C++ Competitive Programming Note(P-1)

Ansari Abrar

August 25, 2024

Input and Output Efficiency

Input and output operations can often be a bottleneck in a program. To make them more efficient, add the following lines at the beginning of your code:

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

Using \n is faster than using `endl` because `endl` causes a flush operation.

The C functions `scanf` and `printf` are faster alternatives to C++ standard streams but are more difficult to use.

Reading Whole Lines

To read an entire input line, possibly containing spaces, use the `getline` function:

```
string s;
getline(cin, s);
```

If the amount of data is unknown, the following loop is useful:

```
while(cin >> x) {
    // code
}
```

File Input/Output

In some contest systems, files are used for input and output. An easy solution is to add the following lines at the beginning of the code:

```
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
```

This way, the program reads input from `input.txt` and writes output to `output.txt`.

C++ Code Template

A typical C++ code template for competitive programming looks like this:

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

int main() {
    // solution comes here
}
```

For taking input:

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

For showing output:

```
int a = 123, b = 456;
string x = "monkey";
cout << a << " " << b << " " << x << "\n";
```

Using `typedef`

The `typedef` keyword can be used to give a short name to a data type. For example:

```
typedef long long ll;
```

This way, instead of writing `long long` frequently, you can use the shorter `ll`.

Comparison:

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

and

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

Both are the same!

```
typedef vector<int> vi; // vector of integers
typedef pair<int,int> pi; // pair of integers
```

Here, we don't need to repeatedly type vector<int>; rather we can type vi and store integers.

Macros

Macros can be used to shorten code. A macro specifies that certain strings in the code will be changed before the compilation. They are defined using the `#define` keyword.

```
#define F first
#define S second
#define PB push_back
#define MP make_pair
```

For example, this code:

```
v.push_back(make_pair(y1,x1));
v.push_back(make_pair(y2,x2));
int d = v[i].first+v[i].second;
```

can be shortened to:

```
v.PB(MP(y1,x1));
v.PB(MP(y2,x2));
int d = v[i].F+v[i].S;
```

Another example:

```
#define REP(i,a,b) for(int i=a; i<=b; i++)
```

This loop:

```
for (int i = 1; i <= n; i++) {
    search(i);
}
```

can be shortened to:

```
REP(i,1,n) {
    search(i);
}
```