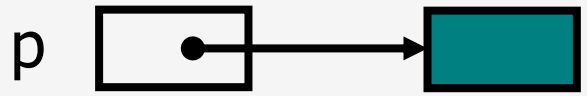
CON TRỎ (POINTER) VÀ FILES

CON TRỎ (POINTER)

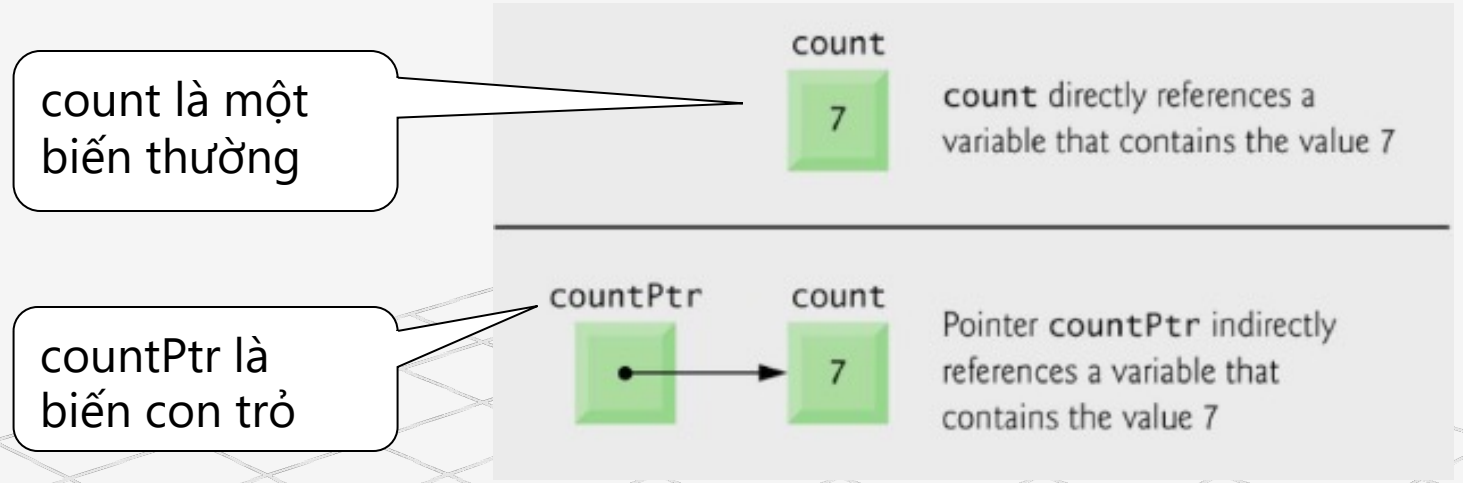
- Con trỏ là gì? Bản chất và ví dụ
- Các phép toán với con trỏ
- Mối quan hệ giữa mảng và con trỏ
- Cấp phát và thu hồi vùng nhớ của biến con trỏ

BIẾN CON TRỎ

- Biến con trỏ là biến mà giá trị của nó là địa chỉ của một ô nhớ, còn giá trị thực của biến được chứa trong địa chỉ mà nó trỏ tới.



- Biến thường: chứa một giá trị cụ thể
- Biến con trỏ: chứa địa chỉ của một biến mà biến đó có giá trị cụ thể



KHAI BÁO BIẾN CON TRỎ

- Thêm kí tự "*" vào trước tên biến

  Ví dụ: int *a; => a là con trỏ kiểu int

- Khai báo nhiều biến con trỏ cùng kiểu: trước mỗi biến đều thêm một dấu "*".

