Langage C

Les fonctions

François Cuvelier

Laboratoire d'Analyse Géométrie et Applications
Institut Galilée
Université Paris XIII.

26 septembre 2012

Plan

Généralités

Les fonctions permettent

- d'automatiser certaines tâche répétitives au sein d'un même programme,
- d'ajouter à la clarté d'un programme,
- l'utilisation de portion de code dans un autre programme,
- de faciliter le debuggage et la validation,
- ...

Plan

Syntaxe

Une fonction est composée de son **prototype** (ou en-tête) et de son **corps**
Le prototype permet au compilateur de vérifier la validité des appels à cette fonction

```
1 int f(int a, int b)
2 {
3    int x;
4    x = a+b;
5    return x;
6 }
```

Plan

Définition de fonctions

Syntaxe

```
<type> <NomFonction>(<arg1>,...,<argN>){
  <declarations eventuelles>
  <instructions>
}
```

On appelle **prototype** ou **en-tête** d'une fonction la partie :

```
<type> <NomFonction>(<arg1>,...,<argN>)
```

La partie entre accolades est appelée **corps** de la fonction : c'est une instruction composée.

Plan

Prototype d'une fonction

Syntaxe

```
<type> <NomFonction>(<arg1>,...,<argN>)
```

- `<NomFonction>` : identifiant de la fonction,
- `<type>` : type retourné par la fonction. Ne peut être un tableau ou une fonction. Si la fonction ne retourne aucune valeur, la déclarer du type `void`.
- `<arg1>,...,<argN>` : arguments de la fonction. Doivent être fourni avec leur type. Si une fonction n'a pas d'argument, utiliser le type `void`.

Plan

Corps d'une fonction

Le corps d'une fonction est une instruction composée :

Syntaxe

```
 {
 <declarations eventuelles>
 <instructions>
}
```

A l'intérieur d'une instruction composée, il est impossible de définir une fonction.

Plan

Exemple

```
1 int f(int a, int b)
2 {
3    int x;
4    x = a+b;
5    return x;
6 }
```

Plan

Déclaration d'une fonction

A pour but d'avertir le compilateur qu'elle va etre utilisée et de l'informer du type de résultat, ainsi que du type des éventuels arguments.
On déclare un prototype de fonction dans deux cas :

- Utilisation d'une fonction définie dans un autre fichier.
- Utilisation d'une fonction définie plus loin dans le même fichier.

Syntaxe

```
<type> <NomFonction>(<arg1>,...,<argN>);
```

Plan

Exemple complet

```c
#include <stdio.h>

int f(int a, int b);

int main(void){
   int x,s=1;
   x=f(s,10);
   printf("x=%d\n",x);
   return 1;
}

int f(int a, int b)
{
   int x;
   x = a+b;
   return x;
}
```

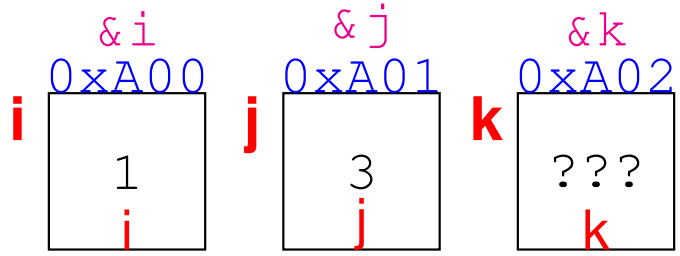
Plan

Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10 ▶  int i=1,j=3,k;
11    k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

```
int i=1,j=3,k;
```

Mémoire : main



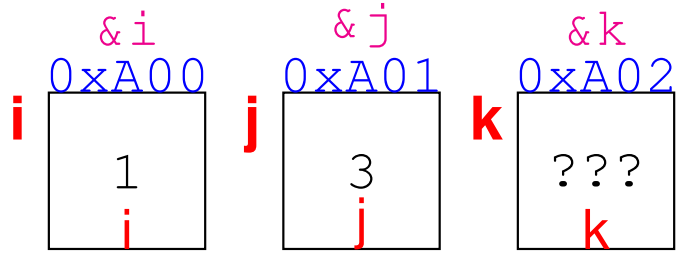
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10 ▶  int i=1,j=3,k;
11    k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

int i=1,j=3,k;

