Các kiến thức cơ bản trong C++

Ban chuyên môn Tin Học – The Gifted Battlefield

Xuất bản vào Ngày 7 tháng 11 năm 2023

Mục lục

LỜI MỞ ĐẦU:

- Chuyên đề nhằm giới những kiến thức cơ bản, các yếu tố cần có trong 1 chương trình C++ và cách cài đặt. Tác giả sẽ tránh giải thích hay đi quá sâu vào phần chi tiết nhằm giúp người đọc dễ tiếp cận và mang tính ứng dụng thực tế cao hơn.

1 Cấu trúc chương trình

1.1 Các kiểu dữ liệu

- Trong một chương trình C++ thường sử dụng rất nhiều kiểu dữ liệu, nhưng để bắt đầu thì chúng ta cần chú ý đến các kiểu dữ liệu sau: **int, long long, double, char, string**. Có thể tham khảo đoạn code sau để hiểu hơn về cách khai báo biến của từng kiểu dữ liệu.

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

int main()
{
    int a = 10; // int dùng để lưu trữ 1 số nguyên
    long long b = 10000000000; // long long cũng tương tự với int nhưng có thể
    lưu số ở giá trị lớn hơn
    double c = 0.77; // double dùng để lưu số thực
    char d = 'q'; // char dùng để lưu 1 ký tự
    string e = "xin chao"; // string dùng để lưu 1 chuỗi gồm nhiều ký tự
    return 0;
}
```

- Với mỗi kiểu dữ liệu đều có 1 giới hạn riêng:

Kiểu	Bit	Giới hạn
char	8	-128 đến 127
unsigned char	8	0 đến 255
int	32	$-2,147,483,648$ đến $2,147,483,647$
unsigned int	32	0 đến $4,294,967,295$
long long	64	-2^{63} đến $2^{63}-1$
unsigned long long	64	0 đến $2^{64}-1$
float	32	-3.4×10^{38} đến 3.4×10^{38}
double	64	-1.7×10^{308} đến 1.7×10^{308}

1.2 Các câu lệnh và biểu thức (biểu thức so sánh, if, while, for)

Biểu thức so sánh

Trong C++ cú pháp để so sánh bao gồm

- a bằng b: a == b
- a khác b: a != b
- a bé hơn b: $a < b$
- a lớn hơn b: $a > b$
- a bé hơn hoặc bằng b: $a <= b$
- a lớn hơn hoặc bằng b: $a >= b$

- Các biểu thức điều kiện:
  - Biểu thức **AND** (VÀ): *<điều kiện 1>* && *<điều kiện 2>* sẽ đúng (trả về ***true***) khi thỏa **cả 2** điều kiện.
  - Biểu thức **OR** (HOẶC): *<điều kiện 1>* || *<điều kiện 2>* sẽ đúng (trả về ***true***) khi thỏa **1 trong 2** điều kiện.
  - Biểu thức **NOT**: *!<điều kiện>* sẽ trả về ***true*** khi *<điều kiện>* sai.

■ Câu lệnh điều kiện (if)

- ***if*** là một câu lệnh điều kiện trong C++ để lựa chọn thực thi hoặc bỏ qua 1 lệnh bằng cách kiểm tra điều kiện có thỏa.
- Cú pháp được sử dụng có dạng:

```
    if (điều kiện) {
        // thực hiện câu lệnh
    }
```

- Bên trong điều kiện sẽ là một phép so sánh để kiểm tra tính đúng đắn của điều kiện đó, ví dụ:

```
    // kiểm tra a có bằng 3 hay không
    if (a == 3) {
        // xuất ra đúng
    }
```

- Ngoài ra còn có cú pháp ***if...else*** được định nghĩa là nếu không thỏa điều kiện này thì ta sẽ thực hiện 1 câu lệnh khác, ví dụ:

```
    // kiểm tra a có bằng 3 hay không
    if (a == 3) {
        // thực hiện khi "a == 3" đúng
    }
    else {
        // thực hiện khi "a == 3" sai, tức là khi "a != 3"
    }
```

- Cú pháp cuối cùng cần nhớ trong câu lệnh if chính là ***else if***. Hiểu đơn giản là thay vì *else* sẽ ngay lập tức thực hiện câu lệnh còn lại, thì *else if* lại cần phải thỏa 1 điều kiện mới thực hiện được, ví dụ:

```
    // kiểm tra a có bằng 3 hoặc bằng 5 hay không
    if (a == 3) {
        // xuất ra đúng, a = 3
    }
    else if(a == 5) {
        // xuất ra đúng, a = 5
    }
    else {
        // xuất ra sai
    }
```

■ Vòng lặp while

- Vòng lặp trong C++ sẽ thực hiện lặp đi lặp lại các câu lệnh miễn sao điều kiện đặt ra vẫn còn thỏa.
- Vòng lặp *while* có cú pháp như sau:

```
    while (điều kiện) {
        // thực hiện câu lệnh
    }
```

- Ngay khi điều kiện trở nên sai, chương trình lập tức chuyển tới dòng sau vòng lặp.
- Để hiểu hơn, có thể tham khảo ví dụ sau:

```
    // đếm từ 1 tới 10
    int dem = 1;
    while (dem <= 10) {
        cout << dem << " "; // xuất ra biến đếm hiện tại
        ++dem;
    }
    // sẽ xuất ra: 1 2 3 4 5 6 7 8 9 10
```

■ Vòng lặp for

- Vòng lặp *for* là 1 cách viết tối ưu và tiện lợi hơn của vòng lặp *while*, vậy nên về bản chất thì vẫn giữ nguyên tư tưởng lặp lại.
- Điểm khác biệt của vòng *for* chính là có tồn tại thêm 2 cú pháp giúp khởi tạo và tăng/giảm biến đếm theo một cách nào đó.
- Cú pháp được sử dụng là:

```
    for (<khởi tạo biến>; <điều kiện>; <tăng / giảm biến>) {
        // thực hiện câu lệnh
    }
```

- Để hiểu rõ hơn, ta sẽ viết lại chương trình đếm từ 1 đến 10 bằng vòng *for*:

```
    // đếm từ 1 tới 10
    for (int dem = 1; dem <= 10; ++dem) {
        cout << dem << " "; // xuất ra biến đếm hiện tại
    }
    // sẽ xuất ra: 1 2 3 4 5 6 7 8 9 10
```

- Ta có thể dễ dàng nhận thấy việc viết vòng *for* như này sẽ tiện hơn vòng *while* trong 1 số trường hợp nhất định.
- Lưu ý: Biến được khởi tạo trong vòng *for* không mang tính cục bộ, đồng nghĩa với việc không thể truy cập nó bên ngoài vòng *for*.

1.3 Hàm

■ Tổng quan về hàm

- Trong C++ và đa số những ngôn ngữ lập trình hiện đại khác, hàm (function) là một dãy các câu lệnh có thể sử dụng lại nhiều lần ở nhiều vị trí khác nhau trong chương trình. Một hàm thường được tạo ra để thực hiện một công việc cụ thể trong chương trình.
- Mọi chương trình C++ đều có ít nhất một hàm, hàm ấy tên là main(). Một chương trình trong C++ sẽ được bắt đầu từ dòng code đầu tiên của hàm main, và kết thúc bằng dòng code cuối cùng của hàm main.
- Một hàm trong C++ sẽ đảm nhiệm một chức năng cụ thể trong chương trình, như là ghi nhận input, xuất ra output hay xử lý các phép toán.
- Cú pháp khai báo hàm trong C++:

```
<kiểu trả về> <tên hàm>([<danh sách tham số>]) {
    <các câu lệnh>
    return <giá trị>;
}
```

- Trong đó:
  - <kiểu trả về> là kiểu dữ liệu mà hàm trả về, nếu hàm không trả về giá trị thì kiểu trả về là void.
  - <tên hàm> là tên của hàm để có thể truy xuất và sử dụng trong chương trình.
  - [<danh sách tham số>] là danh sách các giá trị có thể đưa vào hàm, danh sách này có thể không có.

- <các câu lệnh> là các câu lệnh mà hàm sẽ thực thi nếu được gọi.
- return <giá trị>; là câu lệnh trả về giá trị, câu lệnh này chỉ tồn tại khi hàm không phải là hàm void.

- Ví dụ một chương trình C++ có hàm main() và hàm hello():

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

// Khai báo hàm hello
void hello() {
    cout << "Hello~~~";
}

int main() {
    // Gọi hàm hello
    hello();

    return 0;
}
```

- Trong chương trình trên, hàm main đã gọi hàm hello(). Trong hàm hello() có câu lệnh xuất ra thông điệp "Hello " nên ta thấy dòng chữ này trên console.
- Hàm hello không hề trả về bất kì giá trị gì nên kiểu trả về của hàm này là void.

■ Giá trị trả về

- Trong C++, hàm có thể xử lý những giá trị và trả về giá trị đã được xử lý.
- Nếu hàm của C++ có kiểu trả về khác void, thì sẽ có một giá trị được trả về thông qua câu lệnh return.
- Ví dụ một chương trình nhập vào tên người dùng và xin chào:

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


// Hàm nhập vào và trả về tên người dùng.
string getName() {
    cout << "Enter your name: ";

    string name;
    getline(cin, name);

    return name; // Câu lệnh trả về tên người dùng.
}

// Hàm chào người dùng.
void welcome(string message, string userName) {
    cout << message << " " << userName;
}

int main() {
    string name = getName();
    welcome("Hello", name);

    return 0;
}
```

- Trong chương trình trên, hàm getName() nhận tên người dùng từ bàn phím và trả về tên người dùng. Biến name trong hàm main() sẽ chứa giá trị được trả về từ hàm getName(). Và hàm hello() sẽ nhận thông điệp chào mừng và tên người dùng từ getName() để chào người dùng.
- **Lưu ý**: nếu hàm có kiểu trả về khác void nhưng không có câu lệnh return, compiler có thể báo lỗi hoặc warning. Nếu compiler không báo lỗi thì khi chạy hàm sẽ trả về giá trị rác.

1.4 Thư viện chuẩn C++ (STL)

- Thư viện là những đoạn mã gồm các chức năng, phương thức có sẵn để thực hiện những nhiệm vụ cụ thể. Thư viện gồm các hàm, kiểu dữ liệu,... thông dụng được định nghĩa sẵn giúp việc viết code nhanh và tiện lợi hơn. Thư viện giúp người viết mã không cần bỏ ra công sức thực hiện các chức năng phổ biến như sắp xếp mảng, tìm kiếm,... Từ đó, hạn chế lỗi và code được viết nhanh hơn rất nhiều.
- Thư viện chuẩn (STL - Standard Template Library) trong C++ là một công cụ mạnh mẽ trong việc viết mã với C++. Thư viện này chứa những kiểu dữ liệu cực kì phổ biến và mạnh mẽ, các hàm có sẵn tốc độ cực nhanh và chuẩn xác. STL có thể cung cấp các chức năng từ xử lý chuỗi tới sinh số ngẫu nhiên, các cấu trúc dữ liệu phức tạp được đơn giản hóa còn vài dòng code trong chương trình, các phép toán phức tạp thành những lời gọi hàm cơ bản.
- **Cú pháp khai báo thư viện trong C++:**

