

---

# COMPTE RENDU TP1- STRUCTURES DE DONNÉES

---



03/10/2016

ILISI 1ERE ANNÉE

ZKARA Chaimae, HILALI Abderrahmane.

# Table des matières

---

<u>CHAPITRE</u>	<u>PAGE</u>
LISTES DES FIGURES.....	1
CHAPITRES :	
CHAPITRE 1 – Introduction.....	2
CHAPITRE 2 – Analyse du problème.....	3
CHAPITRE 3 – Analyse fonctionnelle.....	4
CHAPITRE 4 – Exemple d'exécution.....	10

# Table des figures

---

Figures :	Page
Figure 1 : Lecture d une caractère valide.	11
Figure 2 : cas de chaine de caractères non valide.	11
Figure 3 : affichage dans le cas d'un erreur.	12
Figure 4 : l'affichage du nombre convertie.	12

# Introduction

---

Dans ce TP, on va écrire un programme qui va permettre à l'utilisateur de convertir une chaîne de caractères à un nombre réel. L'utilisateur tape le nombre dans la console et le programme le lit en tant que chaîne de caractères et la transforme à un nombre réel.

Il y a plusieurs cas que nous devons gérer : gestion des erreurs, gestion des cas possibles des nombres.... L'un des problèmes que nous avons rencontré pendant la résolution de ce problème est le stockage des caractères tapés par l'utilisateur. Alors dans ce rapport nous avons expliqué la méthode de conversion vers un nombre réel que nous avons adapté, et comment on a géré les erreurs et les cas possibles.

# Analyse du problème

---

L'objectif de ce TP est de permettre à l'utilisateur de taper un nombre, ensuite le programme va le lire caractère par caractère et le convertir en chiffres, comme il ne faut pas utiliser de tableaux. L'astuce pour convertir un caractère en un chiffre est de soustraire la valeur en ASCII du '0' de ce caractère, par exemple si l'utilisateur a tapé le caractère '3', alors pour le convertir en nombre il faut just soustraire sa valeur ASCII du caractère '0'. Comme la valeur de '0' en ASCII est 48 et la valeur de '3' en ASCII est 51, alors le résultat de la soustraction retourne la valeur 3.

La deuxième astuce qui nous permettra de ne pas utiliser les tableaux de caractères (chaîne de caractères) est de lire depuis le buffer. Tous les caractères qui sont saisis par l'utilisateur se trouvent dans la mémoire tampon, Alors tous qu'il faut faire est de lire la chaîne de caractères entrée par l'utilisateur, caractère par caractère depuis la mémoire tampon est de convertir le nombre lu à sa valeur correspondante.

## Mais comment pouvons-nous convertir la chaîne '123' vers le nombre 123 ?

Pour faire convertir la chaîne '123' vers le nombre 123, nous faisons un calcul mathématique simple : La décomposition du nombre vers la somme de plusieurs chiffres.

Exemple :

$$123 = 1 \times 100 + 2 \times 10 + 3 \times 1.$$

Alors D'après cette règle on peut convertir la chaîne '123' vers le nombre 123, en lisant la chaîne caractère par caractère et à chaque fois nous multiplions la valeur convertie par 10. Alors en premier temps nous allons lire le caractère '1', le convertir et le stocker, puis on lit le caractère '2' mais cette fois nous allons multiplier

l'ancienne valeur (1) par 10 et après on ajoute la valeur 2. Le resultat est 21. Nous lisons l'autre caractère. '3' le convertir, multiplier l'ancienne valeur par 10 (120), et puis on ajoute la dernière valeur convertie qui est 3. Alors le résultat sera 123.

Il faut aussi penser à la gestion des erreurs ; les chiffres tapés par l'utilisateur doit être compris entre 0 et 9. Le signe du nombre doit aussi être pris en considération. Si il tape '+123' le programme doit afficher '+123' et non '123' de même pour le signe négatif.

## Analyse fonctionnelle

---

Pour la solution de ce problème, on a choisi d'utiliser la méthode suivante :

La chaîne tapée est stockée dans le buffer,

Notre programme va lire depuis cette mémoire tampon et traiter caractère par caractère.

### ✓ Fonction : isNumber:

Cette fonction teste si un caractère est un chiffre ou non. Si c'est le cas elle retourne 1. Sinon, elle retourne 0.

