Universidade Federal do Rio de Janeiro
Escola Politécnica
Departamento de Engenharia Eletrônica e de Computação

Trabalho de Linguagens de Programação

Alunos	Diego Rodrigues
	Raiano Martins
Professor	Miguel Elias Mitre Campista

Rio de Janeiro, 15 de Outubro de 2021

Conteúdo

1 Introdução

Este programa será avaliado como trabalho final para a disciplina Linguagens de Programação no período de 2021-1 remoto ministrada pelo professor Miguel Elias Mitre Campista. O software consiste em um programa com uma interface em C++ que realiza operações com matrizes em Python, a realização das operações é feita por meio de scripts externos que retornam os resultados para a interface.

2 Objetivos

O objetivo desse projeto é implementar uma calculadora que realiza operações básicas com matrizes de números reais com tamanhos variados, utilizando as bibliotecas já consolidadas na linguagem Python, que devido a seu maior nível de abstração tornam tais operações mais simples de serem programadas.

Como adicional o programa também faz gráficos de funções inseridas pelo usuário através de vetores armazenados em arquivos. Deste modo a integração entre as duas linguagens permite o programador executar tarefas específicas com uma maior facilidade sem precisar escrever um programa inteiro na outra linguagem, portanto mantendo as vantagens das duas linguagens.

3 Caractéristicas e Utilização

3.1 Compilação

A compilacao deve ser feita em um ambiente Linux por meio do Makefile, o usuário digita o comando "make all" então o Makefile realiza a linkedição e compilação e cria um arquivo executável chamado "calculadora_matrizes.exe".

3.2 Utilização

O programa de interface foi escrito em linguagem C++ com o uso de classes, herança e polimorfismo, para os scripts Python externos as operações foram realizadas com funções importadas das bibliotecas Numpy e Matplotlib. A utilização é bem simples, o programa recebe a opção escolhida pelo usuário através de um menu.

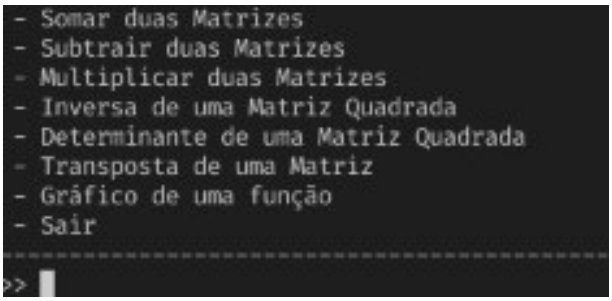

Figura 1: Menu Principal

Para cada opção o programa requesita ao usuário os nomes dos arquivos com os dados das matrizes, o usuário então entra com nomes válidos de arquivo (para nomes inválidos o programa acusa erro e retorna para o menu principal).

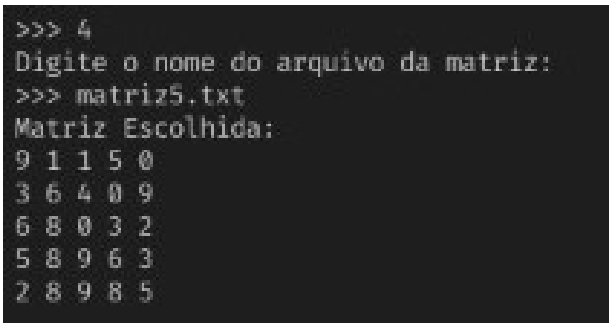

Figura 2: Opção do menu

Consequentemente o script Python realiza as operações e retorna um arquivo com o nome de "result.txt" como resultado, em seguida o programa da interface lê este arquivo com o resultado e armazena em um objeto de modo que consiga retornar ao usuário imprimindo-o na tela.

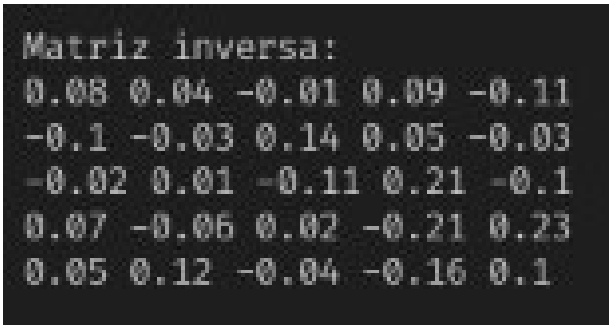

Figura 3: Resultado

A alternativa do menu para imprimir o gráfico de uma função usa dados de um arquivo csv onde na primeira coluna estão os valores do eixo das abscissas e na segunda coluna das ordenadas. O script então importa funções da biblioteca Matplolib e gera uma janela com o gráfico da função e algumas opções já definidas na biblioteca o programa é então posto em segundo plano e fica aguardando até que o gráfico da função seja encerrado para retornar ao menu principal.