```
#include <tên thư viện>
```

- **Thư viện iostream**: Thư viện này cung cấp các phương thức cho việc nhập xuất với chương trình, thao tác với file. Giúp làm đơn giản hóa quá trình nhập xuất dữ liệu. Các thành phần trong thư viện này phải kể đến như cin, cout, cerr, freopen,...
- **Thư viện algorithm**: Cung cấp các thuật toán thông dụng như binary search, sort, find,... Thư viện này rút ngắn chương trình đáng kể bởi các thuật toán thông dụng chỉ cần một câu lệnh gọi hàm để sử dụng. Ngoài ra các thuật toán của thư viện này được cài đặt chuẩn xác nên có thể đảm bảo tính đáng tin cậy của chương trình. Các hàm thông dụng trong thư viện này là sort(), binary_search(), upper_bound(), lower_bound(), find(),...
- **Thư viện cmath**: Là thư viện giúp thực hiện các phép toán số học, thực hiện các phép toán phức tạp như căn, lũy thừa, logarithm,... Những phép toán khá khó cài đặt trong lập trình và phổ biến đều nằm trong thư viện này. Các hàm thông dụng trong thư viện này gồm: sqrt(), log2(), pow(),...
- **Thư viện bits/stdc++.h**: Khi khai báo thư viện này trong chương trình, toàn bộ thư viện của STL sẽ được tự động thêm vào chương trình. Thư viện này chỉ hoạt động với compiler GNU MinGW.
- Ngoài những thư viện kể trên, còn rất nhiều thư viện có sẵn trong C++ như *set, map, iomanip, random, vector, array, queue, list, deque, bitset, complex,...*

2 Input/Output

2.1 cin, cout, cerr

- Cin và cout là 2 câu lệnh được thêm vào trong C++ nhằm phục vụ thao tác input và output
- Để có thể nhập vào 1 dạng dữ liệu, ta sử dụng cin, đi kèm với đó là toán tử (»), ví dụ:

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cin >> a; // nhập vào 1 số
    string s;
    cin >> s; // nhập vào 1 chuỗi
    string t;
    getline(cin, t); // nhập vào 1 dòng, thường sử dụng khi ta cần nhập 1 chuỗi có chứa dấu cách ' '
    return 0;
}
```

- Ngược lại với cin là cout. Câu lệnh cout đi kèm với toán tử («) giúp chúng ta xuất ra các dữ liệu. Ngoài ra các bạn còn phải nhớ 1 vài cú pháp như "\n"là xuống dòng (hoặc endl, nhưng trong lập trình thi đấu không khuyến khích sử dụng vì tốc độ chậm hơn), xem ví dụ sau:

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

int main()
{
    // Chương trình xuất ra số tuổi của An và Bình
    string ten = "An";
    int tuoi = 10;
    cout << ten << " " << tuoi << " tuoi \n"; // Output: An 10 tuoi, sau đó xuống dòng
    ten = "Binh";
    tuoi = 15;
    cout << ten << " " << tuoi << " tuoi \n"; // Output: Binh 15 tuoi
    return 0;
}
```

- Lưu ý rằng ta hoàn toàn có thể gộp các câu lệnh cin/cout:

```cpp
    int a, b;
    cin >> a; cin >> b;
    cout << a; cout << b;
```

- Thành các câu lệnh ngắn gọn hơn:

```cpp
    int a, b;
    cin >> a >> b;
    cout << a << b;
```

- Ngoài ra còn một kiểu xuất dữ liệu được gọi là cerr, sẽ xuất ra luồng stderr. Cách sử dụng y hệt như cout, nhưng thường được dùng để debug vì trong trường hợp quên xóa cerr thì output vẫn không bị ảnh hưởng.

2.2 Fast I/O

- Trong lập trình thi đấu, tốc độ là 1 yếu tố quyết định. Vậy nên việc sử dụng cin/cout sẽ có thể ảnh hưởng khi nó chạy chậm hơn printf/scanf khá nhiều. Nhưng ta có thể cải thiện điều này bằng fast I/O qua câu lệnh sau:

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

int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    return 0;
}
```

3 Nhập xuất với file

- Trong lập trình thi đấu, nhiều khi đề bài sẽ yêu cầu thí sinh nhập xuất với file. Nên

