

Thao tác với filestream

0. File và dòng

- C++ coi file là một chuỗi byte - stream kết thúc bằng ký hiệu end-of-file (EOF)

[IMG]file:///C:/Users/tuanhai/AppData/Local/Temp/msohtmlclip1/01/clip_image001.jpg
[/IMG]

- Khi file mở:
 - một đối tượng được tạo và kết nối với một dòng
 - tạo "đường liên lạc" từ đối tượng tới file
 - các đối tượng cin, cout, v.v... được tạo khi <iostream> được include
- liên lạc giữa chương trình và file/thiết bị qua dòng (stream)

Trong C++ thì ta có 3 loại File stream cơ bản sau :

ifstream : Dùng cho file nhập vào. Loại này chỉ có thể được dùng để đọc dữ liệu vào file vào bộ nhớ mà thôi.

ofstream : Dùng cho file xuất ra. Loại này thì có thể dùng để tạo ra files và chép dữ liệu vào chúng.

fstream : Đây là kênh file.(File stream). Loại này thì có thể vừa tạo file, vừa ghi dữ liệu vào file và đọc dữ liệu từ file vào luôn.

I.Cách sử dụngfstream :

Để định nghĩa 1 đối tượng file ta dùng cú pháp sau :

`fstream dataFile`

Ở đây dataFile chỉ là tên do người dùng đặt mà thôi.

Để mở 1 file ta dùng cú pháp sau :

`dataFile.open("info.txt", ios::out);`

Ở đây hàm open đòi hỏi 2 đối số : đối thứ nhất là 1 chuỗi tên chứa tên file. Đối thứ 2 là 1 flag của file và cái này cho ta biết chế độ nào mà chúng ta dùng để mở file. Ở ví dụ trên thì tên file là **info.txt** còn flag file ở đây là **ios::out**. Cái này nó nói cho C++ biết chúng ta mở file ở chế độ xuất ra.

Chế độ xuất ra cho phép dữ liệu có thể được ghi vào file.

```
datafile.open("info.txt", ios::in);
```

Còn ở ví dụ này thì tức là ta đang mở file ở chế độ nhập vào, tức là cho phép dữ liệu được đọc vào từ file.

Chúng ta có 1 số chế độ cơ bản sau đây :

ios::app Chế độ gắn vào. Nếu file đã được tạo thì nội dung của nó sẽ được gắn vào tận cùng của file. Theo mặc định thì đối với chế độ này thì nếu file chưa được tạo nó sẽ tạo ra 1 file mới.

ios::ate Nếu file đã được tạo, thì chương trình sẽ chạy tới trực tiếp chỗ cuối cùng của file. Xuất ra có thể là được ghi ra bất kì chỗ nào trong file.

ios::binary Chế độ nhị phân. Khi mà file được mở ra ở chế độ này thì dữ liệu sẽ được đọc hay ghi từ 1 định dạng nguyên thủy nhị phân.

ios::trunc Nếu file đã được tạo thì nội dung của nó sẽ bị xóa đi.

Chúng ta còn có thể sử dụng những chế độ trên chung với nhau và chúng sẽ được kết nối với nhau bằng toán tử |. Ví dụ

```
dataFile.open("info.txt", ios::in | ios::out);
```

Dòng lệnh trên cho phép ta mở file info.txt ở cả 2 chế độ xuất và nhập.

Chú ý : Khi dùng riêng lẻ thì ios::out sẽ xóa nội dung của file nếu file đã được tạo sẵn. Tuy nhiên nếu dùng chung với ios::in, thì nội dung file cũ sẽ được giữ lại. Và nếu file chưa được tạo, nó sẽ tạo ra 1 file mới cho chúng ta luôn.

Các phép toán

- !outClientFile hoặc !inClientfile

Trả về nonzero (true) nếu badbit hoặc failbit bật (mở file không tồn tại để đọc, không có quyền mở)

- Toán tử *

Chuyển đổi đối tượng dòng thành con trỏ

Trả về NULL khi failbit hoặc badbit được bật (failbit bật khi gặp EOF)

- while (inClientFile >> myVariable)

–lặp cho đến khi gặpEOF

–Ghi/đọc file (như cout, cin)

- outClientFile << myVariable

- inClientFile >> myVariable

– Đóng file outClientFile.close();

