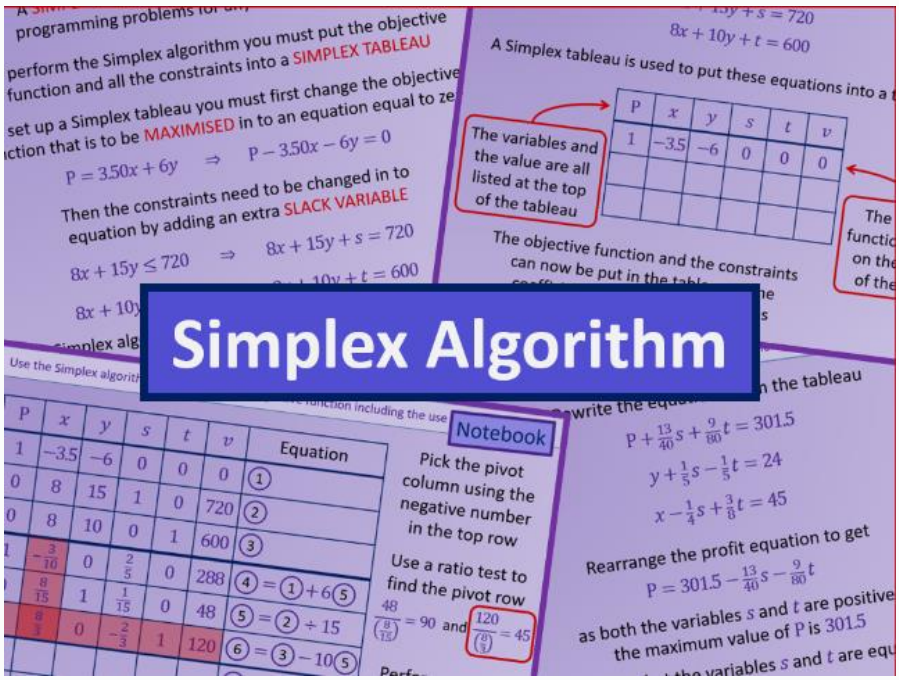
RAPPORT

Simplexe avec C



Réalisé par:

Ameziane Douaa

Kamal Aymane

Encadré par:

Dr. Abdelati REHA

PARTIE CODE

Implémentation des fonctions de Gauss sur les tableaux d'itérations

```c
#include <stdio.h>
//Fonction de gauss
void gauss(float TP[][10], int m, int n, int pivotLigne, int pivotColonne)
    float pivotElement = TP[pivotLigne][pivotColonne];
        // Etape1: Lp=Lp/pivot
    for (int j = 0; j < n; j++) {
        TP[pivotLigne][j] /= pivotColonne;
    }
        // Etape2: Lx=Lx-Va.Lp
    for (int i = 0; i < m; i++) {
        if (i != pivotLigne) {
            float Va = TP[i][pivotColonne];
            for (int j = 0; j < n; j++) {
                TP[i][j] -= Va * TP[pivotLigne][j];
            }
        }
    }
}
```

Détermination de la colonne pivot

```c
//Colonne pivot
int trouverColonnePivot(float TP[][10], int m, int n) {
    int colonnePivot = -1;
    float plusNegatif = 0;
    // Comparer tous les elements a plusNegatif
    //sauf le dernier colonne
    for (int j = 1; j < n - 1; j++) {
        if (TP[0][j] < plusNegatif) {
            plusNegatif = TP[0][j];
            colonnePivot = j;
        }
    }
    return colonnePivot;
}
```

Détermination de la ligne pivot

```
//ligne pivot
int trouverLignePivot(float TP[][10], int colonnePivot, int m, int n) {
    int lignePivot = -1;
    float rapportMin = -1;
    for (int i = 1; i < m; i++) {
        if (TP[i][colonnePivot] > 0) {
      //diviser les cst avec lintersection avec colonne pivot
      //et on compare avec le minimum
            float rapport = TP[i][n - 1] / TP[i][colonnePivot];
            if ((rapportMin == -1 || rapport < rapportMin)
                && TP[i][colonnePivot] != 0) {
                rapportMin = rapport;
                lignePivot = i;
            }
        }
    }
    return lignePivot;
}
```

Création d'un tableau diagonal de 1,1,1 des variables de bases VB

```
//creer un tableau dont le diagonal est 1 et les autres0
//pour remplir notre premier tableau d itteration (VB)
void creerDiagonal(float TD[][10], int NbrCont) {
    for (int i = 0; i < NbrCont; i++) {
        for (int j = 0; j < NbrCont; j++) {
            if (i == j) {
                TD[i][j] = 1;
            } else {
                TD[i][j] = 0;
            }
        }
    }
}
```

