Chapitre 7

Affichage d'un nombre à l'écran ; Valeur de retour d'une fonction

Ce chapitre va nous permettre d'étudier comment utiliser la valeur de retour d'une fonction, ou, plus particulièrement dans l'exemple que nous prendrons, d'un ROM_CALL. Nous verrons auparavant comment afficher une valeur à l'écran, montrerons l'importance du respect des majuscules et minuscules, et finiront ce chapitre par une remarque concernant l'imbrication d'appels de fonctions.

I:\ Afficher un nombre à l'écran

Il existe plusieurs façon d'afficher une valeur à l'écran... Certaines sont standards, d'autres non. Certaines sont simples d'emploi, d'autres non.
Au cours de ce chapitre, et des suivants, nous utiliserons la fonction printf.
Plus loin dans ce tutorial, il est possible que nous voyons d'autres méthodes, peut-être plus flexibles, mais moins pratiques pour le débutant...

A: Utilisation de printf

La fonction printf permet d'afficher des chaînes de caractères formatées ; cela signifie que nous pourrons lui demander d'insérer des valeurs au milieu de la chaîne de caractère, et que nous pourrons préciser comment les formater, c'est-à-dire combien de chiffres afficher au minimum, afficher ou non le signe, ou encore utiliser du binaire ou de l'hexadécimal plutôt que du décimal.

Notons que cette fonction est ANSI, ce qui signifie que tous les compilateurs qui se disent respectant la norme ANSI-C la fournissent ; la norme GNU-C suivie par GCC étant une extension à l'ANSI, il aurait été étonnant que TIGCC ne la fournisse pas. remarquez qu'il peut être bon que vous reteniez comment utiliser cette fonction : si vous êtes un jour amené à programmer sous un autre compilateur que TIGCC, il est quasiment certain que vous la retrouverez (je n'ai jamais rencontré un compilateur C ne la définissant pas !).

printf est une fonction admettant un nombre variable d'arguments ; le premier, obligatoire, est une chaîne de caractère, qui contient ce que l'on veut afficher, plus des sortes de "codes" permettant de définir les valeurs à insérer. Les suivants (qui peuvent être au nombre de 0, 1, ou plus) correspondent aux valeurs à insérer dans la chaîne à l'affichage.
Notez que le nombre d'arguments suivant la chaîne de caractères doit être égal au nombre de codes dans celle-ci ! (S'il y a plus d'arguments que de codes, les arguments en trop seront ignorés ; s'il n'y en n'a pas assez, le résultat est indéfini, ce qui, pour dire les choses clairement signifie qu'il y a un risque non négligeable de plantage. Pour résumer, respectez le nombre d'arguments attendu, c'est la meilleure chose à faire.).

Par exemple, pour afficher une chaîne de caractères, sans insérer la moindre valeur, on pourra utiliser cette syntaxe :

```
printf("Hello World !");
```

Je ne vais pas reproduire ici la liste de toutes les options de formatage possibles, cela rallongerait inutilement ce chapitre, puisqu'elles sont toutes fournies, ainsi que leurs explications, dans la documentation de TIGCC, à l'entrée printf.
Sachez juste que les options de formatage commencent par un caractère '%'. La suite de caractère '\n' (antislash, accessible sur un clavier azerty par la combinaison de touches alt-gr + 8) permet un retour à la ligne.

Notez que, lorsque nous utiliserons printf, nous emploierons la fonction clrscr pour effacer l'écran (voir partie suivante pour plus d'explications).
Sachant que l'option de formatage permettant d'afficher un entier (signed short, pour être précis) est %d, voici un exemple :

```
// C Source File
// Created 08/10/2003; 14:17:20

#include <tigcclib.h>

// Main Function
void _main(void)
{
    clrscr();
    printf("la valeur est : %d !", 100);
    ngetchx();
}
```

Exemple Complet

Lorsque nous exécuterons ce programme, nous aurons à l'écran le message suivant : "La valeur est : 100 !".

Sachant qu'il est possible de remplacer une valeur par une variable contenant une valeur, que nous avons vu comment créer et initialiser des variables, et que l'option de formatage permettant d'afficher un flottant est %f, voici un autre exemple :

```
// C Source File
// Created 09/10/2003; 12:33:35

#include <tigcclib.h>

// Main Function
void _main(void)
{
    float a = 13.456;
    clrscr();
    printf("la valeur est : %f !", a);
    ngetchx();
}
```

Exemple Complet

A l'écran sera affiché... ce à quoi nous pourrions nous attendre.

B: Remarque au sujet de l'importance de la case

Comme vous avez pu le remarquer, nous avons ici utilisé la fonction clrscr, alors que, quelques chapitre auparavant, nous avions employé le ROM_CALL ClrScr (notez l'absence de majuscules dans le premier cas, et leur présence dans le second !).

Le ROM_CALL ClrScr permet, comme nous l'indique la documentation de TIGCC, d'effacer l'écran.
La fonction clrscr, qui n'est pas intégrée à la ROM, et qui prend donc un peu de place dans le programme, à partir du moment où on l'utilise, fait plus que cela : elle réinitialise la position d'écriture utilisée par printf au coin supérieur gauche de l'écran, soit, au pixel de coordonnées (0 ; 0).

Pour avoir la preuve de cette différence, essayez ce programme :

```
// C Source File
// Created 09/10/2003; 12:33:35

#include <tigcclib.h>

// Main Function
void _main(void)
{
    clrscr();
    printf("Salut");
    ngetchx();
    clrscr();
    printf("toi.");
    ngetchx();
}
```

Exemple Complet

Et, maintenant, faites la même chose, en remplaçant les deux appels à clrscr par des appels à ClrScr.
Comme vous pouvez le constater, la différence est... visible.