Bây giờ là 1 ví dụ hoàn chỉnh :

program 1

[CODE=cpp]// This program uses an fstream object to write data to a file.

```
#include <iostream>
#include <fstream>
using namespace std;
```

```
int main()
{
    fstream dataFile;

    cout << "Opening file...\n";
    dataFile.open("demofile.txt", ios::out); // Mở file để ghi vào
    cout << "Now writing data to the file.\n";
    dataFile << "Jones\n"; // Ghi dòng thứ 1
    dataFile << "Smith\n"; // Ghi dòng thứ 2
    dataFile << "Willis\n"; // Ghi dòng thứ 3
    dataFile << "Davis\n"; // Ghi dòng thứ 4
    dataFile.close(); // Đóng file
    cout << "Done.\n";
    return 0;
}[/CODE]
```

Program Output

Code:

Opening file...

Now writing data to the file.

Done.

File Output

Code:

Jones

Smith

Willis

Davis

Bây h là 1 ví dụ khác, chỉnh sửa lại ví dụ trên 1 tí. Và chúng ta sẽ thêm vào chế độ ios::app. Chúng ta sẽ mở file ra ghi vào 2 tên sau đó đóng lại rồi lại mở ra ghi vào tiếp thêm 2 tên nữa.

program 2

[CODE=cpp]// This program writes data to a file, closes the file,
// then reopens the file and appends more data.

```

#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream dataFile;

    cout << "Opening file...\n";
    //Mở file ở chế độ xuất
    dataFile.open("demofile.txt", ios::out);
    cout << "Now writing data to the file.\n";
    dataFile << "Jones\n"; // Ghi dòng thứ 1
    dataFile << "Smith\n"; // Ghi dòng thứ 2
    cout << "Now closing the file.\n";
    dataFile.close(); // Đóng file

    cout << "Opening the file again...\n";
    // Mở file ở chế độ gắn vào.
    dataFile.open("demofile.txt", ios::out | ios::app);
    cout << "Writing more data to the file.\n";
    dataFile << "Willis\n"; // Ghi dòng thứ 3
    dataFile << "Davis\n"; //Ghi dòng thứ 4
    cout << "Now closing the file.\n";
    dataFile.close(); // Đóng file

    cout << "Done.\n";
    return 0;
}[/CODE]

```

Output File

Code:

Jones

Smith

Willis

Davis

Ở ví dụ 2 này ta có thể hình dung như sau :

J O N E S \n S M I T H H \n <EOF>

Khi file được đóng lại thì kí tự end-of-file sẽ được tự động ghi vào. Khi file lại được mở ra thì dữ liệu mới lại được gắn vào chỗ end-of-file đó.

J O N E S \n S M I T H H \n W I L L I S \n D A V I S \n <EOF>

Chú ý : nếu chúng ta thử bỏ thẳng ios::out đứng 1 mình mà không có ios::app thì lần thứ

2 mở file là nội dung đầu sẽ bị xóa hết (tức là 2 cái tên John và Smith sẽ bị xóa) Và lần thứ 2 mở ra ghi lại thì cuối cùng ta chỉ còn 2 tên là Willis and Davis.

Chúng ta đã thấy mặc dù 2 thằng ifstream và ofstream chỉ giới hạn 1 cái được nhập vào và 1 cái xuất ra nhưng nếu sử dụng với những chế độ ios:: khác nhau thì ta có thể vừa xuất vừa nhập như thường.

II.Kiểm tra file có tồn tại hay không trước khi mở :

Đôi khi chúng ta sẽ cần quyết định xem file có tồn tại trước khi chúng ta mở nó ra hay không và sau đây là 1 ví dụ:

PHP Code:

```
fstream dataFile;
dataFile.open("value.txt", ios::in);
if(dataFile.fail())
{
//Nếu file không tồn tại, thì tạo ra 1 file mới
dataFile.open("value.txt", ios::out);
//...
}
else
{
dataFile.close();
}
```

III.Định dạng file xuất ra :

program 3

[CODE=cpp]// This program uses the setprecision and fixed
// manipulators to format file output.

