Lập trình C/C++ online
Tránh lỗi tối đa và debug!

Conan Kudo
nqc290997@gmail.com,
Trần Việt Khoa
tvkhoa.husc@gmail.com
Khoa Công nghệ thông tin, Đại học Khoa học Huế

Ngày 27 tháng 3 năm 2017

1 Lỗi khi lập trình Online

C++ là một ngôn ngữ mạnh, nhanh và gần gũi với lập trình viên (được dạy trong trường Đại học). Lập trình online, lập trình cạnh tranh (Competitive Programming) bằng cách sử dụng C/C++ là phổ biến. Tuy nhiên để có kết quả tốt bạn phải nắm bắt việc sử dụng ngôn ngữ một cách hiệu quả, bằng cách thường xuyên thực hành và tích lũy kinh nghiệm. Những bạn mới bắt đầu với ngôn ngữ này thường gặp phải khó khăn khi lập trình Online hay lập trình cạnh tranh. Những khó khăn này làm họ sớm nản lòng, nhưng thực sự nó là không quá khó khăn. Vì vậy bài viết này sẽ cố gắng giải thích một số kỹ thuật cơ bản nhằm khắc phục những khó khăn ban đầu đối với những ai mới bắt đầu học lập trình C++.

Hệ thống lỗi trong C/C++ cơ bản chia ra hai dạng:

1. Error: Thường là những lỗi cơ bản về ngữ pháp, cú pháp và được xem là lỗi dễ sửa.

2. Warning: Thường là những lỗi về logic, về dùng sai phép toán, lỗi truy cập. Đây là những lỗi sẽ gây ức chế cho người lập trình bởi vì sau khi sửa hết lỗi Error và bạn nộp chương trình, các lỗi sau sẽ xuất hiện: run-time error, Time limit, Memory limit, Wrong answer,...

Chú ý: Một số IDE hỗ trợ lập trình hầu như không thông báo về các lỗi warning trên, nó chỉ gặp sau khi ta cứ nghĩ rằng mình đã đúng (submit - nộp bài).

1.1 Một số lỗi thường xảy ra Warning.

1. Lỗi Off-By-One.

Lỗi xảy ra khi ta sử dụng dấu <= thay cho < khi ta kiểm tra biểu thức điều kiện trong vòng lặp. Xét ví dụ:

```
#include <iostream>
using namespace std;
int main()
{
```

```
    int i, a[5];
    for (i = 0; i <= 5; i++)
        a[i]= i;
    for (i = 0; i <= 5; i++)
        cout << a[i] << " ";
     return 0;
}
```

Chương trình trên xảy ra lỗi khi truy cập ngoài vùng của mảng, trong mã có một phép truy cập a[5], trong khi chỉ số của mảng tối đa chỉ là 4, bởi vì chỉ số mảng của C/C++ bắt đầu từ 0. Chương trình thay lại bởi lệnh for (i = 0; i < 5; i++) cho hai vòng lặp trên.

2. Lỗi vòng lặp vô tận (infinite loop).

Xét chương trình sau:

```
int i = 5;
int sum = 0;

while (i <= 10)
   sum += i;
```

Vòng lặp không thay đổi giá trị biểu thức điều kiện lặp, vòng lặp vô tận xảy ra, đoạn mã sau thay đổi lại như sau:

```
int i = 5;
int sum = 0;

while (i <= 10) {
   sum += i;
   i++;
}
```

3. Lỗi sử dụng phép gán và phép so sánh bằng trong điều kiện lặp.

Đôi khi trong biểu thức so sánh của vòng lặp ta dùng phép so sánh bằng (== trong C/C++, Java), lỗi xuất hiện khi ta gõ thiếu dấu = làm phép toán biến đổi thành phép gán, đoạn mã sẽ bị lặp vô tận. Xét ví dụ:

```
int x, y;

do {
   //action()
} while (x = y);
```

Ngoài ra, lỗi nhầm lẫn giữa && và ||, sử dụng sai mục đích của hai phép toán logic này trong biểu thức. Lỗi chia cho 0 xuất hiện trong các biểu thức của bạn cũng là một lỗi cực kỳ nặng. Tất cả các lỗi trên sẽ làm chương trình của bạn bị lỗi RUN-TIME ERROR.

