1 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

|  |
| --- |
| **BÀI 7: HÀM VÀ LỚP MẪU : Template** |

**1. Khái niệm:**

Mỗi hàm cần có kiểu đối số và kiểu trả về được xác định. Khi gọi hàm cần phải gởi các đối số có kiểu phù  hợp, nếu muốn gởi đối số có kiểu khác với kiểu qui định thì cần phải viết thêm hàm mới.

Ví dụ: Viết hàm hoán vị hai kí tự

#include <iostream>

void hoanvi(char &a, char &b)

{

char c=a; a=b; b=c;

}

int main()

{

 char a='A', b='B';

 hoanvi(a,b);

 cout<<a<<','<<b; // xuat B, A

--------------------------

*float x=1.2, y=2.3;*

*hoanvi(x,y);*

*cout<<endl<<x<<','<<y; // xuat 1.2, 2.3 -> hoan vi khong thanh cong*

}

Ta không thể dùng hàm hoán vị hai kí tự để hoán vị hai số thực mà phải viết hàm khác, mặc dù rất giống  với hàm hoán vị hai kí tự.

//hàm hoán vị hai số thực

*void hoanvi(float &a, float &b)*

*{*

*float c=a; a=b; b=c;*

*}*

Nếu có nhu cầu hoán vị hai cấu trúc sinh viên, ta lại phải víết một hàm hoán vị khác. Để chỉ cần viết một  hàm hoán vị cho tất cả các biến thuộc kiểu khác nhau, C++ cung cấp kỹ thuật hàm mẫu.

2 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

**2. Hàm mẫu (template function)**

• **Khái niệm**

Hàm mẫu là hàm có tham số có kiểu trừu tượng, nên hàm mẫu có thể thực hiện với nhiều kiểu dữ liệu khác  nhau. Khi thực hiện, dựa trên kiểu của tham số truyền vào cho hàm, chương trình dịch sẽ tạo ra một hàm  thể hiện tương ứng.

• **Cú pháp:**

|  |
| --- |
| **template <các kiểu trừu tượng>**  **<định nghĩa hàm>** |

Ví dụ: Viết hàm mẫu hoán vị

#include <iostream>

|  |
| --- |
| template <class Type> //Type la kieu truu tuong |

void hoanvi (Type &a, Type &b)

{

Type c=a; a=b; b=c;

}

struct sinhvien

{

char hoten[30];

float diemthi;

};

int main()

{

char a='A', b='B';

**hoanvi(a,b); //hoan vi hai ki tu**

cout<<a<<','<<b;

--------------------------

float x=1.2, y=2.3;

**hoanvi(x,y); //hoan vi hai so thuc**

cout<<endl<<x<<','<<y;

----------------------

sinhvien sv1={"nguyen hoang", 8};

sinhvien sv2={"tran minh ",10};

**hoanvi(sv1,sv2); //hoan vi hai sinh vien**

cout<<endl<<sv1.hoten<<','<<sv1.diemthi;

cout<<endl<<sv2.hoten<<','<<sv2.diemthi;

}

Có thể xây dựng hàm mẫu với nhiều kiểu trừu tượng

ví dụ:

#include <iostream>

using namespace std;

|  |
| --- |
| template <class type1, class type2> |

void xuat(type1 x, type2 y) //type1, type2 la kieu truu tuong

{

cout<<x<<','<<y<<endl;

}

int main()

{

xuat("hello world",5);

 xuat(1.5,'a');

}

3 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

Khi hàm mẫu không phù hợp với một kiểu dữ liệu nào đó, ta cần tạo một thể hiện mới (viết hàm khác,  giống hàm mẫu nhưng với kiểu dữ liệu cụ thể)

Ví dụ:

Xây dựng hàm mẫu “tìm giá trị nhỏ nhất” sao cho có thể tìm giá trị nhỏ nhất của hai kí tự, hai số thực, hai  sinhvien .