Elle va nous permettre par la suite de gérer le cas où l'utilisateur s'amuse à taper des caractères quelconque.

```
/*  
Fonction   :   isNumber;  
Paramètres :   char c : le caractère à vérifier si c'est un chiffre ou pas;  
Sortie     :   int : on retourne 1 si c'est un chiffre, sinon on retourne 0  
***/
```

```

int isNumber(char c)
{
    //le caractère n'est pas un nombre
    if( c < '0' || c > '9') return (0);
    //le caractère est un nombre
    return (1);
}

```

#### ✓ Fonction : isSigne:

Pour cette fonction, elle permet de tester un caractère s'il est un signe ou non. Dans le cas contraire elle retourne le caractère 0.

```

/*****
Fonction   : isSigne;
Paramètres : char c : le caractère à vérifier si c'est un signe ou pas;
Sortie      : char : si c'est signe retourne 1, sinon on retourne 0
*****/
char isSigne(char c)
{
    if( c == '+' || c == '-') return (1);
    return (0);
}

```

#### ✓ Fonction : isVirgule:

Cette fonction permet de tester si le caractère qui lui est passé en paramètre est une virgule ou non.

```

/*****
Fonction   : isVirgule;
Paramètres : char c : le caractère à vérifier si c'est une virgule ou non;
Sortie      : int : on retourne 1 si c'est une virgule, sinon on retourne 0
*****/
int isVirgule(char c)
{
    (c == ',') ? ( return ( 1 ) ) : ( return ( 0 ) );
}

```

✓ Fonction : conversion:

C'est la fonction la plus importante du programme. Les traitements essentiels aux fonctionnements de ce programme sont implémentés dans cette fonction.

On commence par traité le 1<sup>er</sup> caractère saisi, si ce n'est ni un chiffre, ni un signe ni une virgule alors on affiche un message d'erreur et on quitte le programme. Si l'utilisateur a tapé un signe '+' ou '-' il doit être affiché aussi. La variable `signe` est initialisée à 1, si l'utilisateur tape '+' on lui affecte 2, si il tape '-', on lui affecte -1.

Maintenant, si le 1<sup>er</sup> caractère est un chiffre, on doit le convertir en `double`. C'est ce que nous faisons par l'instruction : `« partieE = buffer - '0'; »`. Finalement, on teste si c'est une virgule, si c'est le cas on saute à la partie flottante de la fonction qu'on va expliquer ultérieurement.

On passe maintenant au restant du nombre. On fait une boucle `while` avec condition d'arrêt saut de ligne ou virgule. Dans cette boucle, on teste chaque caractère s'il n'est pas un nombre on affiche un message d'erreur et on quitte, sinon on ajoute ce caractère à la partie entière du nombre.

Après avoir sorti de la boucle, il ne peut y avoir que deux cas :

Soit on est arrivé au saut de ligne, ce qui veut dire éventuellement que la chaîne de caractère est terminée alors on passe à la partie 'Fin',

Soit l'utilisateur a tapé une virgule, alors on saute à la partie 'flottant'.

Dans la partie 'flottant', on fait la même chose que dans la partie entière. On fait tous les tests et on convertie les caractères à des chiffres on prenant en considération le nombre de chiffres après la virgule (La variable `f`), pour qu'on puisse convertir le nombre proprement.



