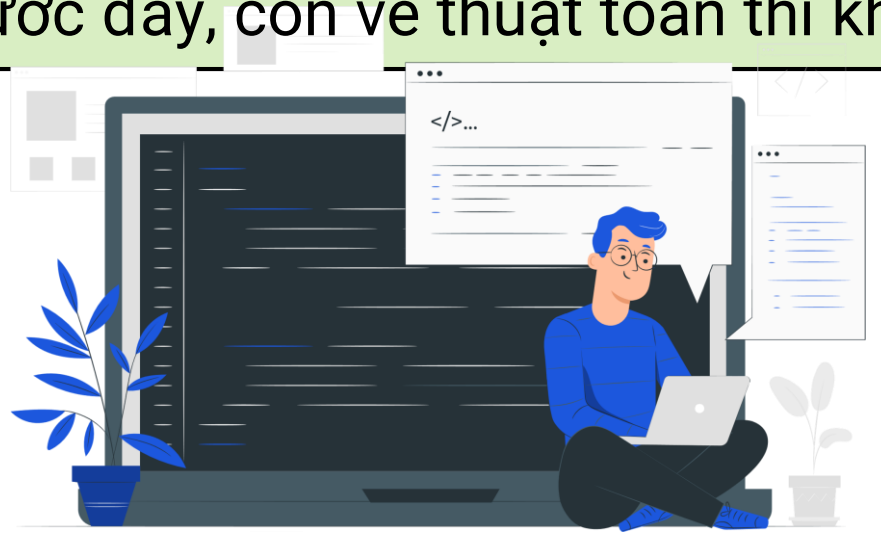
XỬ LÝ FILE TEXT

I. File text:

Các chương trình trước đây của bạn thường được đọc từ bàn phím và in ra màn hình, các dữ liệu đọc ghi đó sẽ mất khi chương trình kết thúc.

File là một phương án lưu trữ các dữ liệu sau khi chương trình kết thúc với dung lượng lớn và tồn tại lâu dài trong bộ nhớ. Các bài toán khi áp dụng vào ra với file chỉ khác về luồng input và output với các bài toán mà bạn đã làm trước đây, còn về thuật toán thì không có gì khác.



I. File text:

1. Tạo file:

Bước đầu tiên khi làm việc với file là tạo một file mới, để tạo một file mới ta sử dụng con trỏ kiểu FILE kết hợp với hàm open.

CÚ PHÁP

```
FILE *f;
//Mở file
f = fopen("fileName", "mode");
//Đóng file
fclose(f);
```

I. File text:

2. Các mode khi mở file:

Mode	Ý nghĩa
r	Mở file để đọc, nếu file chưa tồn tại sẽ báo lỗi
w	Mở file để ghi, nếu file chưa tồn tại nó sẽ tạo một file mới, ngược lại nó sẽ xóa hết nội dung của file trước khi ghi mới nội dung
a	Mở file để ghi, nếu file chưa tồn tại nó sẽ tạo file mới, ngược lại nó sẽ ghi tiếp vào cuối file, nội dung ban đầu của file được giữa nguyên
r+	Mở file để đọc, ghi file từ đầu file, nếu file chưa tồn tại thì không tạo file mới
w+	Mở file để đọc, ghi file, nếu file chưa tồn tại thì tạo mới, nếu file đã tồn tại thì xóa nội dung trước khi đọc ghi.
a+	Mở file để đọc ghi, nếu file chưa tồn tại thì tạo mới. Nếu đọc file thì sẽ đọc từ đầu file, nếu ghi file sẽ ghi từ cuối file

I. File text:

3. Các hàm ghi file:

- **fputc(char, file_pointer):** Ghi 1 kí tự vào file thông qua con trỏ file.
- **fputs(str, file_pointer):** Ghi 1 xâu kí tự vào file thông qua con trỏ file.
- **fprintf(file_pointer, format, variables):** Tương tự hàm printf nhưng ghi vào file thông qua con trỏ file.

