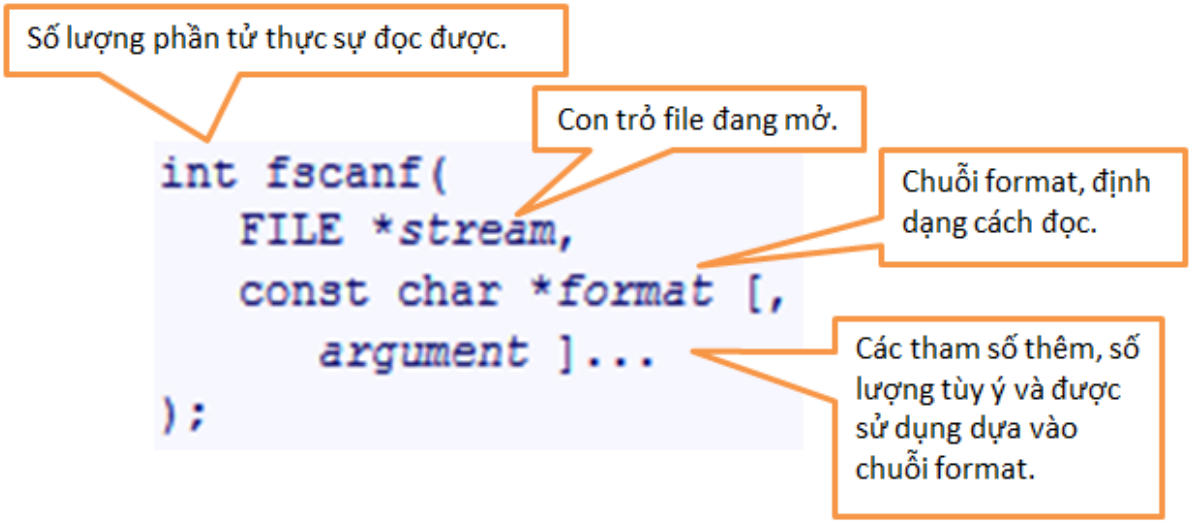
Mục tiêu

Làm bài tập về đọc ghi file text và con trỏ hàm.

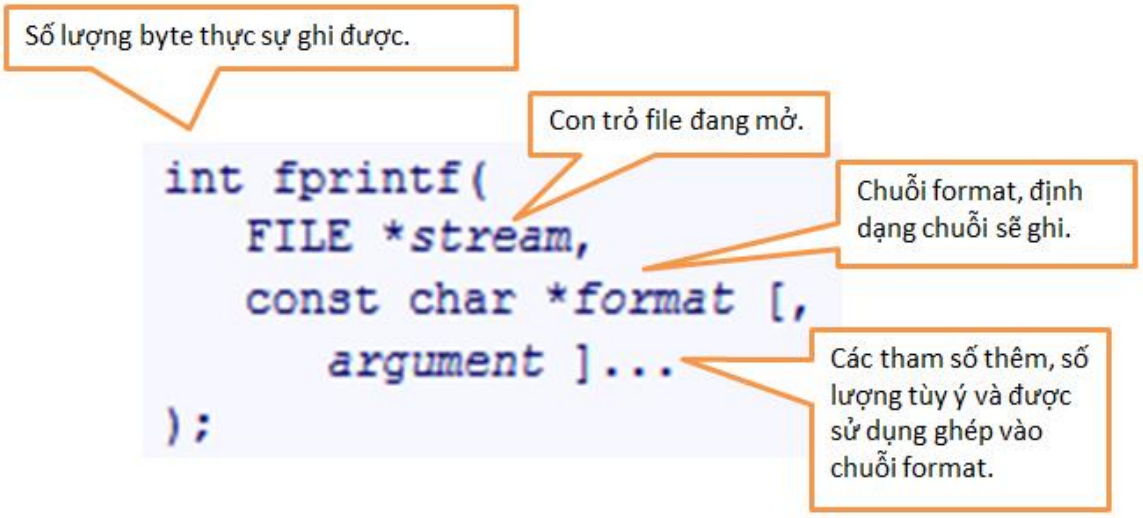
Nội dung

Thao tác đọc ghi file kiểu văn bản trong C

- ✓ Vấn đề thao tác file theo kiểu văn bản là khá dễ tiếp cận vì nó hầu như giống hệt thao tác với màn hình và thiết bị nhập chuẩn (bàn phím).
- ✓ Đọc



