

Xây Dựng "Bộ Công Cụ" Số Học Vạn Năng

Chương 6: Các hàm quan trọng trong Lập trình thi đấu

Ngày 1 tháng 2 năm 2026

Mục tiêu

Thay vì ném cho bạn một tệp code dài dằng dặc, chúng ta sẽ cùng nhau xây dựng một **"bộ công cụ" (Toolbox)** vạn năng cho các bài toán số học.

Trong C++, các hàm này chính là những "trợ thủ" đắc lực giúp bạn giải quyết nhanh các bài toán trên Codeforces.

Chúng ta sẽ chia lộ trình học thành 3 nhóm kỹ năng (Chunks):

- 1 Những "Viên gạch" Cơ bản (Chẵn, Lẻ, Tuyệt đối).
- 2 Nhóm Số Học "Đặc Sản" (Số Nguyên Tố & Số Chính Phương).
- 3 Nhóm "Số học nâng cao" & GCD/LCM.

Chunk 1: Những "Viên gạch" Cơ bản

- **Giá trị tuyệt đối (Absolute Value):** Khoảng cách từ số đó đến số 0. Trong C++, dùng hàm `abs(n)`.
- **Chẵn/Lẻ:** Dựa vào phép chia lấy dư %.
 - Số chẵn: Chia hết cho 2 (dư 0).
 - Số lẻ: Chia cho 2 dư 1 (hoặc -1 với số âm).

Hình ảnh ẩn dụ

Hãy tưởng tượng bạn có một nắm kẹo:

- **Số chẵn:** Chia thành các cặp 2 viên mà không thừa viên nào.
- **Số lẻ:** Sau khi chia cặp, luôn thừa ra 1 viên lẻ loi.

Chunk 1: Mã minh họa (C++)

```
1 // Ham kiem tra chan
2 bool isEven(int n) {
3     return n % 2 == 0;
4 }
5
6 // Ham kiem tra le
7 bool isOdd(int n) {
8     return n % 2 != 0;
9 }
```

Bẫy logic (Trap)

Khi kiểm tra số lẻ, nhiều người viết $n \% 2 == 1$. Điều này sẽ **SAI** nếu n là số âm.

Ví dụ: $-3 \% 2$ sẽ ra -1 .

→ Cách an toàn nhất là $n \% 2 != 0$.

Thử thách tư duy 1

Giả sử tôi có một biến $x = -10$.

- 1 Kết quả của `abs(x)` là bao nhiêu?
- 2 Hàm `isEven(x)` sẽ trả về `true` hay `false`?

Thử thách tư duy 1

Giả sử tôi có một biến $x = -10$.

- 1 Kết quả của `abs(x)` là bao nhiêu?
- 2 Hàm `isEven(x)` sẽ trả về `true` hay `false`?

Đáp án

- `abs(-10)` trả về **10** (khoảng cách đến 0 không phân biệt âm dương).
- `isEven(-10)` trả về **true** vì -10 chia hết cho 2.

Chunk 2: Nhóm Số Học "Đặc Sản"

Đây là nhóm hàm quan trọng để tối ưu thời gian chạy (tránh TLE).

1. Số nguyên tố (Prime Number)

Là số chỉ chia hết cho 1 và chính nó ($n > 1$).

- **Chiến thuật:** Chỉ kiểm tra từ 2 đến \sqrt{n} .
- **Lý do:** Nếu $n = a \times b$, chắc chắn một trong hai số phải $\leq \sqrt{n}$.

2. Số chính phương (Perfect Square)

Là số mà căn bậc hai của nó là một số nguyên (VD: 4, 9, 16).

- **Cách kiểm tra:** Tính \sqrt{n} , làm tròn xuống, bình phương ngược lại xem có bằng n không.

Chunk 2: Mã minh họa (C++)

```
1 // Kiểm tra số nguyên tố
2 bool isPrime(int n) {
3     if (n < 2) return false;
4     // i * i <= n tương đương i <= sqrt(n)
5     for (int i = 2; i * i <= n; i++) {
6         if (n % i == 0) return false;
7     }
8     return true;
9 }
10
11 // Kiểm tra số chính phương
12 bool isPerfectSquare(int n) {
13     if (n < 0) return false;
14     int root = sqrt(n);
15     return (root * root == n);
16 }
```

Bẫy logic (Trap)

Với số nguyên tố, hãy luôn nhớ trường hợp $n < 2$. Rất nhiều bạn quên và kết luận 1 hoặc 0 là số nguyên tố → Wrong Answer.

Thử thách tư duy 2

Xét số **9**.

- 1 Theo thuật toán `isPrime`, vòng lặp kiểm tra các số i nào? Kết quả?
- 2 Với `isPerfectSquare`, giá trị `root` là bao nhiêu? Kết quả?

Hỏi thêm: Số 9 có "vừa là số nguyên tố, vừa là số chính phương" không?

Thử thách tư duy 2

Xét số 9.

- 1 Theo thuật toán `isPrime`, vòng lặp kiểm tra các số i nào? Kết quả?
- 2 Với `isPerfectSquare`, giá trị `root` là bao nhiêu? Kết quả?

Hỏi thêm: Số 9 có "vừa là số nguyên tố, vừa là số chính phương" không?

Đáp án

- **isPrime:** Kiểm tra đến $\sqrt{9} = 3$. Khi $i = 3$, $9\%3 == 0 \rightarrow$ Trả về **false**.
- **isPerfectSquare:** $root = 3$. $3 \times 3 = 9 \rightarrow$ Trả về **true**.
- Kết luận: Số 9 không phải số nguyên tố, nó là số chính phương.

