

Linguagem de Programação II

Prof. Antonio Carlos Sobieranski

DEC7532 | ENC | DEC | CTS



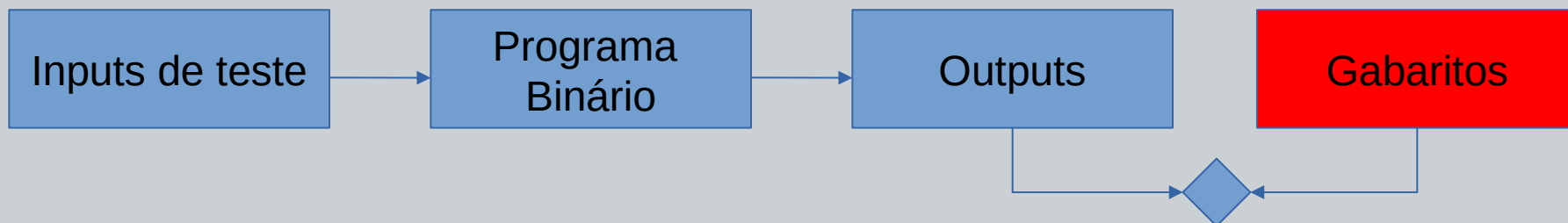
UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Metodologia para Avaliação T#s

Olá ! Aqui será descrito em detalhes como ocorrerá a avaliação automatizada dos trabalhos da disciplina.

A metodologia é baseada na ideia de comparação dos arquivos de saída gerados pelos seus programas.

Os programas deverão ler entradas (**inputs**), e fornecer saídas (**outputs com cout**) em tela conforme solicitadas – variam de acordo com o enunciado. A seguir, as saídas (**outputs**) serão comparadas automaticamente em relação aos **gabaritos**.



Metodologia para Avaliação T#s

Inputs de teste

Serão arquivos (**serão fornecido exemplos**) utilizados na forma de input de teclado sempre, e devem ser lidos com **std::cin**.

Para facilitar e evitar o tempo todo ter de entrar com os valores via teclado, usaremos o conceito de **redirecionamento de I/O**, presente nos SO's.

Programa Binário

Será compilado com compilador C++ padrão (GNUGcc ou MingW) através do comando '**g++**'. Os alunos a disciplina enviarão somente o código fonte em *zip*, nada mais. Será compilado pelo professor via script com a chamada padrão:

```
g++ *.cpp -o exe
```

Não compilou → desclassificado

Metodologia para Avaliação T#s

Outputs

Serão as saídas do programa dos alunos. Sempre em tela com **std::cout**. Devem seguir rigidamente o padrão apresentado pelo professor para cada enunciado. **Será fornecido exemplos.**

Gabaritos

Arquivos de texto de teste do professor utilizados para confrontar com as saídas dos programas dos alunos. **Alguns gabaritos serão fornecidos para cada enunciado para definir os padrões.**

Serão comparados contra as saídas programas dos alunos com um programa que calcula diferenças em arquivos. (ex.: Linux: diff / Windows: fc)

Obs.: o professor converterá a saída do programa dos alunos em arquivos de texto com **pipe (./exe > output.txt)**

Exemplo

Exemplo – Programa do Dicionário, que se encontra no Moodle

Você desenvolveu o programa do slide seguinte, que lê uma lista de palavras, carrega em memória, e faz as seguintes buscas:

- a) busca pela ocorrência exata de palavras
- b) busca por palavras que contém a substring

Exemplo

Exemplo – Programa do Dicionário, que se encontra no Moodle

O programa comunica-se através de entradas de input (**cin**)

```
asobieranski@GentooStrongX ~/Desktop
rcises/aula01-dict $ ./out
dict.txt
Trying to read dict.txt
...done
pizza
Index for pizza is 230960
Substrings found:
lipizzan
lipizzaner
lipizzaners
lipizzans
lippizzaner
lippizzaners
pizza
pizzaiola
pizzalike
pizzas
pizzaz
pizzazes
pizzazz
pizzazzes
pizzazzy
```

```
#include "dict.hpp"

int main(int argc, char* argv[])
{
    string dictionary;
    cin >> dictionary;
    vector<string> list = LoadDictionary(dictionary.c_str());
    if(list.size() == 0)
    {
        cout << "Error, file not found" << endl;
        return 2;
    }

    //searching a word in our dictionary
    string query;
    cin >> query;

    bool success = false;
    size_t index = searchWord(list, query, success);
    if(success)
        cout << "Index for " << query << " is " << index << endl;
    else
        cout << "There is no " << query << " in our dictionary " << endl;

    //how many pizzas do we have as substring ?
    cout << "Substrings found: " << endl;
    for(size_t i=0; i< list.size(); i++)
    {
        std::size_t pos = list.at(i).find(query);
        if(pos < list.at(i).length())
        {
            std::cout << list.at(i) << endl;
        }
    }

    return 0;
}
```