- ✓ Ghi



- ✓ Trong chuỗi format thường sử dụng chuỗi đặc tả chuyển dạng để đổi các kiểu giá trị khác ra chuỗi và ngược lại ("%d" chuyển số int sang chuỗi và ngược lại). Cấu trúc tổng quát của chuỗi đặc tả: %[-][fw][.pp]ký tự chuyển dạng
  - [-]: định dạng việc chuỗi kết quả sẽ dồn về bân phải hay trái trong chiều rộng cho phép (định dạng bởi giá trị fw).
  - [fw]: định dạng bề rộng của chuỗi kết quả.
  - [.pp]: định dạng chỉ áp dụng cho giá trị số thực, quy định số chữ số chính xác phần thập phân.
- ✓ Ngoài ra còn có các hàm làm việc theo từng ký tự: fputc, fgetc.
- ✓ Ví dụ đơn giản

```
void main()

{

        FILE* f = NULL;

        if((f = fopen("test.txt", "w+"))!=NULL)

        {

                int n=2345;

                //ghi vào file tương tự xuất ra màn hình

                fprintf(f, "%d", n);

                //trở về đầu file

                fseek(f, 0, SEEK_SET);

                int m;

                //đọc giá trị vừa ghi

                fscanf(f, "%d", &m);

                //đóng file

                fclose(f);

                printf("%d", m);

                getch();

        }

}
```

Thao tác đọc ghi file kiểu văn bản trong C++

```
int main()
{
        const char *pathFile = "test.txt";
        ifstream fIn;
        ofstream fOut;
        int n = 1234;
```

```
        /// mở file để ghi với cờ tạo mới hoàn toàn
        fOut.open(pathFile, ios::trunc);
        fOut << n;
        /// đóng file
        fOut.close();
        /// mở file để ghi với cờ về vị trí đầu
        fIn.open(pathFile, ios::beg);
        int r = -1;
        fIn >> r;
        cout << "read: " << r << endl;
        fIn.close();
        system("pause");
}
```

Con trỏ hàm (function pointer)

Trong C++, con trỏ hàm (function pointer) được sử dụng để lưu trữ địa chỉ của một hàm. Xem ví dụ sau:

```
#include<iostream>
#include<fstream>
#include<cstring>
#include<cstdlib>
#include<cstdio>

using namespace std;

#define MAXLEN 100

struct SinhVien
{
        int MSSV;
        char HoTen[51];
};

char* SVToString(const SinhVien &sv)
{
        /// xuất thông tin sinh viên ra chuỗi
        /// định dạng: <MSSV> - <HoTen>
}

void PrintArr(SinhVien *arr, const int &n)
{
        for (int i = 0; i < n; i++)
        {
                cout << SVToString(arr[i]) << endl;
        }
}

/// định nghĩa con trỏ hàm
typedef int(*SVComparer)(SinhVien, SinhVien);

/// hàm so sánh sinh viên dựa vào mssv
int SVC1(SinhVien sv1, SinhVien sv2)
{
        if (sv1.MSSV > sv2.MSSV)
        {
```

```
                return 1;
        }
        if (sv1.MSSV < sv2.MSSV)
        {
                return -1;
        }
        return 0;
}

/// hàm so sánh sinh viên dựa vào họ tên
int SVC2(SinhVien sv1, SinhVien sv2)
{
        return strcmp(sv1.HoTen, sv2.HoTen);
}

/// hàm sắp xếp sinh viên
void SortAsc(SinhVien *arrSV, const int &n, SVComparer svc)
{
        for (int i = 0; i < n - 1; i++)
        {
                for (int j = i + 1; j < n; j++)
                {
                        if (svc(arrSV[i], arrSV[j]) > 0)
                        {
                                SinhVien svTemp = arrSV[i];
                                arrSV[i] = arrSV[j];
                                arrSV[j] = svTemp;
                        }
                }
        }
}

/// có thể sử dụng con trỏ hàm không cần định nghĩa trước
void SortDesc(SinhVien *arrSV, const int &n, int (*svSoSanh)(SinhVien, SinhVien))
{
        /// sắp xếp giảm dần
        for (int i = 0; i < n - 1; i++)
        {
                for (int j = i + 1; j < n; j++)
                {
                        if (svSoSanh(arrSV[i], arrSV[j]) < 0)
                        {
                                SinhVien svTemp = arrSV[i];
                                arrSV[i] = arrSV[j];
                                arrSV[j] = svTemp;
                        }
                }
        }
}

SinhVien* LoadSV(const char *path, int &n)
{
        /// đọc danh sách sinh viên từ file
        /// trả về pointer trỏ đến mảng SinhVien đọc được
        /// n là số sinh viên đọc được
}
```

```
int main()
{
	const char *path = "data.txt";
	int n = 0;
	SinhVien *arrSV1 = LoadSV(path, n);
	SinhVien *arrSV2 = LoadSV(path, n);
	cout << "***************Arr begin**************" << endl;
	PrintArr(arrSV1, n);
	SortAsc(arrSV1, n, SVC1);
	cout << "**********Arr sorted mode 1**********" << endl;
	PrintArr(arrSV1, n);
	SortDesc(arrSV2, n, SVC2);
	cout << "**********Arr sorted mode 2**********" << endl;
	PrintArr(arrSV2, n);
	system("pause");
}
```

