "rt");
```

trong đó hàm fopen nhận 2 tham số: tham số đầu tiên (kiểu char*) là tên file (input.txt), tham số thứ 2 là chuỗi "rt" (read + text: đọc dạng file văn bản)

Trong trường hợp muốn viết hàm đọc file và truyền tên file như một tham số vào hàm:

```
void DocFile(char* tenFile)
{
	FILE* fi = fopen(tenFile, "rt");
	…
}
```

B2:

```
//doc du lieu
int n;
fscanf(fi, "%d", &n);
```

dùng hàm fscanf để đọc dữ liệu. Cách dùng hàm fscanf tương tự hàm scanf (chỉ khác có thêm tham số đầu tiên kiểu FILE* là con trỏ đến tập tin đã mở ở trên)

B3:

```
//dong file
fclose(fi);
```

3.2. Xuất

Tương tự với nhập dữ liệu, ta cũng có các bước

B1:

```
	//mo file de ghi
	FILE* fo = fopen(tenFile, "wt"); // wt = write (ghi) + text (dang van ban)
```

B2:

```
	//ghi du lieu ra file
	fprintf(fo, "%d ",n);
```

Dùng hàm fprintf() tương tự printf()

B3:

```
	// dong file
	fclose(fo);
```

3.3. Cách quản lý tập tin nhập/xuất trong project của VS2005

Khi làm việc với tập tin, ta cần phải đặt tập tin ở đúng thư mục để có thể debug/thực thi chương trình. Ta có thể sử dụng VS2005 để tạo tập tin txt, VS2005 sẽ tự động đặt file đó ở đúng thư mục và ta có thể quản lý file dễ dàng hơn.

Để tạo file txt, từ cửa sổ Solution Explorer, nhấn chuột phải vào Resources Files, chọn Add -> New Item

Chọn Text File (.txt) và gõ tên (không gõ đuôi .txt)

Sau đó gõ nội dung tập tin.

Với các file mà ta xuất kết quả ra, ta cũng có thể thêm file đó vào project để dễ quản lý.

Cách làm tương tự, nhấn chuột phải vào Resources Files, chọn Add -> Existing Item

Chọn file mà ta đã xuất ra.

(Xem Demo)

Bài tập (code mẫu: NhapXuatFile)

Đọc từ file "input.txt" mảng một chiều các số thực. Tập tin input.txt có nội dung như sau:

- Dòng đầu chứa 1 số nguyên là số lượng phần tử của mảng
- Dòng sau chứa các phần tử của mảng cách nhau bởi khoảng trắng

Ví dụ:

```
5

1.2 2.3 3.4 4.5 5.6
```

```
#include <stdio.h>


void XuatFile(char* tenFile, float* arr, int n)
{
	//mo file de ghi
	FILE* fo = fopen(tenFile, "wt"); // wt = write (ghi) + text (dang van ban)

	//ghi du lieu ra file
	for (int i = 0; i < n; i++)
		fprintf(fo, "%0.1f ", arr[i]);

	// dong file
	fclose(fo);
```

```
}

void main()
{
	// mo file de doc
	FILE* fi = fopen("input.txt", "rt");

	//doc du lieu
	int n;
	fscanf(fi, "%d", &n);
	float* arr = new float[n];
	for (int i = 0; i < n; i++)
		fscanf(fi, "%f", &arr[i]);

	//dong file
	fclose(fi);

	// in ra man hinh de kiem tra
	for (int i = 0; i < n; i++)
		printf("%0.1f ", arr[i]);
	printf("\n");