  Ví dụ: double *a, *b; => a, b là hai con trỏ kiểu double

- Có thể khai báo kiểu con trỏ cho mọi loại dữ liệu
- Khởi tạo biến con trỏ: gán về 0 hoặc NULL
  - Nghĩa là con trỏ không trỏ vào đâu cả
  - Gán về 0 chỉ với những con trỏ số nguyên
  - Thường thì gán về NULL được dùng nhiều hơn

TOÁN TỬ "&" VÀ TOÁN TỬ "*"

- Lấy giá trị của con trỏ: Sử dụng toán tử "*"
- Lấy địa chỉ của một biến: Sử dụng toán tử "&"
- Ví dụ:

```
int n = 5;
int  *p;
p = &n;      // cả p và n đều trỏ vào cùng 1 ô nhớ
cout << *p;  // đưa ra giá trị của con trỏ p
cout << p;   // đưa ra địa chỉ của con trỏ p
```

  *p tương đương với n

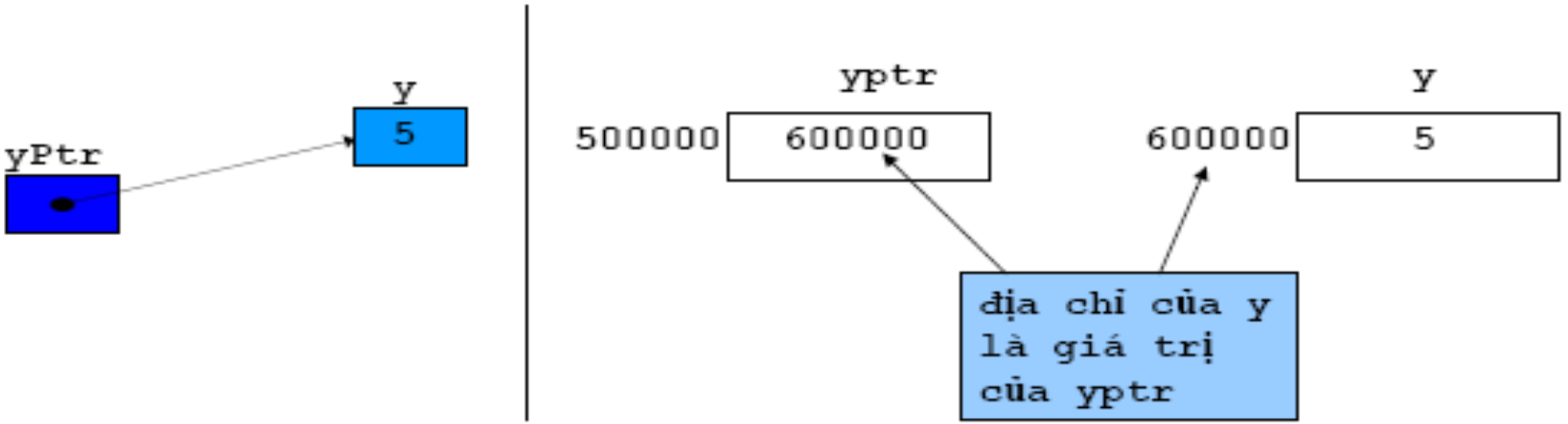
  p tương đương với &n

- Như vậy: Toán tử * và & có vai trò đối ngược nhau, * lấy giá trị của biến con trỏ, còn & lấy địa chỉ của biến thường.

TOÁN TỬ "&" VÀ TOÁN TỬ "*"

```
int y = 5;
int *yPtr;
yPtr = &y;    // yPtr chứa địa chỉ của y
```

