TÀI LIỆU LẬP TRÌNH C++ CƠ BẢN

BUỔI 5:

1/ Đọc và ghi dữ liệu với File

2/ Thủ tục và Hàm

I. ĐỌC VÀ GHI DỮ LIỆU TỪ FILE

1. Giới thiệu

Đối với các kiểu cấu trúc dữ liệu ta đã biết như kiểu mảng, kiểu tập hợp, kiểu Record, dữ liệu được tổ chức trên bộ nhớ trong (RAM) của máy tính nên khi kết thúc chương trình thì dữ liệu cũng bị mất. Điều này thật tai hại vì khi cần ta phải nhập lại dữ liệu - một công việc chẳng thú vị gì và rất mất thời gian. Kiểu FILE cho phép lưu trữ dữ liệu ở bộ nhớ ngoài (trên đĩa cứng) khi kết thúc chương trình hay khi tắt máy thì dữ liệu vẫn còn nên có thể sử dụng nhiều lần.

Trong lập trình, tập tin có 3 loại: Tập tin định kiểu, tập tin văn bản và tập tin tổng quát. Ở đây chúng ta chỉ làm việc với tệp tin dạng văn bản.

2. Đọc dữ liệu từ file theo cấu trúc Mở - Xử lý – Đóng

```
int main()
{
        ifstream inp ("đường dẫn");
        inp >> n;
        inp.close();
        return 0;
}
```

fstream: Thư viện bắt buộc phải khai báo khi đọc ghi file.

ifstream : Lệnh mở file để đọc.

inp : Tên biến, khai báo kiểu ifstream

đường dẫn : Đường dẫn đến tệp dữ liệu. Chẳng hạn tệp **mang.inp** nằm trong ổ đĩa **D** thì đường dẫn sẽ là **D:\mang.inp**. Nếu tệp để chung với nơi lưu file cpp thì chỉ cần gõ **mang.inp**.

close() : lệnh đóng file, nếu không có lệnh này sẽ dễ xảy ra lỗi.

Ví dụ: Giả sử tệp **haiso.txt** nằm trong ổ đĩa D, nội dung file chỉ có một dòng duy nhất gồm 2 số 10 và 20. Chương trình đọc hai số vào biến n và m:

```
int main()
{
        int n; int m;
        ifstream inp ("D:\\haiso.txt");
        inp >> n >> m;
        inp.close();
        cout << "Gia tri cua n : " << n << '\n';
        cout << "Gia tri cua m: " << m ;
        return 0;
}
```

3. Ghi dữ liệu ra tệp theo phương thức Mở - Xử lý – Đóng

```
int main()
{
        ...
        ofstream out ("đường dẫn");
        out << n;
        out.close();
        return 0;
}
```

ofstream : Lệnh mở file để ghi.

out : Tên biến, khai báo kiểu ofstream

Ví dụ: Tạo file **haiso.out** trên ổ đĩa D**,** ghi 2 số n và m ra file, mỗi số nằm trên một dòng

```
int main()
{
        ...
        ofstream out("D:\\haiso.out");
        out << n << '\n' <<m;
        out.close();
        return 0;

}
```

C++ hỗ trợ phương thức ghi thêm nội dung vào một tệp đã có sẵn:

```
ofstream out("D:\\haiso.out" , ios::app);
```

4. Cấu trúc đọc ghi file đặc biệt

C++ hỗ trợ một cấu trúc đặc biệt để đọc và ghi dữ liệu, cho phép thay đổi nhanh chóng giữa phương thức nhập xuất truyền thống sang đọc ghi file và ngược lại. Cấu trúc này cũng được sử dụng trong việc chấm bài tự động khi bạn tham gia các kỳ thi. Để sử dụng cấu trúc này, trước các lệnh nhập xuất ta chỉ cần thêm các dòng lệnh như sau:

- Đọc file

```
freopen(" đường dẫn ","r",stdin);
```

- Ghi file

```
freopen(" đường dẫn ","w",stdout);  //ghi mới
```

hoặc

```
freopen(" đường dẫn ","a",stdout);  //ghi thêm vào file
```

Lưu ý trong cú pháp này, ta chỉ được thay đổi đường dẫn, các từ khóa khác là mặc định.