Certes, vous pourriez être tenté d'utiliser toujours clrscr, et jamais ClrScr... Cela dit, clrscr effectuant plus d'opérations, il est possible qu'elle soit plus lente... et, puisqu'elle n'est pas inclue à la ROM, il est évident qu'elle fera grossir la taille du programme.
Pour résumer mes pensées, utilisez ce dont vous avez besoin ; ni plus, ni moins. Si vous avez besoin d'utiliser printf, il est fort probable que vous souhaitiez réinitialiser la position d'affichage... Dans ce cas, utilisez clrscr. Si vous ne comptez pas utiliser printf, autant appeler ClrScr, qui est tout aussi adaptée, dans ce cas.

Je tenais juste à profiter de cette occasion pour prouver l'intérêt qu'il y a à prêter attention aux majuscules, afin que vous compreniez bien que ce que j'ai dit plus haut n'était pas du vent.

II:\ Utilisation de la valeur de retour d'une fonction

A: Généralités

Une fonction, quelle qu'elle soit, ROM_CALL, fonction définie par vous-même, ou inclue dans TIGCC, a un type de retour. Ce type de retour est précisé devant le nom de la fonction à sa déclaration.
Ce type de retour peut être void, ce qui signifie "vide", ou, plutôt, "néant", auquel cas la fonction de retournera pas de valeur, ou un autre type, parmi ceux que nous avons déjà vu ou ceux qu'il nous reste à étudier.

Par exemple, la fonction ClrScr, dont nous avons parlé juste au dessus, ne retourne rien : son prototype est, si l'on se fie à la documentation, le suivant :

```
void ClrScr(void);
```

La fonction ngetchx, elle, renvoie une valeur de type short, comme nous l'indique son prototype, ainsi défini :

```
short ngetchx(void);
```

Pour une fonction retournant quelque chose, il est possible de récupérer la valeur retournée dans une variable de même type que celle-ci, au moment où on l'appelle (ou, pour être plus précis, au moment où elle s'achève).
Pour cela, nous écrirons le nom de la variable, le symbole d'affectation, et l'appel de la fonction, comme nous aurions fait pour une simple valeur.

Notons que, comme nous l'avons déjà fait, il est possible d'appeler une fonction retournant une valeur sans chercher à utiliser cette valeur !

B: Exemple : ngetchx

Par exemple, pour la fonction ngetchx, nous pourrions agir ainsi :

```
short key;
key = ngetchx();
```

Nous déclarons une variable de type short, ce qui correspond au type de retour du ROM_CALL ngetchx, et, ensuite, nous appelons ngetchx, en précisant que sa valeur de retour doit être mémorisée dans cette variable.

Ensuite, il nous est possible d'afficher le code de cette touche, comme dans l'exemple ci-dessous, dont est tiré l'extrait que nous venons de citer :

```
// C Source File
// Created 08/10/2003; 14:17:20

#include <tigcclib.h>

// Main Function
void _main(void)
{
    clrscr();
    printf("Pressez une touche...");

    short key;
    key = ngetchx();

    printf("\nCode de la touche : %d.", key);

    ngetchx();
}
```

Exemple Complet

Le programme résultat va demander à l'utilisateur d'appuyer sur une touche ; une fois que ce sera fait, il affichera le code correspondant à cette touche.
La code touche renvoyé par ngetchx est généralement le même que celui renvoyé par GetKey en BASIC, mais il différe pour quelques touches ; en particulier, pour les touches directionnelles, il me semble. D'ailleurs, pour ces touches-là, les codes sont inversés selon que le programme tourne sur une TI-89 ou sur une TI-92+ !

C: Remarque concernant l'imbrication d'appels de fonctions

Pour clore ce chapitre, nous finirons par remarquer qu'il est possible d'imbriquer des appels de fonctions.
Pour l'exemple que nous avons pris juste au-dessus, cela nous permettrait d'éviter l'utilisation de la variable key.
Cela dit, le code deviendra peut-être moins lisible... tout en regroupant mieux les parties logiques... question de goût, je suppose.
Toujours est-il qu'il est parfaitement possible d'écrire quelque chose de ce genre :

```
// C Source File
// Created 08/10/2003; 14:17:20

#include <tigcclib.h>

// Main Function
void _main(void)
{
    clrscr();
    printf("Pressez une touche...");

    printf("\nCode de la touche : %d.", ngetchx());

    ngetchx();
}
```

Exemple Complet

Le résultat sera exactement le même que celui obtenu avec l'écriture précédemment présentée.
Comme vous pouvez le remarquer, dans ce genre de situation, c'est l'appel de fonction le plus interne qui est effectué en premier, et l'on termine par le plus externe, puisque celui-ci à besoin du premier.

Voila un chapitre de plus d'achevé. Un chapitre assez simple, et assez facile à supporter, je pense, même s'il nous a permis d'apprendre les bases de l'utilisation d'une fonction extrêmement utile, ainsi que l'emploi de la valeur de retour d'une fonction, chose qu'il est nécessaire de maîtriser.
Je ne peux que vous encourager à consulter la documentation de printf, afin de découvrir les nombreuses options de formatage que cette fonction propose ; nous travaillerons sans nulle doute avec certaines d'entre-elles dans le futur.
Le chapitre prochain nous permettra d'apprendre à calculer en C, maintenant que nous sommes en mesure d'afficher le résultat d'une opération...