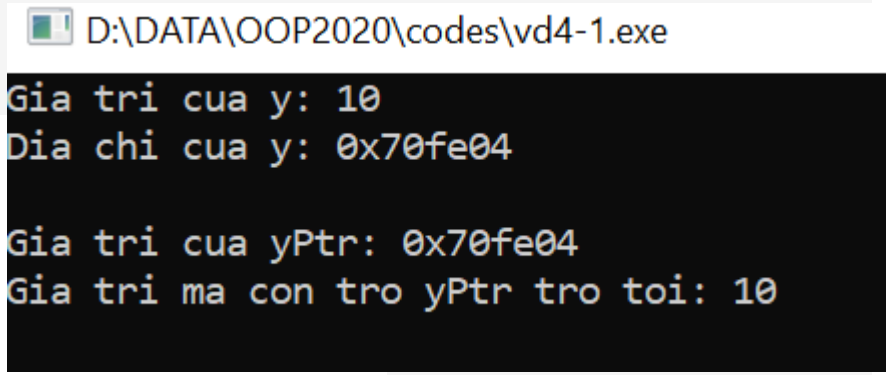
- **yPtr** "trỏ đến" **y**



VÍ DỤ 4-1

- Hiển thị và phân biệt địa chỉ và giá trị của biến con trỏ
- Sự khác nhau của toán tử **&** và ***

```
1  #include <iostream>
2  using namespace std;
3  main() {
4      int y, *yPtr;  // *ptr la bien con tro
5      y = 10;
6      yPtr = &y; // con tro yPtr tro vao dia chi cua bien y
7      cout << "Gia tri cua y: " << y << endl;
8      cout << "Dia chi cua y: " << &y << endl;
9      cout << endl;
10     cout << "Gia tri cua yPtr: " << yPtr << endl;
11     cout << "Gia tri ma con tro yPtr tro toi: " << *yPtr;
12     cout << endl;
13     return 0;
14 }
```



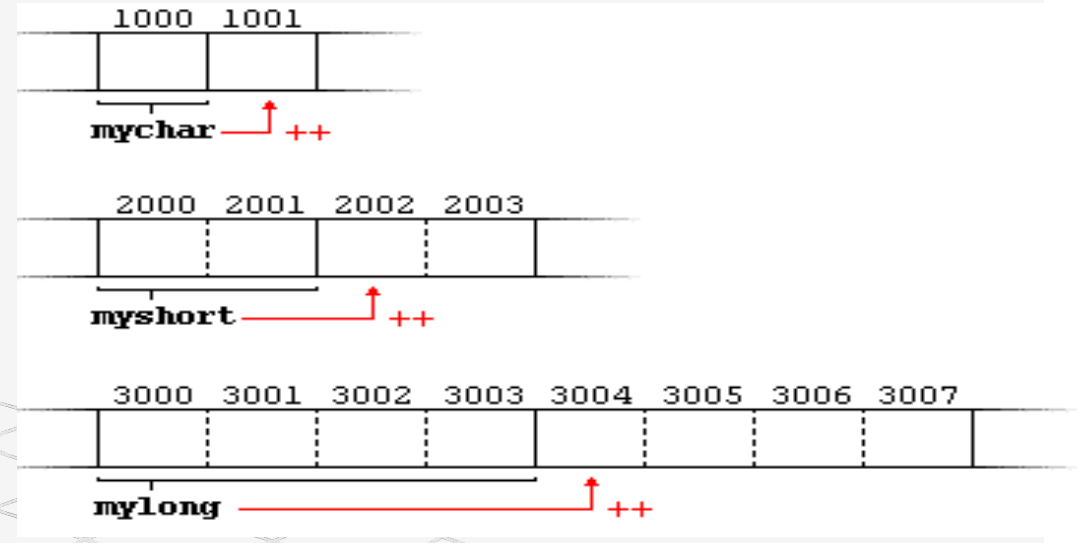
Kết quả chạy chương trình. Chú ý để thấy những giá trị giống và khác nhau.