La dernière partie consiste à calculer le résultat final et à l'afficher en prenant compte du signe de ce nombre.

```
/******
Fonction : conversion;
Paramètres : aucun;
Sortie : affichage du nombre;
Description : Cette fonction permet de taper un nombre en une seule fois, de le lire
caractère par caractère et d'afficher la valeur équivalente tout en traitant le signe et la
virgule flottante;
*****/void conversion(){
    //Déclaration des variables :
    char buffer; //le caractère lu depuis le buffer
    short signe=1; //le signe du nombre, il prend 3 valeur :
        // 1 - si l'utilisaateur a tapé '123' par exemple,
        // 2 - si l'utilisateur a tapé '+123' par exemple,
        // -1 - si l'utilisateur a taper un nombre negatif '-123'

    double partieE = 0.0, //pour stocker la partie entiere du nombre tapé
    partieF=0.0, //pour stocker la partie flottante du nombre tapé
    f=1.0; //la puissance de la partie flottante
    resultat = 0.0;//c'est le resultat finale.

    //on lit la chaine de caractère
    printf("\n\n\tEntrer votre nombre : ...\n");
    printf("\n\n\t");
    buffer = getchar();

    //Traitement du premier caractère :

    //si le caractère n'est pas valide
    if( (!isNumber(buffer)) && (!isSigne(buffer) == 0) &&
        (!isVirgule(buffer)) )
    {
        //on affiche un message d'erreur
        printf("\n\t/!\ /!\ Erreur de saisie /!\ /!\n");
        exit(EXIT_FAILURE);
    }
    else
    {
```

```

        //Si le premier caractere est un signe;
        //S'il est '+' on met dans signe 2 pour indiquer
        //que l'utilisateur à taper le signe + au debut
        if( isSigne(buffer) == 1 )
        {
            (signe == '+' ) ? (signe = 2) : (signe = -1);
        }

        //S'il est un chiffre :
        else if( isNumber(buffer) )
            partieE = buffer - '0';

        //S'il est une virgule, on saut vers la partie flottant. '.123'
        else if( isVirgule(buffer) )
            goto flottant;
    }

    //FIN Traitement du premier char

    while( (( buffer = getchar() ) != '\n') && (!isVirgule(buffer)) )
    {
        //on test si le caaractere est valide
        if( !isNumber(buffer) )
        {
            printf("\n\t/!\ \ /!\ Erreur de saisie /\ \ /\ \n");
            exit(EXIT_FAILURE);
        }

        //calcule du partie entiere courante
        partieE = (double) ((partieE * 10) + (buffer - '0'));
    }

    //On sort de la boucle
    //si il y a saut de la ligne, on saut vers la partie fin
    if( buffer == '\n' )
        goto Fin;
    // si il est un virgule on saut vers la partie Flottant
    else if( isVirgule(buffer) )
        goto flottant;
    //sinon on affiche un message d'erreur et on quite
    else
    {

```

```

        printf("\n\t/!\ \ /!\ Erreur de saisie /!\ \ /!\n");
        exit(EXIT_FAILURE);
    }

//la partie flottant
flottant :
    while( (buffer = getchar()) != '\n' )
    {
        //on verifie que c'est un nombre valide
        if( !isNumber(buffer))
        {
            printf("\n\t/!\ \ /!\ Erreur de saisie /!\ \ /!\n");
            exit(EXIT_FAILURE);
        }
        //on calcule la partie flottant
        partieF = (double) ((partieF*10) + (buffer - '0'));
        // on multiplie f par 0.1
        f *= 0.1;
    }
//fin partie Flottant

//la partie Fin
Fin :
    //calcule du resultat
    resultat = (double) (partieE + (partieF * f));

//affichage du nombre
printf("\n\n\t");
printf("*****\n");
printf("\t");
printf("* la chaine de caractere convertie en nombre : *\n");
printf("\t*****");
printf("\n\n\t");

//le nb est positif sans que le l'utilisateur ait entré '+'
if(signe == 1)
    printf("%f",resultat);
if(signe == -1)
{
    resultat *= -1 ;
    printf("%f",resultat);
}

```

```
    }  
    //l'utilisateur à tapé '+' au debut  
    if(signe == 2)  
        printf("+%f",resultat);  
} //fin conversion
```

✓ Fonction : conversion:

Dans la fonction main, on fait un simple appel de la fonction « conversion ».

```
int main()  
{  
    conversion();  
    return (0);  
}
```

## Exemple d'exécution

---

Pour l'affichage nous avons deux cas importants :

### – le cas d'une chaine valide:

C'est le cas où l'utilisateur tape un caractère valide.