Fonction d'affichage des tableaux d'itération

```
//Affichage des tableaux d itterations
void AfficherTableau(float T[10][10], int NbrVar, int NbrCont, int m,
                     int n) {
    int i, j;
    printf("Z\t ");
    for (i = 0; i < NbrVar; i++) {
        printf("x%d      ", i + 1);
    }
    for (i = 0; i < NbrCont; i++) {
        printf("e%d     ", i + 1);
    }
    printf("|");
    printf("=\n");
    for (i = 0; i < n; i++) {
        printf("%.1f\t", T[0][i]);
    }
    printf("\n");
    for (i = 0; i < n; i++) {
        printf("---------");
    }
    }
    printf("\n");
    for (i = 1; i < m; i++) {
        for (j = 0; j < n - 1; j++) {
            printf("%.1f\t", T[i][j]);
        }
        printf("|%.1f\n", T[i][n - 1]);
    }
}
```

Création d'un tableau ou on stock les coefficients de l'équation Objective

```c
//creer un tableau dont on stock les coefficients du fctn Objectif
void CoeffObjectif(float TO[10], int NbrVar) {
    int i;
    //input des coefficients
    printf("                          Objectif:\n\n");
    printf("Entrer les coefficients :\n\n");
    for (i = 0; i < NbrVar; i++) {
        printf("Coefficient de la variable %d: ", i + 1);
        scanf("%f", &TO[i]);
    }
    //ecrire l equation complete
    printf("\n");
    printf("Z = ");
    for (i = 0; i < NbrVar; i++) {
        printf("%.1fx%d", TO[i], i + 1);
        if (i < NbrVar - 1) {
            printf(" + ");
        }
    }
```

Création du tableau ou on va stocker les coefficients des équations des contraintes

```c
//creer un tableau de coefficients des contraintes
void Contraintes(float TC[][10], int NbrVar, int NbrCont) {
    int i, j;
    //input coefficients
    printf("-----------------------------------------------------------------\n"
    printf("                          Objectif:\n\n");
    for (i = 0; i < NbrCont; i++) {
        printf("Contrainte %d: \n", i + 1);
        for (j = 0; j < NbrVar; j++) {
            printf("Coefficient de x%d : ", j + 1);
            scanf("%f", &TC[i][j]);
        }
     //input resultat
        printf("Entrer la constante de la contrainte %d : ", i + 1);
        scanf("%f", &TC[i][NbrVar]);
    }
    //affichage des contraintes
    printf("\nContraintes:\n\n");
    for (i = 0; i < NbrCont; i++) {
```

```c
        printf("Contrainte %d:\n\n", i + 1);
        for (j = 0; j < NbrVar; j++) {
            printf("%.1fx%d", TC[i][j], j + 1);
            if (j < NbrVar - 1) {
                printf(" + ");
            }
        }
        printf(" <= %.1f\n", TC[i][NbrVar]);
    }
}
```

Création du premier tableau d'itération dont on va stocker les tableaux des objectifs et les contraintes et le résultat

```
//creer le premier tableau d itteration
void PremierTaleau(float TP[][10], float TO[10], float TC[][10],
                   float TD[][10], int NbrVar, int NbrCont, int m, int n) {
    int i, j, c, d;
    //Premier ligne (ligne Objectif):
    TP[0][0] = 1;
    for (i = 1; i < NbrVar + 1; i++) {
        TP[0][i] = -TO[i - 1];
    }
    for (i = NbrVar + 1; i < n; i++) {
        TP[0][i] = 0;
    }
    //les autres lignes(lignes contraintes):
    for (i = 1, d = 0; i < m, d < NbrCont; i++, d++) {
        TP[i][0] = 0;
        for (j = 1, c = 0; j < NbrVar + 1, c < NbrVar; j++, c++) {
            TP[i][j] = TC[d][c];
        }
```

```
 for (j = NbrVar + 1, c = 0; j < NbrVar + 1 + NbrCont, c < NbrCont;
                                        j++, c++) {
             TP[i][j] = TD[d][c];
         }
         TP[i][n - 1] = TC[d][NbrVar];
     }
}
```