i initialisé à 1, j intialisé à 3, k non initialisé

Mémoire : main



Listing 4 – Exemple.c

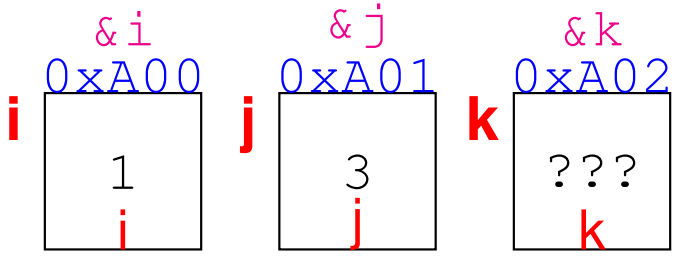
```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11 ▶  k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

k=f(i,j);

Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11 ▶  k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

Mémoire : main



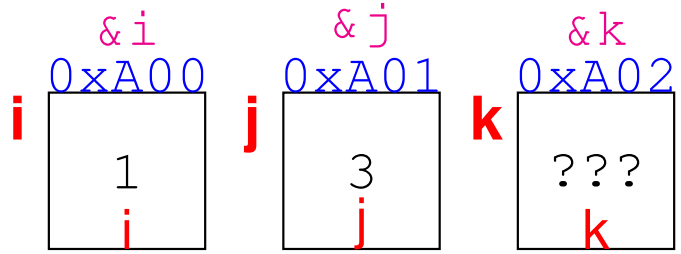
k=f(i,j);

appel de la fonction f avec la valeur de i en premier paramètre et la valeur de j en second

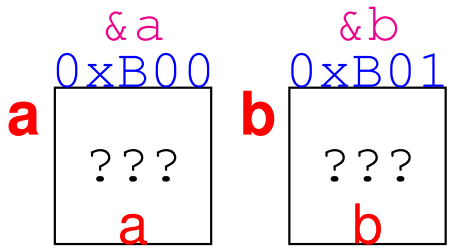
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3  int f(int a, int b){
4    int x;
5    x = a+2*b;
6    return x;
7  }
8
9  int main(){
10   int i=1,j=3,k;
11   k=f(i,j);
12   printf("k=%d\n",k);
13   return 1;
14 }
```

Mémoire : main



Mémoire : f

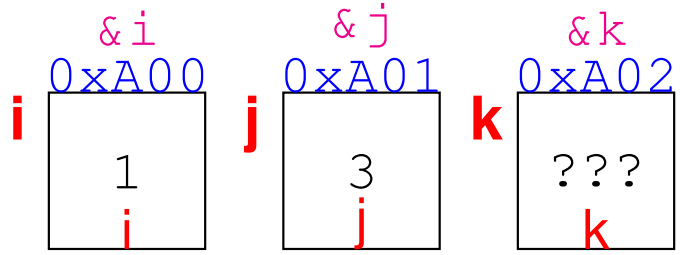



```
int f(int a, int b){
```

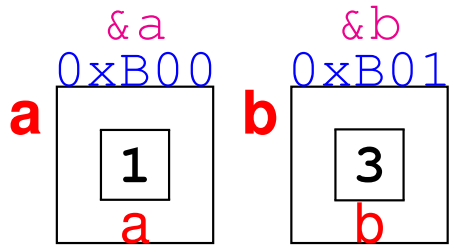
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3  int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11    k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

Mémoire : main



Mémoire : f



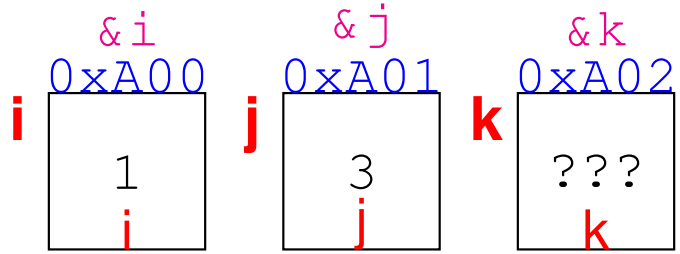
int f(int a, int b){

a est initialisé à 1 (valeur de i) et b est initialisé à 3 (valeur de j)

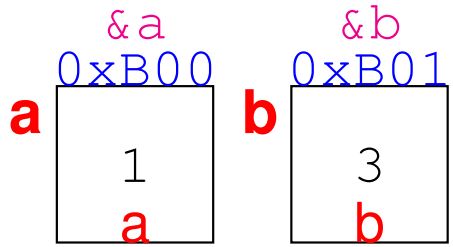
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4   ▶ int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11  ■ k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

