AContas

Nome: Matheus Santos de Moura
Matrícula: 2020102131

Relatório sobre o desenvolvimento do software AContas

A proposta do trabalho foi criar um programa que, lesse uma entrada em forma de arquivo .txt, criasse árvores binárias com as operações, imprimisse um resultado em um arquivo .txt e uma estrutura gráfica que será usada no Graphviz Online, para visualização.

No início do trabalho, o primeiro desafio foi criar a função que faria a leitura de uma árvore e imprimisse o resultado. Para criação das árvores manualmente, foram usadas funções passadas em sala de aula e aplicadas em exercícios passados, como *arv_cria()*, *arv_criaVazia()* etc… Com as árvores prontas, foi necessário criar uma função para calcular o resultado de um nó de operação, que tem, 2 nós filhos folhas, e apenas nesse caso, para isso foi criado a função *arv_calcula()*, concluído os testes, foi necessário aplicar essa função para um nó raiz, que pode ter nós operações com folhas operações, para isso foi criada a função *arv_calcula_raiz()*, que usa recursividade para obter o resultado final de uma árvore raiz, atribuindo o resultado dos nós operações ao próprio nó, modificando a árvores até que não haja filhos para serem calculados.

No início do processo, eu havia definido a estrutura da seguinte forma:

```
struct arv {
    void * cont;
    Arv * esq;
    Arv * dir;
};
```

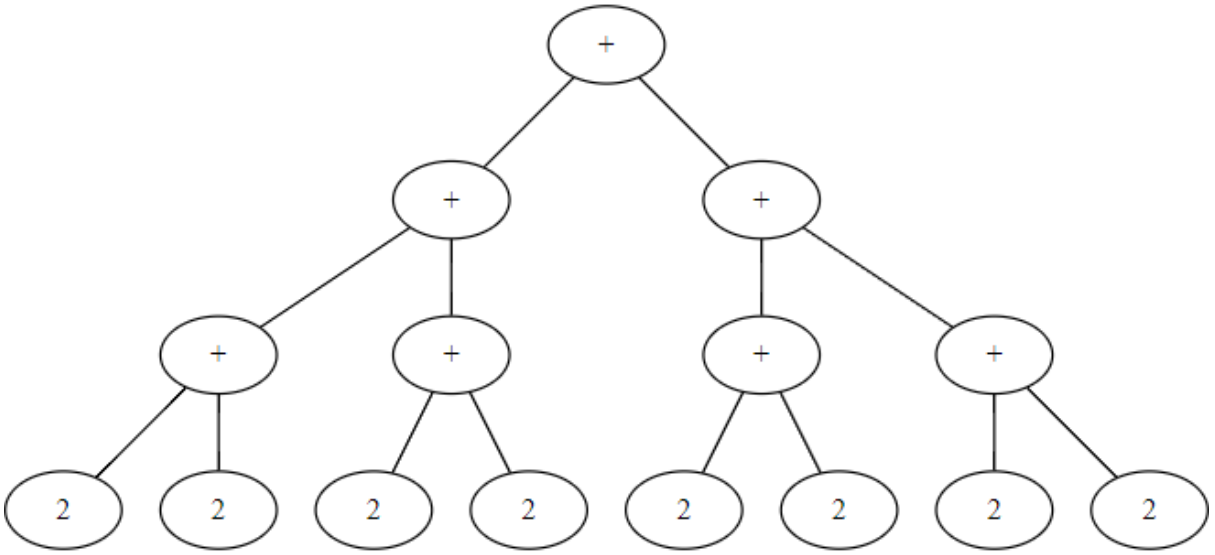
E logo depois fiz a seguinte mudança:

```
struct arv {
    char op; //Guarda a operação que o nó representa;
    float num; // Guarda o número que o nó representa;
    Arv * esq; // Aponta para a árvore á esqueda do nó;
    Arv * dir; // Aponta para o árvore á direita do nó;
    int pos; //Inidca a posição dele na leitura dos arquivos;
};
```

