Algorithmic and Data Structure 2

Functions in C

1. Structure of a function

Syntax:

```
Void or Type_Result Function_Name (Void or Type Param1,…,Type Paramn);
{
Function body  (statements);
}
```

Exemple :

Type void

```
#include <stdio.h>
void afficher (void);
main( )
{
afficher();
}
void afficher(void)
{
      printf("bonjour");
}
```

/*The function "afficher" has neither parameter nor result

/*Display the message « bonjour »

2. Function prototype

→ The function prototype (before the function) gives the rule for using the function: number and type of parameters as well as the type of the returned value.

→ It is recommended not to omit it because it allows the compiler to detect errors during function calls (perform type conversions….).

Syntax:

```
Void or Result_Type Function_Name (Void or Type1,…,Typen);
```

You can insert parameter names but they have no meaning.

3. Calling functions

Syntax:

If the function returns a result and admits parameters

```
Variable = Function_Name (Effective Parameters);
```

If the function returns a result and does not admit parameters

```
Variable = Function_Name ( );
```

If the function returns nothing and admits parameters

```
Function_Name (Effective parameters);
```

If the function returns nothing and does not admit parameters

```
Function_Name ( );
```

4. Passing parameters by value and by variable (by address):

After calling the function, the effective parameters that pass by value keep theirold values from before the call.

A parameter, which passes by variable, must be declared as using *.

Exemple:

Passage par adresse

```
/*Ce programme utilise la fonction triple pour calculer le triple d'un entier*/
      #include <stdio.h>
      void triple (int , int *); /*prototype de la fonction triple qui admet deux paramètres
                                   entiers i et j (j étant le triple de i). i passe par valeur et j passe
                                   par adresse*/
      main( )
      {
      int i,j=0;                 /*i et j variables entiers locales à main et j initialisé à 0*/
      i=2;                       /*i reçoit 2*/
      triple(i,&j);              /*appel de la fonction qui affiche 6, résultat de triple de 2*/
      printf ("%d",j);           /*affiche 6 (nouvelle valeur de j après l'appel de la fonction)*/
      }
      void triple (int i, int *j)      /*définition de la fonction triple*/
      {
      *j=3*i;                          /*j reçoit 6 le triple de 2*/
      printf ("%d",*j);                /*affiche 6*/
      }
```

→Arrays pass by default (and always) by address.

Exemple:

Fonction avec des tableaux comme paramètres

```
/*Ce programme calcule la somme de deux vecteurs (tableaux d'entiers)*/

    #include <stdio.h>
    #define taille 10
    void somme_vecteurs (int [ ], int [ ], int [ ]);  /*prototype de la fonction*/

    main( )
    {

    int a[taille], b[taille], c[taille],i;          /*a, b et c vecteurs d'entiers et i entier*/
    for (i=0; i<taille; i++)
        scanf ("%d%d",&a[i],&b[i]);                 /*lit les éléments de a et b*/
    somme_vecteurs(a,b,c);                          /*c reçoit la somme des vecteurs a et b*/
    for (i=0; i<taille; i++)
        printf ("%d  ",c[i]);                       /*affiche les éléments de c*/

    }

    void somme_vecteurs (int a[ ], int b[ ], int c[ ])      /*définition de la fonction*/
    {
    int i ;
    for (i=0; i<taille; i++)                        /*c reçoit la somme des vecteurs a et b*/
        c[i]=a[i]+b[i]);
    }
```

References

- https://pdfslide.net/documents/asd-algorithmique-et-structures-de-donnees-dbenmerzouge-description-du-cours.html

Thomas H. Cormen, Algorithmes Notions de base *Collection : Sciences Sup, Dunod,* 2013.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Algorithmique - 3ème édition - Cours avec 957 exercices et 158 problèmes Broché, Dunod, 2010.

Rémy Malgouyres, Rita Zrour et Fabien Feschet. *Initiation à l'algorithmique et à la programmation en C : cours avec 129 exercices corrigés*. 2ième Edition. Dunod, Paris, 2011. ISBN : 978-2-10-055703-5.

Damien Berthet et Vincent Labatut. *Algorithmique & programmation en langage C - vol.1 : Supports de cours*. Licence. Algorithmique et Programmation, Istanbul, Turquie. 2014, pp.232.

Damien Berthet et Vincent Labatut. *Algorithmique & programmation en langage C - vol.2 : Sujets de travaux pratiques*. Licence. Algorithmique et Programmation, Istanbul, Turquie. 2014, pp.258. <cel-01176120>

Damien Berthet et Vincent Labatut. *Algorithmique & programmation en langage C - vol.3 : Corrigés de travaux pratiques*. Licence. Algorithmique et Programmation, Istanbul, Turquie. 2014, pp.217. <cel-01176121>

Claude Delannoy. *Apprendre à programmer en Turbo C*. Chihab- EYROLLES, 1994.