**EXPLICAÇÕES MÉTODO SIMPLEX**

O método *Simplex* é um algoritmo utilizado para calcular algebricamente Problemas de Programação Linear (PPL) e tenta encontrar uma ou mais soluções ótimas para cada problema apresentado. A solução ótima de um modelo é uma solução viável do sistema, ou seja, um ponto extremo do hiperplano gerado pelas restrições. Para resolver um PPL é necessário conhecer alguma solução inicial do sistema, ou seja, um dos pontos do hiperplano gerado.

Se a solução inicial for ótima o processo é encerrado, caso contrário um dos pontos adjacentes fornece um valor melhor que o inicial. Neste caso, o método faz a mudança de um ponto que otimize o PPL. Esse procedimento é repetido até que seja obtido um ponto extremo que seja a solução ótima. A seguir, será apresentada a explicação do código do método *simplex*, com as variáveis e funções que ajudarão na compreensão código.

O PPL não precisa estar na forma padrão para o programa funcionar. Para inserir o Problema de Programação Linear (PPL), deve-se fornecer o número de restrições do problema e o número de variáveis. Para colocar a matriz na forma padrão, na função **main**, a variável “aux”, guarda a posição da matriz a partir da qual são colocadas as variáveis de folga, o que leva o aumento do número de colunas da matriz. No código, o tamanho da matriz “restricoes” apenas copia os coeficientes de uma para outra, pois *padrao[i][j] = restricoes[i][j].*

Após algumas condições e comandos de repetição no código, a função “Z” objetivo é colocada na forma padrão. O sinal é alterado caso a função objetivo seja de maximização (equivale a multiplicar todo o Z por -1). Como a matriz “restricoes” não é utilizada na forma padrão, foi usada uma outra variável para desalocar o espaço, que é executada pela seguinte atribuição presente no código: *restricoes = desaloca\_matriz(l, restrições).*

Em seguida, a matriz “tableaux” guarda o *tableau* (inicial e os outros até chegar na solução ótima). Para que seja possível encontrar o *tableau*, é preciso adicionar mais uma coluna (c) para "Z” e uma linha (l) para a “base”. Feito isso, o programa define a primeira linha do *tableau* e caso seja um problema de minimização, multiplica-se a linha por “-1”, inclusive o “Z”.

Ao atribuir a coluna de “Z”, definimos “1” na primeira linha e “0” nas outras. O “Z” é atribuído como a linha base do *tableau* e as restrições também são acrescentadas.

Na função **primeira\_fase**, o parâmetro “W” (que é inicialmente declarado na função **main**), é verificado mediante à variável “isOtimo”, a qual verifica o ótimo do *tableau*, ou seja, caso a variável “isOtimo”seja igual à “1”, o parâmetro “W” não é igual à zero, podendo assim escolher as variáveis que vão entrar (in) e sair (aux), gerando assim uma nova solução.

Se o ótimo (variável “isOtimo”) for igual a 2, a função não tem solução, caso contrário, o “W” é igual a zero e finaliza a primeira fase. No entanto, se “isOtimo” for 0 (zero), inicia-se a segunda fase e o *tableau* é mostrado na tela, assim como o resultado.

Na função **imprime\_tableaux**, o vetor guarda os índices dos “x” para serem impressos embaixo do *tableau*. A função **calloc** cria um vetor de tamanho dinâmico. Após imprimir o *tableau*, no código existe um comando de repetição *for* que percorre o vetor e busca se o índice do “x” em questão que está na base e dependendo das condições apresentadas no código, os valores calculados da matriz “tableaux” são impressos, ou valor igual a 0.

Caso o elemento da linha “0” e coluna “0” da matriz “tableaux” seja igual a “-1” e o elemento da linha “0” e coluna “c -1” (numero de colunas menos 1) da mesma matriz seja diferente de “0”, o valor de “Z” maximizado ou “-Z” minimizado será impresso.

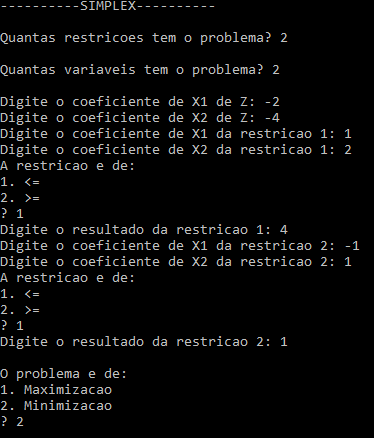
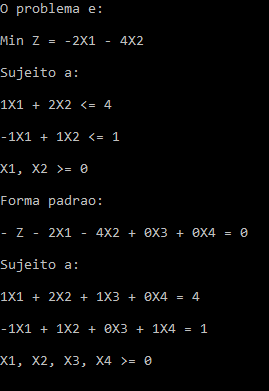
As funções **variable\_in** e **variable\_out** encontram quem deve entrar e sair da base respectivamente, portanto, depois de impresso o *tableau* inicial, a partir da primeira iteração (as interações são feitas na função **iteração\_tableaux**), caso não seja encontrada a solução ótima, uma variável deverá entrar e outra deverá sair, dando continuidade ao procedimento até encontrar o ótimo do problema.

