Chương 7:

XỬ LÝ TỆP TIN (FILE)

Tập tin (FILE)

- Kiểu FILE *
- Tập tin văn bản.
- Tập tin nhị phân.

2

Kiểu FILE *

- Kiểu FILE * (khai báo trong stdio.h) cho phép làm việc với các tập tin (văn bản, nhị phân).
- Khai báo con trỏ tập tin

  FILE * fp;

- chúng ta sử dụng con trỏ tập tin để truy cập (đọc, ghi, thêm thông tin) các tập tin.

3

Mở tập tin

FILE * fopen(const char *FileName, const char *Mode);

- Filename: tên tập tin cần mở. có thể chỉ định một đường dẫn đầy đủ chỉ đến vị trí của tập tin.
- Mode: chế độ mở tập tin: chỉ đọc, để ghi (tạo mới), ghi thêm.
- Nếu thao tác mở thành công, fopen trả về con trỏ FILE trỏ đến tập tin FileName.
- Nếu mở không thành công (FileName không tồn tại, không thể tạo mới), fopen trả về giá trị NULL.

4

Đóng tập tin

int fclose(FILE *filestream);

- filestream: con trỏ đến tập tin đang mở cần đóng.
- Nếu thao tác đóng thành công, fclose trả về 0.
- Nếu có lỗi (tập tin đang sử dụng), fclose trả về giá trị EOF.

5

Ví dụ : Mở, Đóng tập tin

```
1. FILE * fp;

2. // mở VB.TXT “chỉ đọc”
3. if( (fp = fopen( “VB.TXT”, “r” )) == NULL)
4.     { printf( “Tap tin khong mo duoc\n”);
5.       exit(1);
6.     }
   /* ... */

7. fclose( fp );
```

6

Tập tin văn bản

- Tập tin văn bản là kiểu tập tin được lưu trữ các thông tin dưới dạng kiểu ký tự.
- Truy xuất tập tin văn bản:
  - theo từng ký tự
  - theo từng dòng
- chế độ mở trên tập tin văn bản
  - "r" : đọc (tập tin phải có trên đĩa)
  - "w" : ghi (ghi đè lên tập tin cũ hoặc tạo mới nếu tập tin không có trên đĩa)
  - "a" : ghi nối vào cuối tập tin.
  - "r+" : đọc/ghi. Tập tin phải có trên đĩa.
  - "a+" : đọc, ghi vào cuối tập tin. Tạo mới tập tin nếu tập tin chưa có trên đĩa.

7

Ví dụ : Đếm số từ trong tập tin văn bản.

```
1.  #include <iostream.h>
2.  #include <conio.h>
3.  #include <stdlib.h>
4.  int main()
5.     { FILE * fp;
6.     char filename[67], char ch;
7.     int count = 0, isword = 0;
8.     cout<<"Filename: " ;  gets (filename);
9.     if ((fp = fopen (filename, "r" )) == NULL ) // mở tập tin mới để đọc
10.       { cout<<"Open file error"<<endl; exit (1); }
```

8

Ví dụ : Đếm số từ trong tập tin văn bản.

```
11.   while (( ch = getc ( fp ) ) != EOF ) // đọc cho đến hết tập tin
12.      {
13.      if (( ch >= 'a' && ch <= 'z' ) || ( ch >= 'A' && ch <= 'Z' ))
14.           isword =1;
15.      if (( ch == ' ' || ch == '\n' || ch == '\t' ) && isword )
16.           { count ++; isword = 0; }
17.      }
18. Cout<<"Number of word: "<<count<<endl);
19.   fclose ( fp );
20.   }
```

9

fgets()

- fgets: đọc chuỗi ký tự từ tập tin.

  char * fgets (char *Str, int NumOfchar, FILE *fp);

  fgets đọc các ký tự trong tập tin cho đến khi gặp một trong các điều kiện:
  - ***EOF***
  - ***gặp dòng mới***
  - ***đọc được (NumOfchar - 1) ký tự trước khi gặp hai điều kiện trên***.

- fgets trả về chuỗi ký tự đọc được (kết thúc bằng \0) hoặc trả về con trỏ NULL nếu EOF hoặc có lỗi khi đọc.

10

fputs()

- fputs: ghi chuỗi ký tự ra tập tin.

  int fputs (const char *Str, FILE * fp);

- fputs trả về EOF nếu thao tác ghi có lỗi.