Affichage du processus simplexe et les tableaux d'itérations et les variables de base et les lignes pivot et les colonnes pivot (Tout le processus de simplexe)

```
//affichage du processus simplexe en utilisant les fctns precedent
void simplexe(float TP[][10], float TO[10], float TC[][10],
              float TD[][10], int NbrVar, int NbrCont, int m, int n) {
    int iteration = 1;
    while (1) {
    printf("\n\nIteration %d:\n", iteration);
        AfficherTableau(TP, NbrVar, NbrCont, m, n);
        int colonnePivot = trouverColonnePivot(TP, m, n);
        if (colonnePivot == -1) {
          //si le collone pivot n existe plus on sort
            printf("\nLe simplexe est termine.\n");
            break;
        }
        int lignePivot = trouverLignePivot(TP, colonnePivot, m, n);
        if (lignePivot == -1) {
            printf("\nPas de solution optimale.\n");
            break;
        }
```

```
        printf("\nVe: %d\n", colonnePivot);
        printf("Vs: %d\n\n", lignePivot);
        afficherVariablesBase(TP, NbrVar, NbrCont);
        printf("\n");
        afficherVariablesHorsBase(TP, m, n, NbrVar);
        gauss(TP, m, n, lignePivot, colonnePivot);
        iteration++;
    }
}
```

Fonction VB

```
//Fonction Vb
void VariableBase(float TP[][10],int NbrCont,int n ,int m){
   int un=0,emplacement,c=1,i,j,zero=0;
    for(j=1;j<n-1;j++){
        for(i=1;i<m;i++){//calculer si il ya un 1 et des 0
            if(TP[i][j]==1){
                    un++;
                    emplacement=i;
            }
            if(TP[i][j]==0){
                    zero++;
            }}
        if(un+zero==NbrCont && un=1){
            printf("VB%d=TP[emplacement][n-1]",c);
            c++;
        }

    }
}
```

