

TECHNICAL INFO SHEET

These pages contain helpful information on how to avoid slow input/output performance with C++ streams (cin / cout), how to use 64-bit data types (variables) and how to flush the output for interactive tasks.

Slow Input / Output with C++ Streams

When solving tasks with very large amounts of input / output data, you may notice that C++ programs using the cin and cout streams are much slower than equivalent programs that use the scanf and printf functions for input and output processing. Thus, if you are using the cin / cout streams we strongly recommend that you switch to using scanf / printf instead. However, if you still want to use cin / cout, we recommend adding the following line at the beginning of your program:

```
ios::sync_with_stdio(false);
```

and also making sure that you never use `endl` , but use `"\n"` instead.

Please note, however, that including `ios::sync_with_stdio(false)` breaks the synchrony between cin / cout and scanf / printf, so if you are using this, you should never mix usage of cin and scanf, nor mix cout and printf.

64-bit Data Types

For some tasks you may need to deal with numbers too large to fit in 32 bits. In these cases, you would have to use a 64-bit integer data type, such as `long long` in C/C++ or `int64` in Pascal. Here is some sample code that illustrates the usage of these data types:

C/C++

```
int main(void) {
    long long varname;

    scanf("%lld", &varname);
    // Do something with the varname variable
    printf("%lld\n", varname);
    return 0;
}
```

Pascal

```
var
    varname: Int64;
begin
    read(varname);
    { Do something with the varname variable }
    writeln(varname);
end.
```

Flushing the Output

Whenever you solve an interactive task, you always need to flush the buffer of your output after every new line printed on the output. Here is some code to illustrate how to do this under C, C++ and Pascal:

C or C++ with scanf / printf

```
fflush(stdout);
```

C++ with cin / cout

```
cout << flush;
```

Pascal

```
flush(output);
```