11

Hàm chép tập tin văn bản

- Có trong các thư viện

  #include <iostream.h>
  #include <string.h>

- /* chép từ SourceFile sang DestFile và trả về số ký tự đọc được

12

feof(); fscanf(), fprintf(); fflush()

- feof: cho biết đã đến cuối tập tin chư a(EOF).

  int feof (FILE *fp);

  fprintf:

  int fprintf (FILE *fp, const char * Format, ...);

  fscanf:

  int fscanf (FILE *fp, const char * Format, ...);

  fflush:

  int fflush (FILE * fp);

  Buộc ghi ra tập tin các dữ liệu có trong buffer.

13

header file fstream

- Chúng ta tìm hiểu **về fstream, ifstream** và **ofstream** trong C++. Đây là các class mới được thêm vào trong C++ giúp việc thao tác với file trong C++ trở nên dễ dàng hơn so với các phương pháp truyền thống được kế thừa từ ngôn ngữ C.
- Trong đó:
  - **ifstream** dùng để nhập file trong C++
  - **ofstream** dùng để xuất file trong C++
  - **fstream** được gộp lại từ 2 class trên, dùng để nhập xuất file trong C++
- Để dùng được 3 class này, chúng ta cần phải include các header file tương ứng có cùng tên là **ifstream, ofstream và fstream** vào đầu chương trình

14

header file fstream

fstream trong C++

- fstream là một class có chức năng nhập/ xuất file trong C++. fstream được viết tắt từ các cụm từ *file* và *stream* trong tiếng Anh, dịch sang tiếng Việt có nghĩa là **luồng file**.
- fstream được gộp lại từ 2 class là ofstream và ifstream, do vậy bằng cách sử dụng fstream, chúng ta có thể sử dụng tất cả các chức năng của 2 class ofstream và ifstream
- Lưu ý là để sử dụng được fstream trong C++, chúng ta cần include header file **fstream** vào đầu chương trình

15

header file fstream

ofstream trong C++

- ofstream là một class cung cấp chức năng của một luồng file đầu ra. ofstream được viết tắt từ các cụm từ *out*, *file* và *stream* trong tiếng Anh, dịch sang tiếng Việt có nghĩa là **luồng file đầu ra**.
- Giống như trong C chúng ta mở file bằng cách tạo ra một cấu trúc FILE để chứa thông tin của file cần đọc, thì trong C++ chúng ta dùng class ofstream để tạo ra một thực thể **stream** (instance) chứa thông tin về file cần mở, và sau đó thao tác với file thông qua **stream** này.
- *Lưu ý, là để sử dụng được fstream trong C++, chúng ta cần include header file **fstream** hoặc là **ofstream** vào đầu chương trình.*

16

header file fstream

ofstream trong C++

Mở file để ghi bằng ofstream

- Để sử dụng ofstream, trước hết chúng ta cần thêm header file **fstream** vào chương trình, sau đó thì sử dụng cú pháp sau đây để mở file để ghi bằng ofstream trong C++,

```
1. using namespace std;
2. int main(){
3.     ofstream ofs(filepath);
4. }
```

```
1.  #include <iostream>
2.  #include <fstream>

3.  using namespace std;

4.  int main()
5.  {
6.      //Mở file bằng ofstream
7.      ofstream ofs("test.txt");
8.  }
```

17

header file fstream

- Nếu mở file thành công bằng ofstream thì `ofs` sẽ được trả về. Tuy nhiên nếu mở file thất bại thì `ofs` sẽ không tồn tại. Ứng dụng điều này, chúng ta có thể viết xử lý tránh lỗi như sau:

```
1.  if (ofs) //Mở file thành công
2.  {
3.      //Viết xử lý khi mở file thành công
4.  }
5.  if (!ofs) //Mở file thất bại
6.  {
7.      //Viết xử lý khi mở file thất bại
8.  }
```

18

header file fstream

Để mở file bằng hàm open trong ofstream, trước hết chúng ta cần khai báo một stream bằng ofstream, sau đó thì truyền giá trị của đường dẫn file vào hàm như sau:

```
std::ofstream ofs;
ofs.open(filepath);
```

