Fast I/O for Competi ing

In competitive programming, it is important to read input as fast as possible so we save valuable time.

You must have seen various problem statements saying: "***Warning:*** *Large I/O data, be careful with certain languages (though most should be OK if the algorithm is well designed)*". Key for such problems is to use Faster I/O techniques.

It is often recommended to use scanf/printf instead of cin/cout for a fast input and output. However, you can still use cin/cout and achieve the same speed as scanf/printf by including the following two lines in your main() function:

```
ios_base::sync_with_stdio(false);
```

It toggles on or off the synchronization of all the C++ standard streams with their corresponding standard C streams if it is called before the program performs its first input or output operation. Adding ios_base::sync_with_stdio (false); (which is true by default) before any I/O operation avoids this synchronization. It is a static member of function of std::ios_base.

```
cin.tie(NULL);
```

tie() is a method which simply guarantees the flushing of std::cout before std::cin accepts an input. This is useful for interactive console programs which require the console to be updated constantly but

also slows down the program for large I/O. The NULL part just returns a NULL pointer.

Moreover, you can include the standard template library (STL) with a single include:

```
#include <bits/stdc++.h>
```

So your template for competitive programming could look like this:

```
#include <bits/stdc++.h>
using namespace std;

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

It is recommended to use cout << "\n"; instead of cout << endl;. endl is slower because it forces a flushing stream, which is usually unnecessary (See this for details). (You'd need to flush if you were writing, say, an interactive progress bar, but not when writing a million lines of data.) Write '\n instead of endl.

We can test our input and output methods on the problem INTEST – Enormous Input Teston SPOJ. Before further reading, I would suggest you to solve the problem first.

Solution in C++ 4.9.2

Normal I/O : The code below uses cin and cout. The solution gets accepted with a runtime of 2.17 seconds.

```
// A normal IO example code
#include <bits/stdc++.h>
```

```
using namespace std;
int main()
{
    int n, k, t;
    int cnt = 0;
    cin >> n >> k;
    for (int i=0; i<n; i++)
    {
        cin >> t;
        if (t % k == 0)
            cnt++;
    }
    cout << cnt << "\n";
    return 0;
}
```

Fast I/O However, we can do better and reduce the runtime a lot by adding two lines. The program below gets accepted with a runtime of 0.41 seconds.

```
// A fast IO program
#include <bits/stdc++.h>
using namespace std;

int main()
{
    // added the two lines below
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int n, k, t;
    int cnt = 0;
    cin >> n >> k;
    for (int i=0; i<n; i++)
    {
        cin >> t;
        if (t % k == 0)
            cnt++;
    }
    cout << cnt << "\n";
    return 0;
}
```

Now, talking about competitive contests like ACM ICPC, Google CodeJam, TopCoder Open, here is an exclusive code to read integers in the fastest way.

```
void fastscan(int &number)
{
    //variable to indicate sign of input number
    bool negative = false;
    register int c;

    number = 0;

    // extract current character from buffer
    c = getchar();
    if (c=='-')
    {
        // number is negative
        negative = true;
```

```
        // extract the next character from the buffer
        c = getchar();
    }

    // Keep on extracting characters if they are integers
    // i.e ASCII Value lies from '0'(48) to '9' (57)
    for (; (c>47 && c<58); c=getchar())
        number = number *10 + c - 48;

    // if scanned input has a negative sign, negate the
    // value of the input number
    if (negative)
        number *= -1;
}

// Function Call
int main()
{
    int number;
    fastscan(number);
    cout << number << "\n";
    return 0;
}
```

getchar_unlocked() for faster input in C for competitive programming

This article is contributed by **Vinay Garg.** If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Fast I/O in Java in Competitive Programming

Tips and Tricks for Competitive Programmers | Set 2 (Language to be used for Competitive Programming)

How can competitive programming help you get a job?

C++ tricks for competitive programming (for C++ 11)

Python in Competitive Programming

What is Competitive Programming and How to Prepare for It?

A Better Way To Approach Competitive Programming

Bit Tricks for Competitive Programming

What Are The Best Resources For Competitive Programming?

How to begin with Competitive Programming?

How to become a master in competitive programming?

Competitive Programming: Conquering a given problem

Knowing the complexity in competitive programming

Interactive Problems in Competitive Programming | Set 2

Some useful C++ tricks for beginners in Competitive Programming

Article Tags : C++ Competitive Programming GBlog

Practice Tags : CPP

81

2.2

To-do Done

Based on **51** vote(s)

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments