Programmation C

12 - Les fonctions

fonction main et printf

```
include <stdio.h>

int main()
{
   printf(“message 1”);
   printf(“message 2”);
   ...
   return 0;
}
```

```
void printf()
{
   …….
}
```

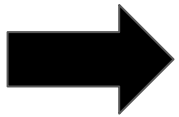
Utilisation des fonctions

```
int main()
{
   printf(“Salut !”);
   printf(“Formation C”);

   printf(“Salut !”);
   printf(“Formation C”);

   printf(“Salut !”);
   printf(“Formation C”);

   return 0;
}
```



```
int main()
{
   maFonction();
   maFonction();
   maFonction();
   return 0;
}
```

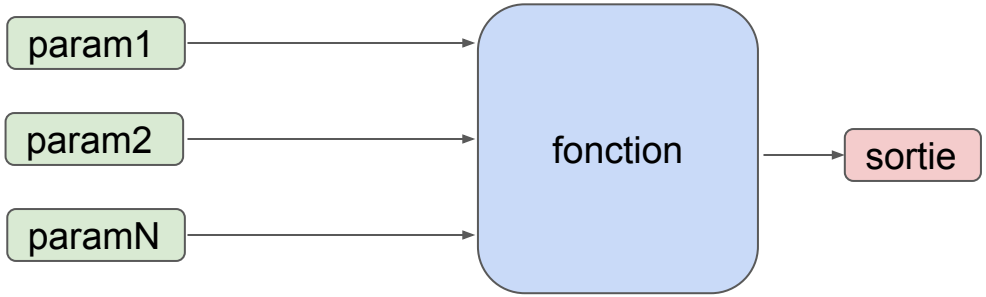
```
void maFonction()
{
   printf(“Salut !”);
   printf(“Formation C”);
}
```

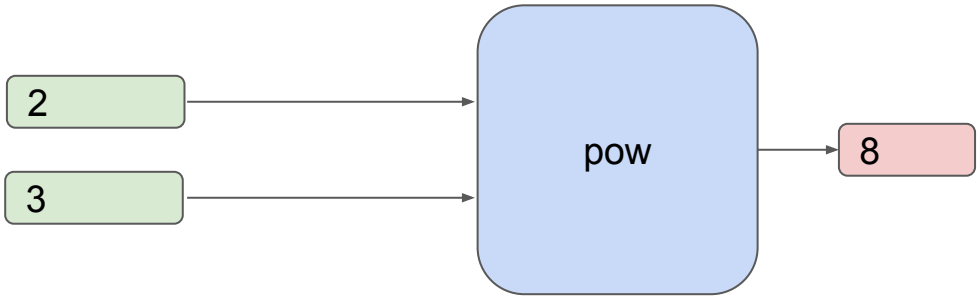
Utilisation des fonctions

```
int main()
{
   maFonction();
   maFonction();
   maFonction();
   return 0;
}
```

```
void maFonction()
{
   printf(“Salut !”);
   printf(“Formation C”);
}
```

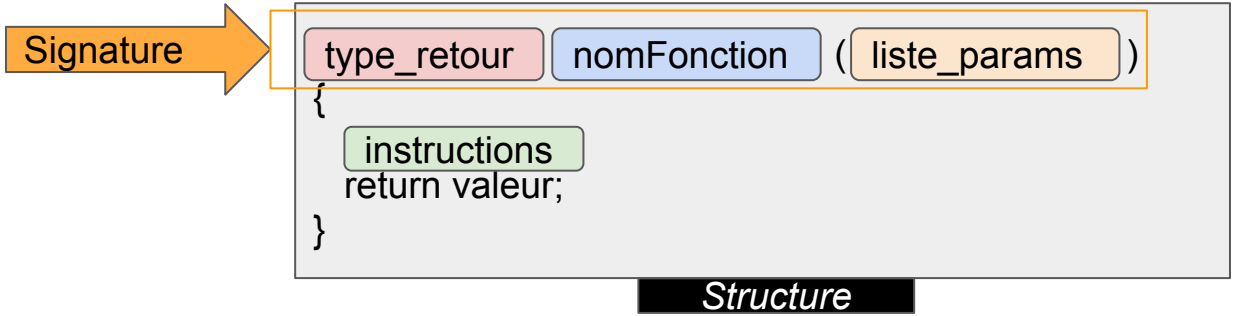
les fonctions





2^3 = 2 * 2 * 2 = 8

Les fonctions




```
int main()
{
   return 0;
}
```

Exemple 1

```
void maFonction()
{
   ...
}
```

Exemple 2

```
void afficherAge(int age)
{
   printf("tu a %d ans\n", age);
   return;
}
```