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
```

```
int main()
{
fstream dataFile;
double num = 17.816392;
```

```
dataFile.open("numfile.txt", ios::out);
```

```

dataFile << fixed; // Định dạng fixed-point
dataFile << num << endl; //Xuất ra num

dataFile << setprecision(4); // Định dạng 4 phần thập phân sau dấu .
dataFile << num << endl; //Xuất ra num

dataFile << setprecision(3); // Định dạng 3 phần thập phân sau dấu .
dataFile << num << endl; // Xuất ra num

dataFile << setprecision(2); // 2 phần
dataFile << num << endl; // xuất ra

dataFile << setprecision(1); // 1 phần
dataFile << num << endl; // xuất ra

cout << "Done.\n";
dataFile.close(); // Đóng file
return 0;
}[/CODE]

```

Output

Code:

```

17.816392
17.8164
17.816
17.82
17.8

```

program 4

```

[CODE=cpp]
// This program writes three rows of numbers to a file.
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;

int main()
{
const int ROWS = 3; //Dòng
const int COLS = 3; //Cột

```

```

int nums[ROWS][COLS] = { 2897, 5, 837,
34, 7, 1623,
390, 3456, 12 };
fstream outFile("table.txt", ios::out);

//Ghi dữ liệu ra với khoảng cách là 8
for (int row = 0; row < ROWS; row++)
{
for (int col = 0; col < COLS; col++)
{
outFile << setw(8) << nums[row][col];
}
outFile << endl;
}
outFile.close();
cout << "Done.\n";
return 0;
}
[/CODE]

```

IV.Cách truyền 1 file name vào hàm :

Chúng ta khi làm việc với những chương trình thực sự thì đôi khi chúng ta cần phải truyền 1 tên file vào hàm nào đó để tiện cho việc quản lý, nhưng khi truyền phải lưu ý là luôn luôn **truyền bằng tham chiếu..**

program 5

```

[CODE=cpp]#include <iostream>
#include <fstream>
#include <string>

using namespace std;
bool OpenFile(fstream &file, char *name);
void ShowContents(fstream &file);

int main()
{
fstream dataFile;
if(!OpenFile(dataFile, "demo.txt"))
{
cout << "Error !" << endl;
return 0;
}
}
[/CODE]

```

```

}
cout << "Successfully.\n";
ShowContents(dataFile);
dataFile.close();

return 0;
}

bool OpenFile(fstream &file, char *name)
{
file.open(name, ios::in);
if(file.fail())
return false;
else
return true;
}

void ShowContents(fstream &file)
{
string line;
while(getline(file, line)){
cout << line << endl;
}
}[/CODE]

```

Nội dung file

Code:

```

wow
ohlalal 32131
hehe
hoho

```

Output

Code:

Successfully.

```

wow
ohlalal 32131
hehe
hoho

```

Press any key to continue . . .

V.Dùng hàm thành viên để đọc và ghi file :

Khoảng trắng (Whitespace) là 1 kí tự mà nó là 1 phần của dữ liệu, vấn đề sẽ nảy sinh khi ta đọc vào bằng toán tử >>. Bởi vì toán tử >> nó xem khoảng trắng như 1 kí tự kết thúc (delimiter), vì thế nó sẽ không đọc chúng vào. Xem ví dụ sau :

program 6

```
[CODE=cpp]// This program demonstrates how the >> operator should not
// be used to read data that contains whitespace characters
// from a file.
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    const int SIZE = 81; // Size của mảng input
    char input[SIZE]; // Mảng lưu input
    fstream nameFile;

    //Mở file ở chế độ đọc
    nameFile.open("murphy.txt", ios::in);
    if (!nameFile)
    {
        cout << "ERROR: Cannot open file.\n";
        return 0;
    }

    //Đọc nội dung file.
    while (nameFile >> input)
    {
        cout << input;
    }

    nameFile.close();
    return 0;
}[/CODE]
```

Nội dung file

Code:

Jayne John
47 Circle street
CA 93065

Output:

Code:

JayneJohn47Circle streetCA93065

Chúng ta sẽ dùng hàm **getline()** để xử lý những trường hợp như vậy.

program 7

[CODE=cpp]// This program uses the file stream object's getline member
// function to read a line of data from the file.

