Home
Login
Rules
Competitions
Caia
Past contests
FAQ
Links
Contact

Technical rules

This section explains how you should write your program. Since the competition is automatically controlled by scripts, it is important that your program works exactly according to these rules.

Sending your program

To enter the contest, you must write a program that is able to play Box. You must submit the *source code* of your program on this website. The source code must consist of one single file. The maximum allowed size of the source file is `1 474 560` bytes. We will compile and run your program under Linux. Your program has to finish compiling in 5 minutes (this should be more than enough, typically compiling programs takes less than 2 seconds).

Input and output

Your program reads input from *standard input* (normally "the keyboard") and writes output to *standard output* (normally "the screen"). Your program is started once at the beginning of a game and keeps running until the end of the game.

You should strictly follow the dialogue described in the game rules. Each element of input or output is on a line by itself. You may assume that all input to your program is completely correct.

Note: when you write a move to standard output, make sure to flush the stream, so the referee is not waiting for the move. Below is explained how you can flush in some of the supported programming language.

For debugging, your program may write messages to *standard error*. Those mesages will become available to you, and only you, on the game result page. However there is a limit to the amount of characters that will be logged (approximately 10 000 characters), so you need to use it sparingly.

Programming languages

You may write your program in C/C++, Python, Java, Pascal, JavaScript, Haskell, OCaml, Go or Rust. The table below shows which compiler and configuration we are using.

Language	Compiler	Version	Command
C	GNU GCC	13.2.0	`gcc -Wall -pipe -O2 -march=native -g --std=c99 -lm`
C++	GNU GCC	13.2.0	`g++ -Wall -pipe -O2 -march=native -g --std=c++20 -lm`
Python		3.12.3	`python3`
Java	OpenJDK	21.0.4	`javac ; java -Xmx2000m -Xms2000m -XX:+UseSerialGC`
Pascal	FreePascal	3.2.2	`fpc -Sog -O2 -viwn -g -Cr-t-`
JavaScript	V8	7.1.0	`d8 --single-threaded`
Haskell	GHC	9.4.7	`ghc --make -O3`
OCaml		4.14.1	`ocamlopt -o`
Go		1.22.2	`go build`
Rust		1.75.0	`rustc -O --edition=2021 -C target-cpu=native`

C and C++

Programs written in C or C++ are compiled with GCC, and linked with the standard math library.

You can use `scanf` to read from standard input, and `printf` to write to standard output. After you write a move to standard output, your program should call `fflush(stdout);` to make sure output is written immediately. Write debugging messages to standard error, using `fprintf(stderr, "Debug info\n");`

In C++ it is also possible to use `cin`, `cout` and `cerr`. Use `cout.flush()` to flush the output buffer.

Python

Programs written in Python are run with the Python interpreter.

You can use `input()` to read from standard input, and `print` to write to standard output. After you write a move to standard output, your program should either supply `flush=True` to `print`, or call `sys.stdout.flush()` to make sure output is written immediately. Write debugging messages to standard error, using `print('Debug info', file=sys.stderr)`.

Your program may use the `numpy` module and all standard Python modules, e.g. `sys`, `re`, `time`. Your own code must be in a single file.

Note that Python programs are typically slower than programs in C, Pascal or Java. Moreover, note that Python 2 is not longer supported.

Java

Programs written in Java are compiled and run with OpenJDK.

You can use the object `System.in` to read from standard input, and

`System.out` to write to standard output. After you write a move to standard output, your program should call `System.out.flush();` to make sure output is written immediately. Write debugging messages to standard error, using `System.err.println("Debug info");`

It is possible to make more than one class for your program, but you will have to put the source for all classes in a single `.java` file (the compiler will produce multiple `.class` files anyway). When you do this, you should not declare your classes `public`, or the compiler will complain about it.

When submitting your Java program on the website, don't forget to enter the exact name of the *main class* of your program (the class which contains the `main` method).

Pascal

Programs written in Pascal are compiled with FreePascal in TurboPascal mode.

FreePascal is a lot like Turbo Pascal, but it is more powerful. You can make an array of a few megabytes size without problems. Please note that the types `Integer` and `Word` can not contain numbers larger than 32767 and 65536 respectively. It is usually better to use the type `LongInt`.

You can use `Read` and `ReadLn` to read from standard input, and `WriteLn` to write to standard output. After you write a move to standard output, your program should make a call to `Flush(Output);` to make sure output is written immediately. Write debugging messages to standard error, using `WriteLn(StdErr, 'Debug info');`

Using units in your program is not recommended. They will not help you very much and may cause problems when your program runs in the competition system. It is not possible to use units you have written yourself, since your source code must consist of a single file. Especially the CRT unit will cause problems. Do not use the CRT unit.

JavaScript

Programs in JavaScript are run with the V8 JavaScript engine.

You can use `readline` to read from standard input, and `write` or `print` to write to standard output. Note that `print` works the same as `write` except that `print` appends a newline. Write debugging messages to standard error, using the `printErr` or `console.error` function.

Runtime environment

Computer	Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz
Memory	8 GB, your program can use 2 GB
Operating system	Ubuntu 24.04.1 LTS

Time limit	30 seconds per game
Compile time limit	5 minutes (should be enough)

Your program is allowed to use at most 30 seconds per game. We measure only the time that your own program spends to select a move, not the time that your opponent spends to select a move. If your program exceeds the time limit, it loses the game and receives a penalty for irregular loss of a game.

There is no time limit per move, only the time limit per game. So your program could use 29 seconds for the first move, but then it would have to do the rest of the game in 1 second.

Remember that the CodeCup systems are 2.00GHz. If your computer at home is slower or faster, you will have to be careful. It is always a good thing to play a game at home exactly as it was played on the CodeCup system. You can see how much time had been used by your program.

To keep the contest fair, some things are not allowed:

- Your program should not read or write any files.
- It is not allowed to make network connections of any sort.
- System dependent tricks like starting other programs or creating extra processes or threads are not allowed.
- It is not possible to do calculations while it is your opponent's turn to make a move. In other words, your program can not "think in the opponent's time".