Segundo Trabalho de Implementação

Algoritmos e Estrutura de Dados II

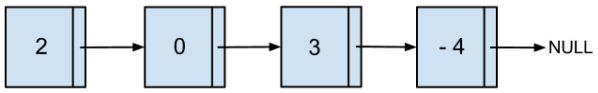
O objetivo deste trabalho é implementar uma biblioteca de operações matemática sobre polinômios. A implementação deve usar necessariamente listas encadeadas. As operações a serem implementadas são soma, subtração, produto, derivada, integral (primitiva) e avaliação em um ponto.

Estrutura de Dados

Cada polinômio será representado por uma referência (ponteiro) para uma *lista encadeada*, como definido no arquivo `polinomio.h`:

```
typedef lista *polinomio;
```

Os nós estão ordenados em ordem crescente de grau. Todos os termos do polinômio são reprentados na lista, mesmo os termos com coeficientes igual a zero. Assim, **por exemplo**, o polinômio $2x^3 + 3x - 4$ teria a **possível** representação gráfica:



Operações

A biblioteca das operações está definida no arquivo `polinomio.h`. As operações são as seguintes:

`polinomio soma(polinomio p, polinomio q)` : soma dois polinômios, devolvendo o resultado em outro polinômio.

`polinomio subtrai(polinomio p, polinomio q)` : subtrai dois polinômios, devolvendo o resultado em outro polinômio. Note que a subtração, diferente da soma, não é comutativa.

`polinomio multiplica(polinomio p, polinomio q)` : multiplica dois polinômios, devolvendo o resultado em outro polinômio.

`polinomio derivada(polinomio p)` : calcula a derivada de um polinômio, devolvendo o resultado em outro polinômio.

`polinomio integra(polinomio p)` : calcula a integral (primitiva) de um polinômio, devolvendo o resultado em outro polinômio.

Após calcular a integral, o termo de menor grau do resultado é uma constante c arbitrária. Para fins de correção do trabalho, assuma que $c = 0$ (Atenção: $c \neq 0$ pode levar à correção incorreta do trabalho).

`float avalia(polinomio p, float x)` : avaliação o polinômio p em um ponto x, ou seja, devolve $p(x)$.

Também deve-se implementar outras funções (não relacionadas com operações sobre polinômios):

`polinomio constroi_polinomio(char *s)` : Essa função recebe uma string, que representa o polinômio, e devolve um polinômio. O formato da string é simples: são k números float separados por espaço, que representam os coeficientes do polinômio (de grau $k-1$), onde o primeiro número representa o coeficiente de termo de grau $k-1$, o segundo representa o coeficiente de termo de grau $k-2$ e assim por diante. Por exemplo, se o parâmetro de entrada da função for a string `‘‘5 0 3 0 1’’`, então deve-se contruir o polinômio $5x^4 + 3x^2 + 1$.

`void destroi_polinomio(polinomio p)` : Função que desaloca a memória ocupada pelo polinômio que foi passado por parâmetro.

`polinomio escreve_polinomio(polinomio p)` Função que imprime em tela o polinômio que foi passado por parâmetro. O formato da impressão deve seguir o seguinte exemplo: se queremos imprimir o polinômio $5x^4 + 3x^2 + 1$ então a saída em tela será `5x^4 + 3x^2 + 1`. Note neste exemplo que, independente da sua implementação interna armazenar ou não os 0's dos coeficientes dos termos x^3 e x^1, eles **não** devem aparecer na impressão em tela. Por exemplo, está errado essa saída: `5x^4 + 0x^3 + 3x^2 + 0x^1 + 1`. Uma observação parecida existe também para o termo x^0, onde **não** deve-se imprimir `x^0`. Por exemplo, `5x^4 + 3x^2 + 1x^0` está errado.

Detalhes de Implementação

São fornecidos 6 (seis) arquivos: `lista.h`, `lista.c`, `polinomio.h`, `polinomio.c`, `teste.c` e `Makefile`.

Os arquivos que **devem** ser preenchido e entregues (faz parte da avaliação) são **três**:

- `lista.h`
- `lista.c`
- `polinomio.c`

Os arquivos `lista.h` e `lista.c` devem **obrigatoriamente** conter as funções e implementações relacionadas à estrutura de dados de lista ligada. Ambos arquivos tem implementação livre, ou seja, você pode definir as funções que quiser, e também implementar como quiser.

O arquivo `polinomio.h` **não deve ser modificado**, pois as declarações (protótipos) das funções nele contidas serão usadas na bateria de testes que o programa será submetido.

O arquivo `polinomio.c` deve ser implementado de acordo com as funções já definidas em `polinomio.h`. Funções adicionais relativas à polinômios podem ser adicionadas a `polinomio.c` (mas NUNCA a `polinomio.h`). Por exemplo, uma função adicional que pode ser útil é pra devolver o *grau* de um polinômio.

O arquivo `teste.c` é um exemplo de bateria de testes que será executada. É **altamente recomendado** fazer seu programa funcionar para essa bateria de testes, mas sabendo que a bateria de teste que o programa será submetido pode ser trocada.

O arquivo `Makefile` é usado para compilar o programa.

O trabalho deve ser feito de forma que possa ser compilado e executado nos computadores do laboratório do Departamento de Informática.

O que você deve entregar

Voce deve entregar os arquivos `lista.h`, `lista.c` e `polinomio.c`. Esses arquivos devem ser compactados num arquivo `.tar.gz` (detalhes abaixo).

Forma de Entrega

O trabalho pode ser feito em grupos de até dois alunos.

Os arquivos devem ser empacotados em um arquivo `grr1-grr2.tar.gz`, onde `grr1-grr2` é uma string com os GRR's dos integrantes da equipe. Ao descompactar este arquivo deverá ser criado um diretório de nome grr1-grr2 que conterá todos os demais arquivos.

Este arquivo deve ser enviado como anexo por e-mail ao endereço do professor com o assunto "CI056-trab2" (**exatamente**).

Data de Entrega

O trabalho pode ser entregue até às 23h59m da data estipulada no site da disciplina. Entregas feitas após esse prazo **não serão** avaliadas (recebem nota zero).

Critério de Avaliação

Os critérios de avaliação são os seguintes:

- O trabalho deve ser entregue no formato que foi especificado, assim como explicado na Seção "Forma de Entrega".
- O trabalho deve compilar e executar sem problemas, inclusive sendo compilado com Makefile e podendo ser testado com o arquivo `teste.c`.
- O trabalho está correto, ou seja, as implementações das operações devolvem resultados iguais às definições matemáticas correspondentes para cada operação sobre polinômios.
- Clareza do código.