3.1 freopen

- freopen() là hàm duy nhất trong C++ để thay đổi hoàn toàn hướng của luồng nhập xuất. Nghĩa là freopen() có thể định hướng chương trình từ nhập hoàn toàn với bàn phím thành nhập xuất hoàn toàn với file. Hàm này nằm trong thư viện iostream.
- Trong C++, cout sẽ hoạt động qua stdout và cin sẽ hoạt động trên stdin. Nếu ta cho stdin và stdout trở thành một file thì cin, cout nghiễm nhiên nhận input, xuất output thông qua file.
- Hàm freopen() được khai báo với các tham số như sau:

```
    FILE* freopen(const char* filename, const char* mode, FILE* stream);
```

- Hàm này trả về một con trỏ kiểu FILE. Tuy nhiên trong bài viết này, ta sẽ không sử dụng tới giá trị trả về của nó.
- filename là tên file mà ta muốn mở để nhập/xuất. VD: "input.inp", "output.out"
- mode là tên chế độ mở file, trong bài viết này, ta chỉ quan tâm tới mode "r" và "w".
  - "r": viết tắt của read, là chế độ đọc file. Nếu file chưa tồn tại thì sinh lỗi.
  - "w": viết tắt của write, là chế độ ghi file. Nếu file đã tồn tại thì nội dung của file sẽ bị xóa sạch hoàn toàn. Nếu file chưa tồn tại thì tự động tạo file ấy.
- stream là luồng dữ liệu cần gán với file đó. Ví dụ ta muốn đọc file với cin thì stream lúc này sẽ là stdin, nếu muốn ghi file với cout thì stream sẽ là stdout.
- Hay nói cách khác, chúng ta có cú pháp khai báo file nhập xuất như sau:

```
    freopen("<tên file input>", "r", stdin);
    freopen("<tên file output>", "w", stdout);
```

- Ví dụ về nhập/xuất file với freopen():
  - Giả sử ta có file input.inp ở cùng một folder với file exe. Nội dung của file input.inp như sau:

```
HSG_VNU-HCM
```

  - Và nội dung file code main.cpp như sau:

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

int main() {
    freopen("input.inp", "r", stdin); // Gán file input.inp cho cin với mode r (đọc file).
    freopen("output.out", "w", stdout); // Gán file output.out cho cout với mode w (ghi file).

    // Lúc này cin đã được gán thành file input.inp nên cin sẽ đọc file input.inp
    string school;
    cin >> school;

    // cout đã được gán với file output.out nên cout sẽ ghi vào file output.out
    cout << "Your school is: " << school;
}
```

  - Sau khi chạy chương trình, file output.out được tạo nhờ chế độ "w" và có nội dung như sau:

```
Your school is: HSG_VNU-HCM
```

- Tóm gọn lại, freopen() giúp ta thay vì cin, cout với console thì thao tác cin, cout sẽ được thực hiện với file được chọn.

3.2 Đọc hết file (while(cin » str))

- Khi sử dụng cin với file, nếu cin được đặt làm điều kiện của vòng lặp while thì cin sẽ đọc liên tục cho đến khi hết file.
- Giả sử ta có file input.inp như sau:

```
this is a sentence.
```

- Ví dụ cách sử dụng while(cin » str)

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

int main() {
```

```
    freopen("input.inp", "r", stdin); // Gán file input.inp cho cin với mode r (đọc file).

    string str;
    // Nhập từng từ từ file input.inp cho tới khi hết file
    while (cin >> str) {
        cout << str << endl;
    }
}
```

- Kết quả nhận được sau khi chạy code là:

```
this
is
a
sentence.
```

- Vòng lặp while sẽ tự động dừng khi cin đạt đến cuối file. Và mỗi lần cin ta nhận được một từ trong file input.inp nên các từ ở trong output của ta được cách nhau bởi ký tự xuống dòng (endl).
- Ngoài cin, ta cũng có thể dùng getline(cin, str) làm điều kiện cho vòng lặp while, lúc này thay vì nhận được str là từng từ, ta nhận được str là từng dòng trong file. Vòng lặp lúc này cũng sẽ tự động ngừng khi đọc tới cuối file.