CÁC PHÉP TOÁN TRÊN CON TRỎ

- Các biến trỏ có các phép toán: cộng và trừ con trỏ (không có phép toán nhân chia)
- Tuy nhiên, phép cộng/trừ khác với cộng/trừ các số bình thường, tùy thuộc vào kiểu dữ liệu của biến con trỏ, mà mỗi lần cộng thêm 1 đơn vị thì con trỏ dịch chuyển 1byte, 2 bytes, 4 bytes
- Ví dụ

```
1 char *mychar;
2 short *myshort;
3 long *mylong;
```

```
1 mychar++;
2 myshort++;
3 mylong++;
```



- Cần phân biệt các phép toán trên con trỏ với các phép toán trên giá trị mà con trỏ trỏ tới.
- Ví dụ: p là 1 con trỏ int, thì *p++ khác với (*p)++

MỐI QUAN HỆ GIỮA CON TRỎ VÀ MẢNG

- Mảng và con trỏ có quan hệ chặt chẽ.
- Tên mảng chính là hằng con trỏ trỏ vào phần tử đầu tiên của mảng.
- Ví dụ: int A[100];
- Thì: A chính là &A[0] và *A chính là A[0]

  (A + i) chính là &A[i] và *(A+i) chính là A[i]
- Có thể sử dụng con trỏ để truy cập vào các phần tử của mảng, ví dụ:

```
for (i = 0; i < n; i++) {
   cout << *(A+i) <<endl;
}
```

Sử dụng con trỏ để truy cập vào phần thứ *i* của mảng A

VÍ DỤ 4-2

- Sử dụng con trỏ để nhập mảng 1 chiều, tính tổng các phần tử của mảng đã nhập và đưa ra màn hình kết quả

```
#include <iostream>
#include <conio.h>
using namespace std;
main() {
    int *p, n, A[100];
    cout << "Cho n: ";
    cin >> n;
    p = A; // con tro p tro vao mang A
    for (int i = 0; i < n; i++) {
        cout << "A[" << i << "]= ";
        cin >> *(p+i);
    }
    //tinh tong cac phan tu cua mang A
    int S = 0;
    for (int i = 0; i < n; i++) {
        S += *(p+i);
    }
    cout << "Tong cac phan tu: " << S;
    getch();
    return 0;
}
```

TRUYỀN THAM BIẾN CHO HÀM SỬ DỤNG CON TRỎ

- Có thể sử dụng con trỏ để truyền tham biến cho hàm: đây là cách truyền theo kiểu tham biến biến
- Cú pháp: khi khai báo cần thêm dấu * vào trước tham biến; khi gọi hàm cần thêm dấu **&** vào trước tên biến truyền vào.
- Ví dụ:

```
void nhapdulieu(int *n, int *m);
double tong(int *a, int b);
```

Gọi hàm:

```
nhapdulieu(&a, &b);
z = tong(&a, b);
```

VÍ DỤ 4-3

```
1  #include <iostream>
2  using namespace std;
3  int cong2so(int *a, int b) {
4      *a = *a + 2; // tang a them 2
5      b = b + 4; // tang b them 4
6      return *a + b;
7  }
8
9  main() {
10     int a, b, c;
11     a = 15; b = 20;
12     c = cong2so(&a, b); // goi ham cong2so
13     cout << "a la: " << a << endl;
14     cout << "b la: " << b << endl;
15     cout << "Tong cua a+b la: " << c;
16     return 0;
17 }
```

```
D:\DATA\OOP2020\codes\vidu4-3.exe
a =  17
b = 20
Tong cua a+b la: 41
```

Giải thích kết quả?

Con trỏ a là tham biến biến; b là tham biến trị

Khi gọi hàm, đối với tham biến sử dụng con trỏ, cần thêm & vào trước tham biến

CON TRỎ HÀM

- Có thể truyền một hàm như là một tham số đến một hàm khác, nhằm mục đích linh động và mềm dẻo trong việc gọi hàm; thao tác này sử dụng con trỏ trỏ tới hàm
- Để có thể khai báo một con trỏ trỏ tới một hàm chúng ta phải khai báo nó như là khai báo khuôn mẫu (prototype) của một hàm nhưng phải để tên hàm trong ngoặc đơn () và thêm dấu sao (*) vào phía trước tên hàm.
- Ví dụ: int pheptoan(int x, int y, (*ham_phep_toan)(int, int));