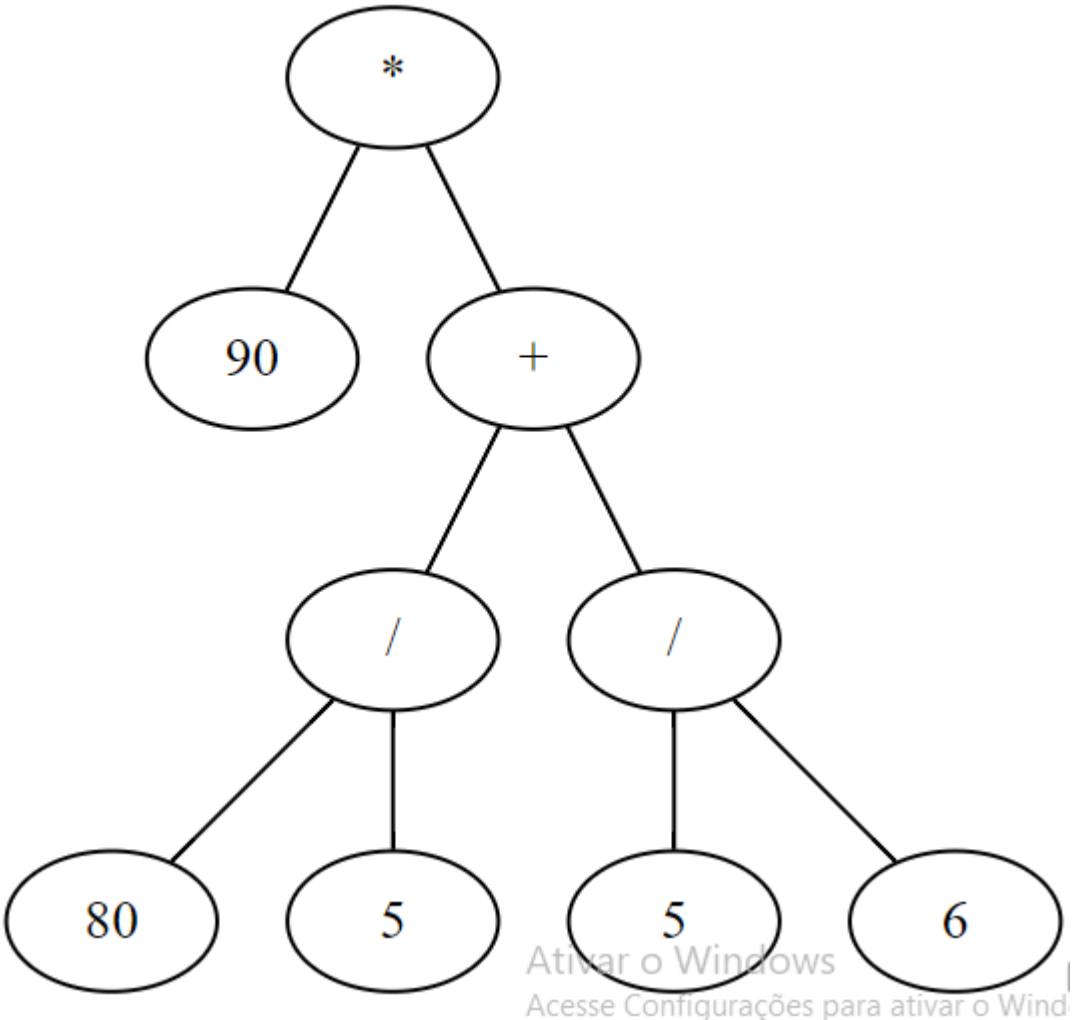
Sob a dúvida de qual seria a melhor forma, fui aconselhado pelos monitores que o void* pode acarretar erros no Valgrind, que culminam em perda de pontos, por isso optei pela struct com 2 variáveis, mesmo que uma delas fique inútil, na falta de proposta melhor, tanto por minha parte como pelo dos monitores. Nesse ponto ainda