Dùng hàm ofs.close(); để đóng tệp

19

header file fstream

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    std::ofstream ofs;                  //Khai báo stream
    ofs.open(test.txt);                 //Mở file bằng stream
    ofs << "abc" << std::endl;          //Ghi vào file
    ofs.close();                        //Đóng file
}
```

20

header file fstream

Cũng giống như khi sử dụng hàm fopen() để mở file, thì trong ofstream cũng chuẩn bị sẵn các mode để mở file bằng hàm thành viên open() như sau:

mode	Cách dùng
std::ios::out	Mở để ghi(chế độ mặc định) Khi viết, nội dung cũ bị xóa
std::ios::in	Mở để đọc
std::ios::app	Mở để ghi chèn
std::ios::ate	Sau khi mở thì di chuyển về vị trí cuối file
std::ios::trunc	Mở để ghi, tương tự mode out
std::ios::binary	Mở file nhị phân

21

header file fstream

Ghi file trong C++ bằng ofstream

- Kết hợp các kiến thức ở trên, chúng ta mở file bằng cách tạo **stream** từ **ofstream**, sau đó tiến hành ghi file và đóng file như ví dụ sau đây

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{   ofstream ofs("test.txt"); //Mở file bằng ofstream
    //Kiểm tra file đã mở thành công hay chưa
    if (!ofs) {
        cerr << "Error: file not opened." << endl;
        return 1;
        }
//ghi và in lần lượt các ký tự vào file
    ofs << "I am a big big girl!"<< endl;
    ofs << "in a big big World";
    //Đóng file
    ofs.close();
    return 0;
}
```

22

header file fstream

Lấy và thay đổi vị trí đọc file trong C++
Để lấy và thay đổi vị trí đang đọc và ghi file hiện tại, chúng ta sử dụng tới hàm thành viên seekp() trong class ofstream. Hàm này cũng có chức năng tương tự như hàm fseek() trong C

mode	Cách dùng
std::ios::beg	Di chuyển tới đầu file
std::ios::cur	Lấy vị trí hiện tại trong file
std::ios::end	Di chuyển tới cuối file

23

header file fstream

Hãy lấy vị trí cuối file, sau đó thêm nội dung vào file và kiểm tra lại vị trí này như sau:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{   //Mở file và di chuyển tới cuối file
  std::ofstream ofs("hello.txt", std::ios::in | std::ios::ate);
  if (!ofs)
  {       cout << "Cant open file" << std::endl;
          return 0;
  }
            //Hiển thị vị trí hiện tại trong file
  cout << "Vi tri hien tai: " << ofs.tellp() << std::endl;
            //Ghi chèn dòng abcde vào file
  ofs << "\nVietnam" << std::endl;
  cout << "Vi tri hien tai: " << ofs.tellp() << std::endl;
           //Dịch chuyển vị trí hiện tại về đầu file 2 ký tự
  ofs.seekp(-2, std::ios::cur);
  cout << "Vi tri hien tai: " << ofs.tellp() << std::endl;
  return 0;
}
```

24

header file fstream

ifstream trong C++ là gì
ifstream là một class cung cấp chức năng của một luồng file đầu vào. ifstream được viết tắt từ các cụm từ *in*, *file* và *stream* trong tiếng Anh, dịch sang tiếng Việt có nghĩa là **luồng file đầu vào**.
Cách dung:

```
1. #include <iostream>
2. #include <fstream>
3. using namespace std;
4. int main()
5. {
6.     //Mở file bằng ifstream
7.     ifstream ifs("test.txt");
8. }
```

25

header file fstream

Kết hợp các kiến thức ở trên, chúng ta mở file này bằng cách tạo stream từ ifstream, sau đó tiến hành đọc file và đóng file như ví dụ sau đây:

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
int main()
{   string filename("sample.txt");
    vector<string> lines; // Khai báo vector để lưu các dòng đọc được
    string line;
    ifstream input_file(filename); //Mở file bằng ifstream
    //Kiểm tra file đã mở thành công chưa
    if (!input_file.is_open()) {
        cerr << "Could not open the file - '" << filename << "'" << endl;
        return EXIT_FAILURE;   }
    //Đọc từng dòng trong
    while (getline(input_file, line)){
        lines.push_back(line);//Lưu từng dòng như một phần tử vào
vector lines.  }
    //Xuất từng dòng từ lines và in ra màn hình
    for (const auto &i : lines)
            cout << i << endl;
    input_file.close();
}
```

26

header file fstream

- Đọc file trong C++. Ngoài cách dùng các hàm được kế thừa từ ngôn ngữ C thì chúng ta cũng có thể đọc file trong C++ bằng các phương pháp mới được tích hợp trong **header file fstream**,
- ví dụ như sử dụng toán tử >>, hoặc là hàm **getline()** trong C++
- **class fstream** bao gồm cả **class ifstream** và có khả năng vừa đọc vừa ghi file, nên thông thường chúng ta sẽ chọn class này để tạo ra stream chứa thông tin file cần đọc.