x và y là các tham số

Tham số thứ 3 là một con trỏ hàm, trỏ đến một hàm khác

VÍ DỤ 4-4

```
1  #include <iostream>
2  using namespace std;
3  int phepcong(int x, int y) {
4      return x + y;
5  }
6  int pheptru(int x, int y) {
7      return x - y;
8  }
9  // khai bao ham co su dung tham so la con tro ham
10 int pheptoan(int x, int y, int (*phep_toan_can_goi)(int, int)) {
11     int z;
12     z = (*phep_toan_can_goi)(x, y);
13     return z;
14 }
```

Tham số này là con trỏ của một hàm có 2 tham số int

VÍ DỤ 4-4

```
15  // ham main
16  int main ()
17  {
18  int m,n;
19  // khai bao mot con tro ham, tro den ham pheptru
20  int (*pheptru_Ptr)(int, int) = pheptru;
21  // goi ham pheptoan voi tham so thu 3 la con tro cua 1 ham khac
22  m = pheptoan(15, 20, &phepcong);
23  n = pheptoan(40, 18, pheptru_Ptr);
24  cout << "m = " << m << endl;
25  cout << "n = " << n << endl;
26  return 0;
27  }
```

Khai báo một con trỏ hàm và trỏ đến hàm pheptru

Gọi hàm với tham số là địa chỉ của 1 hàm khác, hoặc là một con trỏ trỏ đến một hàm khác

TOÁN TỬ NEW VÀ DELETE

- Toán tử new
  - Ý nghĩa: Cấp phát vùng nhớ cho biến con trỏ; có thể sử dụng trong chương trình để tối ưu việc sử dụng bộ nhớ; sử dụng con trỏ thay cho mảng.
  - Cú pháp: bien_tro = new kieu_du_lieu [n];
  - Trong đó N là số lượng phần tử được cấp phát.
  - Ví dụ: int *p, *q;

    q = new int; // cấp phát vùng nhớ cho biến q

    p = new int [10]; // cấp phát vùng nhớ gồm 10 phần tử cho p
- Toán tử delete
  - Ý nghĩa: thu hồi vùng nhớ đã cấp phát cho biến con trỏ.
  - Cú pháp: delete bien_tro;
  - Ví dụ: delete p; // giải phóng vùng nhớ đã cấp phát cho p

VÍ DỤ 4-5

```
1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4  main() {
5      int *p, n;
6      cout << "Cho biet n: ";
7      cin >> n;
8      // cap phat vung nho cho p
9      // p tuong duong voi mot mang gom n phan tu
10     p = new int [n];
11     for (int i = 0; i < n; i++) {
12         cout << "A[" << i << "]= ";
13         cin >> *(p+i);
14     }
```

Làm lại Ví dụ 4-2, sử dụng toán tử cấp phát vùng nhớ cho biến con trỏ

Cấp phát vùng nhớ cho biến con trỏ p

VÍ DỤ 4-5

```
15      // vi du tinh tong cac phan tu cua p
16      int S = 0;
17      for (int i = 0; i < n; i++) {
18          S += *(p+i);
19      }
20      cout << "Tong cac phan tu: " << S << endl;
21      // thu hoi vung nho cua p
22      // sau khi su dung p nen thu hoi vung nho
23      delete p;
24      getch();
25      return 0;
26  }
```

Thu hồi vùng nhớ của con trỏ p để dọn dẹp bộ nhớ

BÀI TẬP

Bài 4-1: Lập trình nhập vào một mảng gồm N số nguyên, tính tổng các phần tử chia hết cho 2 và 3. Đưa tổng tìm được lên màn hình.

