# Les Fonctions

### Exercice 1 Calculatrice de Grands Nombres Entiers

Les types de base du langage ne permettent pas d'effectuer des opérations sur des nombres entiers de très grande taille.

On se propose d'écrire des fonctions qui vont permettre de réaliser l'addition de deux grands nombres entiers positifs dont la taille maximale sera définie par une constante 'NbChiffresMax'.

Les 'Grands entiers' sont représentés par des tableaux dont chaque case contient un chiffre, les unités étant le plus à droite.

Par exemple, pour coder le nombre entier 78 001 581, nous avons la représentation suivante :

valeur	0	0	7	8	0	0	1	5	8	1
indice	0	1	2	3	4	5	6	7	8	9

Il est utile de faire les déclarations suivantes pour construire notre calculatrice :

```
#define NbChiffresMax 30
// taille max pour representer un grand entier

typedef int GrandEntier[NbChiffresMax];

/* representation des grands entiers à l'aide d'un tableau */
/* les positions qui ne contiennent pas de chiffre significatif */
/* sont initialisées à zero */
```

**Question 1**: Ecrire une fonction void Initialise (GrandEntier  $\mathbb{N}$ ); qui a pour paramètre un *GrandEntier N* et qui assigne à 0 tous les chiffres de  $\mathbb{N}$ .

**Question 2**: Ecrire une fonction void Affiche (GrandEntier N); qui affiche à l'écran un GrandEntier N.

Améliorer l'affichage en groupant les chiffres par trois avec un espace.

**Question 3**: Ecrire une fonction int AjouteADroite (GrandEntier N, int c); qui a pour paramètres un *chiffre c* et GrandEntier N et qui ajoute le chiffre à droite du nombre N. Les autres chiffres de N sont décalés vers la gauche.

Par exemple, si N représente la valeur 78 001 581, après l'appel de la fonction

AjouteADroite (N, 5), N représentera 780 015 815.

La fonction renvoie 0 dans le cas général ou -1 si le décalage a provoqué la perte d'un digit.

Question 4: Ecrire une fonction int addition (GrandEntier A, GrandEntier B, GrandEntier result) qui additionne deux GrandEntiers A et B. La fonction renvoie un code d'erreur -1 quand il y a un dépassement de capacité.

(Attention à la gestion de la retenue).

Question 5: Tester la calculatrice.

#### Code fourni:

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <stdbool.h>
#define NbChiffresMax 30
// taille max pour representer un grand entier
typedef int GrandEntier[NbChiffresMax];
/* representation des grands entiers à l'aide d'un tableau */
/* les positions qui ne contiennent pas de chiffres significatifs */
/* sont initialisées à zero
/* Procedure d'initialisation d'un grand entier a zero */
void Initialise(GrandEntier Nombre)
{
/* Procedure d'affichage d'un grand entier à l'ecran */
void Affiche(GrandEntier Nombre)
      /* affichage d'espace à la place des zéros à gauche
  Sauf s'il s'agit du digit de droite, au cas ou GrandEntier vaut 0,
   il faut afficher 0 */
               /* Procedure d'ajout d'un chiffre à droite dans un grand entier */
int AjouteADroite(GrandEntier Nb, int Chiffre)
      /* décallage à gauche des chiffres */
      //écriture du nouveau chiffre
      // dans la case de droite ainsi libérée
      }
```

Pascal RICQ

```
/* Procedure de saisie d'un grand entier au clavier
/* FONCTION COMPLETE: NE DOIT PAS ETRE MODIFIEE */
int EntreeClavier( GrandEntier Nombre)
  int NbChiffresLus = 0;
 wchar_t car; // touche frappée en entrée
 wchar_t chiffre[2]; // variable type chaine de caractère Wide pour conversion
                    // de car en entier avec _wtoi
  bool EstUnChiffre = false;
  Initialise(Nombre); // Nombre prend la valeur zero
  do
   {
   while ((car = _getwch()) == 0) {
     car = _getwch(); // touche de fonction ou direction enfoncée
   }; // lire la touche frappee au clavier
   if (EstUnChiffre = (bool)isdigit(car)) // s'il s'agit bien d'un chiffre
     {
     wprintf(L"%c", car); // echo ecran du chiffre valide
     chiffre[0]=car; // conversion du caractere en chaine null terminated pour _wtoi()
     chiffre[1]=0;
     AjouteADroite(Nombre, _wtoi(chiffre)); // conversion du caractere en valeur numerique et NbChiffresLus++; // affectation dans le tableau qui contient le grand entier
   else { // s'il s'agit d'un caractère de controle CLEAR (touche 'C' )
     if (( car == 'C') || (car == 'c') ) {
       Initialise(Nombre); // effacement des caractères saisies
printf("\nClear\n");
       EstUnChiffre = true; // pour continuer la saisie de ce nombre
     }
   }
  while (EstUnChiffre && (NbChiffresLus < NbChiffresMax));</pre>
 printf("\n");
 if (( car == 'X') || (car == 'x') ) return -1;
 return 0;
 }
/* Procedure de calcul de la somme de deux grands entiers
int Addition( GrandEntier Nb1, GrandEntier Nb2, GrandEntier Result)
       {
       }
```