27

header file fstream

- Ví dụ cụ thể, chúng ta include header file **fstream** và mở file trong C++ bằng ifstream như sau:

```
#include <iostream>
#include <fstream>
 using namespace std;
int main()
{
    //Mở file bằng ifstream
ifstream ifs("test.txt");
}
```

- Sau khi đã mở file thành công bằng ifstream, một stream ifs chứa thông tin file đã được tạo ra, và lúc này chúng ta đã có thể tiến hành **đọc dữ liệu từ file trong C++** bằng cách thao tác với stream này với các phương pháp sau đây:

28

header file fstream

- **Đọc file trong C++ bằng toán tử >> trong ifstream**
- Toán tử >> trong ifstream được sử dụng để gán từng nội dung đọc từ file vào một biến, hoặc một chuỗi chỉ định.
  Cú pháp sử dụng toán tử >> để đọc file trong C++ như sau:
  ifstream ifs >> variable;
- Trong đó ifs là luồng chứa file được tạo ra khi mở file bằng ifstream, và variable là một biến kiểu string hoặc char để lưu dữ liệu đọc từ file ra.

Nếu chỉ định variable là một biến thuộc kiểu *string* thì chúng ta có thể đọc được từng từ trong file. Và nếu chỉ định variable là một biến thuộc kiểu *char* thì chúng ta có thể đọc được từng ký tự trong file.
Lưu ý, toán tử >> trong ifstream chỉ có thể đọc từng từ hoặc từng ký tự từ file trong C++, do đó chúng ta cần sử dụng toán tử này kết hợp với một vòng lặp để có thể đọc toàn bộ dữ liệu trong file.

29

header file fstream

Đọc từng ký tự trong file bằng toán tử >>
Chúng ta chỉ định **variable** là một biến thuộc kiểu *char* để đọc từng ký tự trong file bằng **toán tử >>**.
Ví dụ cụ thể, chúng ta có file **sample.txt** với nội dung sau đây:
"123abc"
Chúng ta sẽ đọc từng ký tự từ file này và xuất ra màn hình bằng cách sử dụng toán tử >> trong C++ như sau:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{ ifstream ifs("sample.txt");//Mở file bằng ifstream
//Kiểm tra file đã mở thành công hay chưa
    if(!ifs){
            cerr << "Error: file not opened." << endl;
            return 1;    }
char chr; //Khai báo biến kiểu char để lưu ký tự đọc được
//Đọc và in lần lượt các ký tự trong file
while(ifs >> chr){
            cout << chr << endl;
            }
ifs.close();//Đóng file
}
```

30

header file fstream

Chúng ta chỉ định variable là một biến thuộc kiểu ***string*** để đọc từng từ trong file bằng **toán tử >>**.
Ví dụ cụ thể, chúng ta có file **sample.txt** với nội dung sau đây:
"I am a big big girl!"
Chúng ta sẽ dùng ifstream để mở file, sau đó đọc từng từ trong file này và xuất ra màn hình bằng cách sử dụng toán tử >> trong C++ như sau

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{   //Mở file bằng ifstream
    ifstream ifs("sample.txt");
    //Kiểm tra file đã mở thành công hay chưa
    if(!ifs){
            cerr << "Error: file not opened." << endl;
            return 1;
            }
    string str;  //Khai báo biến kiểu string để lưu từng từ đọc được
        //Đọc và in lần lượt các từ trong file
    while(ifs  >> str){
            cout << str << endl;
            }
    ifs.close();          //Đóng file
    return 0;
}
```


header file fstream

Hàm getline() trong C++
Hàm getline() trong C++ là một hàm có sẵn trong thư viện chuẩn, có tác dụng **đọc từng dòng trong file** chỉ định.
Hàm getline đọc liên tục các ký tự từ file và lưu trữ nó trong một chuỗi cho đến khi nó tìm thấy ký tự phân cách.
Chúng ta có thể chỉ định ký tự phân cách này, và giá trị mặc định của ký tự phân cách là ký tự xuống dòng \n.
Chúng ta sử dụng hàm getline trong C++ với cú pháp sau đây:

std::getline(input, str, delimiter)

Trong đó:
•input: là luồng chứa file được tạo ra khi mở file bằng ifstream
•str: là chuỗi để lưu kết quả đọc file
•delimiter: là ký tự phân cách, và chúng ta có thể lược bỏ đối số này khi muốn sử dụng ký tự phân cách mặc định là ký tự xuống dòng \n.