Mémoire : main



Mémoire : f

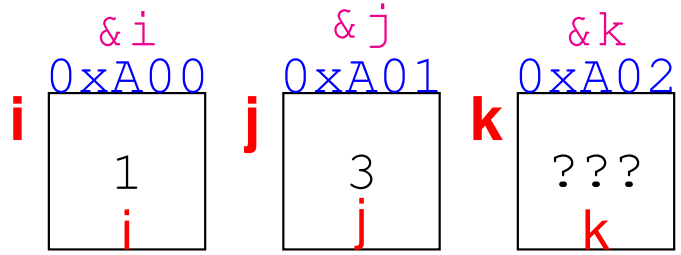


int x;

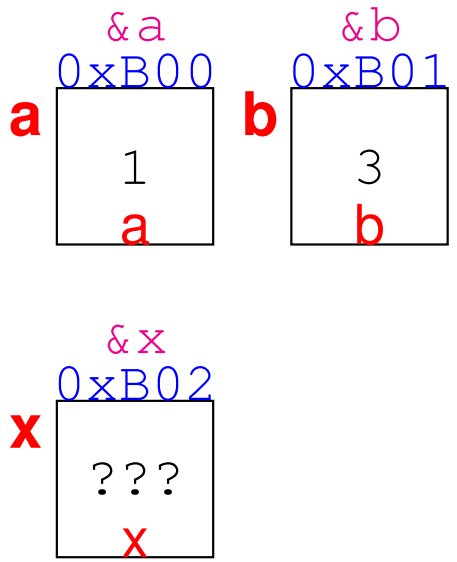
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4   int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11  k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

Mémoire : main



Mémoire : f



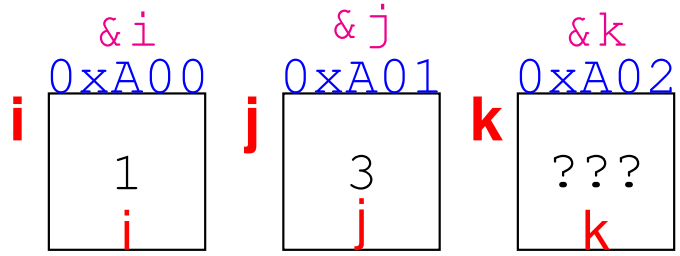
int x;

Déclaration de x (non initialisé)

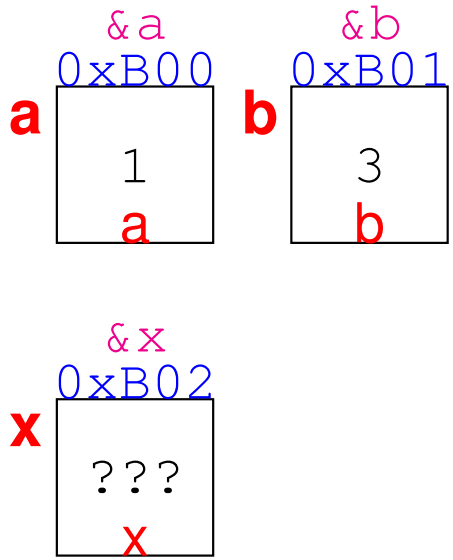
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5   ▶ x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11 ■  k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

Mémoire : main



Mémoire : f

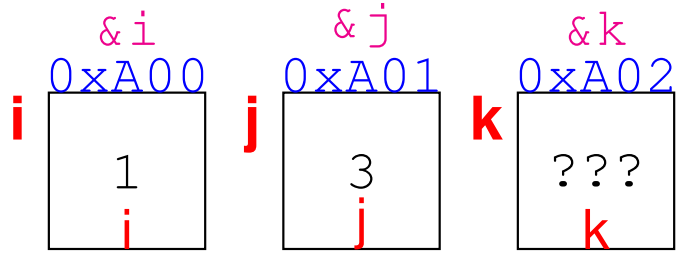


x=a+2*b;