Bài tập

Bài 1

Làm hoàn thiện bài ví dụ con trỏ hàm với file data.txt là:

```
23120001;Phan Van A
23120017;Nguyen Van C
23120051;Tran Thi B
23120005;Le Thi M
```

Bài 2

Viết chương trình thao tác với danh sách các tài khoản người dùng

Thông tin tài khoản:

1. Tên đăng nhập
    a. Tên đăng nhập ko được chứa chữ “admin”
2. Mật khẩu (khi nhập thì cần nhập mật khẩu 2 lần)
    a. Từ 6 kí tự trở lên
    b. Có kí tự in hoa
    c. Có kí tự a→z, A→Z
    d. Có số
    e. Có 1 trong các dấu phẩy, chấm, hỏi, hai chấm, chấm phẩy, ngã, gạch ngang, gạch dưới
3. Họ tên
    a. Khi nhập, nhập chung 1 chuỗi họ tên, vì là họ tên nên ít nhất phải có 2 từ. Khi xuất, xuất riêng 3 phần: họ, tên lót và tên
4. Email
    a. Phải chứa dấu @ và dấu .
5. Ngày tháng năm sinh

a. Khi nhập thì nhập ngày tháng năm sau, khi xuất thì xuất tuổi

Các chức năng:

1. Nhập vào danh sách tài khoản.
2. Xuất danh sách tài khoản ra màn hình.
3. Nhập vào 1 chuỗi, xuất tất cả các tài khoản mà trong họ tên có chứa chuỗi đó. VD: "phan" thì xuất tài khoản "Phan Van N".
4. Nhập vào 1 tài khoản mới
   a. Nếu tên đăng nhập đã có, thì xuất thông tin tài khoản đã có ra màn hình
   b. Nếu tên đăng nhập chưa có, thì thêm tài khoản vào cuối danh sách
5. Nhập vào 1 tên đăng nhập. Xóa tài khoản có tên đăng nhập đó ra khỏi danh sách
6. Sử dụng con trỏ hàm để sắp xếp danh sách tài khoản cho phép lựa chọn chế độ:
   a. Tăng dần hay giảm dần
   b. Theo tên đăng nhập
   c. Theo tuổi
   d. Theo họ tên
7. Chúc mừng sinh nhật các tài khoản có ngày sinh trong tháng này (tháng 4)

❖ Hướng dẫn:

Hàm xuất thông tin 1 tài khoản

```
// Tên hàm: xuất tài khoản
// Tham số: tài khoản cần xuất thông tin
// Trả về: ko trả về gì hết
void Xuat(TaiKhoan tk);
```

```
// Tên hàm: xuất ngày tháng năm
// Tham số: ngày tháng năm cần xuất
// Trả về: ko có
void Xuat(Ngay ng);
```

```
void Xuat(TaiKhoan tk)
{
    // printf("Ten dang nhap: %s\n", tk.tenDangNhap);
    cout<<"Ten dang nhap: "<< tk.tenDangNhap<<"\n";
    // printf("Mat khau: %s\n", tk.matKhau);
    cout<<"Mat khau: "<<tk.matKhau << "\n";
    // printf("Ho ten: %s\n", tk.hoTen);
    cout<<"Ho ten: "<<tk.hoTen << "\n";
    // SV tự làm phần tách họ, tên lót và tên ra 3 phần riêng
    // printf("Email: %s\n", tk.email);
    cout<<"Email: "<<tk.email << "\n";
    // printf("NTNS: ");
    cout<<"NTNS: ";
    Xuat(tk.ntns);
    // printf("\nTuoi: %d\n", 2015 - tk.ntns.nam);
    cout<<"\nTuoi: "<<2015 - tk.ntns.nam << "\n";
}
```

```
void Xuat(Ngay ng)
{
    // printf("%d/%d/%d", ng.ngay, ng.thang, ng.nam);
    cout << ng.ngay << "/" << ng.thang << "/" << ng.nam;
}
```

Bây giờ, chúng ta sẽ viết các hàm trong 2 file MangTaiKhoan.h và MangTaiKhoan.cpp

```h
#ifndef _MANGTK_H_
#define _MANGTK_H_

#include "TaiKhoan.h"

// Tên hàm: Nhập mảng tài khoản
// Tham số: mảng, số phần tử
// Trả về: ko có
void Nhap(TaiKhoan *&a, int &n);

// Tên hàm: xuất mảng tài khoản
// Tham số: mảng, số phần tử
// Trả về: ko có
void Xuat(TaiKhoan *a, int n);
```

```h
// Tên hàm: liệt kê tài khoản theo tên
// Tham số: mảng, số phần tử, chuỗi tìm kiếm
// Trả về: ko có
void LietKeTheoTen(TaiKhoan *a, int n, char ten[]);

// Tên hàm: thêm tài khoản
// Tham số: mảng, số phần tử, tài khoản mới cần thêm
// Trả về: nếu tài khoản đã có: trả về vị trí;
// nếu tài khoản chưa có: trả về -1
int ThemTaiKhoan(TaiKhoan *a, int &n, TaiKhoan tk);

// Tên hàm: xóa tài khoản
// Tham số: mảng, số phần tử, tên đăng nhập
// Trả về: ko có
void XoaTaiKhoan(TaiKhoan *a, int &n, char tenDangNhap[]);
#endif
```

```
void Nhap(TaiKhoan *&a, int &n)
{
    // B1: Nhập số lượng tài khoản
    cout << "Nhap n: ";
    cin >> n;
    // B2: Cấp vùng nhớ cho mảng a
    a = new TaiKhoan[n];
    // B3: Kiểm tra cấp vùng nhớ có thành công ko
    if (a == NULL)
        return;
    // B4: Duyệt từ 0 đến n-1, gọi hàm nhập 1 tài khoản
    for (int i = 0; i < n; i++)
    {
        cout << "Nhap tai khoan thu " << i << "\n";
        Nhap(a[i]);
    }
}
```

```
void Xuat(TaiKhoan *a, int n)
{
    // B1: Kiểm tra n=0, báo ko có tài khoản nào
    if (n == 0)
    {
        cout << "Ko co tai khoan nao trong danh sach\n";
        return;
    }
    // B2: Duyệt từ đầu đến cuối mảng, gọi hàm xuất tài khoản
    for (int i = 0; i < n; i++)
    {
        cout << "Tai khoan thu " << i << "\n";
        Xuat(a[i]);
        cout << "\n";
    }
}
```

```
void LietKeTheoTen(TaiKhoan *a, int n, char ten[])
{
    // B1: Kiểm tra n=0, báo ko có tài khoản nào
    // B2: Duyệt từ đầu đến cuối mảng,
    for (int i = 0; i < n; i++)
    {
        char s[51];
        // Copy sang s để tránh thay đổi họ tên của a[i]
        strcpy(a[i].hoTen, s);
        // Chuyển s thành chữ thường
        // Kiểm tra ten có nằm trong s
        char *p = strstr(s, ten);
        if (p != NULL)
        {
            // Tìm thấy, gọi hàm xuất tài khoản a[i]
        }
    }
}
```