1.2 Lỗi của Online Judge

Trong hệ thống bài thi online (máy chấm), các lỗi sau (dạng warning) cũng thường xuyên xảy ra với các bạn:

1. Lỗi TIME LIMIT - lỗi này thường xảy ra khi mã chương trình của bạn gặp:

+ Vòng lặp vô tận, hoặc bị lặp vô tận khi bạn vô tình sử dụng các lệnh như getch(), system("pause").

+ Quan trọng hơn cả là thuật toán của bạn không vượt qua được thời gian quy định của máy chấm. Như vậy một thuật toán tối ưu sẽ phải được tìm kiếm và thay thế cho thuật toán hiện đang dùng của bạn. Việc đánh giá nhanh, chậm có thể được chứng minh qua thuật ngữ "Độ phức tạp thuật toán".

2. Lỗi MEMORY LIMIT - Lỗi giới hạn bộ nhớ được thiết lập bởi quản trị hệ thống Online judge nhằm hạn chế người sử dụng biến, hằng quá nhiều trong chương trình.

+ Ví dụ để liệt kê, đếm các số nguyên tố trong một khoảng nhất định nào đó, người lập trình có thể dùng thuật toán brute force để tính sau đó ghi lại kết quả vào các biến hằng nhằm dễ xử lý trong chương trình.

+ Lỗi cũng có thể xảy ra khi bạn dùng biến static với dung lượng quá lớn.

3. Lỗi WRONG ANSWER - Lỗi sai thuật toán, hoặc giả kết quả out put của bạn không đúng định dạng yêu cầu.

2 Debug

Một số ý tưởng tốt cho việc sửa bài (debug).

1. Kiểm tra test dữ liệu đầu vào của mục ví dụ.

- Hầu hết các bài toán đều cho ví dụ mẫu, bạn phải đúng ít nhất trường hợp này mới có thể thử nộp. Nhưng quá nhanh dẫn đến nguy hiểm bởi vì qua được test này không có nghĩa là thuật toán bạn đúng.

2. Kiểm tra test có tính không đúng (incorrect).

- Phương pháp chứng minh phản ví dụ là một phương pháp tốt trong toán học cũng như tin học, việc bạn suy nghĩ và tìm ra một phản ví dụ để kiểm chứng thuật toán của mình là việc cần làm.

3. Kiểm tra điều kiện biên.

- Nhiều lỗi trong chương trình là do lỗi "off-by-one" xảy ra như dữ liệu vào bị rỗng (empty), chưa khởi tạo giá trị mặc định, chia cho 0, truy cập ngoài vùng dữ liệu (mảng).

4. Test các trường hợp đặc trưng khi ta biết chính xác câu trả lời.

Thật dễ dàng kiểm tra các trường hợp riêng mà ta có thể giải quyết bằng tay, với hầu hêt các trường hợp riêng có vẻ hợp lý cho bài toán dựa theo việc phân tích hành vi mong muốn.

5. Kiểm tra trường hợp dữ liệu lớn khi mà bạn không biết chính xác câu trả lời đúng.

- Thông thường ta có thể kiểm tra bằng tay (giấy và bút) cho các ví dụ nhỏ. Điều khó khăn hơn cho ta là với dữ kiện lớn việc tính toán bằng tay là không thể. Trong trường hợp này ta thử vài trường hợp dữ liệu lớn được xây dựng bằng ngẫu nhiên hoặc bằng quy tắc mà sao cho có thể kiểm chứng được để đảm bảo chương trình không bị rơi vào trường hợp lỗi.

