

# Juízes Eletrônicos

---

Prof. Edson Alves

2018

Faculdade UnB Gama

1. Juízes Eletrônicos
2. Entrada e Saída

# Juízes Eletrônicos

---

- Juízes eletrônicos são programas que fornecem mecanismos de correção automática para problemas de programação competitiva
- A correção é feita através de testes unitários, e contempla desde a compilação e execução até a validação dos resultados de cada teste unitário
- Uma solução só é considerada correta se passar, de forma bem sucedida, pelo processo de compilação e por todos os testes unitários
- Um juiz online é uma plataforma ou site que agrega um juiz eletrônico, uma base de problemas e mecanismos de autenticação e gerência de usuários

- O URI Online Judge<sup>1</sup> é o maior juiz online do Brasil, criado e mantido pela equipe da Universidade Regional Integrada do Alto Uruguai e das Missões, que fica no Rio Grande do Sul
- Conta com mais de 1.800 problemas (2018)
- O Codeforces<sup>2</sup> é um juiz online russo, que hospeda uma série *contests* semanalmente
- Ao contrário do URI, o sistema de ranqueamento não é baseado no número de problemas acertados
- O Codeforces disponibiliza editoriais e códigos de soluções corretas para análise e estudo
- Conta com mais de 500 *rounds* (2018)

---

<sup>1</sup><https://www.urionlinejudge.com.br/>

<sup>2</sup><http://codeforces.com/>

# Feedback dos juízes eletrônicos

- A cada solução submetida por parte do usuário, o juiz retornará um *feedback* sobre a solução
- Caso a solução esteja correta, a resposta o juiz será *Accepted* (AC)
- Caso a solução esteja incorreta, será retornada uma dentre várias respostas de erro possíveis, a depender da característica do erro
- Importante ressaltar que o juiz não diz exatamente qual foi o erro, e sim uma categorização possível do erro
- Cabe ao usuário interpretar este retorno e tentar localizar e corrigir o erro antes de sua próxima submissão

## Respostas para soluções incorretas

Código	Erro	Descrição
WA	<i>Wrong Answer</i>	Uma ou mais saídas geradas estão incorretas. O juiz não informa as entradas que geraram o erro nem a resposta correta para tais entradas
PE	<i>Presentation Error</i>	As saídas do programa estão corretas, mas a apresentação (formatação, espaçamento, etc) está diferente do que foi especificado
CE	<i>Compilation Error</i>	O programa não compila corretamente. Em geral, os juízes listam os parâmetros de compilação utilizados na correção

## Respostas para soluções incorretas

Código	Erro	Descrição
RE	<i>Runtime Error</i>	O programa trava durante a execução, geralmente por conta de falhas de segmentação, divisão por zero, etc
TLE	<i>Time Limit Exceeded</i>	Os programas devem gerar as saídas válidas dentro de um limite de tempo especificado. Caso o programa exceda este tempo, esta será a resposta do juiz
MLE	<i>Memory Limit Exceeded</i>	O programa requer mais memória em sua execução do que o juiz permite



## Respostas para soluções incorretas

Código	Erro	Descrição
RF	<i>Restricted Functions</i>	O programa faz uma chamada a uma função considerada ilegal (por exemplo, <code>fork()</code> e <code>fopen()</code> )
SE	<i>Submission Error</i>	O formulário de envio da submissão tem campos vazios ou incorretos
OLE	<i>Output Limit Exceeded</i>	O programa tentou imprimir mais informações do que o permitido. Geralmente causado por laços infinitos

# Linguagens Permitidas

- Cada juiz tem um conjunto de linguagens aceitas para a resolução dos problemas
- Em geral, as linguagens aceitas são C, C++, Java e Pascal, embora alguns juízes aceitem centenas de linguagens diferentes
- Na Maratona de Programação da SBC são aceitos: C, C++, Java e Python
- Nos juízes online são listados os processos de compilação, de escrita e leitura em console e outras peculiaridades de cada linguagem
- C++ é a linguagem mais utilizada pelos maratonistas

# **Entrada e Saída**

---

# Entrada e Saída em Console

- Os problemas de programação competitiva requerem que as soluções leiam suas entradas e escrevam suas saídas em arquivos específicos
- na maioria dos casos, estes arquivos são a entrada e a saída padrão do sistema
- Cada linguagem tem mecanismos para ler a entrada e escrever na saída padrão do sistema (console)
- As entradas se encaixam em quatro categorias:
  1. Uma única instância do problema
  2.  $T$  instâncias do problema (o valor de  $T$  é dado na primeira linha)
  3.  $N$  instâncias do problema, a entrada termina com um valor sentinela
  4.  $N$  instâncias do problema, a entrada termina com fim de arquivo (EOF)

## Exemplo das 4 Categorias de Entrada

**Problema:** dados dois inteiros positivos  $X$  e  $Y$ , determine sua soma.

**Exemplos das 4 categorias de entrada:**

**1**

$X \ Y$

**2**

$T$

$X_1 \ Y_1$

$\dots$

$X_T \ Y_T$

**3**

$X_1 \ Y_1$

$X_2 \ Y_2$

$\dots$

$-1 \ -1$

**4**

$X_1 \ Y_1$

$X_2 \ Y_2$

$\dots$

$X_N \ Y_N$

# Solução C: Categoria 1

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int X, Y;
6     scanf("%d %d", &X, &Y);
7
8     printf("%d\n", X + Y);
9
10    return 0;
11 }
```

## Solução C: Categoria 2

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int T;
6     scanf("%d", &T);
7
8     while (T--)
9     {
10         int X, Y;
11         scanf("%d %d", &X, &Y);
12
13         printf("%d\n", X + Y);
14     }
15
16     return 0;
17 }
```