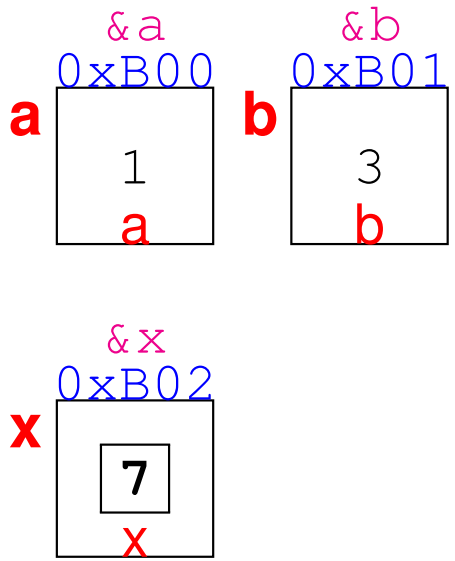
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5  ▶  x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11 ■  k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

Mémoire : main



Mémoire : f



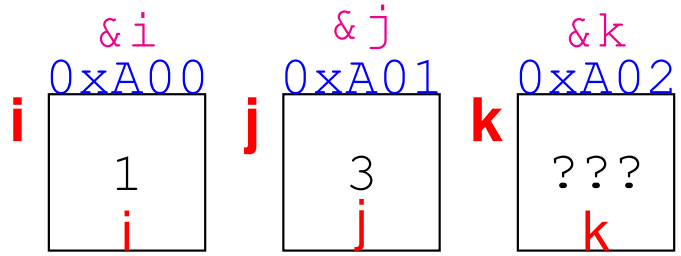
x=a+2*b;

x ← a+2*b=7

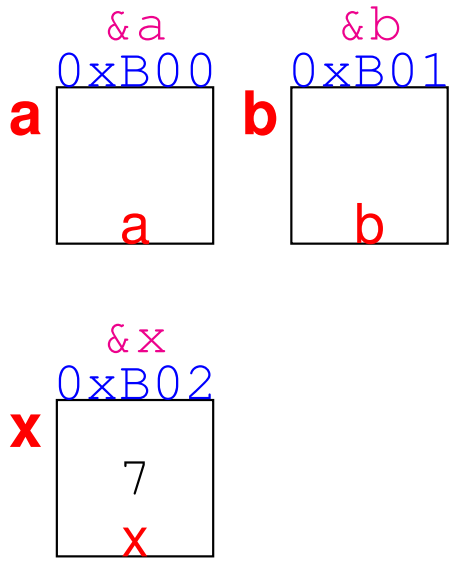
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6   ▶ return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11 ■  k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

Mémoire : main



Mémoire : f

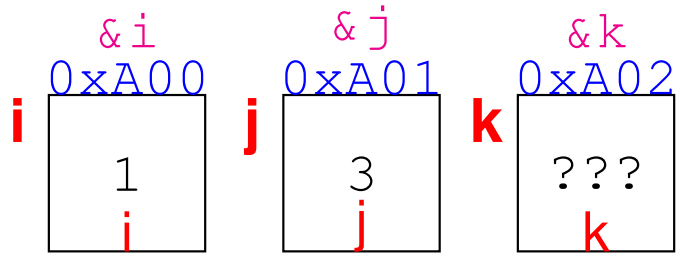


return x;