Bài 4-2: Lập chương trình gồm các hàm

- Nhập vào một dãy số nguyên
- Đưa ra dãy số nguyên đã nhập
- Tính số lượng phần tử là số nguyên tố trong dãy đã nhập

Bài 4-3: Cho một dãy gồm N số nguyên, hãy tìm một đoạn con (gồm các phần tử liên tiếp) không giảm dài nhất của dãy đã cho.

Chú ý: Phải sử dụng con trỏ để thay cho việc sử dụng mảng một chiều.

FILES

- File là gì? Vì sao cần sử dụng file
- Cách khai báo và mở file
- Cách đọc/ghi dữ liệu vào file

FILE LÀ GÌ?

- File: là một tập hợp dữ liệu liên quan được máy tính ghi lên bộ nhớ thứ cấp (HDD, FDD,…)
- Khi máy tính đọc file: copy dữ liệu từ bộ nhớ thứ cấp vào bộ nhớ trong của máy.
- Khi máy tính ghi file: copy dữ liệu từ bộ nhớ trong ra bộ nhớ thứ cấp
- File được dùng để lưu trữ dữ liệu
- Buffers: là các vùng đặc biệt làm trung gian để đọc/ghi dữ liệu giữa máy tính và bộ nhớ thứ cấp (chẳng hạn: khi ghi dữ liêu, chương trình sẽ ghi vào buffers trước, rồi sau đó mới ghi vào HDD)

VÌ SAO CẦN SỬ DỤNG FILE

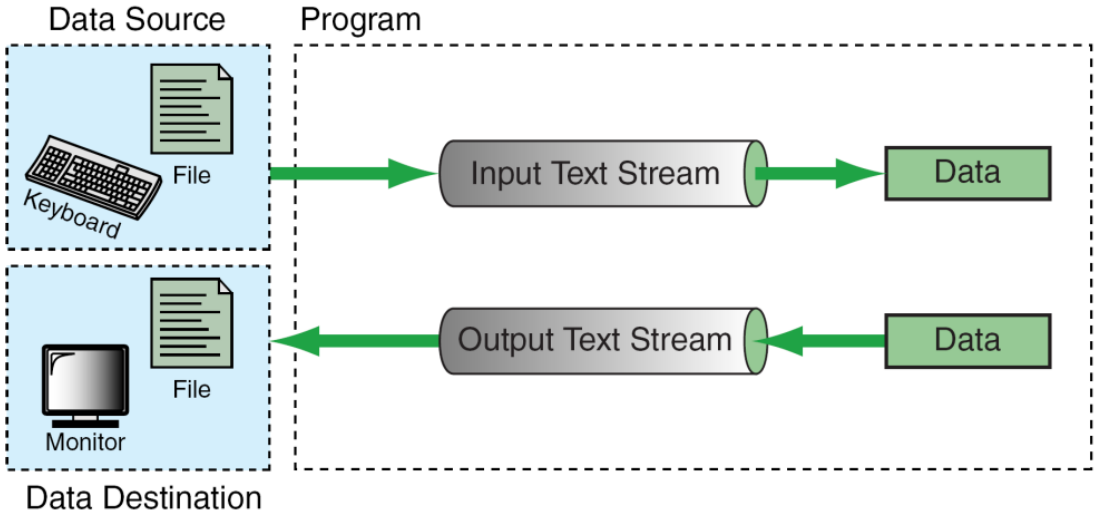
- File chứa thông tin dữ liệu lâu dài trên các thiết bị lưu trữ, dễ dàng sử dụng nhiều lần
- Thuận tiện để lưu trữ và xử lý dữ liệu lớn
- Tránh nhập dữ liệu vào chương trình nhiều lần
- Dễ dàng chia sẻ dữ liệu với các chương trình khác

=> Cần thiết phải sử dụng file để nhập/xuất dữ liệu trong khi viết các chương trình máy tính.