## Solução C: Categoria 3

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int X, Y;
6
7     while (scanf("%d %d", &X, &Y), X != -1 && Y != -1)
8     {
9         printf("%d\n", X + Y);
10    }
11
12    return 0;
13 }
```



## Solução C: Categoria 4

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int X, Y;
6
7     while (scanf("%d %d", &X, &Y) == 2)
8     {
9         printf("%d\n", X + Y);
10    }
11
12    return 0;
13 }
```

# Solução C++: Categoria 1

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     ios::sync_with_stdio(false);
8
9     int X, Y;
10    cin >> X >> Y;
11
12    cout << X + Y << endl;
13
14    return 0;
15 }
```

## Solução C++: Categoria 2

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     ios::sync_with_stdio(false);
8
9     int T;
10    cin >> T;
11
12    while (T--)
13    {
14        int X, Y;
15        cin >> X >> Y;
16
17        cout << X + Y << endl;
18    }
19
20    return 0;
21 }
```

# Solução C++: Categoria 3

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     ios::sync_with_stdio(false);
8
9     int X, Y;
10
11     while (cin >> X >> Y, X != -1 and Y != -1)
12     {
13         cout << X + Y << endl;
14     }
15
16     return 0;
17 }
```

# Solução C++: Categoria 4

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     ios::sync_with_stdio(false);
8
9     int X, Y;
10
11     while (cin >> X >> Y)
12     {
13         cout << X + Y << endl;
14     }
15
16     return 0;
17 }
```

# Solução Java: Categoria 1

```
1 import java.util.Scanner;  
2  
3 public class C1 {  
4  
5     public static void main(String[] args) {  
6  
7         Scanner scanner = new Scanner(System.in);  
8  
9         int X = scanner.nextInt();  
10        int Y = scanner.nextInt();  
11  
12        System.out.println(X + Y);  
13    }  
14  
15 }
```

## Solução Java: Categoria 2

```
1 import java.util.Scanner;
2
3 public class C2 {
4
5     public static void main(String[] args) {
6
7         Scanner scanner = new Scanner(System.in);
8
9         int T = scanner.nextInt();
10
11         for (int i = 0; i < T; ++i) {
12             int X = scanner.nextInt();
13             int Y = scanner.nextInt();
14
15             System.out.println(X + Y);
16         }
17     }
18
19 }
```

## Solução Java: Categoria 3

```
1 import java.util.Scanner;
2
3 public class C3 {
4
5     public static void main(String[] args) {
6
7         Scanner scanner = new Scanner(System.in);
8
9         while (true) {
10             int X = scanner.nextInt();
11             int Y = scanner.nextInt();
12
13             if (X == -1 && Y == -1)
14                 break;
15
16             System.out.println(X + Y);
17         }
18     }
19
20 }
```



## Solução Java: Categoria 4

```
1 import java.util.Scanner;
2
3 public class C4 {
4
5     public static void main(String[] args) {
6
7         Scanner scanner = new Scanner(System.in);
8
9         while (scanner.hasNext()) {
10             int X = scanner.nextInt();
11             int Y = scanner.nextInt();
12
13             System.out.println(X + Y);
14         }
15     }
16
17 }
```

## Solução Python: Categoria 1

```
1 line = input()
2 X, Y = [int(n) for n in line.split()]
3
4 print(X + Y)
```

## Solução Python: Categoria 2

```
1 T = int(input())
2
3 for _ in range(T):
4     line = input()
5     X, Y = [int(n) for n in line.split()]
6
7     print(X + Y)
```

## Solução Python: Categoria 3

```
1 while True:
2     line = input()
3     X, Y = [int(n) for n in line.split()]
4
5     if X == -1 and Y == -1:
6         break
7
8     print(X + Y)
```

## Solução Python: Categoria 4

```
1 while True:
2     try:
3         line = input()
4         X, Y = [int(n) for n in line.split()]
5
6         print(X + Y)
7     except EOFError:
8         break
```

# Teste das soluções

Assuma que a entrada do problema esteja no arquivo `in.txt`, e que a solução desta entrada esteja no arquivo `out.txt`.

Para gerar o arquivo `ans.txt` referente à saída produzida pela solução proposta, podemos usar um dos três comandos abaixo:

```
$ ./a.out < in.txt > ans.txt      # C/C++  
$ java Main < in.txt > ans.txt    # Java  
$ python sol.py < in.txt > ans.txt # Python
```

Para verificar se a soluções está correta, basta usar o comando `diff`:

```
$ diff ans.txt out.txt
```

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. **LAAKSONEN**, Antti. *Competitive Programmer's Handbook*, 2018.
3. **SKIENA**, Steven S.; **REVILLA**, Miguel A. *Programming Challenges*, 2003.
4. Codeforces<sup>3</sup>.
5. URI Online Judge<sup>4</sup>.

---

<sup>3</sup><http://codeforces.com/>

<sup>4</sup><https://www.urionlinejudge.com.br/>