Sau khi có 2 dòng lệnh trên, phần còn lại sẽ sử dụng các lệnh nhập, xuất như đối với nhập xuất từ bàn phím. Nghĩa là để đọc dữ liệu từ file, ta sử dụng lệnh *cin*. Tương tự đối với ghi ra file là *cout*.

Đối với C++, nếu không khai báo dòng **stdout** (dòng dưới), nội dung sẽ được in ra màn hình như bình thường. Hoặc nếu không khai báo dòng **stdin** (dòng đầu), ta sẽ phải nhập dữ liệu từ bàn phím.

Điều này có nghĩa là, ta có thể chuyển đổi qua lại giữa 2 phương thức nhập xuất một cách nhanh chóng, chỉ cần khóa lại 2 dòng lệnh trên, ta sẽ trở lại cách thực hiện truyền thống.

Thêm một điểm khác biệt khi sử dụng cấu trúc dạng này là không cần khai báo thư viện **fstream**.

Ví dụ chương trình đọc giá trị cho biến n dưới đây:

```
int main()
{
	int n;
	freopen("mang.txt","r",stdin);
	freopen("mang.out","w",stdout);
	cin >> n ;
	cout << "Gia tri cua n : " << n ;
	return 0;

}
```

Lúc này, dòng chữ **Gia tri cua n :** sẽ ghi trực tiếp ra tệp **mang.out**

5. Tối ưu đọc ghi file

Cần bổ sung các lệnh sau phía trước lệnh đọc ghi file

```
ios_base::sync_with_stdio(0);
cin.tie(0); cout.tie(0);
```

Hai lệnh này giúp tối ưu đọc ghi dữ liệu, nếu không có các lệnh trên, việc đọc ghi đối với các file dữ liệu lớn sẽ làm quá thời gian chạy chương trình, còn cụ thể như thế nào, bạn có thể tìm hiểu trên Google. Như vậy, lệnh đọc ghi file cần thể hiện như sau:

```
ios_base::sync_with_stdio(0);
cin.tie(0); cout.tie(0);
freopen("mang.txt","r",stdin);
freopen("mang.out","w",stdout);
```

6. Một số hàm và thủ tục hỗ trợ đọc ghi file

Hàm eof(): trả về true khi con trỏ đã trỏ tới cuối file và false thì ngược lại.

Hàm fail() : trả về true nếu đọc học viết lên file bị lỗi hoặc khi ta đọc 1 số nguyên từ file nhưng dữ liệu được đọc là một string; trả về false khi không có lỗi phát sinh khi đọc viết file.

Hàm is_open() : Để kiểm tra file đã được mở hay chưa ta dùng hàm is_open() hàm này trả về true nếu mở thành công và false nếu thất bại.

Thật ra các lệnh trên rất ít khi được sử dụng.

7. Cách đặt file có phần mở rộng .inp hoặc định dạng bất kỳ

Xem cuối tài liệu

8. Một số ví dụ về đọc ghi file

DỮ LIỆU TRONG FILE	LỆNH ĐỌC
10 5	cin >> n >> m;
10 5	cin >> n >> m;
10 5 3 6 8 1 4 9 7 10 2	cin >> n; for (int i=0; i<n; i++) cin >> a[i];
5 3 6 8 1 4 9 7 10 2 19 11 … (không xác định số phần tử)	n:=0; while (cin >> a[n]) n++;
Đếm số ký tự của 1 văn bản hơn 255 ký tự	char a[1000006]; int n=0; while (cin>>a[n]) n++;

II. CHƯƠNG TRÌNH CON

Chương trình con là một đoạn chương trình nằm ngoài chương trình chính, giúp chương trình chính thực hiện hoàn chỉnh một công việc nào đó.

Nói về chương trình con thì có rất nhiều vấn đề từ cơ bản đến phức tạp. Ở nội dung tài liệu này tôi chỉ nói về những vấn đề cơ bản nhất.

Chương trình con gồm 2 dạng: Không trả về giá trị và trả về giá trị sau khi thực hiện. Đối với tôi thì dạng không trả về giá trị tôi gọi là **THỦ TỤC**, dạng trả về giá trị tôi gọi là **HÀM**.

Lợi ích của chương trình con:

- Viết một lần, sử dụng được nhiều lần.