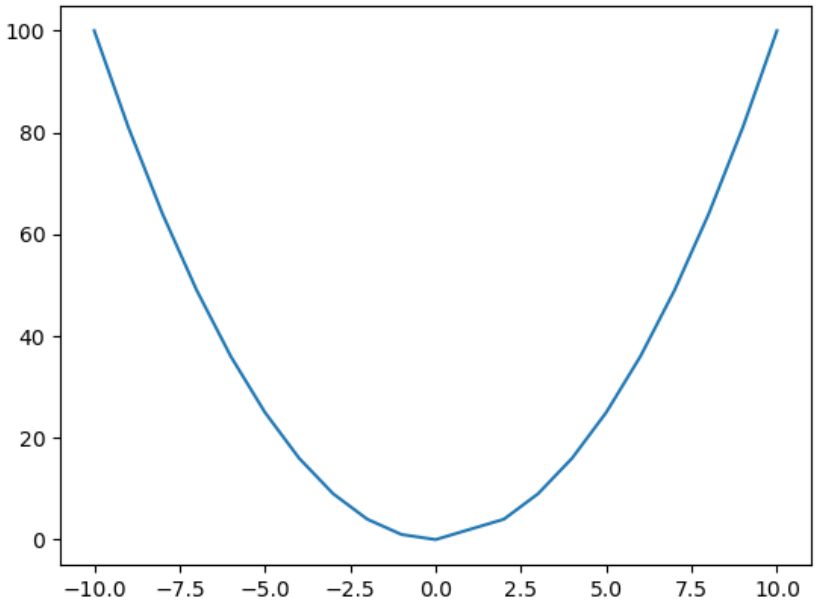

Figura 4: Gráfico da função

Uma validação para a opção que foi inserida do menu foi implementada permitindo que o usuário só consiga prosseguir depois de ter digitado uma opção válida.

4 Implementação

4.1 Arquivos pyClass

4.1.1 Classes e métodos

Classe CPyInstance:
Possui apenas um construtor que inicializa uma instância de Python e um destrutor que a finaliza.
Classe PyArray:
É uma classe derivada de CPyInstance. Possui atributos privados para armazenar o nome do arquivo associado à matriz e um vector< *vector* < *double* >> para armazenar os valores da

matriz. Além disso possui métodos para leitura do arquivo e para adicionar elementos à matriz.
Classe PyFunction:
Similar à classe PyArray com a diferença de a função ser armazenada em um vector< $pair <$ $double, double >>$.

4.1.2 Funções

void runPyScriptArgs(const char *file, int argc, char *argv[]):
Serve para manipular or argumentos que serão passados ao scripts Python e definir os seus tamanhos em memória.

4.2 Arquivos functions

4.2.1 Funções

template $< class\ T >$
void printVector(vector $< vector < T >>$ const &matrix)
template $< class\ T >$
void printFunction($vector < Pair < T, T >>$ const &matrix):
São funções Template que servem para imprimir o conteúdo da matriz e função (discreta) respectivamente. Apenas percorrem a STL recebida como argumento através de um for e imprimem na tela os valores.

4.3 Arquivo add.py

array_1 = np.loadtxt(sys.argv[1], dtype=float, delimiter=' ')
array_2 = np.loadtxt(sys.argv[2], dtype=float, delimiter=' ')

Lê um arquivo de texto e retorna uma matriz. Seu primeiro argumento é o nome do arquivo de texto a ser lido, o segundo argumento é o tipo de dados da matriz e o terceiro argumento é o delimitador que separa os elementos da matriz no arquivo texto.

result = add(array_1, array_2)

Armazena a adição das duas matrizes passadas como argumento.

np.savetxt("result.txt", result, fmt='%.2f', delimiter=' ', newline='\n')

Armazena o resultado em um arquivo, o primeiro argumento é o nome do arquivo, o segundo a variável com o resultado a ser armazenado, o terceiro as casas decimais mostradas, o quarto o delimitador entre os itens e o quinto argumento o marcador para pular linha.

4.4 Arquivo det.py

array = np.loadtxt(sys.argv[1], dtype=float, delimiter=' ')

Lê um arquivo de texto e retorna uma matriz. Seu primeiro argumento é o nome do arquivo de texto a ser lido, o segundo argumento é o tipo de dados da matriz e o terceiro argumento é o delimitador que separa os elementos da matriz no arquivo texto.

result = det(array)

Armazena o determinante da matriz passada como argumento.

np.savetxt("result.txt", result, fmt='%.2f', delimiter=' ', newline='\n')

Armazena o resultado em um arquivo, o primeiro argumento é o nome do arquivo, o segundo a variável com o resultado a ser armazenado, o terceiro as casas decimais mostradas, o quarto o delimitador entre os itens e o quinto argumento o marcador para pular linha.