#include <iostream>

|  |
| --- |
| template <class T> //T la kieu truu tuong |

T min(T &a, T &b) //ham mau nay khong the dung cho kieu sinhvien

{

return (a<b)?a:b; //vi phep toan < chi dung de so sanh hai so

}

struct sinhvien

{

char hoten[30];

float diemthi;

};

//tao mot the hien cua ham mau min

sinhvien min(sinhvien &sv1, sinhvien &sv2) //ham the hien cua ham mau

{

return (sv1.diemthi<sv2.diemthi)?sv1:sv2;

}

void main()

{

char a='a', b='B', c;

**c=min(a,b); //tim gia tri nho nhat giua hai ki tu**

cout<<c<<endl;

float x=1.2, y=-2.3, z;

**z=min(x,y); //tim gia tri nho nhat giua hai so thuc**

cout<<z<<endl;

sinhvien sv1={"tran van", 8};

sinhvien sv2={"tran minh ",1};

**sinhvien sv=min(sv1,sv2); //tim gia tri nho nhat giua hai sinh vien**

cout<<sv.hoten<<','<<sv.diemthi;

}

4 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

**3. Lớp mẫu:**

Lớp mẫu là lớp có các thành phần dữ liệu hay phương thức sử dụng kiểu trừu tượng, kiểu dữ liệu sẽ được  truyền đến hàm dưới dạng tham số nên lớp mẫu có thể sử dụng với nhiều kiểu dữ liệu khác nhau. Các hàm  trong lớp mẫu, mặc định là các hàm mẫu. Khi định nghĩa biến thuộc lớp mẫu, cần chỉ rõ kiểu cụ thể.

• **Cú pháp**

|  |
| --- |
| **template <tham số> <định nghĩa lớp>** |

• **Ví dụ**

Xây dựng lớp tập hợp sao cho có thể nhập/xuât hai tập hợp dạng số (nguyên, thực) hoặc hai tập hợp  dạng kí tự.

#include <iostream>

//T la kieu truu tuong va duoc xac dinh khi khai bao doi tuong

template <class T, int n> // n goi la tham so bieu thuc, gia tri truyen cho n phai la hằng

|  |
| --- |
|  |

class taphop

{

public:

T data[n]; //chua du lieu cua tap hop, n la so phan tu toi da

int count; //so phan tu cua tap hop

void nhap();

void xuat();

};

|  |
| --- |
| template <class T, int n> |

void taphop<T,n>::nhap() // nhap() la ham mau

{

cout<<"\nSo phan tu cua tap hop:";

do

{

cin>>count;

if (count>n)cout<<"\nKhong du bo nho";

}while (count>n);

for (int i=0; i<count; i++) cin>>data[i];

}

|  |
| --- |
| template <class T, int n> |

void taphop<T,n>::xuat() // xuat() cung la ham mau

{

cout<<endl<<'[';

for (int i=0; i<count-1; i++) cout<<data[i]<<',';

cout<<data[count-1]<<']';

}

void main()

{

cout<<"\nNhap tap ki tu:";

**taphop <char, 20> A;**

A.nhap(); A.xuat();

cout<<"\nNhap tap so nguyen:";

**taphop <int, 20> B;**

B.nhap(); B.xuat();

cout<<"\nNhap tap so thuc:";

**taphop <float, 20> C;**

C.nhap(); C.xuat();

}

5 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

|  |
| --- |
| **BÀI 6: HỆ THỐNG NHẬP XUẤT** |

**1. Nhập xuất chuẩn**

Nhập dữ liệu từ bàn phím vào bộ nhớ và đọc dữ liệu từ bộ nhớ ra màn hình gọi là nhập xuất chuẩn.  Luồng nhập (đối tượng nhập) cin thuộc lớp istream dùng để nhập dữ liệu,

luồng xuất (đối tượng xuất) cout thuộc lớp ostream dùng để xuất dữ liệu.

Lớp istream, ostream, iostream thừa kế lớp ios.