- Có thể đưa sang sử dụng trong một chương trình khác.

- Tránh được các rủi ro khi sử dụng biến toàn cục.

…

Một chương trình con sẽ được thực thi thông qua lời gọi từ chương trình chính. Từ chương trình con cũng có thể gọi đến một chương trình con khác. Về cơ bản, chương trình con được gọi phải nằm phía trên chương trình chính hoặc chương trình con chứa lệnh gọi nó. Có thể mô tả về chương trình con trong chương trình như sau:

```
<Khai báo các thư viện>
<Khai báo biến toàn cục>

Chương trình con 1 ();
        {
                <Các lệnh trong chương trình con 1>;
        }
Chương trình con 2 ();
        {
                <Các lệnh trong chương trình con 2>;
        }

Chương trình chính
        {
                Chương trình con 1;
                Chương trình con 2;
        }
```

Trong Chương trình con 2, ta còn có thể gọi thực hiện Chương trình con 1.

1. Chương trình con không trả về giá trị - Thủ tục

Thủ tục là một đoạn chương trình không trả về giá trị mà chỉ là lời gọi một chiều từ chương trình chính hoặc chương trình con khác, giúp thực hiện một chức năng độc lập nào đó của chương trình.

Cách cài đặt Thủ tục như sau:

```
void <tên thủ tục> ( <các biến tham số> )
{
        <các câu lệnh>;
}
```

Các biến tham số có thể có hoặc không. Nếu có, ta lại xét 2 trường hợp: Một trường hợp các biến được đưa vào sau khi hoàn thành thủ tục sẽ không thay đổi giá trị, ta gọi là truyền biến kiểu tham trị, một trường hợp là biến được đưa vào sé thay đổi giá trị sau khi thực hiện thủ tục, gọi là truyền biến kiểu tham chiếu.

Ví dụ 1: Chương trình con không có biến tham số dưới đây thực hiện in ra giá trị bình phương của số a

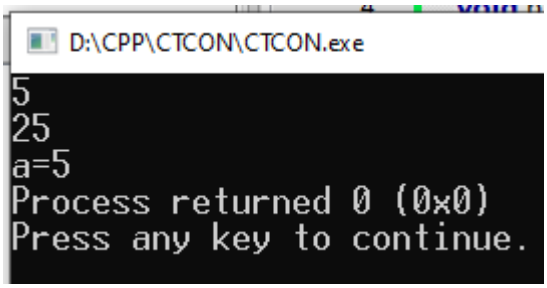
```
int a;

void binhphuong()
{
        a=a*a;
        cout <<  a;
}

int main()
{
        cin >> a;
        binhphuong();
        return 0;
}
```

Ví dụ 2: Chương trình con có biến tham số kiểu truyền tham trị (giá trị của biến tham số không thay đổi sau khi thực hiện)

```
void binhphuong(int num)
{
        num=num*num;
        cout << num << '\n';
}
int main()
{
   int a;
   cin >> a;
   binhphuong(a);
   cout << "a=" << a;
   return 0;
}
```



Giá trị của a được gửi vào biến num, sau khi thực hiện xong giá trị của a vẫn không thay đổi.

Ví dụ 3: Chương trình con có biến tham số kiểu truyền tham chiếu (giá trị của biến tham số sẽ thay đổi sau khi thực hiện)

```
void binhphuong(int &num)
{
        num=num*num;
        cout << num << '\n';
}
int main()
{
   int a;
   cin >> a;
   binhphuong(a);
   cout << "a=" << a;
   return 0;
}
```

2. Hàm

Hàm cũng tương tự như thủ tục, nhưng khác ở chỗ hàm trả về một giá trị cho chương trình chính.