	// xuat ra file
	XuatFile("output.txt", arr, n);
}
```

1. Biên dịch đoạn chương trình trên.

2. Tạo tập tin dữ liệu mới “MSSV.txt” thay cho file “input.txt”. Nhập dữ liệu cho file MSSV.txt và chạy chương trình.

3. Xuất ra file “out_MSSV.txt” thay cho file “output.txt”. Thêm file output.txt vào project để có thể xem kết quả xuất từ Visual Studio thay vì phải dùng Windows Explorer và Notepad.

3. Sửa lại chương trình để chỉ xuất ra file các tập tin có chỉ số lẻ của mảng. (chỉ in ra arr[1], arr[3], …)

4. Sửa lại chương trình để tính tổng các phần tử và xuất ra file tổng đó.

3.4. Đọc đến hết file

Nếu bài toán đọc mảng các số thực ở trên không có thông tin số lượng phần tử thì ta sẽ giải quyết theo hướng cứ đọc vào đến khi nào hết file thì dừng. Vậy ta cần phải biết dấu hiệu kết thúc file: hàm feof()

Xem đoạn chương trình mẫu sau: đọc một mảng không biết trước số lượng các số thực và in ra màn hình.

```
void DocHetFile1(char* tenFile)
{
	FILE* fi = fopen(tenFile, "rt");
	float temp;
	while (!feof(fi)) // khi chua ket thuc file
	{
		fscanf(fi, "%f", &temp); // doc so thuc vao bien temp
		printf("%0.1f ", temp);
	}
```

```
	printf("\n");
	fclose(fi);
}
```

Trong đó, ta sử dụng vòng while và điều kiện để thực hiện là chưa hết file: !feof(fi)

Hàm feof() nhận 1 tham số kiểu FILE* là file đang đọc và trả về true/false nếu kết thúc/chưa kết thúc file.

Bài tập (code mẫu: NhapXuatFile)

```
#include <stdio.h>


void DocHetFile1(char* tenFile)
{
	FILE* fi = fopen(tenFile, "rt");
	float temp;
	while (!feof(fi))
	{
		fscanf(fi, "%f", &temp);
		printf("%0.1f ", temp);
	}
	printf("\n");
	fclose(fi);
}

void DocHetFile2(char* tenFile)
{
	FILE* fi = fopen(tenFile, "rt");
	float temp;
	while (!feof(fi))
	{
		if (fscanf(fi, "%f", &temp)>0)
			printf("%0.1f ", temp);
		else
			break;
	}
	printf("\n");
	fclose(fi);
}


void main()
{

	printf("Doc khi khong biet so luong phan tu, doc den het file thi
dung:\n");

	DocHetFile1("input2.txt");
	DocHetFile1("input3.txt");

}
```

Trong đó

input2.txt có nội dung

```
1.2 2.3 3.4 4.5 5.6
```

input3.txt có nội dung (có 1 dấu khoảng trắng ở cuối file)

```
1.2 2.3 3.4 4.5 5.6
```

1. Biên dịch đoạn chương trình trên.

2. Nhận xét về kết quả xuất ra màn hình (2 dòng tương ứng với 2 file input2.txt và input3.txt). 2 dòng kết quả xuất ra có giống nhau không?

3. Nếu 2 dòng kết quả xuất ra không giống nhau, giải thích tại sao với dữ liệu vào như nhau (xem nội dung file input2.txt và input3.txt) kết quả xuất lại không giống nhau.

Gợi ý: file input3.txt có một dấu khoảng trắng ở cuối file nên ở vòng lặp cuối sau khi đọc 5.6 điều kiện (!feof(fi)) vẫn trả về đúng (vì hết số 5.6 chưa là cuối file) .

4. Sửa lại 2 dòng

```
	DocHetFile1("input2.txt");
	DocHetFile1("input3.txt");
```

thành

```
	DocHetFile2("input2.txt");
	DocHetFile2("input3.txt");
```

và chạy lại chương trình. Nhận xét kết quả xuất. Từ đó rút ra kết luận hàm nào đọc hết file chính xác hơn.

Gợi ý:

Hàm DocHetFile2 và DocHetFile1 chỉ khác nhau ở chỗ hàm DocHetFile2 có kiểm tra lệnh đọc số thực từ file có thành công hay không như sau:

Hàm fscanf() có giá trị trả về là số lượng biến đọc thành công

```
if (fscanf(fi, "%f", &temp)>0)
```

Ở đây ta đọc 1 biến , do đó chỉ cần kiểm tra giá trị trả về > 0 là đọc thành công.