**2. Định dạng dữ liệu Xuất/Nhập chuẩn**

Dùng các hàm trong tập tin tiêu đề **iomanip**

|  |  |
| --- | --- |
| **Hàm** | **Ý nghĩa** |
| dec/oct/hex/bin | xuất dạng cơ số 10,8,16, 2 |
| setw(int w) | đặt độ rộng là w |
| setfill(int ch) | dùng ch là kí tự lấp đầy |
| setprecision(int p) | đặt độ chính xác là p số lẻ |
| setiosflags(long f) | đặt cờ f |
| resetiosflags(long f) | xoá cờ f |
| Flush | Làm sạch luồng xuất |

**Cờ f gọi là cờ định dạng (format flags):** dùng để xác định cách hiện dữ liệu, cờ là một số long định nghĩa  trong lớp ios và gồm các hằng static sau:

|  |  |
| --- | --- |
| **Flag** | **Ý nghĩa** |
| left/ right | canh trái/phải (mặc định là canh trái) |
| Showpoint | hiện dấu chấm và phần lẻ của số nguyên |
| Scientific | hiện số dạng khoa học 0.xxx e+n hay 0.xxx e-n |
| Fixed | hiện dạng dấu chấm cố định có 6 số lẻ (mặc định) |
| Uppercase | viết hoa kí tự e hoặc kí tự hệ 16 |
| Showbase | hiện cơ số 0(oct), 0X(hex) |

**ví dụ:**

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{ int x; cout<<"Nhap so nguyen can thu nghiem (nen nhap >10):"; cin>>x;

 float y; cout<<"Nhap so thuc can thu nghiem (nen nhap co so thap phan):"; cin>>y;  char c; cout<<"Nhap ký tu can thu nghiem (VD: $):"; cin>>c;

 char z[10]; cout<<"Nhap chuoi can thu nghiem:"; cin.ignore(); cin.getline(z,10);

 cout<<oct<<x<<'\t'<<hex<<x<<"\t"<<dec<<x<<endl;

 cout<<setw(10)<<z<<endl;

 cout<<setw(20)<<setfill(c)<<x<<setfill(c)<<endl;

 cout<<setprecision(3)<<y/3<<endl;

 cout<<setw(10)<<setiosflags(ios::left)<<setfill(c)<<z<<z<<endl;

 cout<<setiosflags(ios::showpoint)<<y/2<<endl;

 cout<<setiosflags(ios::scientific|ios::uppercase)<<y<<c<<y<<c<<endl;

 cout<<setiosflags(ios::fixed)<<y<<endl;

 cout<<setiosflags(ios::uppercase|ios::showbase)<<hex<<x<<endl;

}

6 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

=====================

cout<<oct<<10<< '\t' <<hex<<10<<dec<< endl; //12 a

cout<<setw(10)<<"chao"<<endl; // chao

cout<<setw(5)<<setfill('$')<<255<<setfill(‘ ‘); //$$255

cout<<setprecision(3)<<20.0/3; //6.67

cout<<setw(10)<<setiosflags(ios::left)<<setfill(‘.’)<<"chao"<<"ban";//chao……ban cout<<setiosflags(ios::showpoint)<<6.0/2; //3.000000

cout<<setiosflags(ios::scientific|ios::uppercase)<<0.012<<’,’<<12.34<<',';

cout<<setiosflags(ios::fixed)<<0.07; //1.2E-02,1.234E+01,0.07 cout<<setiosflags(ios::uppercase|ios::showbase)<<hex<<255; //0XFF

**3. Nhập xuất tập tin**

**3.1 Phân loại tập tin:**

Có hai loại tập tin là tập tin văn bản và tập tin nhị phân . Tập tin văn bản khi đọc/ghi sẽ thực hiện chuyển  đổi theo bảng Ascii, ví dụ kí tự ‘\n’ lưu bằng hai byte 10,13. Tập tin nhị phân khi đọc/ghi không thực hiện  bất cứ sự chuyển đổi nào.

