

Mise à niveau en C

Les fonctions en C

Enseignant: P. Bertin-Johannet

Qu'est ce qu'une fonction

- Une fonction associe un nom à un bloc de code
- On peut **appeler** la fonction pour executer ce bloc de code :
- Nous avons déjà utilisé printf et scanf qui sont des fonctions

Arguments d'une fonction

- Une fonction peut **accepter** des arguments.
 - Ce sont des variables dont la valeur sera copiée et utilisée par la fonction.
 - Pour cela on passe les arguments entre parenthèses après le nom de la fonction.
- Par exemple lorsqu'on utilise la fonction `printf` on passe en argument une chaîne de format ainsi qu'une série d'objets à afficher.

```
printf("Bonjour %s tu as %d ans\n", name, age);
```

Un appel de fonction est une expression

- Une fonction peut renvoyer une valeur, l'appel à la fonction se comportera comme n'importe quelle expression.
- Par exemple on peut utiliser la fonction `strlen` pour connaître la longueur d'une chaîne de caractères ou la fonction `pow` pour calculer des puissances.

```
int longueur_bonjour = strlen ("bonjour");
```

```
float trois_puissance_quatre = pow(3, 4);
```

Déclaration d'une fonction

- Comme pour les variables, avant de l'utiliser, on doit déclarer le type d'une fonction en précisant:
 - Le **type de la valeur renvoyée** par la fonction
 - Le **nom** de la fonction
 - Les **types et noms des arguments** de la fonction
- Par exemple, les fonctions `strlen` et `pow` sont déclarées ainsi:

```
int strlen(char* text);  
double pow(double x, double y) ;
```

Définition d'une fonction

- Une fois la fonction déclarée, nous pouvons définir son comportement.
- Pour cela on écrit entre accolades le code exécuté par la fonction.
- On peut utiliser le mot clé return pour renvoyer une valeur.
- Par exemple une fonction add qui ajoute 3 entiers peut être définie ainsi:

```
int add(int a, int b, int c){  
    return a + b + c;  
}
```

Fonctionnement des arguments

- Les arguments d'une fonction sont des variables qui n'existent que dans le code de celle ci.
- Par exemple dans le code ci-dessous, la variable **b** existe uniquement dans le code de la fonction main et la variable **a** uniquement dans le code de la fonction triple

```
int triple (int a){  
    return a + a + a;  
}  
  
int main () {  
    int b = 5;  
    printf ("%d \n" , triple(b)) ;  
}
```

Fonctionnement des arguments

- Lors de l'appel à une fonction, les valeurs passées sont **copiées** dans les arguments, il n'est donc pas possible de modifier une variable extérieure depuis une fonction.
- Par exemple dans le code ci-à-côté, la valeur **b** n'est pas modifiée par la fonction ajoute .

```
int ajoute ( int a) {  
    a = 6;  
    return a + 5;  
}  
  
int main () {  
    int b = 5;  
    int c = triple(b) ;  
    printf("%d %d \n" , b, c) ;  
}
```


Variable statique

- Il est possible de créer une variable dite **statique** dans une fonction, cette variable existera pour toute la durée du programme.
- La variable sauvegardera sa valeur entre plusieurs appels à la fonctionLa variable sauvegardera sa valeur entre plusieurs appels à la fonction.
- Pour la déclarer, on écrit le mot clé **static** avant le type de la variable, l'initialisation s'exécutera au premier appel à la fonction.

// cette ligne de code n'est exécutée que la première fois que la fonction est appelée

```
static char c = 'k';
```

Variable statique - Exemple

```
int fonc(int a) {  
    static char c = 7;  
    c ++;  
    return a + c;  
}  
  
int main(){  
    printf("%d \n", fonc(3));  
    printf("%d \n", fonc(3));  
}
```

Variable statique - Exemple

```
int fonc(int a) {  
    static char c = 7;  
    c ++;  
    return a + c;  
}  
  
int main(){  
    printf("%d \n", fonc(3));  
    printf("%d \n", fonc(3));  
}
```

Sortie après le premier appel : 11

Variable statique - Exemple

```
int fonc(int a) {  
    static char c = 7;  
    c ++;  
    return a + c;  
}  
  
int main(){  
    printf("%d \n", fonc(3));  
    printf("%d \n", fonc(3));  
}
```

Sortie après le premier appel : 11

Sortie après le second appel : 12

Mémoire et fonctions

- Lorsqu'on appelle une fonction, un espace mémoire à la suite de celui utilisé par la fonction courante est alloué le temps de son execution.
- Cet espace mémoire contient assez d'octets pour:
 - Les arguments de la fonction.
 - Les variables locales à la fonction.
 - Des informations permettant de reprendre le programme dans son état avant l'appel de la fonction.
- À la suite de cet espace mémoire, un autre espace sera alloué pour la fonction appelée.

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
	Octet 1
	Octet 2
	Octet 3
	Octet 4
	Octet 5
	Octet 6
	Octet 7
	Octet 8
	Octet 9
	Octet 10
	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
Réservé pour la fonction main	Octet 1
	Octet 2
	Octet 3
	Octet 4
	Octet 5
	Octet 6
	Octet 7
	Octet 8
	Octet 9
	Octet 10
	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
	Octet 3
	Octet 4
	Octet 5
	Octet 6
	Octet 7
	Octet 8
	Octet 9
	Octet 10
	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
	Octet 3
	Octet 4
Contexte d'appel	Octet 5
	Octet 6
	Octet 7
	Octet 8
	Octet 9
	Octet 10
	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
	Octet 3
	Octet 4
Contexte d'appel	Octet 5
	Octet 6
	Octet 7
	Octet 8
6	Octet 9
	Octet 10
	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
	Octet 3
	Octet 4
Contexte d'appel	Octet 5
	Octet 6
	Octet 7
	Octet 8
6	Octet 9
	Octet 10
7	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
13	Octet 3
	Octet 4
Contexte d'appel	Octet 5
	Octet 6
	Octet 7
	Octet 8
6	Octet 9
	Octet 10
7	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
13	Octet 3
	Octet 4
	Octet 5
	Octet 6
	Octet 7
	Octet 8
6	Octet 9
	Octet 10
7	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
13	Octet 3
	Octet 4
Contexte d'appel	Octet 5
	Octet 6
	Octet 7
	Octet 8
6	Octet 9
	Octet 10
7	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
13	Octet 3
	Octet 4
Contexte d'appel	Octet 5
	Octet 6
	Octet 7
	Octet 8
14	Octet 9
	Octet 10
7	Octet 11

Exemple

Programme:

```
int fonc(int a){  
    char c = 7;  
    return a + c;  
}  
  
int main(){  
    int b = 5;  
    int a = fonc(b + 1);  
    affiche(a + 1);  
}
```

Mémoire:

Valeurs	Octets
5	Octet 1
	Octet 2
13	Octet 3
	Octet 4
	Octet 5
	Octet 6
	Octet 7
	Octet 8
14	Octet 9
	Octet 10
7	Octet 11

Mise en pratique

```
printf("< TP 3 >  
-----  
      \  ^  ^  
        ^  
      \ (oo)\_____  
        (__) \      )\ /\   
           ||-----w ||  
           ||         ||  
")
```