Kiểm tra (testing) là nghệ thuật gỡ lỗi của người lập trình, việc tìm và fix lỗi hoàn toàn là một công việc khó khăn đòi hỏi nhiều kinh nghiệm. Được rèn luyện, thực hành qua nhiều bài toán, nhiều tình huống (codeforces có chức năng hacked). Viết một chương trình mà tránh được các lỗi là cả một nghệ thuật và chiến thuật. Làm thế nào để tránh và gỡ được lỗi, các hiểu biết sau sẽ giúp bạn:

1. Sử dụng được kỹ năng debug.

Bất kỳ IDE nào cũng đều hỗ trợ chức năng này, giúp bạn quan sát được giá trị của biến ở bất kỳ thời điểm chạy (runtime) nào của mã nguồn, giúp bạn kiểm soát được việc tính toán, logic của thuật toán thông qua quan sát giá trị của các biến.

2. Hiển thị cấu trúc dữ liệu của bạn.

Xuất ra file dữ liệu của chương trình được lưu trữ trong cấu trúc dữ liệu của bạn ra tại một thời điểm runtime nào đó của chương trình và quan sát.

3. Kiểm tra các biến cố bất biến.

Một bất biến là đúng với bất kỳ đầu vào nào của dữ liệu.

4. Kiểm tra mã chương trình.

Kiểm tra mã chương trình một cách cẩn thận có lẽ là cách đơn giản nhất, bằng cách đặt câu hỏi với 5 W cũng sửa được hầu hết các lỗi đơn giản, ngớ ngẩn.

5. Thêm các lệnh in (print).

Bổ sung thêm lệnh print để in ngữ nghĩa cũng như giá trị tính toán ở từng block lệnh nhằn báo cáo cho chúng ta biết trong từ khối lệnh của từng thời điểm.

6. Sử dụng mảng lớn hơn so với điều kiện cần thiết (ràng buộc) của đề.

Cuối cùng giới thiệu với các bạn khái niệm hacked là gì?

Trường hợp này chỉ có duy nhất trên codeforces.com. Nếu là đang trong quá trình thi. Hệ thống chỉ đưa ra một số test để kiểm tra tính đúng của thuật toán.

Một vài trường hợp còn lại, thường là troll để cho các bạn suy nghĩ về nó, tăng thêm độ vui vẻ của cuộc thi. Nếu code bạn bị hack, có nghĩa là code bạn đã sai trong một số trường hợp mà người đọc code của bạn biết được bạn làm thiếu nó. Nên bị hack.

Thế hack code người khác như thế nào?

Đầu tiên bạn vào problem, chọn vấn đề mình đã giải được và khóa nó, sau đó nhấp vào room. Là phòng mà codefor ngẫu nhiên cho bạn vào, trong đó có bảng rank và bạn có thể xem code của người khác,ở câu bạn đã làm được và đã khóa. Sau đó đọc code, nếu nghĩ nó sai, ấn vào hack rồi nhập test mà bạn nghĩ là code người ta cho ra kết quả sai. Nếu hack thành công bạn được cộng 100đ. Không thành công bị trừ 50đ. Vui lắm!!!

3 Sinh test

3.1 Cơ bản C++

1. Mẫu chương trình.

Nhắc lại cấu trúc của một file giải một bài toán tin viết bằng C++11.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
   // solution comes here
   return 0;
}
```

Chương trình trên sẽ được biên dịch bằng câu lệnh sau, giả sử bạn dùng DEVC bạn có thể vào option để thay đổi lại lệnh biên dịch:

```
g++ -std=c++11 -O2 -Wall code.cpp -o bin
```

Trong đó, file code.cpp là file chứa mã chương trình và file bin là file chạy của chương trình, sẽ hợp lý hơn nếu bạn dùng hệ điều hành Ubuntu cho công việc này.

2. Vào ra dữ liệu.

Hầu hết các bài toán thi online trên máy chấm đều sử dụng vào ra chuẩn, do vậy nếu dùng C++ ta sẽ dụng 2 lớp vào ra chuẩn là cin/cout. Có thể thay đổi bằng hai hàm scanf() và printf() của C cho chức năng này.

Đối với cin/cout trong quá trình sử dụng có thể sinh ra hiệu ứng nút cổ chai, để đồng bộ hóa bạn sử dụng câu lệnh sau:

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

Ngoài ra bạn nên thay endl bằng "\n".

Ví dụ:

```
int a, b;
string x;
cin >> a >> b >> x;
```

Trong quá trình debug trên máy cá nhân trước khi nộp bài, người ta khuyến khích bạn sử dụng text file thay cho input/output chuẩn. Một phần đỡ công nhập và quan sát kết quả, một phần có thể debug với nhiều testcase khác nhau. Bằng cách chuyển định hướng stdin/stdout hoặc ifstream/ofstream cho in/out chuẩn. Mẫu chương trình như sau:

```
   cin.sync_with_stdio(0);
   cout.sync_with_stdio(0);
   cin.tie(0);
   cout.tie();
   #ifndef ONLINE_JUDGE
      //ifstream cinf("INPUT.txt");
      //ofstream coutf("OUTPUT.txt");
      freopen("INPUT.txt", "r", stdin);
      freopen("OUTPUT.txt", "w", stdout);
   #endif
```

Như vậy dữ liệu được tổ chức lưu trữ ở file INPUT.txt được đọc vào và kết quả được in ra trên OUTPUT.txt, dòng tiền định nghĩa trên cho phép các máy chấm bỏ qua việc điều hướng này để chọn đúng dòng vào ra chuẩn (#ifndef ONLINE_JUDGE ... #endif). Dĩ nhiên bạn phải trả về lại môi trường cũ cho viều điều hướng bằng các lệnh

như fclose(stdin), fclose(stdout).

3. Làm gọn code.

Để nhanh chóng triển khai code khi thi người ta sử dụng #typedef để định nghĩa lại các tên dài, các cấu trúc làm cho chương trình gọn hơn khi dùng, sau đây là một số định nghĩa quen thuộc:

Thay vì dùng:

```
long long a = 123456789;
long long b = 987654321;
cout << a*b << "\n";
```

Ta định nghĩa lại như sau:

```
typedef long long ll;
ll a = 123456789;
ll b = 987654321;
//chu y: voi kieu long long 64 bit phep gan so lon co kieu sau:
//ll x = 123456789123456789LL; phia sau co 2 ky tu LL.
cout << a*b << "\n";
```

Tương tự ta có các kiểu dữ liệu như pair, vector sẽ được định nghĩa:

```
typedef vector<int> vi;
typedef pair<int,int> pi;
```

Ngoài ra còn có thể sử dụng macro, ví dụ như:

```
#define F first
#define S second
#define PB push_back
#define MP make_pair
#define REP(i,a,b) for (int i = a; i <= b; i++)
//sau khi dinh nghia, doan ma sau
v.push_back(make_pair(y1,x1));
v.push_back(make_pair(y2,x2));
int d = v[i].first+v[i].second;
//co the thay bang
v.PB(MP(y1,x1));
v.PB(MP(y2,x2));
int d = v[i].F+v[i].S;
//vong lap
for (int i = 1; i <= n; i++) {
   search(i);
}
//co the thay bang
REP(i,1,n) {
   search(i);
}
```

3.2 Sinh test

Ngoài các testcase đơn giản, testcase cho bởi ví dụ mẫu có thể làm bằng tay. Các testcase phức tạp đều được tạo ra bằng các quy tắc sau:

- Sinh dữ liệu ngẫu nhiên. Sử dụng hàm rand() của C/C++ để tạo dữ liệu ngẫu nhiên.

- Sinh dữ liệu theo quy tắc.

+ Sinh dãy tăng dần, có thể giải quyết bằng ngẫu nhiên rồi sắp thứ tự.

+ Sinh dãy bằng nhau có giá trị MAX, MIN

+ Sinh cấp số cộng.

+ Sinh dãy tăng dần đôi một khác nhau.

+ Sinh hoán vị.

+ Sinh tổ hợp.

+ Sinh xâu ký tự theo mẫu.

Tất cả các dữ liệu sinh trên được lưu vào file input.txt (hoặc i.in, i=1..n) để tạo các bộ test cho chương trình.

4 Bài tập

Hãy viết các đoạn chương trình cho các bài toán sinh dữ liệu ở trên.