FILE TRONG C++

- Sử dụng thư viện fstream.h
- Là thư viện vào/ra dữ liệu với file
- Đã định nghĩa sẵn một số thao tác với file
- Stream (luồng) đóng vai trò giao tiếp giữa chương trình và file
- Mỗi luồng được định nghĩa sẵn một số thao tác với luồng dữ liệu

Minh họa luồng giao tiếp giữa file và dữ liệu



CÁC LỚP FILE TRONG C++

- C++ sử dụng 3 lớp dưới đây để đọc/ghi file
  - ofstream – ghi vào file
  - ifstream – đọc từ file
  - fstream – cả đọc và ghi file
- Khác với vào/ra chuẩn?
  - Khai báo <iostream>, thì đối tượng ra ostream là cout, chỉ đến stdout (màn hình) được tự động xác định
  - ofstream kế thừa ostream và sử dụng lại thao tác >> để ghi ra file thay vì màn hình

KHAI BÁO VÀ SỬ DỤNG FILES

- Cần khai báo các luồng để sử dụng trong chương trình

  ifstream finp; // khai báo biến finp là một luồng vào

  ofstream fout; // khai báo biến fout là một luồng ra

- Mở file để thực hiện việc đọc/ghi dữ liệu, tương ứng với loại file đang mở

  ten_bien_file.open(<tên file cần mở>);

  Ví dụ:

  finp.open("data.inp");

  fout.open("danhsach.out")';

  f1.open("number.txt").

- Kiểm tra việc mở file có thành công không, sử dụng hàm fail().

  Ví dụ: if (finp.fail()) cout << "Khong the mo duoc file !";

- Đóng file: ten_bien_file.close();

ĐỌC DỮ LIỆU TỪ FILE

- Sau khi mở file, có thể tiến hành đọc dữ liệu từ file với các hàm như sau:

Hàm	Mô tả
finp.fail()	Trả về true nếu mở file không thành công
finp.open(ten_file)	Mở file được truyền vào bởi ten_file
finp.close()	Đóng file
finp.get() finp.get(var)	Đọc và trả về một kí tự Đọc vào trả về một kí tự vào biến var – Trả EOF (kết thúc file) khi đến cuối file
getline(finp, str)	Đọc một dòng dữ liệu vào biến str. Trả về true nếu thành công, false nếu thất bại
finp >> var	Đọc dữ liệu từ luồng vao vào biến var, sẽ tự động đọc dữ liệu theo kiểu dữ liệu của biến var
finp.eof()	Trả về true nếu đã đọc đến cuối file

VÍ DỤ 4-6

Cho file dữ liệu gồm danh sách họ tên trong file danhsach.txt, đọc dữ liệu từ file và hiển thị lên màn hình

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  main() {
5      ifstream finp;
6      finp.open("danhsach.txt");
7      if (finp.fail()) {
8          cout << "Khong mo file thanh cong !";
9          return 0;
10     }
11     string s;
12     cout << "Du lieu doc duoc tu file la: " << endl;
13     while (!finp.eof()) {
14         getline(finp, s); // doc mot dong trong file
15         cout << s << endl; // hien thi ket qua len man hinh
16     }
17     return 0;
18 }
```

Đọc một dòng dong trong file finp, kết quả đưa vào xâu s

VÍ DỤ 4-7

Đọc một dãy số từ file dayso.txt và tính tổng của dãy số này.