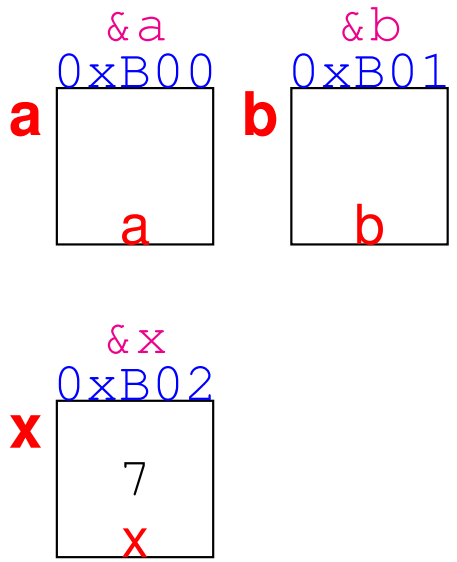
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3  int f(int a, int b){
4    int x;
5    x = a+2*b;
6    return x;
7  }
8
9  int main(){
10   int i=1,j=3,k;
11   k=f(i,j);
12   printf("k=%d\n",k);
13   return 1;
14 }
```

Mémoire : main



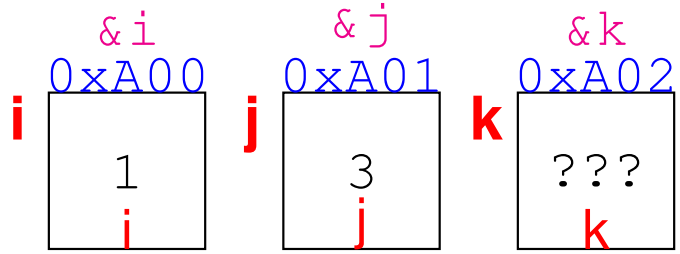
Mémoire : f



return x;

La valeur de x est retounée : 7

Mémoire : main

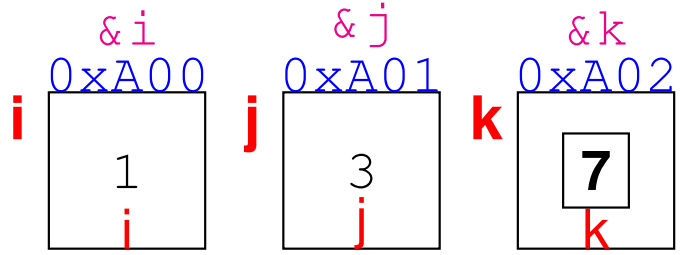


Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11 ▶  k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

k=f(i,j);

Mémoire : main



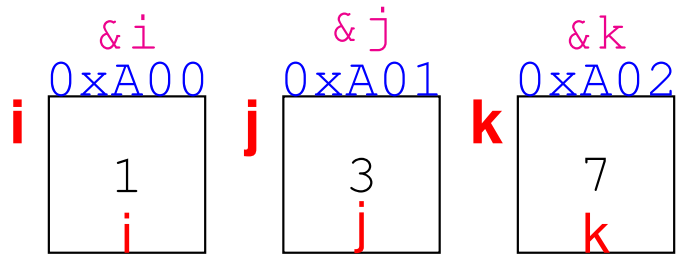
Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11 ▶  k=f(i,j);
12    printf("k=%d\n",k);
13    return 1;
14 }
```

k=f(i,j);

k reçoit la valeur retournée par f(i,j) : 7

Mémoire : main



Listing 4 – Exemple.c

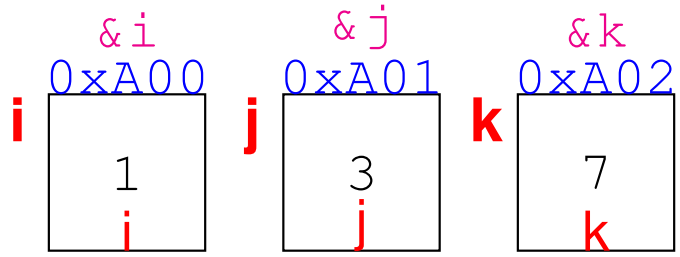
```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11    k=f(i,j);
12 ▶  printf("k=%d\n",k);
13    return 1;
14 }
```

```
printf ("k=%d\n",k);
```

Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11    k=f(i,j);
12 ▶  printf("k=%d\n",k);
13    return 1;
14 }
```

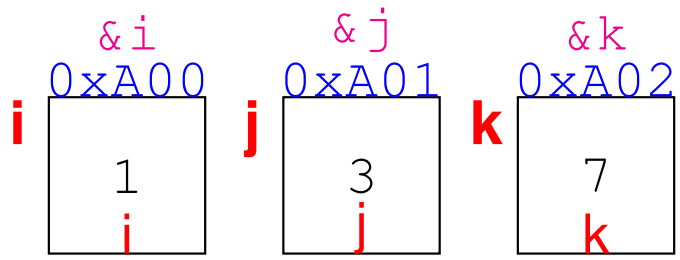
Mémoire : main



printf ("k=%d\n",k);

Affichage de : k=7

Mémoire : main

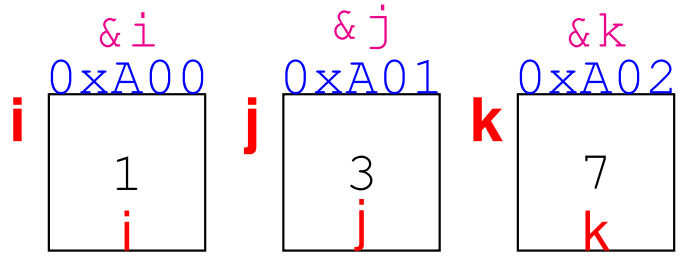


Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11    k=f(i,j);
12    printf("k=%d\n",k);
13 ▶  return 1;
14 }
```

return 1;