```
#include <iostream>
#include <fstream>
using namespace std;
```

```
int main()
{
    const int SIZE = 81;
    char input[SIZE];
    fstream nameFile;

    nameFile.open("murphy.txt", ios::in);
    if (!nameFile)
    {
        cout << "ERROR: Cannot open file.\n";
        return 0;
    }
```

```
    nameFile.getline(input, SIZE); //Dùng kí tự mặc định \n như kí tự kết thúc.
    while (!nameFile.eof())
    {
        cout << input << endl;
        nameFile.getline(input, SIZE); //Chỗ này cũng vậy.
    }
```

```
    nameFile.close();
    return 0;
}[/CODE]
```

Output

Code:

Jayne John
47 Circle street
CA 93065

Hoặc nếu không thích dùng kí tự mặc định '\n' của hàm getline() thì ta có thể dùng 1 kí tự bất kì khác như sau ví dụ :

program 8

[CODE=cpp]

```
// This file demonstrates the getline function with a user-  
// specified delimiter.
```

```
#include <iostream>  
#include <fstream>  
using namespace std;
```

```
int main()  
{  
    const int SIZE = 81;  
    char input[SIZE];
```

```
    fstream dataFile("names2.txt", ios::in);
```

```
    // Bây h dùng kí tự $ là kí tự kết thúc.  
    dataFile.getline(input, SIZE, '$');  
    while (!dataFile.eof())  
    {  
        cout << input << endl;  
        dataFile.getline(input, SIZE, '$');  
    }
```

```
    dataFile.close();  
    return 0;  
}  
[/CODE]
```

Nội dung file

JayneJohn\n\$47Circle street\n\$CA93065\n\$

Output

Code:

Jayne John
47 Circle street

I.Hàm get() :

Hàm get() là 1 hàm rất hữu dụng trong việc thao tác với file. Ví dụ :

```
inFile.get(ch);
```

Trong ví dụ trên thì ch là 1 biến kiểu char. Một kí tự sẽ được đọc vào từ file và lưu vào ch. Chương trình sau sẽ là 1 demo cho cách dùng get.

program 1

```
[CODE=cpp]// This program asks the user for a file name. The file is  
// opened and its contents are displayed on the screen.
```

```
#include <iostream>  
#include <fstream>  
using namespace std;
```

```
int main()  
{  
    const int SIZE = 51; // Array size for file name  
    char fileName[SIZE]; // To hold the file name  
    char ch; // To hold a character  
    fstream file; // File stream object
```

```
    // Get the file name  
    cout << "Enter a file name: ";  
    cin >> fileName;
```

```
    // Open the file.  
    file.open(fileName, ios::in);  
    if (!file)  
    {  
        cout << fileName << " could not be opened.\n";  
        return 0;  
    }
```

```
    // Get each character from the file and display them.  
    file.get(ch);  
    while (!file.eof())  
    {  
        cout << ch;  
        file.get(ch);  
    }
```

```
// Close the file.  
file.close();  
return 0;  
}[/CODE]
```

Chương trình trên sẽ xuất ra nội dung của bất kì file như thế nào. Hàm get() sẽ đọc luôn những kí tự trắng vì thế nội dung file sẽ y chang như nó xuất hiện trong file.

II. Hàm put()

Hàm put sẽ ghi 1 kí tự vào file. Ví dụ :

```
outFile.put(ch);
```

Trong ví dụ trên thì biến ch là kiểu char.

Ví dụ về cách dùng put()

program 2

```
[CODE=cpp]// This program demonstrates the put member function.
```

```
#include <iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
char ch; // To hold a character
```