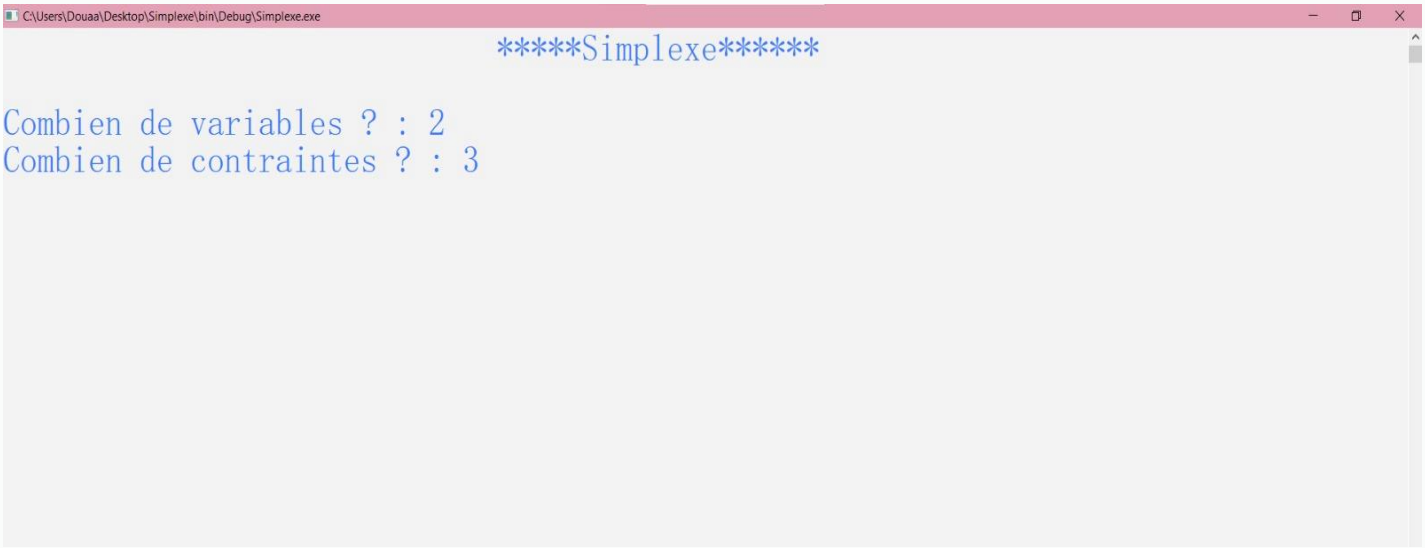
IMPLEMENTATION DES FONCTIONS DANS LE MAIN

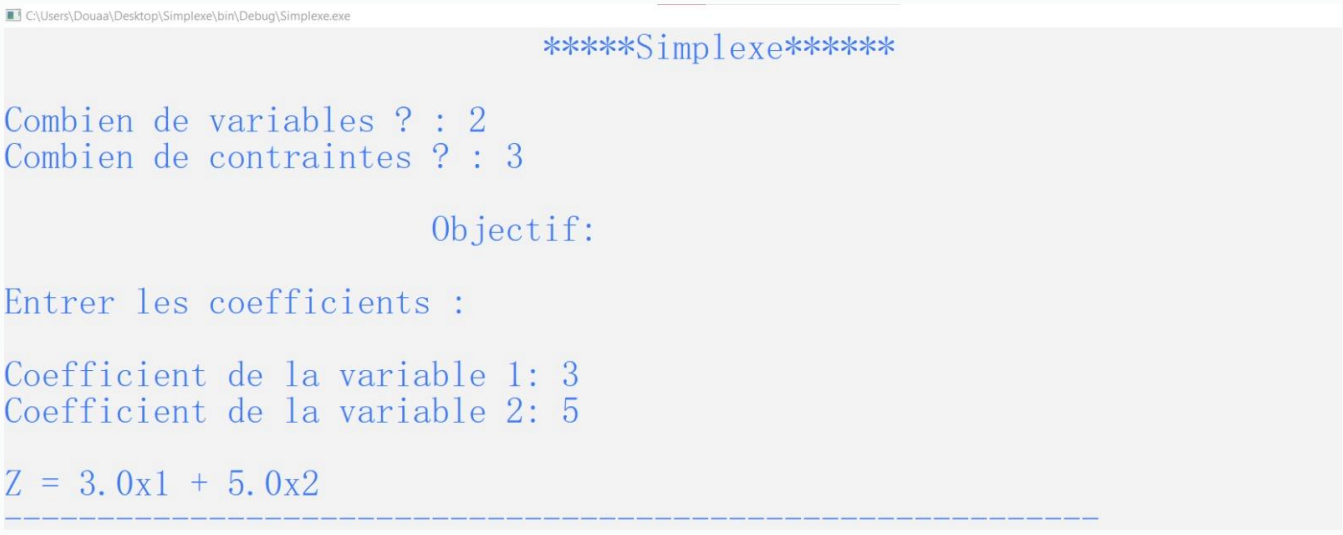
```c
int main() {
    int NbrVar, NbrCont;
    printf("                                    *****Simplexe******\n\n");
    //limiter la saisie des utilisateurs
    do {
        printf("Combien de variables ? : ");
        if (scanf("%d", &NbrVar) != 1) {
            printf("Veuillez entrer un nombre entier.\n");
            scanf("%*s"); // tampon effacee
        }
    } while (NbrVar <= 0);
    //limiter la saisie des utilisateurs
    do {
        printf("Combien de contraintes ? : ");
        if (scanf("%d", &NbrCont) != 1) {
            printf("Veuillez entrer un nombre entier.\n");
            scanf("%*s"); // tampon effacee
        }
    } while (NbrCont <= 0);
```