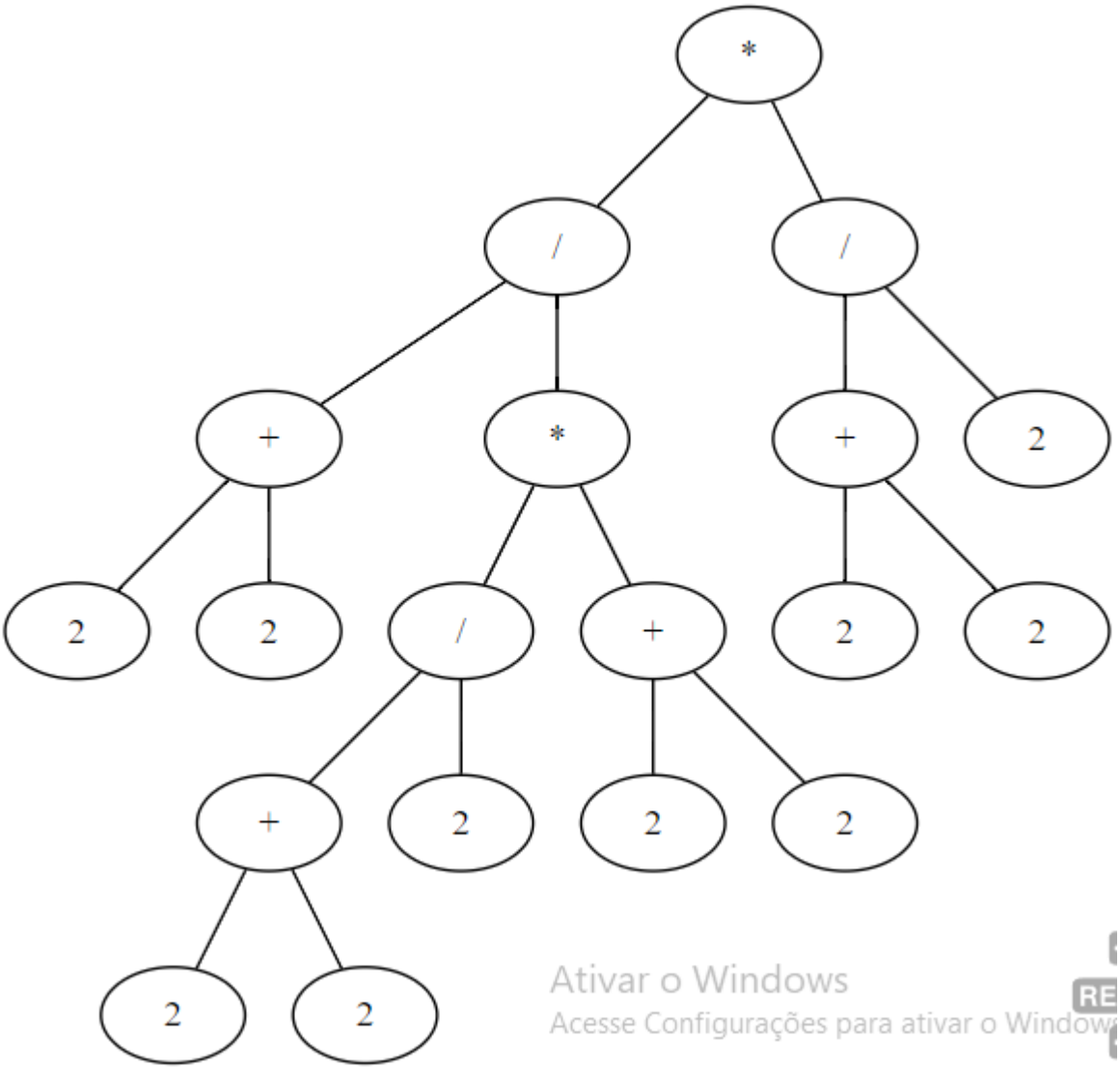
não havia definido a variável que guardaria a posição para ser usada na função que imprime a árvore.
Concluído esse desafio, parti para implementação da função que faria a leitura dos dados do *entrada.txt*, de começo foi bem difícil conceber a ideia e ajustar para que a leitura fosse precisa, mas após muitas tentativas, consegui criar um algoritmo para sanar esse problema, ignorando o primeiro parêntese, a cada novo parêntese (, é criado um nó, à esquerda do nó pai, e, após a função retornar com recursividade, para o nó pai, lê o conteúdo (operação ou número), e, lendo o próximo (, inicia o nó da direita do nó pai, da função recursiva anterior à atual, isso se repete até que o nó esteja preenchido ou que o nó seja folha (número), criando assim a árvore com os nós corretamente. Em seguida, precisou-se criar a função de leitura e escrita para o programa , que lesse todas as linhas e imprimesse todas as respostas (até o momento as funções faziam apenas a leitura e cálculo de uma linha, por cada vez que fosse chamada), pra isso foi implementada a função *leEntradaEPrintaSaida()*, que faz a leitura dos dados com um loop e imprime logo em seguida.
Concluído essa parte, o próximo desafio foi criar o arquivo graphviz.txt, conteudo a construção gráfica das árvores raiz das operações, essa parte foi bem simples, usando recursividade, imprime o nó no estilo *nolabel=””*, onde há um número identificador e o conteúdo do nó, seja ele número ou operação, em seguida, a relação com o nó da esquerda do nó pai e imprime o próprio conteúdo desse nó filho (junto com seus respectivos nós à esquerda e à direita até achar uma folha), logo em seguida immindo a relação do nó com o nó da direita e repetindo mesmo processo do nó da esquerda.
Um desafio encontrado nessa etapa foi para como numerar os nós sem que a recursividade interferisse que valor (reiniciando a variável de posição), para resolver esse problema, foi criada uma variável global, que pode ser acessada em qualquer escopo do arquivo para ser zerada, com autorização da professora.
Concluído isso, foi feita uma alteração nos valores da estrutura dos nós, para que comportem floats, podendo realizar assim operações com números racionais, e também uma alteração na saída, para que a impressão respeite os padrões estabelecidos nos casos de teste.