```
// Open the file for output.
```

```
fstream dataFile("sentence.txt", ios::out);
```

```
cout << "Type a sentence and be sure to end it with a ";
```

```
cout << "period.\n";
```

```
// Get a sentence from the user one character at a time
```

```
// and write each character to the file.
```

```
cin.get(ch);
```

```
while (ch != '.')
```

```
{
```

```
dataFile.put(ch);
```

```
cin.get(ch);
```

```
}
```

```
dataFile.put(ch); // Write the period.
```

```
// Close the file.  
dataFile.close();  
return 0;  
}[/CODE]
```

III. Các hàm định vị cho file tuần tự

- con trỏ vị trí ghi số thứ tự của byte tiếp theo để đọc/ghi
- các hàm đặt lại vị trí của con trỏ:
 - seekg (đặt vị trí đọc cho lớp istream)
 - seekp (đặt vị trí ghi cho ostream)
 - seekg và seekp lấy các đối số là offset và mốc (offset: số byte tương đối kể từ mốc)
- Mốc(ios::beg mặc định)
 - ios::beg - đầu file
 - ios::cur - vị trí hiện tại
 - ios::end - cuối file
- các hàm lấy vị trí hiện tại của con trỏ:
 - tellg và tellp
- Ví dụ

PHP Code:

`fileObject.seekg(0)`

đến đầu file (vị trí 0), mặc định đối số thứ hai là ios::beg

`fileObject.seekg(n)`

đến byte thứ n kể từ đầu file

`fileObject.seekg(n, ios::cur)`

tiến n byte

`fileObject.seekg(y, ios::end)`

lùi y byte kể từ cuối file

`fileObject.seekg(0, ios::cur)`

đến cuối file

seekp tương tự

`location = fileObject.tellg()`

lấy vị trí đọc hiện tại của fileObject

IV. Các xử lý với nhiều file

program 3

[CODE=cpp]// This program demonstrates reading from one file and writing
// to a second file.

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <cctype> // Needed for the toupper function.
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
const int SIZE = 51; // Array size for file name
```

```
char fileName[SIZE]; // To hold the file name
```

```
char ch; // To hold a character
```

```
ifstream inFile; // Input file
```

```
// Open a file for output.
```

```
ofstream outFile("out.txt");
```

```
// Get the input file name.
```

```
cout << "Enter a file name: ";
```

```
cin >> fileName;
```

```
// Open the file for input.
```

```
inFile.open(fileName);
```

```
if (!inFile)
```

```
{
```

```
cout << "Cannot open " << fileName << endl;
```

```
return 0;
```

```
}
```

```
// Process the files.
```

```
inFile.get(ch); // Get a char from file 1
```

```
while (!inFile.eof()) // Test for end of file
```

```
{
```

```
outFile.put(toupper(ch)); // Write uppercase char to file 2
```

```
inFile.get(ch); // Get another char from file 1
```

```
}
```

```
// Close the files.
```

```
inFile.close();
```

```
outFile.close();
cout << "File conversion done.\n";
return 0;
}[/CODE]
```

VI. File nhị phân :

Định nghĩa : File nhị phân là file chứa nội dung không nhất thiết phải là ASCII text. Tất cả những file từ đầu tới h chúng ta thao tác đều là text files. Có nghĩa là dữ liệu trong những file này đều đã được định dạng dưới mã ASCII. Thậm chí là số đi chăng nữa khi nó được lưu trong file với toán tử << thì nó đã đc ngầm định chuyển về dạng text. Ví dụ :

PHP Code:

```
ofstream file("num.dat");
short x = 1297;
file << x;
```

Dòng lệnh cuối cùng của ví dụ trên sẽ ghi nội dung của x vào file. Và chúng được lưu vào ở dạng ký tự '1', '2', '9', '7'.

Thực sự là con số 1297 không hề được lưu trong bộ nhớ. Nó đã được định dạng thành 1 số nhị phân, và chiếm 2 byte trong bộ nhớ máy tính.

Vì x kiểu short nó sẽ được lưu như sau :

00000101 | 00010001

Và đây chính là dữ liệu nguyên thủy được lưu trong bộ nhớ.

Và để làm được như vậy chúng ta sẽ có cú pháp như sau :

```
file.open("stuff.dat", ios::out | ios::binary);
```

Ghi chú : mặc định của compiler thì file sẽ được mở ở định dạng text.

V. Hàm write và read :

1. Write

-Hàm write dùng để ghi 1 file stream ở định dạng nhị nhận. Dạng tổng quát của hàm write như sau :

```
fileObject.write(address, size);
```

Ở đây chúng ta có những lưu ý sau :

-fileObject là tên của đối tượng file stream.

-address là địa chỉ đầu tiên của 1 vùng nhớ được ghi vào file. **Đối số này có thể là địa chỉ của 1 ký tự hoặc là con trỏ tới kiểu char.**

-size là số lượng byte của vùng nhớ mà nó được write. Đối số này bắt buộc phải là kiểu integer(số nguyên dương)

Chúng ta xét ví dụ sau :

PHP Code:

```
char letter = 'A';  
file.write(&letter, sizeof(letter));
```

-Đối thứ nhất ở đây là địa chỉ của biến letter. Và đối này sẽ nói cho hàm write biết rằng dữ liệu được ghi vào file ở đâu trong vùng nhớ.

-Đối thứ 2 sẽ là kích thước của biến letter, và đối này sẽ nói cho hàm write biết số lượng byte của dữ liệu sẽ ghi vào file. Bởi vì dữ sẽ được lưu khác nhau trên tùy hệ thống khác nhau, nên cách tốt nhất là chúng ta dùng toán tử sizeof để quyết định số bytes được ghi. Và sau khi hàm này được thực hiện, nội dung của biến letter sẽ được ghi vào file nhị phân của đối tượng "file" đó.