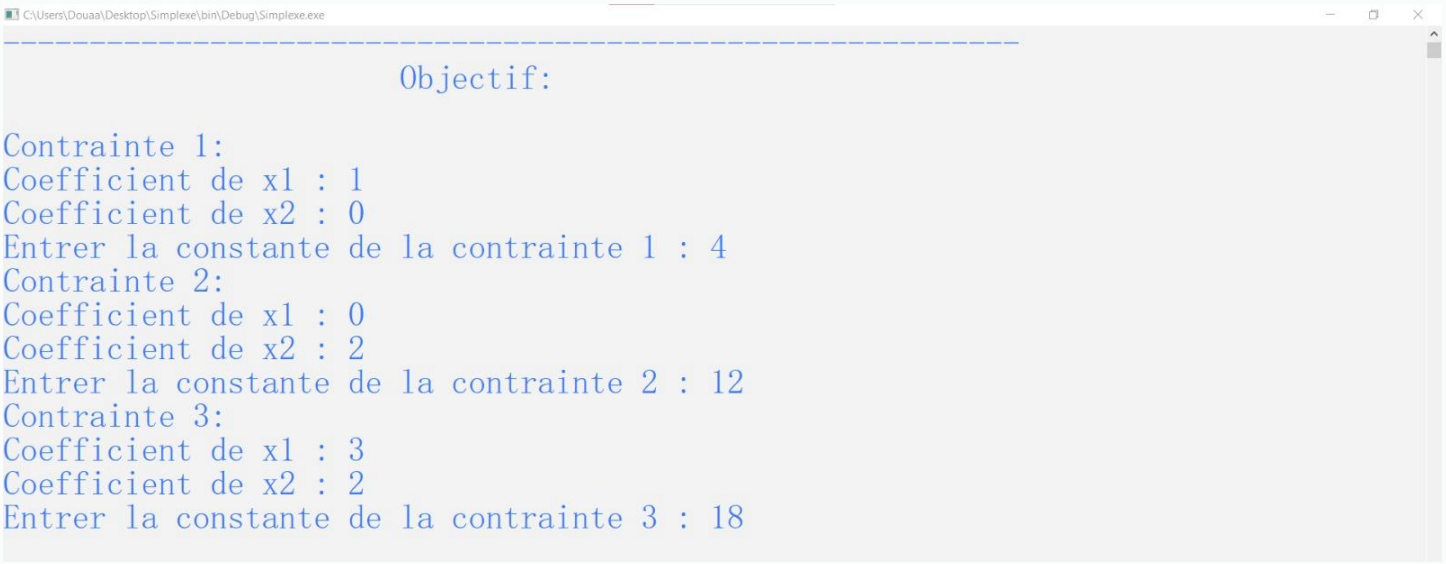
```c
    printf("\n");
    //des tableaux statiques dont on stock nos inputs en fcts
    float TO[10]; // tableau objectif
    float TC[10][10];  //tableau contraintes
    float TP[10][10];  //tableau premier
    float TD[10][10];  //tableau diagonal
    //les dimensions des tables d itteration
    int m = NbrCont + 1;// nbr des contraintes + Objectif
    int n = NbrCont + NbrVar + 2; // Objectif+ Nbr Contraintes+Nbr Vars+Cs
    //utilisation des fonctions:
    creerDiagonal(TD, NbrCont);
    CoeffObjectif(TO, NbrVar);
    Contraintes(TC, NbrVar, NbrCont);
    PremierTaleau(TP, TO, TC, TD, NbrVar, NbrCont, m, n);
    printf("-------------------------------------------------------------\n")
    simplexe(TP, TO, TC, TD, NbrVar, NbrCont, m, n);

    return 0;
}
```

