Tabalho de Polinômios

Você deve implementar um conjunto de programas que realizam operações com polinômios que são representados como arquivo.

Formato do arquivo

Cada polinômio é expresso como um arquivo binário formado por um código id do tipo de arquivo, aqui definido por 4 caracteres com o valor `"poly"`, um número inteiro p, indicando a potência máxima do polinômio, seguido de p `doubles`, um vetor de coeficientes coef, cada elemento representando um coeficiente do polinômio. Esses coeficientes estão dispostos em ordem crescente de potência.

Assim, define-se o seguinte formato de arquivo com a estrutura:

- id: 4 `bytes` com o conteúdo *'p', 'o', 'l' e 'y'*, representando que o arquivo em questão é para armazenar polinômios;
- p: 1 `int` para armazenar a potência máxima do polinômio representado;
- coef: vetor com p elementos do tipo `double`.

A estrutura a ser usada deve ser a seguinte:

```
#define ID_SIZE 4 typedef struct { char code[ID_SIZE]; int p;
double* coef; } Poly;
```

Programas a serem implementados:

1. **pbuild**

   Este programa recebe, via linha de comando, uma `string` contendo um polinômio e o nome do arquivo que ele deve ser armazenado. O uso desse programa deve seguir o seguinte padrão:

   ```
   pbuild <poly_string> <file_name> <poly_string>: Lista de pares de números que definem o coeficiente e a respectiva potência Exemplo: "2x^0+0.2x^4" <file_name>: nome do arquivo Exemplo: "polyA"
   ```

   Exemplos de execução:

   ```
   $./pbuild "2x^0+0.2x^4-1x^2" polyA
   ```

   O comando acima cria o arquivo `polyA` com o polinômio $2 - x^2 + 0.2x^4$.

2. **pview**

   Este programa recebe uma `string` com o nome do arquivo que contém um polinômio e o apresenta no terminal. O uso desse programa deve seguir o seguinte padrão:

   ```
   pview <file_name> <file_name>: nome do arquivo
   ```

   Exemplos de execução:

   ```
   $./pview p1 $ 2-x^2+0.20x^4
   ```

   Este exemplo mostra a execução do programa `pview` que recebe o nome do arquivo de polinômio `p1` já existente na pasta corrente e o apresenta no terminal.

3. **psum, psub**

   Estes programas realizam a soma e a subtração de dois polinômios e salva o resultado em um arquivo. Ele recebe o nome de três arquivos, sendo os dois primeiros os arquivos que contém os polinômios a serem somados e o último nome representa o nome do arquivo resultante. O uso desse programa deve seguir o seguinte padrão:

```
psum <file_name_1> <file_name_2> <file_name_out> psub
<file_name_1> <file_name_2> <file_name_out> <file_name_1>: nome
do arquivo do primeiro polinômio <file_name_2>: nome do arquivo
do segundo polinômio <file_name_out>: nome do arquivo para o
polinômio resultante
```

Testando os seus programas

Seu trabalho será testado por um `script bash` que realiza uma sequência de comandos no terminal chamando os programas que você codificou com os argumentos apropriados. Abaixo segue um exemplo de `script bash` para teste do seu conjunto de programas:

```
#!/usr/bin/bash ./pbuild "2x^2-2x^0" p1 ./pview p1 ./pbuild "2x^1-
2x^0" p2 ./pview p2 ./psum p1 p2 soma ./psub p1 p2 sub ./pview soma
./pview sub
```

Aquivo teste.sh que contém um exemplo de `script` para teste. Para executar esse script no terminal é necessário torná-lo executável com o comando `chmod a+x teste.sh` . Feito isso, o arquivo pode ser executado assim como outro programa via terminal: `./teste.sh`

Entrega do trabalho

O trabalho é individual e substitui a nota da prova-05 que cobre o conteúdo de **Arquivos**. O trabalho deve ser entregue em um arquivo zipado (.zip) contendo os códigos de cada programa até o dia **14 de Junho de 2021** via o link de submissão no SIGAA.

Dicas e observações

1. Fique a vontade para colocar mensagens de erro que desejar
2. Lembre-se que todo dado é passado por linha de comando. Assim, não há necessidade de leitura do terminal, ou seja, não é necessário o uso do `scanf`
3. Para converter `strings` em double você pode usar a função que implementou nas listas de exercícios ou usar a função `atof` presente na `stdlib.h`
4. O desenvolvimento de todos os programas fica mais fácil se você construir uma biblioteca de polinômios, formada pelos arquivos `poly.h` e `poly.c`. Desse modo, para cada programa a ser implementado, você inclui o seu `header` e chama as funções apropriadas