

MARATONA

ROBÓTICA PAULA SOUZA



2018-2

CETEC – Centro Paula Souza

Atualização da Competição

Maratona de Programação 2018 – 2º Semestre

Finalidade

Este documento tem por finalidade informar aos professores responsáveis pelos times das unidades participantes da Maratona de Programação das ETECs do CPS, as mudanças implementadas nas competições, a partir da competição do 2º Semestre de 2018.

Justificativa

Devido ao aumento das equipes participantes, ocorrido na última edição da competição, e conseqüentemente o aumento do número de submissões de problemas resolvidos pelos times que e ainda, a grande dificuldade de analisar e julgar todas estas submissões pelo time de juízes alocado pela equipe de robótica, decidiu-se pela implementação da rotina de autojulgamento (Autojudge) do software BOCA, que controla a competição.

Assim, fez-se necessário mudar a forma como os dados são inseridos nos programas e como os resultados dos processamentos são mostrados para a posterior análise e julgamento. Como o autojudge do BOCA captura suas entradas pelas entradas padrões das linguagens e verifica o resultado dos processamentos pelas saídas padrões, faz-se necessário informar aos professores responsáveis pelos times e aos seus alunos sobre essa importante mudança.

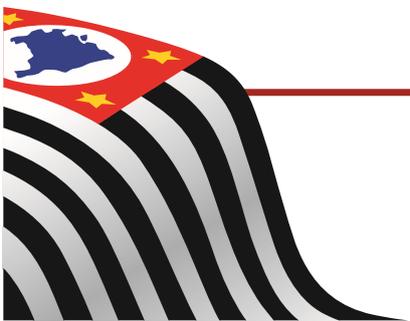
Entradas e Saídas Padrões

Vale frisar, que a partir da edição da competição do 2º semestre de 2018, os códigos que resolvem os problemas propostos, **não** mais deverão ler e nem gravar arquivos textos para as entradas e saídas, respectivamente. Tudo será feito pelas entradas e saídas padrões das linguagens.

A seguir segue um tutorial em Java de como isso irá funcionar:

Entrada padrão no Java

Existem várias formas de se ler os valores digitados em Java. A



CETEC – Centro Paula Souza

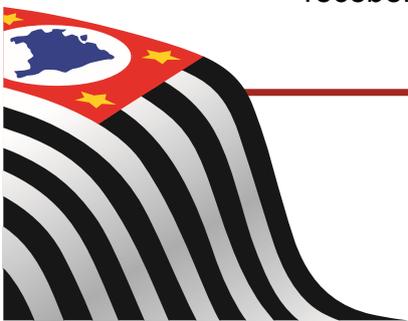
seguir mostraremos a mais comum e mais utilizada. No entanto, qualquer outra que cumpra a mesma função poderá ser utilizada.

```
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4
5 public class EntradaPadrao {
6
7     public static void main(String[] args) throws IOException {
8
9         String linha = "";
10
11         BufferedReader entrada;
12
13         try {
14
15             entrada = new BufferedReader(new InputStreamReader(System.in));
16
17             while ((linha = entrada.readLine()) != null) {
18
19                 // seu código aqui
20             }
21
22         }
23         catch (Exception e) {
24             System.out.println("Ocorreu um erro durante a leitura!\n" + e);
25         }
26
27         System.exit(0);
28     }
29 }
30 }
```

Explicando:

- Utilizamos a classe `BufferedReader` que recebe um objeto da classe `InputStreamReader`, que por sua vez recebe outro da classe `System.in`. Essa combinação nos permite capturar uma entrada, via entrada padrão (no caso o teclado) até que a tecla Enter seja pressionada. Ou seja, cada linha capturada pela rotina, tem seu final identificado pelo Enter;

- Na linha 9 criamos uma variável do tipo `String` chamada `linha`, que receberá cada uma das linhas lidas pela rotina;



CETEC – Centro Paula Souza

- Na linha 11 instanciamos um objeto da classe `BufferedReader` chamado entrada;
- Perceba que da linha 13 até a linha 25, fomos obrigados a “circundar” com `try/catch` toda a rotina de entrada pois, poderá ocorrer algum erro de leitura na entrada e isso precisa ser tratado;
- Na linha 15 iniciamos objeto entrada, criado na linha 11, como explicado no início;
- Na linha 17, iniciamos uma laço `while` que a cada volta irá atribuir à variável linha o valor lido pelo objeto entrada através do método `readLine()`. Isso permitirá que cada linha colocada na entrada padrão possa ser lida e passada à variável linha. Além disso, o laço fará a comparação do conteúdo lido para que, se o valor for nulo (`null`) o laço seja encerrado, indicando o final da leitura, ou seja, quando não houver mais nenhuma linha para ser lida na entrada;
- Perceba então, que tudo aquilo que você irá tratar dentro do seu código, deverá ser feito dentro do laço, a menos que você capture todas as entradas e as guarde em arrays ou outros objetos de coleções diferentes;
- Caso haja qualquer exceção (erro), este será captura pela rotina `try/catch` e deverá ser tratado a partir da linha 23;
- O código se encerra na linha 27 através do comando `System.exit(0)`.