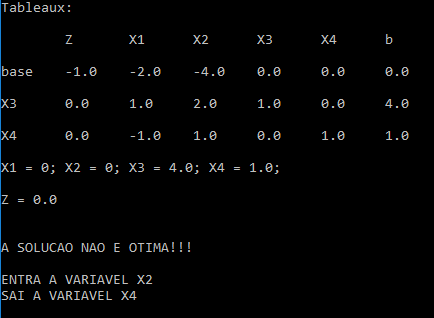
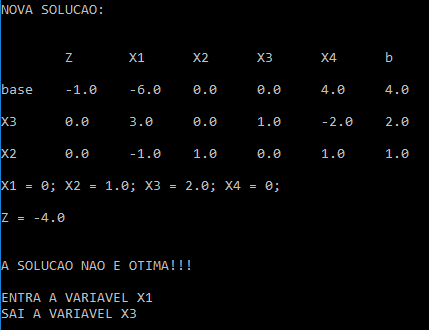
A função responsável por testar as duas funções descritas anteriormente (**variable\_in** e **variable\_out**) é a **resultado\_tableaux**, a qual determina se a solução é degenerada, se o problema possui múltiplas soluções, ou se a função tem solução infinita, ou mesmo se é solução ótima, ou não, pois essa função aplica os cálculos do *tableau*.

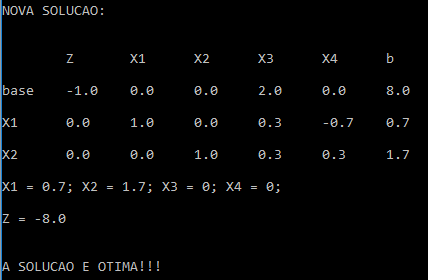
Ainda sobre a função **resultado\_tableaux**, um comando de repetição *for* é responsável por realizar o cálculo da nova linha pivô e essa nova linha será dividida pela intersecção entre a linha e a coluna pivô*.* Um outro *for* inicia o percurso pela primeira linha do *tableau* e existe uma condição que determina se prossegue com os cálculos, caso a linha atual não seja a linha pivô, pois ela já foi calculada antes. Dentro desse *for*, existe um que caso a condição não seja a linha pivô, ele percorrerá a linha toda. Uma variável auxiliar presente no for interno guarda a intersecção entre a linha atual com a coluna pivô. Os novos valores de cada linha, são cada valor subtraído da multiplicação de “aux” (variável auxiliar) com o valor correspondente na mesma coluna, na linha pivô.

Nos exemplos que sucedem, foi-se detalhado cada um deles e o programa mostra quando a solução ótima é encontrada, além de dizer qual o tipo de solução, assim como foi apresentado no código.

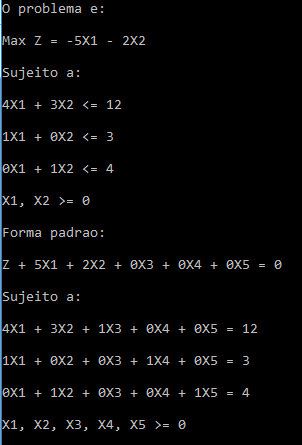
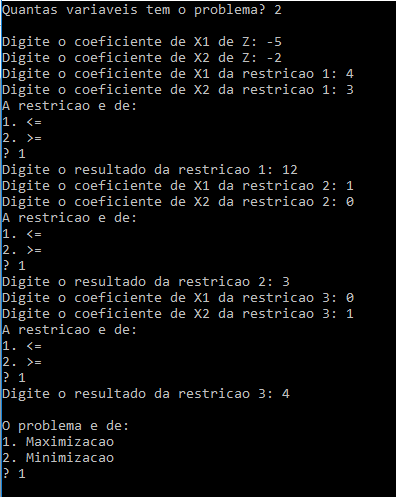
* Primeiro exemplo

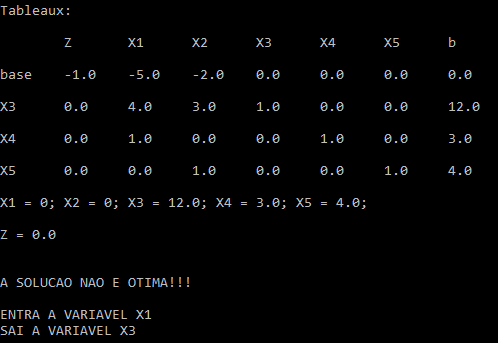
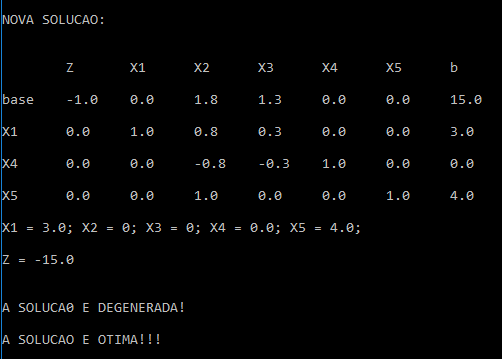
 



* Segundo exemplo



* Terceiro Exemplo