Chúng ta xem tiếp 1 ví dụ sau :

PHP Code:

```
char data[] = {'A', 'B', 'C', 'D'};  
file.write(data, sizeof(data));
```

Trong ví dụ này thì đối thứ 1 là tên của mảng (data). Vì khi ta truyền tham số là tên của mảng thì tức là ta đã truyền con trỏ tới vị trí đầu tiên của mảng. Đối thứ 2 cũng có ý nghĩa tương tự như ví dụ 1. Và sau khi gặp này thực hiện thì nội dung của mảng sẽ được ghi vào file nhị phân tương ứng với đối tượng file.

2.Read:

Hàm read thì sẽ dùng đọc vào số dữ liệu nhị phân từ file vào bộ nhớ máy tính.

Dạng tổng quát là :

```
fileObject.read(address, size);
```

-Ở đây fileObject là tên của đối tượng file stream.

-address là địa chỉ đầu tiên mà vùng nhớ mà dữ liệu được đọc vào được lưu.

Và đối này có thể là địa chỉ của 1 kí tự hay 1 con trỏ tới kiểu char.

-size cũng là số lượng byte trong bộ nhớ được đọc vào từ file. Và đối này bắt buộc cũng phải là số kiểu integer (nguyên dương)

Tương tự hàm read ta cũng có 2 ví dụ sau :

PHP Code:

```
char letter;  
file.read(&letter, sizeof(letter));
```

Và :

PHP Code:

```
char data[4];  
file.read(data, sizeof(data));
```

Chương trình sau sẽ minh họa cách sử dụng 2 hàm read and write :

program 4

[CODE=cpp]// This program uses the write and read functions.

```
#include <iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
const int SIZE = 4;
```

```
char data[SIZE] = {'A', 'B', 'C', 'D'};
```

```
fstream file;
```

```
// Open the file for output in binary mode.
```

```
file.open("test.dat", ios::out | ios::binary);
```

```
// Write the contents of the array to the file.
```

```
cout << "Writing the characters to the file.\n";
```

```
file.write(data, sizeof(data));
```

```
// Close the file.
```

```
file.close();
```

```
// Open the file for input in binary mode.
```

```
file.open("test.dat", ios::in | ios::binary);
```

```
// Read the contents of the file into the array.
```

```
cout << "Now reading the data back into memory.\n";
```

```
file.read(data, sizeof(data));
```

```
// Display the contents of the array.
```

```
for (int count = 0; count < SIZE; count++)
```

```
cout << data[count] << " ";
```

```
cout << endl;
```

```
// Close the file.
```

```
file.close();
```

```
return 0;
```

```
}[/CODE]
```

Nếu chúng ta muốn ghi các kiểu khác vào file nhị phân thì ta phải dùng cú pháp có 1 tí đặc biệt sau đây.

`reinterpret_cast<dataType>(value)`

Ở cú pháp trên thì dataType sẽ là kiểu dữ liệu mà chúng ta muốn ép về, và value sẽ là giá trị mà chúng ta muốn ép.

Ví dụ :

PHP Code:

```
int x = 65;  
file.write(reinterpret_cast<char *>(&x), sizeof(x));
```

Đối với mảng thì :

PHP Code:

```
const int SIZE = 10;  
int numbers[SIZE] = {1,2,3,4,5,6,7,8,9,10};  
file.write(reinterpret_cast<charr *>(numbers), sizeof(numbers));
```

program 5

[CODE=cpp]// This program uses the write and read functions.

```
#include <iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
const int SIZE = 10;
```

```
fstream file;
```

```
int numbers[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

```
// Open the file for output in binary mode.
```

```
file.open("numbers.dat", ios::out | ios::binary);
```

```
// Write the contents of the array to the file.
```

```
cout << "Writing the data to the file.\n";
```

```
file.write(reinterpret_cast<char *>(numbers), sizeof(numbers));
```

```
// Close the file.
```

```
file.close();
```

```
// Open the file for input in binary mode.
```

```
file.open("numbers.dat", ios::in | ios::binary);
```

```
// Read the contents of the file into the array.
```

```
cout << "Now reading the data back into memory.\n";
```

```
file.read(reinterpret_cast<char *>(numbers), sizeof(numbers));
```

```
// Display the contents of the array.  
for (int count = 0; count < SIZE; count++)  
    cout << numbers[count] << " ";  
cout << endl;
```

```
// Close the file.  
file.close();  
return 0;  
}[/CODE]
```