Tipps und Beispiele für das Lesen und Schreiben von Daten in C++, Java und Python

Einen Überblick zur Bedienung von DOMJUDGE finden Sie im 'Team manual' auf https://domjudge.informatik.uni-ulm.de/aud2020/public. Ergänzend dazu werden hier weitere Tipps zur Programmgestaltung gegeben.

Im Folgenden wird zunächst für die drei Sprachen jeweils ein Programm gegeben, das einen Input liest und verändert ausgibt. Input ist hier eine 2 x 2 Matrix M aus Integer. Ausgabe ist die Matrix $M + I$ (Elemente auf der Hauptdiagonalen werden inkrementiert).

Beispiel:

Input:	Output:
1 4	2 4
3 7	3 8

1 C++

```cpp
#include <iostream>
using namespace std;

// Ein- und Ausgaben können mit '>>' und '<<' in bzw. aus den
// Standardstreams std::cin und std::cout gestreamt werden.

int main() {
   int number_of_rows = 2;
   int number_of_columns = 2;
   int value;

   for(int i = 0; i < number_of_rows; i++){
      for(int j = 0; j < number_of_columns; j++){
         cin >> value;
         if(i == j) value++;
         cout << value;

         if( j+1 != number_of_columns)
            cout << ' ';
         else
            cout << '\n';
      }
   }
   return 0;
}
```

2 Java

```java
//java.util.Scanner ist nicht zu empfehlen, da er für unsere Zwecke zu langsam sein kann.
//Ein BufferedReader ist deutlich schneller.
//Wenn nicht viel ausgegeben wird kann auch System.out.print verwendet werden.
//Wird viel ausgegeben empfielt es sich auch einen BufferedWriter zu verwenden.

import java.io.*;

public class XY{

   public static void main(String []args) throws IOException{
      BufferedReader in  = new BufferedReader(new InputStreamReader (System.in ));
      BufferedWriter out = new BufferedWriter(new OutputStreamWriter(System.out));

      int number_of_rows = 2;
      int number_of_columns = 2;

      for(int i = 0; i < number_of_rows; i++){
            String[] line = in.readLine().split(" ");
            for(int j = 0; j < number_of_columns; j++){
               int value = Integer.parseInt(line[j]);
               if(i == j) value++;
               out.write(Integer.toString(value));
               if( j+1 != number_of_columns)
                  out.write(" ");
               else
                  out.write("\n");
            }
       }

       //wenn der Writer nicht geschlossen wird, kann der Output unvollständig bleiben.
       in.close();
       out.close();
   }
}
```

3 Python

```python
#stdin , stdout zu Nutzen ist meines Wissens nach die schnellste Methode für IO bei Python

from sys import stdin , stdout

number_of_rows = 2
number_of_columns = 2

for i in range(number_of_rows):
   line = stdin.readline().split()

   for j in range(number_of_columns):
      value = int(line[j])
      if i == j:
         value = value+1

      stdout.write(str(value))

      if j+1 != number_of_columns:
         stdout.write(" ")
      else:
         stdout.write("\n")
```

4 Anmerkungen zur IO

In C++ ist es am einfachsten die nächste Zahl (oder den nächsten String etc.) aus einen Stream auszulesen und man muss nicht zwingend Zeile für Zeile lesen. Bei Java und Python empfielt es sich ein Zeile zu lesen und zu splitten. IO ist bei C++ am schnellsten und mit BufferedReader/Writer auch bei Java schnell.

Bei Python wird das meiner Erfahrung nach etwas langsamer. In Domjudge bekommen Python-Lösungen daher einen kleinen Zeitbonus.

5 Umleiten von Streams auf der Kommandozeile

Die Programme sollten direkt auf der Kommandozeile ausgeführt werden können. Zum Testen bietet es sich aber manchmal an den Input in einer Datei vorzubereiten und den Output in einer anderen Datei zu speichern um ihn so zu sichern und Vergleichen zu können. Im Folgenden wird kurz erklärt wie das geht (Anleitung für die Linux-Kommandozeile).

Mit `'<'` leiten Sie eine Datei in den Standart-Inputstream und mit `'>'` leiten Sie den Standart-Outputstream in eine Datei.

Beispiel: Sie haben die Programme `test.cpp`, `test.java` und `test.py`, der Input steht in der Datei `in.txt` und der Output soll in der Datei `out.txt` gespeichert werden.

```
#C++ (g++ ist hier der Kompiler, dieser kann bei Ihnen auch aubweichen)
g++ test.cpp -o test.x  #Kompilieren
./test.x < in.txt > out.txt #Ausführen

#Java
javac test.java #Kompilieren
java test < in.txt > out.txt #Ausführen

#Python
python test.py < in.txt > out.txt #Kompilieren und Ausführen
```