Chunk 3: Số hoàn hảo & Số đối xứng

1. Số hoàn hảo (Perfect Number)

Tổng các ước thực sự (không tính chính nó) bằng chính nó.

VD: $6 = 1 + 2 + 3$.

2. Số đảo (Reverse) & Đối xứng (Palindrome)

- **Số đảo:** Viết ngược lại (VD: $123 \rightarrow 321$).
- **Số đối xứng:** Đảo ngược vẫn là chính nó (VD: 121).
- **Thuật toán:** Dùng $\%$ 10 để lấy số cuối và $/$ 10 để cắt số cuối.

Chunk 3: Mã minh họa (C++)

```
1 // Kiểm tra số hoàn hảo
2 bool isPerfectNumber(int n) {
3     if (n <= 1) return false;
4     int sum = 0;
5     for (int i = 1; i <= n / 2; i++) {
6         if (n % i == 0) sum += i;
7     }
8     return (sum == n);
9 }
10
11 // Hàm lấy số đảo ngược
12 int getReverse(int n) {
13     int rev = 0;
14     while (n > 0) {
15         rev = rev * 10 + (n % 10);
16         n /= 10;
17     }
18     return rev;
19 }
20
21 bool isPalindrome(int n) { return (n == getReverse(n)); }
```

Bẫy logic (Trap)

Thử thách tư duy 3

"Chạy tay" thuật toán getReverse với số **123**.

❶ **Vòng 1:** $n \% 10$ lấy ra số mấy? rev bằng bao nhiêu?

Thử thách tư duy 3

"Chạy tay" thuật toán `getReverse` với số **123**.

- ❶ **Vòng 1:** $n \% 10$ lấy ra số mấy? `rev` bằng bao nhiêu?
- ❷ **Vòng 2:** Sau khi $n /= 10$, `n` còn bao nhiêu? `rev` tiếp theo?

Thử thách tư duy 3

"Chạy tay" thuật toán getReverse với số **123**.

- ❶ **Vòng 1:** $n \% 10$ lấy ra số mấy? rev bằng bao nhiêu?
- ❷ **Vòng 2:** Sau khi $n /= 10$, n còn bao nhiêu? rev tiếp theo?
- ❸ **Kết quả:** Số **28** có phải số hoàn hảo không? (Ước: 1, 2, 4, 7, 14).

Thử thách tư duy 3

"Chạy tay" thuật toán getReverse với số **123**.

- ① **Vòng 1:** $n \% 10$ lấy ra số mấy? rev bằng bao nhiêu?
- ② **Vòng 2:** Sau khi $n /= 10$, n còn bao nhiêu? rev tiếp theo?
- ③ **Kết quả:** Số **28** có phải số hoàn hảo không? (Ước: 1, 2, 4, 7, 14).

Đáp án

- V1: Lấy **3**, $rev = 3$. $n = 12$.
- V2: Lấy **2**, $rev = 3 \times 10 + 2 = 32$. $n = 1$.
- V3: Lấy **1**, $rev = 321$.
- Số 28: $1 + 2 + 4 + 7 + 14 = 28 \rightarrow$ **Là Số hoàn hảo.**

Chunk 4: GCD (UCLN) và LCM (BCNN)

- **GCD (Greatest Common Divisor):** Thuật toán Euclid ($GCD(a, b) = GCD(b, a \% b)$).
- **LCM (Least Common Multiple):** $LCM(a, b) = \frac{|a \times b|}{GCD(a, b)}$.

```
1 int getGCD(int a, int b) {  
2     while (b != 0) {  
3         int temp = a % b;  
4         a = b; b = temp;  
5     }  
6     return a;  
7 }  
8  
9 int getLCM(int a, int b) {  
10    if (a == 0 || b == 0) return 0;  
11    return abs(a * b) / getGCD(a, b);  
12 }
```

Mẹo Codeforces

Trong C++17, dùng hàm có sẵn `__gcd(a, b)` hoặc `std::gcd(a, b)` để tiết kiệm thời gian!

Thử thách tư duy cuối cùng

Tìm UCLN và BCNN của **12** và **8**.

- 1 Theo Euclid: $12 \% 8$ dư mấy? Số dư này trở thành b hay a?
- 2 UCLN cuối cùng là bao nhiêu?
- 3 Tính BCNN dựa trên UCLN đó.

Thử thách tư duy cuối cùng

Tìm UCLN và BCNN của **12** và **8**.

- 1 Theo Euclid: $12 \% 8$ dư mấy? Số dư này trở thành b hay a?
- 2 UCLN cuối cùng là bao nhiêu?
- 3 Tính BCNN dựa trên UCLN đó.

Đáp án

- $12 \% 8 = 4$. Số dư 4 trở thành số chia mới (b).
- Tiếp: $8 \% 4 = 0$. Dừng lại. → **UCLN = 4**.
- BCNN: $(12 \times 8) / 4 = 96 / 4 = \mathbf{24}$.

Tổng kết bộ công cụ số học

Loại số	Đặc điểm chính	Kỹ thuật C++
Chẵn/Lẻ	Chia dư cho 2	$n \% 2 == 0$
Số Nguyên Tố	Chỉ có 2 ước	Loop từ 2 đến \sqrt{n}
Chính Phương	Căn bậc 2 nguyên	$\text{sqrt}(n)$ và bình phương lại
Hoàn Hảo	Tổng ước bằng nó	Loop tìm ước và cộng dồn
Đối Xứng	Đọc xuôi ngược như nhau	Đảo ngược bằng $\% 10, / 10$
UCLN/BCNN	Ước chung/Bội chung	Thuật toán Euclid

Chúc bạn code vui vẻ và chinh phục các bài toán số học!