Exemplo

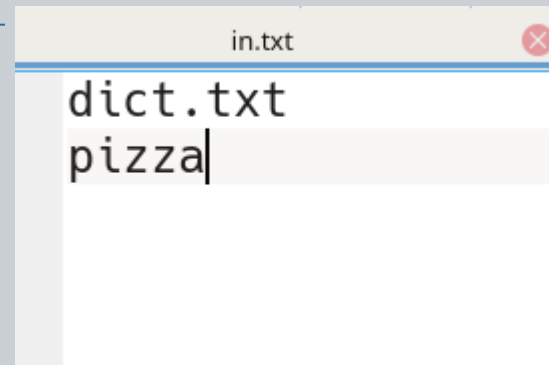
Uma vez que o professor forneceu o arquivo de teste, o mesmo pode ser utilizado para testar o programa com:

1. Input manual com o teclado como usualmente feito
- 2*. Input automatizado com **redirecionamento de I/O**

Considere o binário **out** que aguarda para realizar leitura do teclado com **std::cin**

O redirecionamento de input é realizado com '<'

```
rcises/aula01-dict $ ./out < in.txt
Trying to read dict.txt
...done
Index for pizza is 230960
Substrings found:
lipizzan
lipizzaner
lipizzaners
lipizzans
lippizzaner
lippizzaners
pizza
pizzaiola
pizzalike
pizzas
pizzaz
pizzazes
pizzazz
pizzazzes
pizzazzy
```



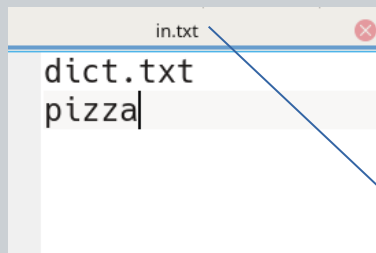
***Atenção: testes em CLI (command line interface) ou configurando na IDE (pesquisar como fazer, varia para cada IDE). Em windows, basta abrir o 'cmd' no diretório do executável gerado.**

Exemplo

Uma vez que o professor forneceu o arquivo de teste, o mesmo pode ser utilizado para testar o programa com:

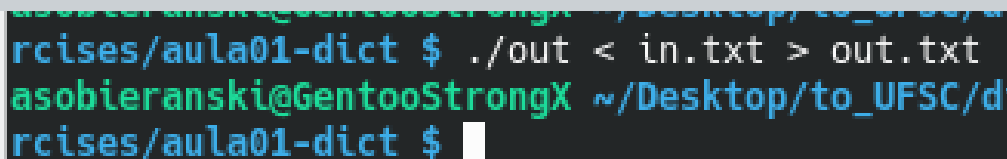
1. Input manual com o teclado como usualmente feito
- 2*. Input automatizado com **redirecionamento de I/O**

Da mesma forma, podemos utilizar **redirecionamento de output** para gerar um arquivo com a saída do programa com '>'

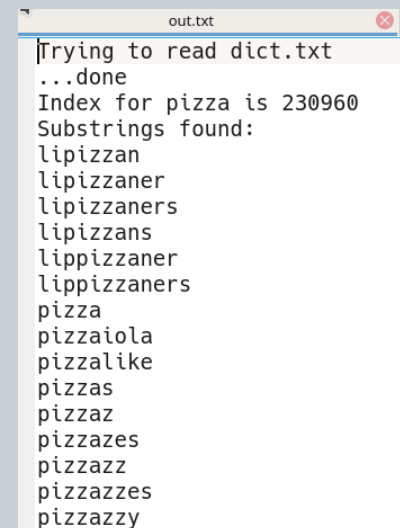


in.txt

dict.txt
pizza



```
asobieranski@GentooStrongX ~/Desktop/to_UFSC/drcises/aula01-dict $ ./out < in.txt > out.txt
asobieranski@GentooStrongX ~/Desktop/to_UFSC/drcises/aula01-dict $
```



out.txt

Trying to read dict.txt
...done
Index for pizza is 230960
Substrings found:
lipizzan
lipizzaner
lipizzaners
lipizzans
lippizzaner
lippizzaners
pizza
pizzaiola
pizzalike
pizzas
pizzaz
pizzazes
pizzazz
pizzazzes
pizzazzy

***Atenção: testes em CLI (command line interface) ou configurando na IDE (pesquisar como fazer, varia para cada IDE). Em windows, basta abrir o 'cmd' no diretório do executável gerado.**

Metodologia para Avaliação T#s

Em resumo, o sistema de autocorreção consiste em compilar o programa dos alunos com:

```
g++ *.cpp -o exe
```

E testar o programa com um input **in.txt** e gerar um arquivo de saída do programa **out.txt**:

```
./exe < in.txt > out.txt
```

E comparar em relação aos gabaritos (um gabarito para cada entrada)

Os alunos enviarão somente código fonte, ou arquivo compactado contendo todos os *sources*, e no mesmo diretório, (hpp e cpp)

Contato

Prof. Antonio Carlos Sobieranski – DEC | A316

E-mail: a.sobieranski@ufsc.br

Inst: @antonio.sobieranski



UNIVERSIDADE FEDERAL
DE SANTA CATARINA