Exemple 3

```
int somme(int var1, int var2)
{
   int total= var2 + var1;
   return total;
}
```

Exemple 4

Règles de nommage des fonctions

Règles	Exemples
Pas de nombre en début de nom	2fonction
Pas de caractères spéciaux à l'exception de _	nom-de-fonction
Pas de mots réservés en c (ex type de variable)	for
Pas d'accent (à, é, è , …)	début
Pas d'espace	nom de fonction
Nom explicite (très important pour la lisibilité)	nom_de_fonction
Minuscule sauf 1ere lettre mots séparateurs	nomDeFonction

Paramètres de fonction

```
type_retour nomFonction (type param1, type param2, …)
{
   instructions
   return valeur;
}
```

```
void somme(int var1, int var2)
{
   int somme = var1+var2;
   printf("%d+%d=%d", var1, var2, somme);
}
```

```
int main()
{
   int ma_variable = 2;
   somme(3,1);
   somme(3, ma_variable);
}
```

```
3+1=4
3+2=5
```

Fonctions: paramètres optionnels

```
void FonctionCorecte(int var=0)
{
}
```

```
void FonctionIncorecte(int var1=0, int var2)
{
}
```

```
void FonctionCorecte(int var1=0, int var2=2)
{
}
```

```
void FonctionCorecte(int var1, int var2)
{
}
```

```
void FonctionCorecte(int var1, int var2=2)
{
}
```

```
void FonctionCorecte()
{
}
```

Les returns

```
void balrog()
{
   return;
   printf(“Vous ne passerez pas!”);
}

int main()
{
   printf(“Salut \n”);
   balrog();
   return 0;
}
```

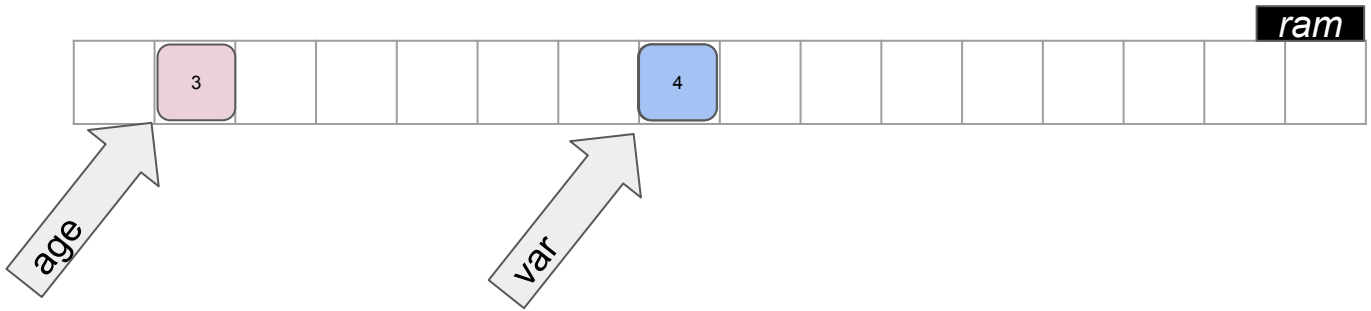
Exemple

```
Salut
```

Passage de paramètres par copie

```
int main()
{
   int age= 3;
   anniversaire(age);
}
```