4.5 Arquivo inv.py

array = np.loadtxt(sys.argv[1], dtype=float, delimiter=' ')

Lê um arquivo de texto e retorna uma matriz. Seu primeiro argumento é o nome do arquivo de texto a ser lido, o segundo argumento é o tipo de dados da matriz e o terceiro argumento é o delimitador que separa os elementos da matriz no arquivo texto.

result = inv(array)

Armazena a inversa da matriz passada como argumento.

np.savetxt("result.txt", result, fmt='%.2f', delimiter=' ', newline='\n')

Armazena o resultado em um arquivo, o primeiro argumento é o nome do arquivo, o segundo a variável com o resultado a ser armazenado, o terceiro as casas decimais mostradas, o quarto o delimitador entre os itens e o quinto argumento o marcador para pular linha.

4.6 Arquivo mult.py

array = np.loadtxt(sys.argv[1], dtype=float, delimiter=' ')

Lê um arquivo de texto e retorna uma matriz. Seu primeiro argumento é o nome do arquivo de texto a ser lido, o segundo argumento é o tipo de dados da matriz e o terceiro argumento é o delimitador que separa os elementos da matriz no arquivo texto.

result = mult(array)

Armazena a multiplicação da matriz passada como argumento.

np.savetxt(”result.txt”, result, fmt=’%.2f’, delimiter=’ ’, newline=’\n’)

Armazena o resultado em um arquivo, o primeiro argumento é o nome do arquivo, o segundo a variável com o resultado a ser armazenado, o terceiro as casas decimais mostradas, o quarto o delimitador entre os itens e o quinto argumento o marcador para pular linha.

4.7 Arquivo operations.py

def add(array_1, array_2)
def sub(array_1, array_2)
def mult(array_1, array_2)
def inv(array)
def det(array)
def transp(array)

Definição das funções das operações entre as matrizes.

4.8 Arquivo sub.py

array_1 = np.loadtxt(sys.argv[1], dtype=float, delimiter=’ ’)
array_2 = np.loadtxt(sys.argv[2], dtype=float, delimiter=’ ’)

Lê um arquivo de texto e retorna uma matriz. Seu primeiro argumento é o nome do arquivo de texto a ser lido, o segundo argumento é o tipo de dados da matriz e o terceiro argumento é o delimitador que separa os elementos da matriz no arquivo texto.

result = sub(array_1, array_2)

Armazena a subtração das duas matrizes passadas como argumento.

np.savetxt(”result.txt”, result, fmt=’%.2f’, delimiter=’ ’, newline=’\n’)

Armazena o resultado em um arquivo, o primeiro argumento é o nome do arquivo, o segundo a variável com o resultado a ser armazenado, o terceiro as casas decimais mostradas, o quarto o delimitador entre os itens e o quinto argumento o marcador para pular linha.

4.9 Arquivo transp.py

np.loadtxt(sys.argv[1], dtype=float, delimiter=’ ’)

Lê um arquivo de texto e retorna uma matriz. Seu primeiro argumento é o nome do arquivo de texto a ser lido, o segundo argumento é o tipo de dados da matriz e o terceiro argumento é o delimitador que separa os elementos da matriz no arquivo texto.

result = transp(array)

Armazena a transposta da matriz passada como argumento.

np.savetxt("result.txt", result, fmt='%.2f', delimiter=' ', newline='\n')

Armazena o resultado em um arquivo, o primeiro argumento é o nome do arquivo, o segundo a variável com o resultado a ser armazenado, o terceiro as casas decimais mostradas, o quarto o delimitador entre os itens e o quinto argumento o marcador para pular linha.

4.10 Arquivo main

string entry;
Entrada digitada pelo usuário.
uint8_t option;
Entrada digitada pelo usuário após verificação de tipo, para tratamento de opção inválida. Opção que será passada ao switch para escolha de opção do menu.
int py_argc = 3;
Número máximo de argumentos passados ao script Python.
char *py_argv[py_argc];
Vetor de strings que serão passadas como argumentos ao script Python.
string firstEntry, secondEntry;
Nomes de arquivos passados pelo usuário ao programa.
PyArray *matriz_1, *matriz_2, *resultado;
Objetos em C++ para receber as matrizes.

4.11 Arquivo consts.h

Arquivo com a definição de algumas constantes.

5 Conclusão

Podemos constatar que é possível integrar duas linguagens que funcionam com sintaxes distintas em um mesmo programa e que as funções que fazem as operações em matrizes e a plotagem do gráfico são muito mais simples em Python em comparação ao C++. Cabe mencionar também que o programa fez a troca de informações entre as linguagens por meio de arquivos, porém poderia ter sido implementado, de uma forma mais eficiente, através de funções, contudo o programa obteve os resultados que se esperavam.