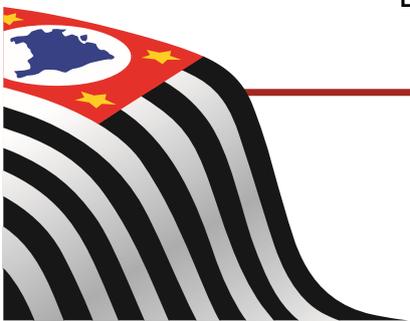
Para fazer a entrada ser inserida no código, basta copiar e colar uma sequência de texto qualquer, com uma ou várias linhas, sempre terminadas com `Enter`, e colar na entrada. No caso do NetBeans, isso deverá ser feito ao executar o código, clicando na caixa de Saída, onde aparecerá o cursor piscando, colar e dar `Enter`. A partir daí todas as linhas serão lidas e enviadas para dentro do programa. Isso vale também para aqueles que utilizarem o terminal de comandos no Windows (`cmd`).

Vale lembrar que o código que resolve o problema não deverá apresentar nenhuma interação com o usuário, como perguntas, dicas ou qualquer informação. Deverá se limitar apenas a receber o que está descrito no exemplo de entrada do problema.

Saída padrão no Java

A saída padrão no Java é ainda mais fácil. Bastando utilizar o `System.out.println()` ou `System.out.print()` quando se deseja mostrar algo pulando linha ou não.

Lembrando que a saída deve estar formatada exatamente como é



CETEC – Centro Paula Souza

indicada no problema. Com caracteres espaçados ou não e com as linhas, sempre terminadas com Enter. No caso do Java o comando `System.out.println()` já coloca o Enter ao final da linha (pular linha). Outra maneira é utilizar o caractere especial “\n” ao final de uma String.

No caso do NetBeans na caixa de saída ou da tela de terminal do Windows, pode-se ver o resultado da saída padrão.

Também vale lembrar aqui que o código que pretende resolver o problema em questão, não deverá apresentar na saída qualquer tipo de interação com o usuário, limitando-se apenas a mostrar o que se pede no exemplo de saída do problema.

Agora em Python:

Entrada padrão em Python até 2.7

Em Python é ainda mais simples, basta usar o comando `input` para valores inteiros ou `raw_input()` para strings, sem argumentos a serem mostrados:

```
varnum = input() // valores inteiros
varcar = raw_input() // valores string
```

Para converter valores de inteiro para decimais pode-se utilizar o `float()`.

Entrada padrão em Python 3.0 e posterior

No caso do Python a partir da versão 3.0, o comando `input` aceita tanto valores numéricos com alfanuméricos.

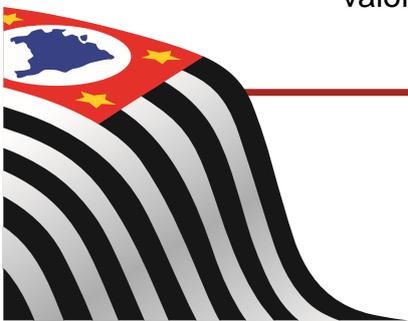
Saída padrão em Python 2.7, 3.0 e posterior

A saída padrão em Python também é bem simples, bastando usar o comando `print`:

```
print("texto aqui")
```

O `print` imprime o conteúdo entre parênteses e pula uma linha.

Lembrando que no Python utiliza-se a vírgula para concatenar valores:



CETEC – Centro Paula Souza

```
print("texto"),varnum
```

Agora vamos ver como fica tudo isso em C:

Entrada padrão em C

A função `scanf()` é utilizada para fazer a leitura de dados formatados via teclado.

```
scanf("expressão de controle", lista de argumentos);
```

Exemplo:

```
scanf("%f", &salario);
```

Explicação: este comando efetua uma leitura do teclado onde é esperada uma variável float (indicada por "%f"). O valor lido será armazenado na variável `salario`.

Na lista de argumentos devemos indicar os endereços das variáveis. Para fazer isso adicionamos o símbolo "&" como prefixo na frente do nome da variável.

Tabela - Tipos de dados básicos e representação

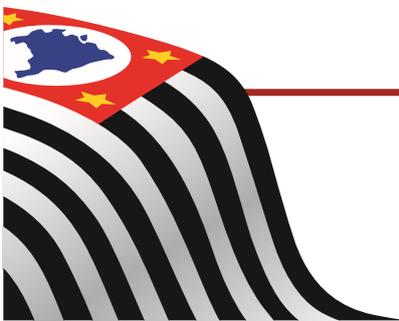
Linguagem C	Formato	Tipo de dados
<code>char</code>	<code>%c</code>	<u>caracter</u>
<code>int</code>	<code>%d</code>	<u>inteiro</u>
<code>float</code>	<code>%f</code>	<u>real</u>
<code>char[]</code>	<code>%s</code>	<u>cadeia de caracteres (string)</u>

Saída padrão em C

Chamamos de saída de dados a exibição de textos ou valores de variáveis no vídeo.

A função `printf()` é usada para exibir valores na saída padrão.

```
printf("Mensagem a ser escrita na tela");
```



MARATONA

ROBÓTICA PAULA SOUZA



2018-2

CETEC – Centro Paula Souza

Também é possível mostrar texto e valores de variáveis usando argumentos.

```
printf("Mensagem a ser escrita na tela", lista de argumentos);
```

Exemplo de mensagem que inclui o valor de uma variável:

```
printf("Total a pagar: R$ %f", total);
```

onde:

%f representa o local onde será escrita uma variável float

total é a variável float que será mostrada na posição marcada por %f

Qualquer dúvida favor entrar em contato. Obrigado!