**3.2 Các hàm nhập xuất tập tin:**

Khai báo trong tập tin tiêu đề **fstream**

**a) Tạo luồng**

Trước khi truy xuất tập tin cần tạo luồng nhập hoặc luồng xuất hoặc luồng nhập/xuất, rồi sử dụng luồng để  mở file

- **Luồng nhập**: là đối tượng thuộc lớp ifstream, dùng luồng này để đọc dữ liệu từ tập tin - **Luồng xuất**: là đối tượng thuộc lớp ofstream, dùng luồng này để ghi dữ liệu vào tập tin - **luồng nhập xuất**: là đối tượng thuộc lớp fstream, dùng luồng này để đọc, ghi dữ liệu vào tập tin

**b) Mở file**

**void open (const char\* filename, int mode)**

filename là tên file cần mở, mode là chế độ mở, mode có những giá trị hằng được khai báo trong lớp ios như  sau:

|  |  |
| --- | --- |
| **Mode** | **Ý nghĩa** |
| App | có rồi ghi vào cuối, nếu không có tạo mới. |
| binary/text | File nhị phân/ văn bản (mặc định là văn bản) |
| In | nhập (luồng ifstream mặc định là nhập) |
| Out | xuất (luồng ofstream mặc định là xuất), có rồi ghi đè, nếu không có tạo mới. |
| Ate | chuyển con trỏ tới cuối file |

**Ví dụ:**

ifstream fin; //khai bao doi tuong **fin** de doc file ( luồng nhập)

//dùng luồng fin mở file autoexec.bat để đọc

fin.**open**(“c:/autoexec.bat”);// hoặc có thể ghi fin.open(“c:\\autoexec.bat”);

ofstream fout; //tạo luồng xuất

//dùng luồng fout mở file config.sys để ghi vào cuối

fout.open(“c:\\config.sys”,ios::out|ios::app);// hoặc có thể ghi fout.open(“c:\\config.sys”,ios::app);

fstream f; //tạo luồng nhập,xuất

//dùng luồng f mở file sinhvien.dat dạng nhị phân để đọc,ghi

f.open(“sinhvien.dat”,ios::binary|ios::in|ios::out);// in, out là bắt buộc

Khi mở không thành công thì luồng sẽ bằng 0, do đó cần kiểm tra luồng trước khi truy xuất file.

7 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

**ví dụ:**

if (fin)

{

//các lệnh truy xuất file fin

}

Có thể dùng phương thức thiết lập có sẵn để tạo luồng đồng thời mở file.

**Ví dụ:**

//tạo luồng nhập fin và dùng luồng fin mở file autoexec.bat để đọc

**ifstream fin (“c:\\autoexec.bat”);**

//tạo luồng xuất fout và dùng luồng fout mở file config.sys để ghi vào cuối

**ofstream fout (“c:\\config.sys”,ios::app);**

//tạo luồng nhập,xuất f và dùng luồng f mở file sinhvien.dat dạng nhị phân để đọc,ghi **fstream f (“sinhvien.dat”,ios::binary|ios::in|ios::out) ;**

**c) Đóng file**

Khi file không sử dụng nữa, cần phải đóng lại để cập nhật dữ liệu từ buffer vào file. **void close();**

**ví dụ:**

**fin.close();** //đóng file autoexec.bat

**d) Đọc ghi file văn bản**

• **Dùng các phép toán nhập (>>), xuất (<<):**

Thay các luồng cin, cout bằng các luồng nhập/xuất file. Thông tin lưu trên file có định dạng giống như  xuất trên màn hình.

**Ghi chú**: Đọc bằng phép toán >> thì kí tự khoảng cách, kí tự xuống dòng bị bỏ qua. **Ví dụ:**

#include <fstream>

int main()

{

char ch;

ifstream fin ("d:/quehuong.txt");

while (1)

{

**fin>>ch;** //doc mot ki tu tu file quehuong.txt

**if (!fin) break;** //neu cuoi tap tin thi thoat

cout<<ch; //xuat ki tu ra màn hình

}

}

Ghi chú:

- Mỗi tập tin có một biến gọi là con trỏ file để ghi nhận vị trí đọc hoặc ghi file. Khi mới mở file, con trỏ file =0 nghĩa là sẽ được đọc/ghi từ đầu file. Khi thực hiện việc đọc/ghi file thì con trỏ file tự động được  cập nhật lại vị trí mới.