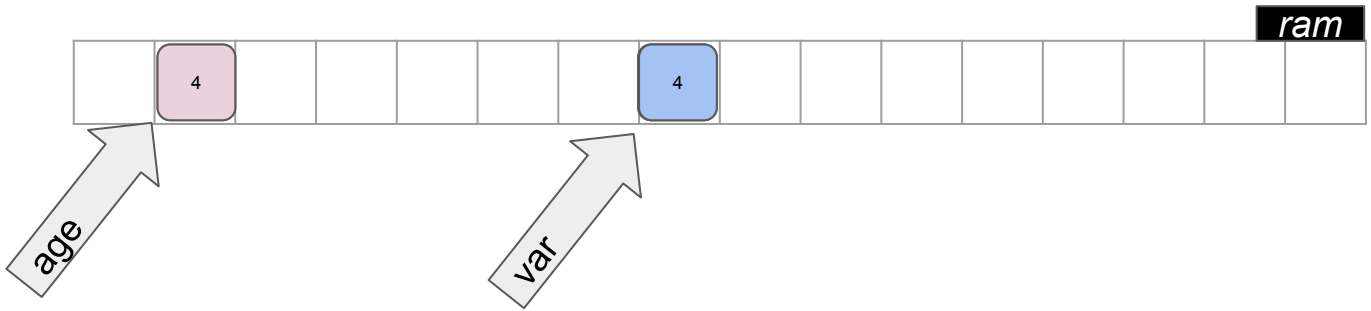
```
void anniversaire(int var)
{
   var++;
   printf("%d", var);
}
```



Récupération du retour de fonction

```c
int main()
{
   int age= 3;
   age = anniversaire(age);
}
```

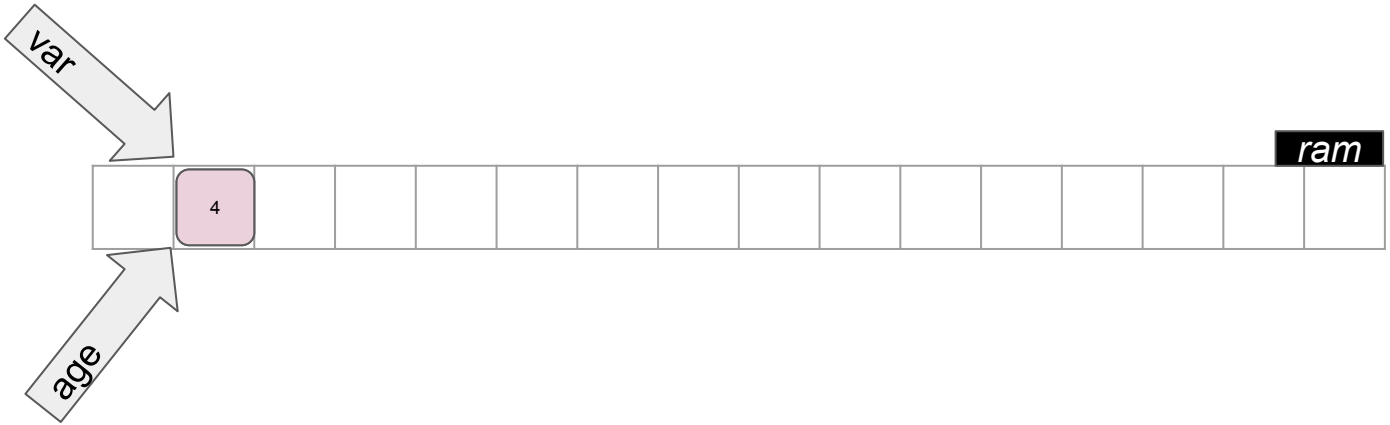
```c
int anniversaire(int var)
{
   var++;
   return var;
}
```



