**Relatório do Projeto I de Estrutura de Dados**

*Alunos: Victor Botin Avelino (18172) e Pedro Gomes Moreira (18174)*

**Introdução:**

O relatório descrito aqui reflete o trabalho de nossa dupla no projeto número I, referente a matrizes esparsas. O que consta neste documento engloba nossos dias de desenvolvimento do código mais os erros e dificuldades encontradas no decorrer do preso.

**Desenvolvimento:**

* **14/03:** criação do projeto no GitHub, commit inicial com enunciado, relatório e programa;
* **21/03:** início e fim da classe *Celula*;
* **22/03:** começo da parte visual;
* **25/03:** começo do desenvolvimento da classe *ListaCruzada*;
* **26/03:** método de inclusão de uma nova célula e conserto dos erros por distração;
* **27/03:** desenvolvimento do método *Exibir()* e alteração em alguns pontos da interface;
* **28/03:** método de leitura do arquivo texto que contém a matriz a ser analisada, bem como mais alterações visuais e o método *ValorDe()* para procura pelo botão “Pesquisar”. Depois, fizemos os métodos *Excluir(), Limpar()* e prosseguimos nas alterações visuais;
* **02/04:** esqueleto dos métodos *SomaMatriz()*, *MultiplicacaoMatriz()* e *SomarEmColuna()*, faltando apenas a codificação interna, além de escrever cabeçalho do *DataGridView* no método *Exibir()*;
* **03/04:** desenvolvimento do método *SomarEmColuna()* e criação da classe *CelulaInexistenteException*, disparada quando a célula não existe;
* **05/04:** fizemos o método *ValorDe()* retornar 0 caso a célula não exista na matriz esparsa. Além disso, o método *SomarEmColuna()* verifica se o valor real passado é 0, porque caso ele seja não é preciso fazer nada. Também foi concluído o método *SomaMatriz()*, e foi implementada a inclusão de célula por meio da edição direta do *DataGridView*;

**Erros e dificuldades:**

* **21/03:** Percebemos o que seria a matriz dita pelo enunciado a partir do conceito dos ponteiros. Isso evitou a construção de uma classe equivocada;
* **26/03:** No método de inclusão, foi essencial inicializar o Direita na *novaCelula* como sendo a *cabecaLinha* e o Abaixo dela sendo a *cabecaColuna*, evitando assim que essas propriedades ficassem nulas caso não houvesse nenhuma célula abaixo ou à direita dela;
* **27/03:** com a criação do método, compreendemos sobre a diminuição do número de loops;
* **02/04:** o método *Exibir()* joga uma exceção quando o tamanho da matriz é muito grande. Fizemos o formulário pegar a exceção com o catch e exibir a mensagem ao usuário;

**Conclusão:**