*•Lưu ý, hàm getline chỉ có thể **đọc từng dòng file trong C++**, do đó chúng ta cần sử dụng hàm này kết hợp với một vòng lặp để có thể đọc toàn bộ các dòng trong file*

header file fstream

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
int main()
{    string filename("sample.txt");
     vector<string> lines;     // Khai báo vector để lưu các dòng đọc được
    string line;
    ifstream input_file(filename); //Mở file bằng ifstream
   //Kiểm tra file đã mở thành công chưa
   if (!input_file.is_open()) {
cerr << "Could not open the file - '"
          << filename << "'" << endl;
    return EXIT_FAILURE;
   }
   //Đọc từng dòng trong
   while (getline(input_file, line)){
      lines.push_back(line);
//Lưu từng dòng như một phần tử vào vector lines.
   }
  //Xuất từng dòng từ lines và in ra màn hình
   for (const auto &i : lines)
      cout << i << endl;
    input_file.close();
}
```


header file fstream

Ghi file trong C++ bằng toán tử << trong ofstream

Toán tử << trong ofstream được sử dụng để ghi dữ liệu vào file trong C++.
Cú pháp sử dụng toán tử << để ghi file trong C++ như sau:

```
ofs << data;
```

Trong đó `ofs` là luồng chứa file được tạo ra khi mở file bằng ofstream, và `data` là dữ liệu cần ghi vào file. Dữ liệu này có thể là một ký tự, hoặc một chuỗi trong C++

Cần chú ý, toán tử << trong ofstream chỉ có thể ***ghi từng ký tự hoặc từng chuỗi vào file trong C++***, do đó chúng ta cần sử dụng toán tử này kết hợp với một vòng lặp để có thể ghi dữ liệu nhiều lần vào file.

header file fstream

Ghi từng ký tự vào file bằng toán tử <<
Chúng ta chỉ định data là một ký tự để ghi từng ký tự vào file bằng **toán tử <<**.
Chúng ta sẽ ghi từng ký tự vào file này bằng cách sử dụng toán tử << trong C++ như sau:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{   ofstream ofs("test.txt"); //Mở file bằng ofstream
    //Kiểm tra file đã mở thành công hay chưa
    if(!ofs){
        cerr << "Error: file not opened." << endl;
        return 1;
    }
    //Ghi lần lượt các ký tự vào file
    ofs << 'H';    ofs << 'e';    ofs << 'l';
    ofs << 'l';    ofs << 'o';    ofs << endl;
    ofs.close(); //Đóng file
    return 0;
}
```


header file fstream

Ghi từng dòng vào file trong C++ bằng toán tử <<
Chúng ta chỉ định data là một chuỗi ký tự để ghi từng dòng vào file trong C++ bằng **toán tử <<**.
Ví dụ cụ thể, chúng ta muốn tạo file `sample.txt` và ghi vào file với nội dung sau đây:

```
I am a big big girl!
in a big big World
```

Chúng ta sẽ ghi từng dòng vào file này bằng cách sử dụng toán tử << trong C++ như sau. Lưu ý là để xuống dòng thì chúng ta cần phải ghi thêm ký tự xuống dòng vào cuối mỗi chuỗi ký tự

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{   ofstream ofs("test.txt"); //Mở file bằng ofstream
    //Kiểm tra file đã mở thành công hay chưa
    if(!ofs){
        cerr << "Error: file not opened." << endl;
        return 1;
    }
      //ghi và in lần lượt các ký tự vào file
    ofs << "I am a big big girl!"<< endl;
    ofs << "in a big big World";
    ofs.close();   //Đóng file
    return 0;
}
```

header file fstream

Ghi file trong C++ bằng hàm write() trong fstream

Hàm **write()** trong C++ là một hàm có sẵn trong thư viện chuẩn, có tác dụng **ghi chuỗi vào file trong C++**. Hàm write nhận địa chỉ của một chuỗi tại bộ nhớ, và ghi n ký tự được chỉ định từ chuỗi này vào trong file.

Chúng ta sử dụng hàm write trong C++ với cú pháp sau đây:

```
write (const char* s, streamsize n);
```

CHú ý: để dung hàm write(), chúng ta cần khai báo namespace

using namespace std;

*Chú ý, hàm write chỉ có thể **ghi từng dòng file trong C++**, do đó chúng ta cần sử dụng hàm này nhiều lần nếu muốn ghi nhiều dòng, và đừng quên ghi thêm cả ký tự xuống dòng ở cuối chuỗi để xuống dòng trong file.*

37

header file fstream

Chúng ta mở file cần ghi bằng ofstream, sau đó ghi từng dòng vào file trong C++ bằng hàm write như ví dụ sau đây:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{   string text1("I am a big big girl!\n");
    string text2( "in a big big World");
    ofstream ofs("test2.txt");    //Mở file bằng ofstream
    //Kiểm tra file đã mở thành công hay chưa
    if (!ofs){
            cerr << "Error: file not opened." << endl;
            return 1;
           }
    //Ghi từng dòng vào file bằng hàm write
    ofs.write(text1.data(), text1.size());
    ofs.write(text2.data(), text2.size());
    //Đóng file
    ofs.close();
    return 0;
}
```

38

header file fstream

Ghi mảng vào file trong C++

Ứng dụng một trong hai phương pháp ở trên, chúng ta có thể thực hiện việc **ghi mảng vào file trong C++**.

Ví dụ chúng ta sử dụng vòng lặp để lấy giá trị từng phần tử, sau đó ghi vào file bằng toán tử << như sau:

```
#include <iostream>
#include <fstream>
using namespace std;
//Tạo macro tìm số phần tử trong mảng
int main()
{    ofstream ofs("nums.txt"); //Mở file bằng ofstream
    //Kiểm tra file đã mở thành công hay chưa
    if(!ofs){
        cerr << "Error: file not opened." << endl;
        return 1;
      }
    int num[ ] = {10, 20, 44, 60, 82};
    //Ghi từng phần tử từ mảng vào file
    for (int i =0; i < 5; i++)
        ofs << num[i] << "    ";
     ofs.close();  //Đóng file
    return 0;
}
```

39

Tập tin Nhị phân

- Tập tin nhị phân là một chuỗi các ký tự, không phân biệt ký tự in được hay không in được.
- Tập tin nhị phân thường dùng để lưu trữ các cấu trúc (struct) hoặc union.
- Khai báo:

  FILE * fp;
- Truy xuất tập tin nhị phân theo khối dữ liệu nhị phân.

40

Tập tin Nhị phân

- các chế độ mở tập tin nhị phân:
  - "rb" : mở chỉ đọc
  - "wb" : ghi (ghi đè lên tập tin cũ hoặc tạo mới nếu tập tin không có trên đĩa)
  - "ab" : ghi nối vào cuối tập tin.
  - "rb+" : đọc/ghi. Tập tin phải có trên đĩa.
  - "wb+" : tạo mới tập tin cho phép đọc ghi.
  - "ab+" : đọc, ghi vào cuối tập tin. Tạo mới tập tin nếu tập tin chưa có trên đĩa.

41

Đọc ghi tập tin Nhị phân

- **fread() : đọc**

  size_t fread (void *Ptr, size_t ItemSize, size_t NumItem, FILE * fp);

  fread đọc NumItem khối dữ liệu, mỗi khối có kích thước ItemSize từ fp và chứa vào vùng nhớ xác định bởi Ptr.

  fread trả về số khối dữ liệu đọc được.

  Nếu có lỗi hoặc EOF thì giá trị trả về nhỏ hơn NumItem
- **fwrite() : ghi**

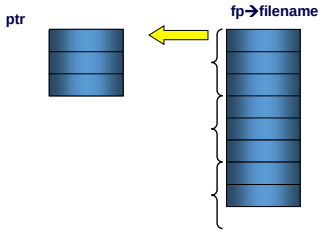
  size_t fwrite (const void *Ptr, size_t ItemSize, size_t NumItem, FILE * fp);

  fwrite ghi khối (NumItem x ItemSize) xác định bởi Ptr ra fp.