```
3  using namespace std;
4  main() {
5      ifstream finp;
6      finp.open("dayso.txt");
7      if (finp.fail()) {
8          cout << "Khong mo file thanh cong !"; return 0;
9      }
10     cout << "Day so doc duoc tu file la: " << endl;
11     int x, tong = 0;
12     while (!finp.eof()) {
13         finp >> x;
14         if (finp.eof()) {break;} // dung luon khi eof() = true
           tong = tong + x;
16         cout << x << " "; // hien thi ket qua len man hinh
17     }
18     cout << endl << "Tong cua day so doc duoc la: " << tong;
19     return 0;
20 }
```

Đọc từng số nguyên có mặt trong file, tuần tự từ đầu file cho đến hết

GHI DỮ LIỆU VÀO FILE

- Khai báo biến file fout thuộc lớp ofstream
- Sau khi mở file, các hàm làm việc với fout như sau

Hàm	Mô tả
fout << bieu_thuc;	Ghi biểu thức vào luồng ra file có biến file là fout.
fout.put(ch);	Ghi một kí tự ch ra file fout.
fout.fail()	Trả về true nếu file không được mở thành công
fout.close()	Đóng file, cần đóng file trước khi kết thúc chương trình

VÍ DỤ 4-8

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  main() {
5      int n;
6      ofstream fout;
7      fout.open("dayso.txt");
8      if (fout.fail()) {
9          cout << "Khong mo file thanh cong !"; return 0;
10     }
11     cout << "Cho so luong so: ";
12     cin >> n;
13     cout << "Nhap cac so: " << endl;
14     int i = 0, x;
15     while (i < n) {
16         cout << "So thu " << i << ": ";
17         cin >> x;  // nhap so tu ban phim
18         fout << x << " "; // ghi vao file fout
19         i++;
20     }
21     cout << "Chuong trinh da ket thuc !" << endl;
22     cout << "Mo file dayso.txt de xem ket qua !" << endl;
23     fout.close();
24     return 0;
25 }
```

Nhập một dãy số từ bàn phím và ghi vào file dayso.txt

Đọc dữ liệu từ bàn phím và ghi vào file fout

Cần đóng file trước khi kết thúc chương trình

BÀI TẬP

Bài 4-4: Cho file dayso.txt, mỗi dòng chứa một số số nguyên dương. Hãy đếm số lượng số chẵn có mặt trong file đã cho. Ghi kết quả lên màn hình.

Bài 4-5: Một lớp học có n sinh viên, mỗi sinh viên có 2 thông tin: *tên* và *điểm tổng kết*. Hãy viết chương trình nhập thông tin của n sinh viên từ bàn phím. Kiểm tra và lưu những sinh viên có tên là "Hoang" và có *điểm tổng kết* ≥ 8.0 vào file sinhvien.txt; mỗi dòng ghi thông tin của 1 sinh viên, gồm: *họ tên đầy đủ* và *điểm tổng kết*.

Bài 4-6: (trang bên)

BÀI TẬP

Bài 4-6: Một công ty sản xuất n mặt hàng, mỗi mặt hàng có các thông tin: *mã mặt hàng, tên mặt hàng, số lượng, giá bán*. Hãy lập trình đọc dữ liệu về các mặt hàng, sắp xếp các mặt hàng theo chiều giảm dần của số lượng, nếu hai mặt hàng bằng nhau về số lượng thì sắp xếp theo chiều giảm dần của giá bán.

- Dữ liệu vào được cho trong file: **mathang.inp** với cấu trúc như sau:
  - Dòng đầu ghi số n, là số lượng mặt hang
  - Cứ bốn dòng tiếp theo ghi thông tin về một mặt hang theo thứ tự:
    mã mặt hàng, tên mặt hàng, số lượng, giá bán
- Dữ liệu ra được ghi vào file: **mathang.out** với cấu trúc như sau:
  - Mỗi dòng ghi thông tin về một mặt hàng, bao gồm các thông tin: *mã mặt hàng, số lượng, giá bán*. Các mặt hàng được ghi theo thứ tự đã sắp xếp như yêu cầu bên trên.
- Ví dụ: <trang bên>

DỮ LIỆU MINH HỌA

mathang.inp	mathang.out
3 MH01 Dieu hoa Panasonic 20 600000 MH02 May tinh xach tay Dell 78 150000 MH03 Xe may Honda Vision 78 240000	Xe may Honda Vision 78 240000 May tinh xach tay Dell 78 150000 Dieu hoa Panasonic 20 600000