Mémoire : main



Listing 4 – Exemple.c

```
1  #include <stdio.h>
2
3   int f(int a, int b){
4     int x;
5     x = a+2*b;
6     return x;
7  }
8
9  int main(){
10    int i=1,j=3,k;
11    k=f(i,j);
12    printf("k=%d\n",k);
13 ▶  return 1;
14 }
```

return 1;

La fonction main est terminée et retourne 1

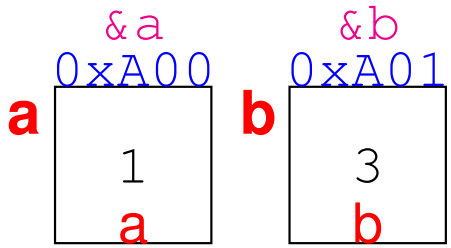
Plan

Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9  ▶  int a=1,b=3;
10    permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

```
int a=1,b=3;
```

Mémoire : main



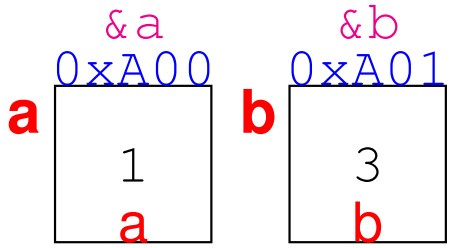
Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3    int x;
4    x=*a;
5    *a=*b;
6    *b=x;
7  }
8  int main(){
9  ▶ int a=1,b=3;
10   permut(&a,&b);
11   printf("a=%d,b=%d\n",a;b);
12   return 1;
13 }
```

int a=1,b=3;

a initialisé à 1, b intialisé à 3.

Mémoire : main

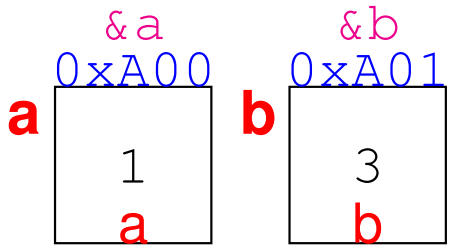


Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ▶  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

permut(&a,&b);

Mémoire : main



Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ▶  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

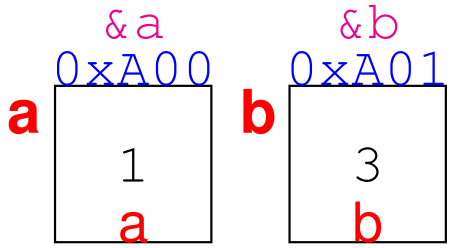
permut(&a,&b);

appel de la fonction permut avec &a $\Longleftrightarrow$ 0xA00 et &b $\Longleftrightarrow$ 0xA01

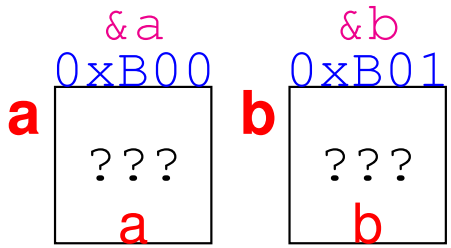
Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2  void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

Mémoire : main



Mémoire : permut

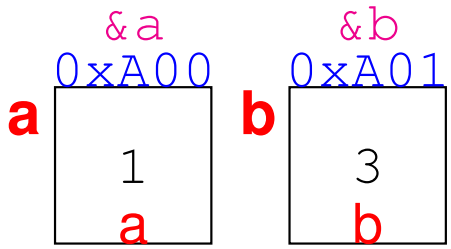