Cách cài đặt Hàm như sau:

<kiểu giá trị> <tên hàm> (<các biến tham số>)

{

<các lệnh xử lý>;

return <giá trị trả về>;

}

Ví dụ 1: Chương trình con không có biến tham số dưới đây trả về giá trị bình phương của số a

```
int a;

int binhphuong()
{
        return a*a;
}

int main()
{
   cin >> a;
   cout << binhphuong();
   return 0;
}
```

Ví dụ 2: Chương trình con có biến tham số kiểu truyền tham trị

```
int a;

int binhphuong(int num)
{
        return num*num;
}

int main()
{
   cin >> a;
   cout << binhphuong(a);
   return 0;
}
```

III. BÀI TẬP

1. Viết chương trình nhập vào từ file hai số nguyên N và M. In ra file tổng và tích của hai số.

2. Viết chương trình nhập từ file BCC.inp một số nguyên N. In ra file BCC.out bảng cửu chương N.

3. Viết chương trình nhập từ file hai số nguyên N và M. In ra file hình vuông kích thước NxM tạo bởi các dấu *.

Vuongnxn.inp	Vuongnxn.out
3 2	* * * * * *

4. Viết chương trình nhập vào từ file một số nguyên N ($0<N<10^9$). In ra file:

- Dòng 1: Tổng từ 1 đến N
- Dòng 2: Các số từ 1 đến N

Dayso.inp	Dayso.out
10	55 1 2 3 4 5 6 7 8 9 10

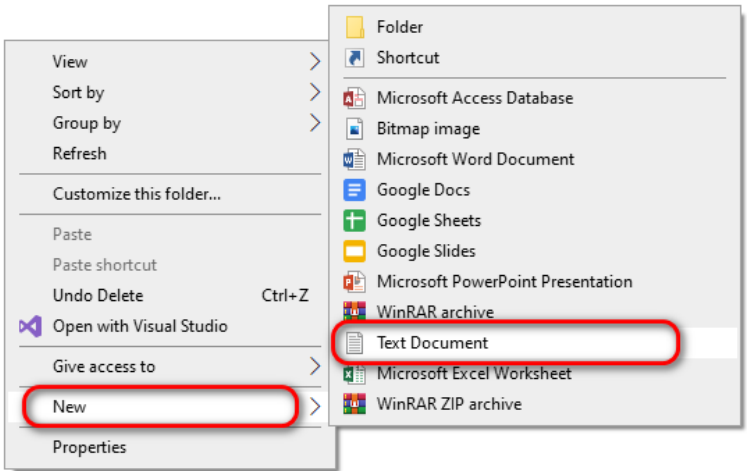
5. ĐỌC SỐ

Nhập từ file một số nguyên N có 3 chữ số. Đọc 3 chữ số đó thành chữ.

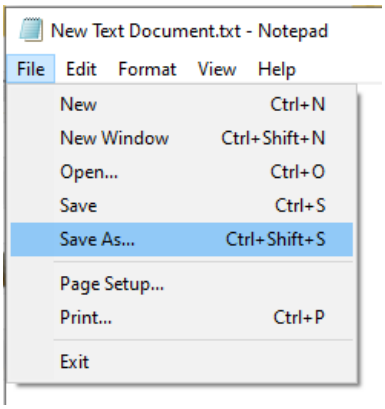
docso.inp	docso.out
452	Bon Nam Hai

CÁCH TẠO FILE .INP THEO YÊU CẦU ĐỀ BÀI

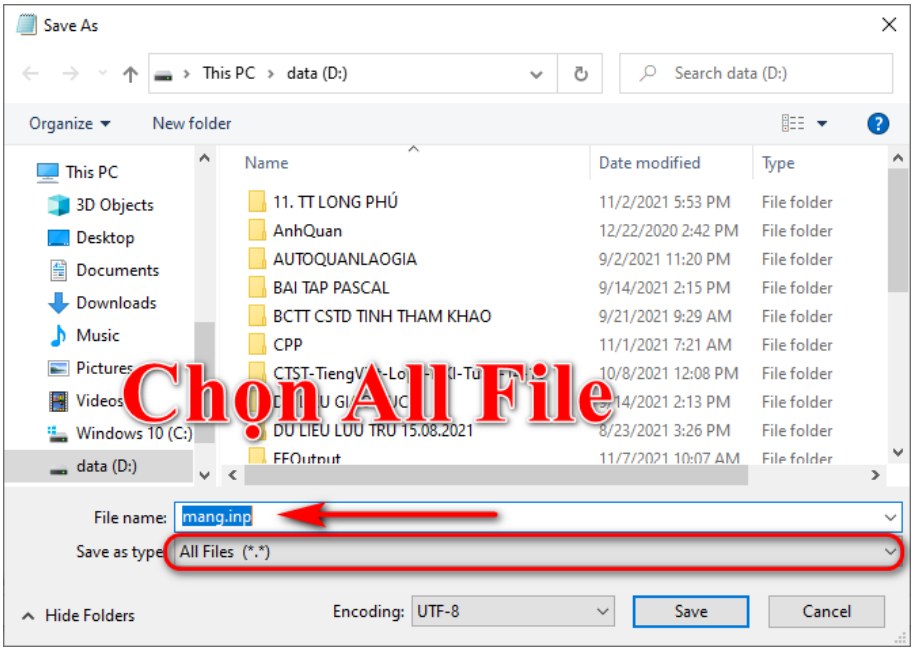
Bước 1: Nhấn chuột phải tại một chỗ trống trong ổ đĩa, chọn New → Text Document