```
/* Programme principal: COMPLET: rien à modifier
int main()
{
         GrandEntier Nb1, Nb2, Nb3;
         int i;
         bool Fin = false;
         int Code; // valeur de retour de la fonction EntreeClavier
         while (!Fin)
                   if ( (Code = EntreeClavier(Nb1)) < 0) Fin=true;</pre>
                   if ( !Fin && (Code = EntreeClavier(Nb2)<0)) Fin=true;</pre>
                   if (!Fin)
                   {
                            printf("\n ");
                            Affiche(Nb1);
                            printf("+");
                            Affiche(Nb2);
                            printf(" ");
                            for (i=0; i < NbChiffresMax ; i++)</pre>
                                      if ((NbChiffresMax - i ) % 3 == 0) printf("-");
                                      printf("-");
                            printf("\n= ");
                            Addition(Nb1, Nb2, Nb3);
                            Affiche(Nb3);
                            printf("\n");
                   else
                   {
                            printf("\n Bye !\n");
         }
         return (EXIT_SUCCESS);
}
```

## Exercice 2

Ecrire une fonction factorielle(n) améliorée qui prend en paramètre un nombre entier positif de type **int** et affiche le résultat du calcul du produit 1 x 2 x 3...x n.

Si un factoriel a déjà été calculé précédemment pour une valeur de **n** inférieure, on reprend le calcul là où il s'était arrêté.

On distingue donc trois cas:

- le factoriel a déjà été calculé pour cette valeur de **n**.
- **n** est plus petit que celui du calcul précédent et on reprend le calcul au début.
- **n** est supérieur et on poursuit le calcul en reprenant la boucle là où elle s'était arrêtée.

Les variables utilisées pour le calcul seront déclarées de la manière suivante :

```
static int produit ; /* la valeur du dernier factoriel calculé.*/
static int N;/* la valeur de n pour laquelle on a calculé ce factoriel */
```

Afficher le nombre de boucles effectuées avant la sortie de la fonction. Qu'observe-t-on?

A partir de quelle valeur de **n**, le résultat du factoriel est-il faux ?

Pascal RICQ TDPRG1521 v1.0 Page 5 | 7

# Annexe: seulement si vous utilisez le compilateur GCC

#### Lecture de caractères au clavier avec GCC :

Les fichiers conio.c et conio.h permettent, sans utiliser la librairie ncurses, de lire un caractère frappé au clavier sans attendre l'appui sur la touche entrée.

*conio.h* existe dans l'environnement DOS et fournit la fonction *getch()*. Cette fonction n'est pas standard et pose un problème de portabilité du code vers un autre environnement.

Voici une implémentation de getch() pour Linux.

**Ajouter** 

*getche()* permet, en plus de la lecture au clavier d'un caractère, de faire un 'écho' de ce caractère sur la console.

Pour utiliser les deux fonctions *getch()* et *getche()*, ajouter les deux fichiers à votre projet. Le fichier .h est un fichier d'entêtes à ajouter dans "Header Files" et conio.c est ajouté dans "Sources Files"

```
#include "conio.h" dans main.c
et faire un appel à la fonction getch() pour lire un caractère. Par exemple :
char carlu;
carlu = (char) getch();
conio.h
#ifndef CONIO H
   int getch(void);
   int getche(void);
#define
              CONIO H
#endif
conio.c
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
    /* reads from keypress, doesn't echo */
    int getch(void)
    struct termios oldattr, newattr;
    int ch;
    tcgetattr( STDIN_FILENO, &oldattr );
    newattr = oldattr;
    newattr.c_lflag &= ~( ICANON | ECHO );
    tcsetattr( STDIN_FILENO, TCSANOW, &newattr );
    ch = getchar();
    tcsetattr( STDIN_FILENO, TCSANOW, &oldattr );
    return ch;
    /* reads from keypress, echoes */
```

Pascal RICQ

```
int getche(void)
{
struct termios oldattr, newattr;
int ch;
tcgetattr( STDIN_FILENO, &oldattr );
newattr = oldattr;
newattr.c_lflag &= ~( ICANON );
tcsetattr( STDIN_FILENO, TCSANOW, &newattr );
ch = getchar();
tcsetattr( STDIN_FILENO, TCSANOW, &oldattr );
return ch;
}
```

Pascal RICQ TDPRG1521 v1.0 Page 7 | 7