Chapitre 5: Les fonctions

5.1 Fonctions et procédures

Dans un code source, il est parfois nécessaire de reproduire plusieurs fois les mêmes actions. Pour éviter de taper plusieurs les instructions identiques, on factorise le code en morceau appelés fonctions. Cela permet d'avoir un code plus maintenable (si une modification est à faire, on ne la fait qu'à un endroit) et plus lisible.

5.1.1 Définition

1. Une fonction est une liste d'instructions
2. Elle peut prendre des paramètres en entrée et peut retourner une valeur.
3. Un paramètre est une variable locale à la fonction qui aura une valeur définie lors de l'appelle de celle-ci
4. Une fonction qui n'a pas de retour est appelée procédure.

5.1.2 Syntaxe

```
type_retour nomFonction (type1 param1, type2 param2,…) {
  //corps de la fonction
}
```

- Le nomFonction est ce qui permettra d'identifier la fonction et l'appelé dans le reste du programme
- Les param1, param2, etc. sont la liste des paramètres passé à la fonction. Ils doivent être typés et sont séparés par des virgules. Si il n'y a aucun paramètre en entrée, on ne précisera rien entre les parenthèses ou on mettra le mot clef void
- Le corps de la fonction est l'ensemble des instructions de la fonction. Il exploitera des variables locales, les paramètres de la fonctions et éventuellement des variables globales.
- Le type_retour est simple le type de la valeur retourner en fin de fonction. Si la fonction ne retourne rien (qu'il s'agit donc d'une procédure), on indiquera le mot clef void

5.2 Passages de paramètres

5.2.1 Passage par valeurs

En C, lorsqu'on utilise les paramètre d'une fonction, on fait un passage par valeur. C'est à dire que la valeur transmise sera une copie de la valeur initiale, et donc sera stockée dans une variable locale à la fonction. La conséquence est que si une fonction de renvoi aucune valeur, les valeurs passée en entrée lors de l'appelle d'une fonction ne seront pas modifiés à l'endroit ou la fonction est appelée.

Pour mieux comprendre ça, nous allons utiliser l'exemple suivant :

```
void incremente(int nbr) {
   nbr++;
}
int main() {
   int x=2;
   incremente(x);
   printf("La valeur de x est %d\n",x);
}
```

Dans cet exemple, la fonction incremente utilise le paramètre en entrée nbr de type int. nbr est donc une variable locale à la fonction.

Lorsque celle-ci est appelée dans la fonction main, on lui passe la valeur de x. Cette valeur est copiée dans la variable locale nbr.

A la fin du traitement, lors de l'affichage, la valeur de x est toujours 2. Seul la variable locale nbr de la fonction incremente aura changer de valeur.

5.3 Portés des variables, visibilités et durée de vie

1. Toutes variables déclarées en dehors de la fonctions main sont des variable globales
2. Une variable globale est limitée à la partie de son programme source suivant sa déclaration
3. L'emplacement mémoire d'une variable globale est fixe (allocation statique)
4. Une variable définie dans une fonction est dite locale
5. La portée d'une variable locale est la fonction où elle est définie
6. L'emplacement mémoire de ce type de variable est défini à l'entrée de la fonction et libéré à la sortie.