Bước 2: Mở file vừa tạo và chọn Save As

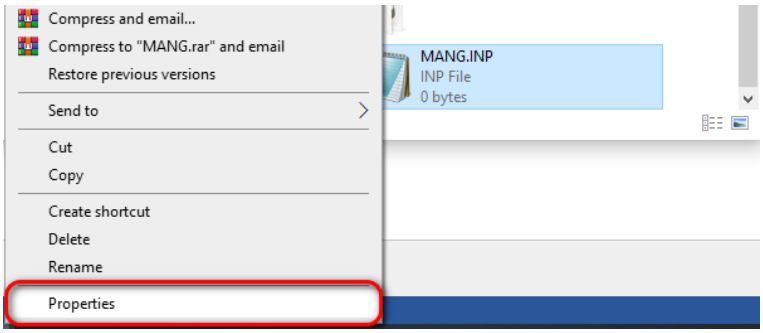


Bước 3: Trong mục Save as type chọn dòng All File, sau đó đặt tên file theo yêu cầu đề bài rồi nhấn Save. Ví dụ gõ tên file là **mang.inp**

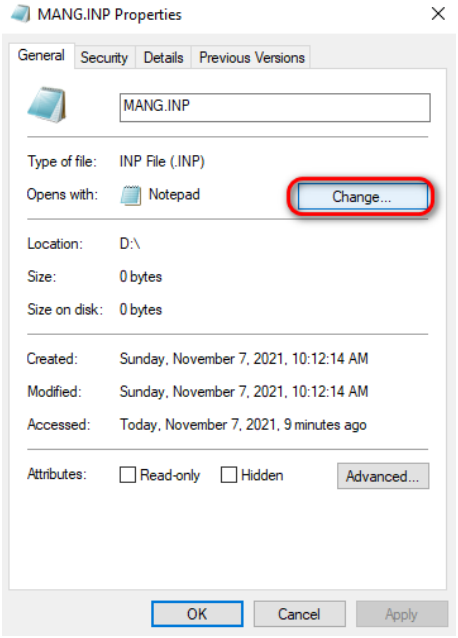


CÁCH ĐỌC FILE .INP, .OUT TRÊN MÁY TÍNH

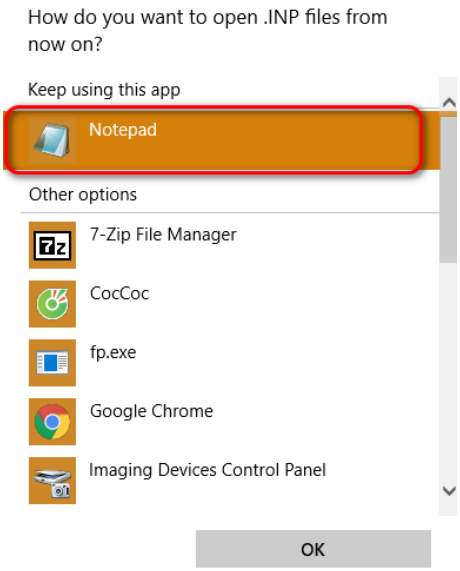
Bước 1: Nhấn chuột phải vào file chọn Properties



Bước 2: Nhấn vào nút Change…



Bước 3: Chọn vào **Notepad** và nhấn OK, sau đó tiếp tục OK trong cửa sổ như ở bước 2



```
#pragma warning( disable : 4996 )
```