42

Đọc ghi tập tin Nhị phân

- chép 3 mục từ tập tin filename vào vùng nhớ trỏ bởi ptr



43

con trỏ FILE

- Một tập tin sau khi mở được quản lý thông qua một con trỏ FILE.
- Khi mở tập tin (**wb, rb**), con trỏ FILE chỉ đến đầu tập tin.
- Khi mở tập tin (**ab**), con trỏ FILE chỉ đến cuối tập tin.
- con trỏ FILE chỉ đến từng byte trong tập tin nhị phân.
- Sau mỗi lần đọc tập tin, con trỏ FILE sẽ di chuyển đi một số byte bằng kích thước (byte) của khối dữ liệu đọc được.

44

fseek()

- các hằng dùng trong di chuyển con trỏ FILE

```
#define     SEEK_SET    0
#define     SEEK_cUR    1
#define     SEEK_END    2
```

int fseek(FILE *fp, long int offset, int whence);

fseek di chuyển con trỏ fp đến vị trí offset theo mốc whence

- offset: khoảng cách (byte) cần di chuyển tính từ vị trí hiện tại. (offset > 0: đi về phía cuối tập tin, offset < 0: ngược về đầu tập tin)
- whence: SEEK_SET: tính từ đầu tập tin
  SEEK_cUR: tính từ vị trí hiện hành của con trỏ
  SEEK_END: tính từ cuối tập tin
- fseek trả về: 0 nếu thành công, <>0 nếu di chuyển có lỗi

45

ftell() và rewind()

- rewind đặt lại vị trí con trỏ về đầu tập tin.

void rewind (FILE * fp);

tương đương với *fseek (fp, 0L, SEEK_SET);*

- ftell trả về vị trí offset hiện tại của con trỏ

long int ftell (FILE * fp);

Nếu có lỗi, ftell trả về -1L

46

Ví dụ: xác định kích thước tập tin.

```
...// khai báo biến cẩn thận

1.  if ( fp = fopen ( FileName, "rb" ) ) == NULL )
2.      fprintf( stderr, "cannot open %s\n", FileName );
3.  else
4.      {
5.      fseek( fp, 0, SEEK_END );
6.      FileSize = ftell( fp );
7.      cout<<"File size :"<<FileSize<<"byte";
8.      fclose( fp );
9.      }
```

47

Vị trí con trỏ: fgetpos() và fsetpos()

- với các tập tin có kích thước cực lớn fseek và ftell sẽ bị giới hạn bởi kích thước của offset.
- Dùng

int fgetpos (FILE *fp, fpos_t *position);

int fsetpos (FILE *fp, const fpos_t *position);

48

Xóa và đổi tên tập tin

- Thực hiện xoá

  int remove(const char *filename);

- Đổi tên tập tin

  int rename(const char *oldname, const char *newname);

49

chú ý khi làm việc với tập tin

- Khi mở tập tin filename, tập tin này phải nằm cùng thư mục của chương trình hoặc
- Phải cung cấp đầy đủ đường dẫn đến tập tin
  vd: c:\baitap\taptin.dat
  viết như thế nào?

  fp = fopen("c:\baitap\taptin.dat", "rb");

SAI RỒI

- vì có ký tự đặc biệt '\' nên viết đúng sẽ là:
  fp = fopen("c:\\baitap\\taptin.dat", "rb");

50

Tham số dòng lệnh chương trình

- Khi gọi chạy một chương trình, chúng ta có thể cung cấp các tham số tại dòng lệnh gọi chương trình.
  ví dụ: dir A:*.c /w
  "copy", "A:*.c" và "/w" là hai tham số điều khiển chương trình.
- command-line arguments.
- các tham số dòng lệnh được tham chiếu qua hai đối số khai báo trong hàm main.
  main (int argc, char * argv[ ])
  { ... }

  argc: argument count
- *argv: argument vector*

51

Tham số dòng lệnh – ví dụ

- ... // addint.c → addint.exe

```
1. int main (int argc, char * argv [ ])
2.   {
3.     int res = 0;
4.     cout<<" Program "<<argv[0];
5.     for ( int i = 1; i < argc; i++ )
6.           res += atoi(argc[i]);
7.     cout<<res );
8. }
```

- G:\>addint 3 –2 1 5 6 -2 1
  12

52

Bài tập

- Bài 1: viết chương trình đếm số từ có trong một tệp văn bản
- Bài 2: viết chương trình đếm số ký tự có trong một tệp văn bản (không tính các ký tự kết thúc dòng)
- Bài 3: viết chương trình tách đều một tệp làm hai tệp con
- Bài 4: viết chương trình tách một tệp thành các tệp có kích thước không lớn hơn MAX cho trước

53