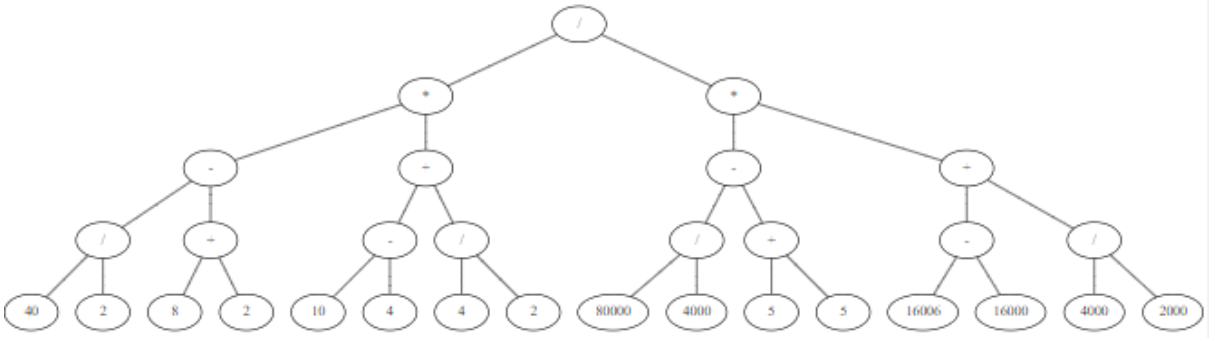
Segue algumas imagens de árvores que construí como caso de teste:



(maior árvore construída para testes)







No geral o trabalho foi relativamente tranquilo, me ajudou muito a fixar o conceito de recursividade, que inicialmente apresentei dificuldade, mesmo estando relativamente familiarizado com esse conceito, vindo de uma bagagem de JavaScript que tenho, e gostei bastante do resultado visual, que exemplificou como funciona a estrutura das árvores binárias na prática.

- StackOverflow
- Graphviz Online
- Aulas de ED
- Monitoria de ED