- Ở cuối file có kí tự kết thúc file (gọi là kí tự EOF: End Of File). Đối tượng fin ở ví dụ trên sẽ =0 khi con  trỏ file chỉ vào vị trí sau kí tự kết thúc file.

- Có thể viết lại ví dụ trên như sau:

while (!fin.eof()) //hàm eof() tra ve 0 neu chua ket thuc file

{

fin>>ch; //doc mot ki tu tu tap tin que huong

if (fin) //neu chua het file

cout<<ch; //xuat ki tu ra man hinh

}

8 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

**e) Đọc ghi file bất kỳ:** có ba cách: theo byte, theo dòng, theo khối

• **Đọc ghi theo byte**

|  |  |
| --- | --- |
| istream &get(char &ch) | đọc một byte từ luồng nhập vào ch, luồng nhập sẽ bằng 0 nếu đến cuối file |
| istream &put(char &ch) | ghi một byte từ ch vào luồng xuất. |

**Ghi chú:** Đọc bằng get() sẽ không bỏ qua bất cứ kí tự nào.

**Ví dụ:**

**Viết chương trình tạo tập tin quehuong.txt và xuất nội dung tập tin ra màn hình.**

#include <fstream>

void main()

{

char ch;

**ofstream fout("d:/quehuong.txt");** //mo de ghi

cout<<"\nNhap noi dung tap tin:\n";

while (1)

{

//nhap ki tu tu ban phim, nhap duoc khoang trang, ki tu xuong dong

**cin.get(ch);** //khong nen dung cin>>ch;

**if (ch==26) break;** //neu nhan F6 hoac ctrl-Z thi ngung nhap

**fout<<ch;** // ghi ki tu vao tap tin que huong, co the dung **fout.put(ch);**

}

fout.close(); //dong tap tin

cout<<"\nXuat noi dung tap tin:\n";

**ifstream fin ("d:/quehuong.txt");** //mo ra de doc

while (1)

{

**fin.get(ch);** //doc mot ki tu tu tap tin que huong, khong nen dung fin>>ch;

**if (!fin) break;**//neu cuoi tap tin thi thoat, co the ghi đk la (fin==0) hoac ch==EOF cout<<ch; //xuat ki tu ra man hinh

}

}

• **Đọc theo dòng**

istream & getline(char \*buf, int n, char ch=’\n’)

Đọc một dòng tối đa n kí tự vào buf hoặc cho tới khi gặp kí tự ch. Lệnh sẽ bỏ qua kí tu ch. **Ví dụ:** Đọc tập tin quehuong.txt theo dòng và hiện ra màn hình

#include <fstream>

int main()

{

char buf[512];

cout<<"\nXuat:\n";

ifstream fin ("d:/quehuong.txt");

while (1)

{

**fin.getline(buf,512);**

**if (!fin) break;**

cout<<buf<<endl;

}

}

9 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân) • **Đọc ghi theo khối (nhiều byte)**

|  |  |
| --- | --- |
| istream &read(unsigned char \*buf, int n) | đọc n byte từ file vào buf |
| ostream &write(unsigned char \*buf, int n) | ghi n byte từ buf vào file |
| int gcount() | số byte đọc/ghi được |

**Ví dụ:**

#include <fstream>

#include <iomanip>

#include <conio.h>

#include <iostream>

using namespace std;

struct sinhvien

{

char hoten[30]; float dlt,dth;

};

int main()

{

sinhvien sv;

**ofstream fout("d:/sinhvien.dat",ios::binary);**

cout<<"\nNhap ds sinh vien:\n";

while (1)

{

cout<<"\nHo ten:"; cin.getline(sv.hoten,30);

if (sv.hoten[0]==0) break;

cout<<"\nDtl:"; cin>>sv.dlt;

cout<<"\nDth:"; cin>>sv.dth;

cin.get();

**fout.write((char \*)&sv,sizeof(sv));** //ghi sv vao file

}

fout.close();