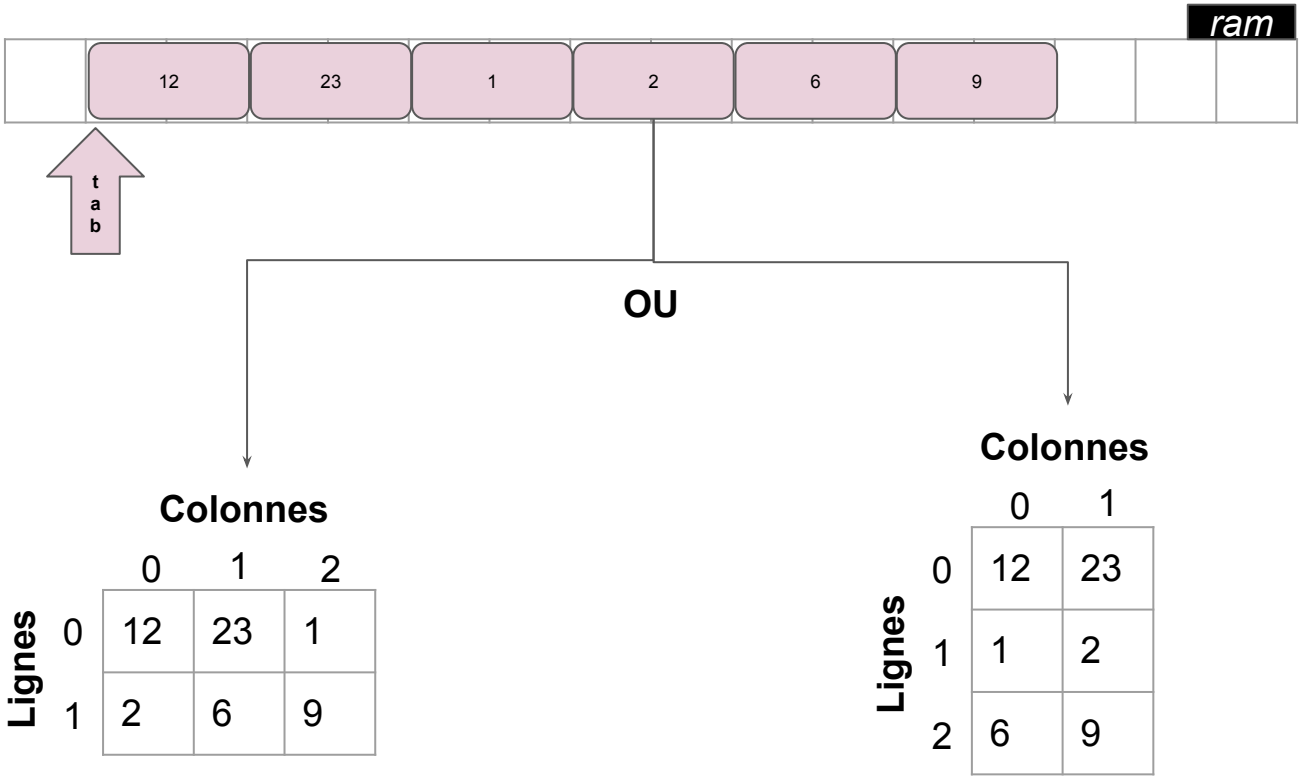
Passage de paramètres par pointeur

```
int main()
{
   int age= 3;
   anniversaire(&age);
}
```

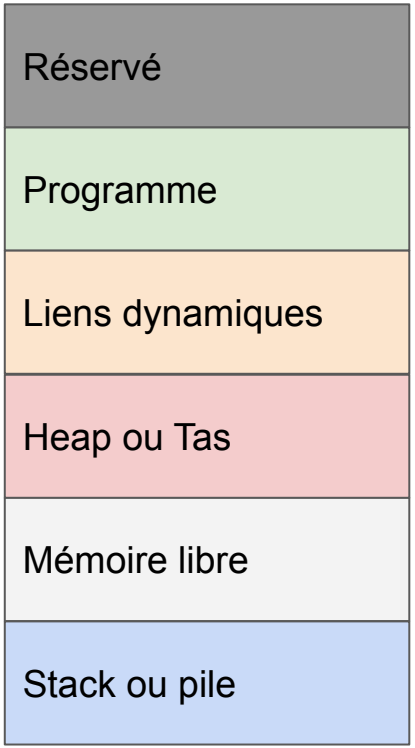
```
void anniversaire(int* var)
{
   *var++;
   printf("%d", *var);
}
```



structure d’un tableau



Mémoires et programme



Zone réservée à d'autres programmes

Zone des instructions et données statique globales

Zone mémoire pour charger dynamiquement les librairies

Données allouées dynamiquement

Mémoire libre, utilisable pour étendre le tas et la pille

Zone mémoire de données spécifiques à une procédure

La Stack (pile)

Stack ou pile

Zone mémoire de données spécifiques à une procédure

Stack

- frame x
- frame ...
- frame 2
- frame 1
- frame 0

frame X — Procédure ou fonction (paramètres, variables, valeur de retour)

frame 1 — Correspond à la fonction main

La Stack: exemple

```
void MaFonction()
{
   int var = 12;
   var = AutreFonction(2, 63);
}

int AutreFonction(int var1, int var2)
{
    return var1 + var2;
}

int main()
{
   char toto='E';
   MaFonction();
   return 0;
}
```

Exemple