I. File text:

3. Các hàm ghi file:

Ví dụ hàm fputc:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "w");
    char c[] = "28tech c programming !";
    for(int i = 0; i < strlen(c); i++){
        fputc(c[i], f);
    }
    fclose(f);
}
```

28tech_demo.txt

28tech c programming !

I. File text:

3. Các hàm ghi file:

Ví dụ hàm fputs:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "w");
    fputs("28tech C programming !\n", f);
    fputs("Become a better developer!\n", f);
    fputs("28tech.com.vn\n", f);
    fclose(f);
}
```

28tech_demo.txt

28tech c programming !
Become a better developer!
28tech.com.vn

I. File text:

3. Các hàm ghi file:

Ví dụ hàm fprintf:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "w");
    for(int i = 1; i <= 5; i++){
        fprintf(f, "%d ", i);
    }
    fprintf(f, "\n%s %s", "28tech", "28tech.com.vn\n");
    fclose(f);
}
```

28tech_demo.txt

1 2 3 4 5
28tech.com.vn

I. File text:

3. Các hàm ghi file:

Ví dụ hàm fprintf khi sử dụng mode là a:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "a");
    for(int i = 6; i <= 10; i++){
        fprintf(f, "%d ", i);
    }
    fprintf(f, "\n%s", "Noi dung duoc ghi vao cuoi file \n");
    fclose(f);
}
```

28tech_demo.txt

```
1 2 3 4 5
28tech.com.vn
6 7 8 9 10
Noi dung duoc ghi vao cuoi file
```

I. File text:

4. Các hàm đọc file:

Khi mở file lên để đọc, các bạn cần chú ý xem có mở được file hay không.

fgetc(file_pointer): Đọc 1 kí tự từ file, nếu tới cuối file và không đọc được nữa thì hàm này trả về EOF.

fscanf(file_pointer, format, variables): Tương tự như hàm scanf khi đọc từ bàn phím.

fgets(buffer, size, file_pointer): Tương tự như hàm gets, được dùng để đọc cả 1 dòng trong file.

I. File text:

4. Các hàm đọc file:

Ví dụ hàm fgetc:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "r");
    if(f == NULL){
        printf("Cannot open file !\n"); return 0;
    }
    char c = fgetc(f);
    while(c != EOF){
        printf("%c", c);
        c = fgetc(f);
    }
    fclose(f);
}
```

28tech_demo.txt

1 2 3 4 5
28tech.com.vn

OUTPUT

1 2 3 4 5
28tech.com.vn

I. File text:

4. Các hàm đọc file:

Ví dụ hàm fscanf:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "r");
    if(f == NULL){
        printf("Cannot open file !\n"); return 0;
    }
    int a[5];
    for(int i = 0; i < 5; i++){
        fscanf(f, "%d", &a[i]);
    }
    for(int i = 0; i < 5; i++){
        printf("%d ", a[i]);
    }
    char c[1000];
    fscanf(f, "%s", c);
    printf("\n%s", c);
    fclose(f);
}
```

Để đọc được file sử dụng fscanf bạn cần phải nắm được nội dung của file được ghi như thế nào.

OUTPUT

```
1 2 3 4 5
28tech.com.vn
```

I. File text:

4. Các hàm đọc file:

Giả sử trong file đang có các số nguyên, để đọc hết mọi số nguyên ta sử dụng cách sau:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "r");
    if(f == NULL){
        printf("Cannot open file !\n"); return 0;
    }
    int n;
    while(fscanf(f, "%d", &n) != -1){
        printf("%d ", n);
    }
    fclose(f);
}
```

28tech_demo.txt

```
1 2 3 4 5
6 7 8 9 10
11 12
13
```

OUTPUT

```
1 2 3 4 5 6 7 8 9 10 11 12 13
```

I. File text:

4. Các hàm đọc file:

Để đọc hết mọi từ trong file:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "r");
    if(f == NULL){
        printf("Cannot open file !\n"); return 0;
    }
    char c[1000];
    while(fscanf(f, "%s", c) != -1){
        printf("%s ", c);
    }
    fclose(f);
}
```

28tech_demo.txt

28tech become
python developer
c programming

OUTPUT

28tech become python developer c programming

I. File text:

4. Các hàm đọc file:

Ví dụ hàm fgets: Chú ý lệnh bị trôi

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "r");
    if(f == NULL){
        printf("Cannot open file !\n"); return 0;
    }
    char c[1000];
    while(fgets(c, 1000, f) != NULL){
        printf("%s", c);
    }
    fclose(f);
}
```

28tech_demo.txt

```
28tech become
python developer
c programming
```

28tech_demo.txt

```
28tech become
python developer
c programming
```

I. File text:

5. Hàm fseek và ftell:

Hàm fseek được sử dụng để di chuyển con trỏ file tới vị trí mong muốn

Cú pháp:

```
int fseek(FILE *pointer, long int offset, int position)
```

- pointer: con trỏ file
- offset: số lượng byte di chuyển
- position: vị trí bắt đầu di chuyển

Các vị trí bắt đầu cho con trỏ file:

- SEEK_END : Bắt đầu từ cuối file
- SEEK_SET : Bắt đầu từ đầu file
- SEEK_CUR : Bắt đầu từ vị trí hiện tại

Hàm ftell được sử dụng để chỉ ra vị trí của con trỏ file hiện tại, được tính bằng số byte từ đầu file từ vị trí hiện tại của con trỏ file.

Cú pháp: `ftell(FILE *pointer)`

I. File text:

5. Hàm fseek và ftell:

Ví dụ:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.txt", "r");
    if(f == NULL){
        printf("Cannot open file !\n"); return 0;
    }
    printf("%d\n", ftell(f));
    fseek(f, 10, SEEK_SET);
    printf("%d\n", ftell(f));
    fseek(f, 0, SEEK_END);
    printf("So byte cua file : %d\n", ftell(f));
    fclose(f);
}
```

28tech_demo.txt

28tech become

OUTPUT

0
10
So byte cua file: 13

II. File nhị phân:

- File nhị phân thường được lưu với đuôi file .bin, khác với file text thì nội dung của file nhị phân không ở dạng bản rõ (plain text).
- File nhị phân có nhiều ưu điểm: Lữu trữ dữ liệu tốn ít bộ nhớ hơn, không thể xem được nội dung file, bảo mật tốt hơn, tốc độ đọc ghi cao hơn
- Cách tạo file và mode mở file tương tự như file text, bạn chỉ cần thêm chữ b vào các mode như wb, rb, ab,...

II. File nhị phân:

Thông thường đối với file nhị phân sẽ sử dụng struct để đọc ghi, ví dụ đọc ghi 1 loạt thông tin sinh viên vào file nhị phân.

Struct sử dụng trong phần đọc ghi file nhị phân

```
struct SinhVien{
    char id[100];
    char ten[100];
    double gpa;
};

typedef struct SinhVien SV;
```

II. File nhị phân:

1. Hàm fwrite():

Cú pháp: fwrite(addressData, sizeData, numbersData, pointerToFile);

- addressData : Địa chỉ của biến mà bạn muốn ghi
- sizeData : Kích thước theo byte của biến mà bạn muốn ghi
- numbersData : Số lượng biến mà bạn muốn ghi
- pointerToFile : Con trỏ file

II. File nhị phân:

1. Hàm fwrite():

Ví dụ:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.bin", "wb");
    SinhVien sv;
    strcpy(sv.id, "CNTT1");
    strcpy(sv.ten, "Tran Xuan Loc");
    sv.gpa = 2.5;
    fwrite(&sv, sizeof(sv), 1, f);
    fclose(f);
}
```

Nội dung file 28tech_demo.bin:

28tech_demo.txt | 28tech_demo.bin | 0.cpp

```
1  434e 5454 3100 0000 e825 4000 0000 0000
2  2870 4000 0000 0000 1100 0110 0000 0000
3  5807 1800 0000 0000 86aa e445 f87f 0000
4  1020 4000 0000 0000 0000 0000 0000 0000
5  1020 4000 0000 0000 0800 0000 0000 0000
6  0000 0000 0000 0000 0000 0000 0000 0000
7  0800 0000 5472 616e 2058 7561 6e20 4c6f
8  6300 4000 0000 0000 1d00 0000 0000 0000
9  5013 1800 0000 0000 0100 0000 0000 0000
10 ffff ffff ffff ffff 1d00 0000 0000 0000
11 0100 0000 0000 0000 1924 4000 0000 0000
12 0000 0000 0000 0000 1d00 0000 0000 0000
13 0000 0000 0000 0000 0000 0000 0000 0440
```

II. File nhị phân:

2. Hàm fread():

Hàm read() được sử dụng khi bạn muốn đọc file nhị phân trong ngôn ngữ C

Cú pháp: fread(addressData, sizeData, numbersData, pointerToFile);

- addressData : Địa chỉ của biến mà bạn muốn ghi
- sizeData : Kích thước theo byte của biến mà bạn muốn ghi
- numbersData : Số lượng biến mà bạn muốn ghi
- pointerToFile : Con trỏ file

II. File nhị phân:

2. Hàm fread():

Ví dụ:

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

int main(){
    FILE *f;
    f = fopen("28tech_demo.bin", "rb");
    SinhVien sv;
    fread(&sv, sizeof(sv), 1, f);
    printf("%s %s %.2lf\n", sv.id, sv.ten, sv.gpa);
    fclose(f);
}
```

OUTPUT

```
CNTT1 Tran Xuan Loc 2.50
```