**ifstream fin ("d:/sinhvien.dat",ios::binary);**

cout<<setw(30)<<setiosflags(ios::left)<<"HO TEN"<<resetiosflags(ios::left);

cout<<setw(10)<<"DLT"<<setw(10)<<"DTH"<<setw(10)<<"DTB"<<endl;

cout<<setfill('-')<<setw(60)<<'-'<<setfill(' ')<<endl;

while (1)

{

**fin.read((char \*)&sv,sizeof(sv));** //doc sv tu file

if (fin.gcount()==0) break; //neu het file thi thoat while

cout<<setw(30)<<setiosflags(ios::left)<<sv.hoten<<resetiosflags(ios::left);

cout<<setw(10)<<setiosflags(ios::showpoint)<<setprecision(2)<<sv.dlt;

cout<<setw(10)<<sv.dth<<setw(10)<<(sv.dlt+sv.dth)/2<<endl;

}

cout<<setfill('-')<<setw(60)<<'-'<<setfill(' ')<<endl;

fin.close(); getch();

}

10 TS. GVC. VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

**f) Di chuyển con trỏ tập tin**

Con trỏ dùng để xác định vị trí đọc/ghi file. Nếu file mở ra để đọc sẽ dùng con trỏ get, file mở ra để ghi sẽ  dùng con trỏ put, file mở ra để đọc và ghi sẽ dùng cả hai con trỏ. Khi đọc/ghi thì con trỏ sẽ tự động di chuyển  sang vị trí đọc/ghi mới, để tự di chuyển con trỏ get, put và xác định vị trí của chúng ta có thể dùng các lệnh  sau:

|  |  |
| --- | --- |
| istream & seekg(int n, int pos) | dời con trỏ đọc (get pointer), n byte tính từ vị trí pos |
| istream & seekp(int n, int pos) | dời con trỏ ghi (put pointer), n byte tính từ vị trí pos |
| long tellg() | trả về vị trí hiện tại của con trỏ get |
| long tellp() | trả về vị trí hiện tại của con trỏ put |

pos là hằng trong lớp **ios** và có các giá trị sau:

|  |  |
| --- | --- |
| beg | Tính từ vị trí đầu file |
| cur | Tính từ vị trí hiện hành |
| end | Tính từ vị trí cuối file |

**Ví dụ:**

#include <fstream.h>

#include <iomanip.h>

#include <conio.h>

struct sinhvien

{ char hoten[30];

 float dlt,dth;

};

void main()

{ sinhvien sv;

ifstream fin ("d:/sinhvien.dat",ios::binary);

fin.seekg(sizeof(sv),ios::beg);

fin.read((char \*)&sv,sizeof(sv)); //doc sinhvien thu 2

cout<<sv.hoten<<setw(10)<<sv.dlt<<setw(10)<<sv.dth;

cout<<setw(10)<<(sv.dlt+sv.dth)/2<<endl;

fin.seekg(0,ios::end); //doi con tro doc toi cuoi

long filesize=fin.tellg(); //vi tri con tro doc chinh la kich thuoc file

cout<<"file size="<<filesize;

fin.close(); getch();

}

**Ví dụ:**

#include <fstream.h>

#include <iomanip.h>

#include <conio.h>

**struct sinhvien**

{ char hoten[30];

 float dlt,dth;

};

void main()

{ sinhvien sv={"tran minh kho chuoi",9.5,8};

**fstream f ("d:/sinhvien.dat",ios::binary|ios::in|ios::out);**

**f.seekp(0,ios::beg); //doi con tro put ve mau tin dau tien**

**f.write((char \*)&sv,sizeof(sv)); //ghi sv vao mau tin dau**

**f.seekg(0,ios::beg); //doi con tro get ve mau tin dau**

**f.read((char \*)&sv,sizeof(sv)); //doc sinhvien thu 1**

cout<<sv.hoten<<setw(10)<<sv.dlt<<setw(10)<<sv.dth;

cout<<setw(10)<<(sv.dlt+sv.dth)/2<<endl;

f.close(); getch();

}

• **Các hàm khác:**

remove(char \* filename): xoá file có tên là filename

rename(char \* filename1, char filename2): đổi tên filename1 thành filename2