### –le cas d'une chaine non valide:

C'est le cas où l'utilisateur tape un caractère non valide

#### 1. Lecture du nombre:

\* exemple de chaine de caractère valide

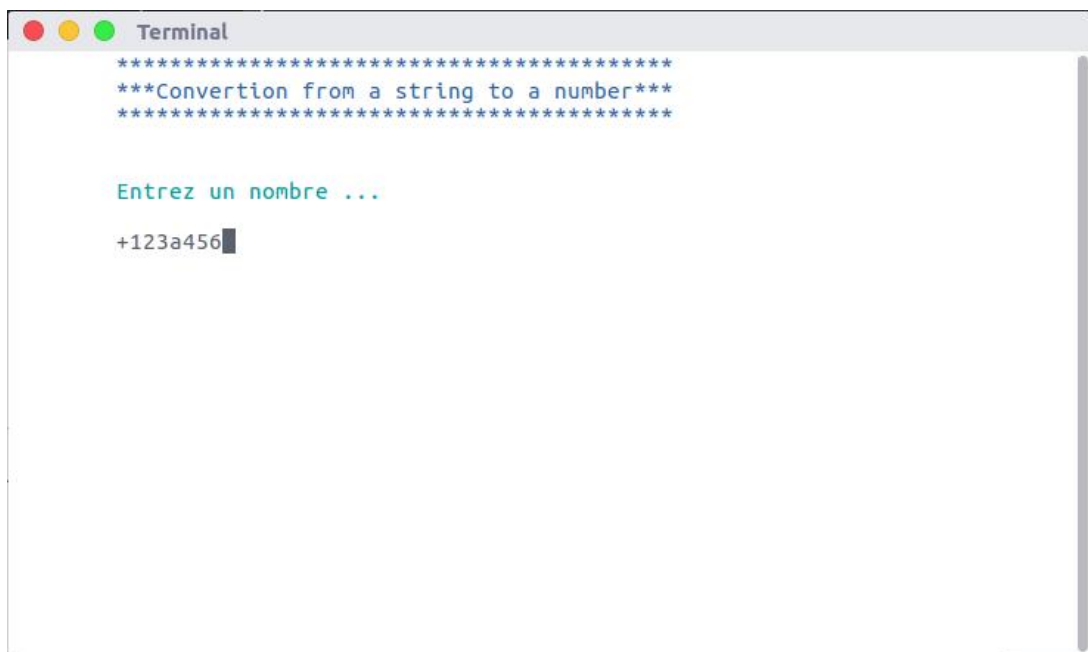


```
Terminal
*****
***Conversion from a string to a number***
*****

Entrez un nombre ...
+123.456
```

Figure 1 : Lecture d une caract re valide.

\* exemple de chaine de caract re non valide :



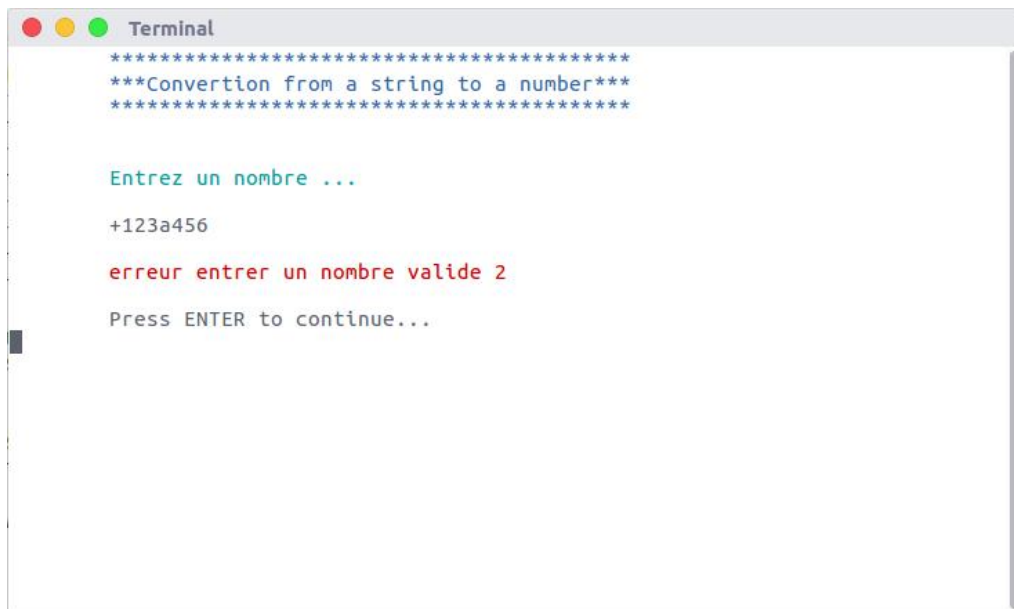
```
Terminal
*****
***Conversion from a string to a number***
*****

Entrez un nombre ...
+123a456
```

Figure 2 : cas de chaine de caract res non valide.

## 2. Affichage du r sultat:

\* exemple d'affichage dans le cas d'une chaine non valide:



```
*****
***Conversion from a string to a number***
*****

Entrez un nombre ...
+123a456

erreur entrer un nombre valide 2

Press ENTER to continue...
```

Figure 3 : affichage dans le cas d'un erreur.

\* exemple d'affichage dans le cas d'une chaine valide



```
*****
***Conversion from a string to a number***
*****

Entrez un nombre ...
+123.456

*****
* la chaine de caractere Transformée en nombre : *
*****

+123.456000

Press ENTER to continue...
```

Figure 4 : l'affichage du nombre convertie.