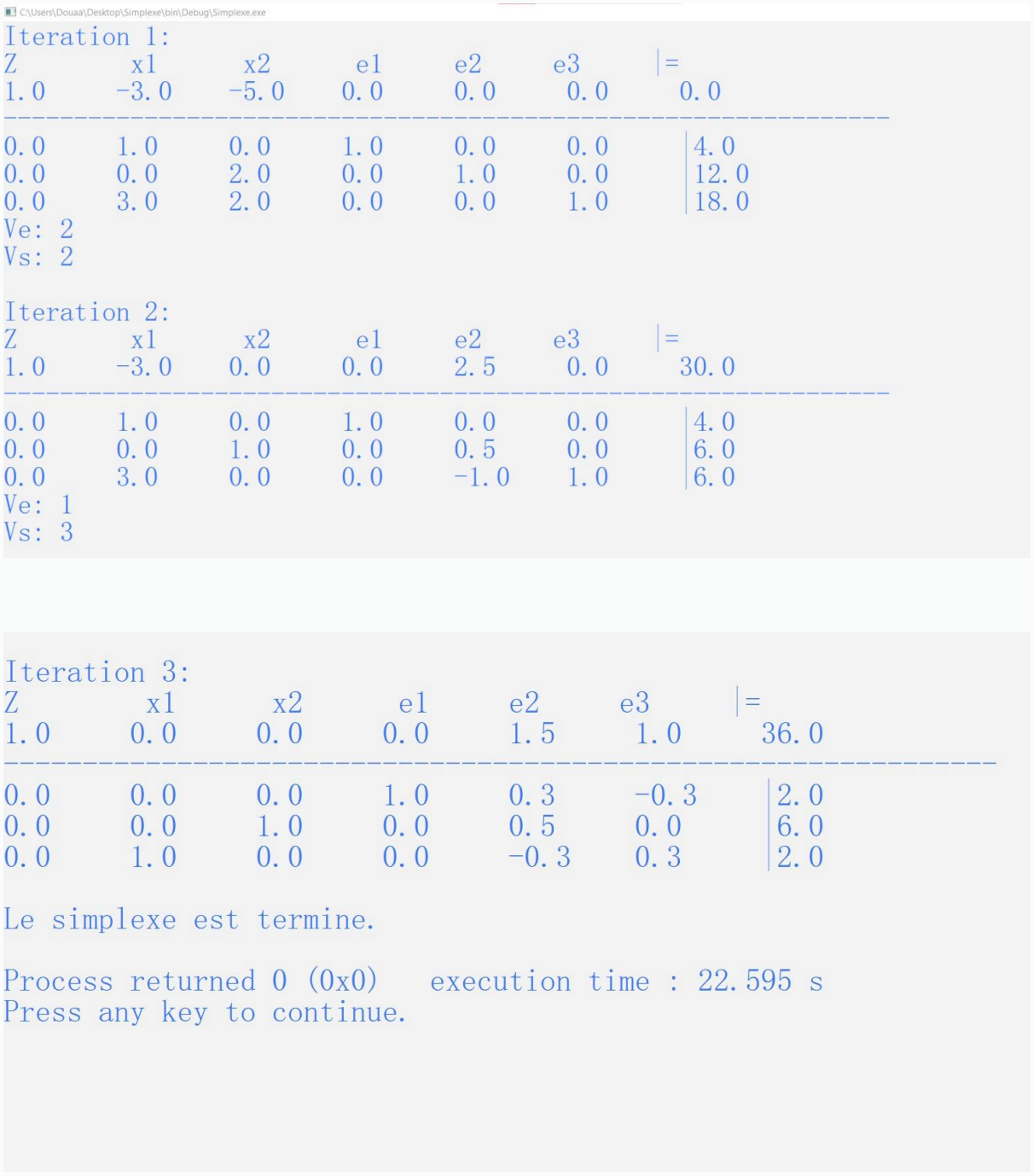
PARTIE EXECUTION


C:\Users\Douaa\Desktop\Simplexe\bin\Debug\Simplexe.exe
*****Simplexe******
Combien de variables ? : 2
Combien de contraintes ? : 3


C:\Users\Douaa\Desktop\Simplexe\bin\Debug\Simplexe.exe
*****Simplexe******
Combien de variables ? : 2
Combien de contraintes ? : 3
Objectif:
Entrer les coefficients :
Coefficient de la variable 1: 3
Coefficient de la variable 2: 5
Z = 3.0x1 + 5.0x2


C:\Users\Douaa\Desktop\Simplexe\bin\Debug\Simplexe.exe
Objectif:
Contrainte 1:
Coefficient de x1 : 1
Coefficient de x2 : 0
Entrer la constante de la contrainte 1 : 4
Contrainte 2:
Coefficient de x1 : 0
Coefficient de x2 : 2
Entrer la constante de la contrainte 2 : 12
Contrainte 3:
Coefficient de x1 : 3
Coefficient de x2 : 2
Entrer la constante de la contrainte 3 : 18


C:\Users\Douaa\Desktop\Simplexe\bin\Debug\Simplexe.exe
Contraintes:
Contrainte 1:
1.0x1 + 0.0x2 <= 4.0
Contrainte 2:
0.0x1 + 2.0x2 <= 12.0
Contrainte 3:
3.0x1 + 2.0x2 <= 18.0


C:\Users\Douaa\Desktop\Simplexe\bin\Debug\Simplexe.exe
Iteration 1:
Z x1 x2 e1 e2 e3 |=
1.0 -3.0 -5.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 1.0 0.0 0.0 |4.0
0.0 0.0 2.0 0.0 1.0 0.0 |12.0
0.0 3.0 2.0 0.0 0.0 1.0 |18.0
Ve: 2
Vs: 2
Iteration 2:
Z x1 x2 e1 e2 e3 |=
1.0 -3.0 0.0 0.0 2.5 0.0 30.0
0.0 1.0 0.0 1.0 0.0 0.0 |4.0
0.0 0.0 1.0 0.0 0.5 0.0 |6.0
0.0 3.0 0.0 0.0 -1.0 1.0 |6.0
Ve: 1
Vs: 3
Iteration 3:
Z x1 x2 e1 e2 e3 |=
1.0 0.0 0.0 0.0 1.5 1.0 36.0
0.0 0.0 0.0 1.0 0.3 -0.3 |2.0
0.0 0.0 1.0 0.0 0.5 0.0 |6.0
0.0 1.0 0.0 0.0 -0.3 0.3 |2.0
Le simplexe est termine.
Process returned 0 (0x0) execution time : 22.595 s
Press any key to continue.