```
void permut(int *a,int *b){
```

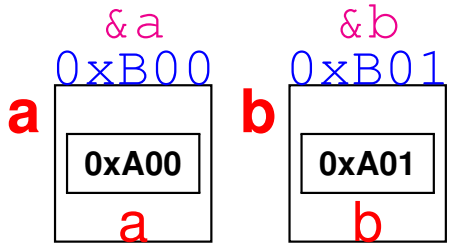
Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2  void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

Mémoire : main



Mémoire : permut

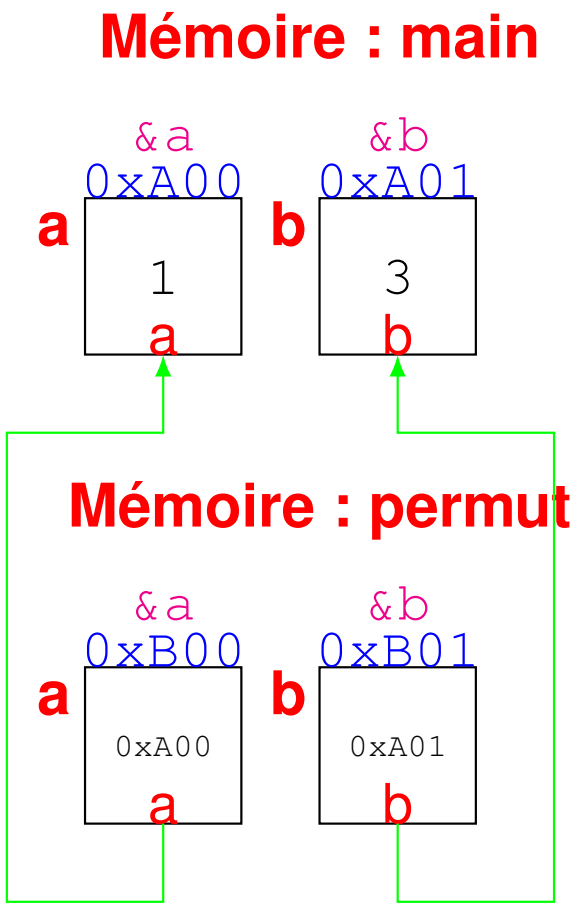


void permut(int *a,int *b){

a est initialisé à 0*xA*00 (valeur de &a du main) et b est initialisé à 0*xA*01 (valeur de &b du main)

Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2  void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

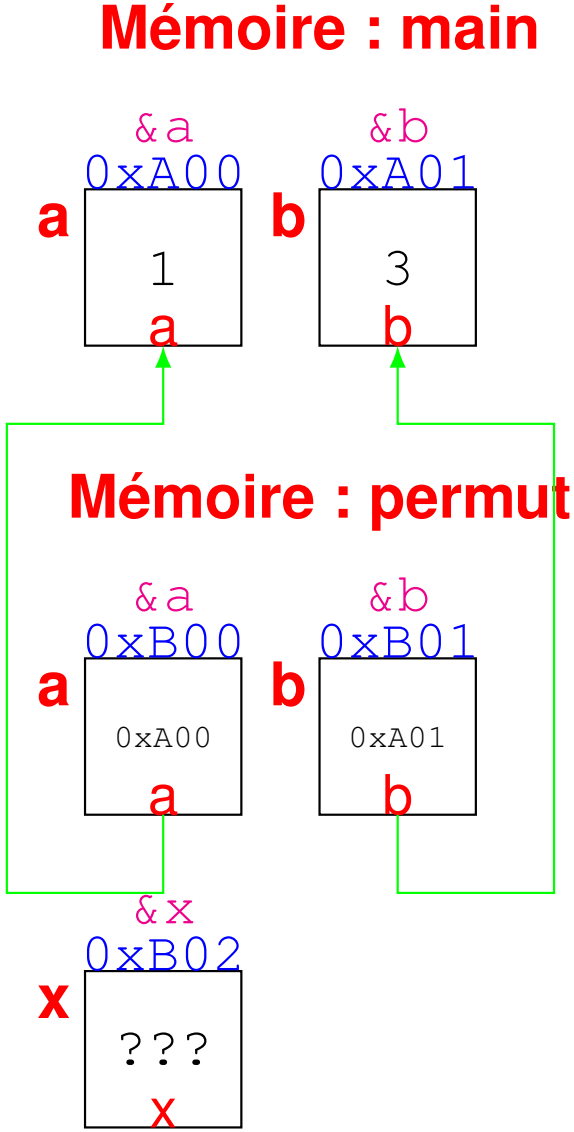


void permut(**int** *a,**int** *b){

*a (permut) $\iff$ a (main) $\iff$ 1 et *b (permut) $\iff$ b (main) $\iff$ 3

Listing 4 – Exemple2.c

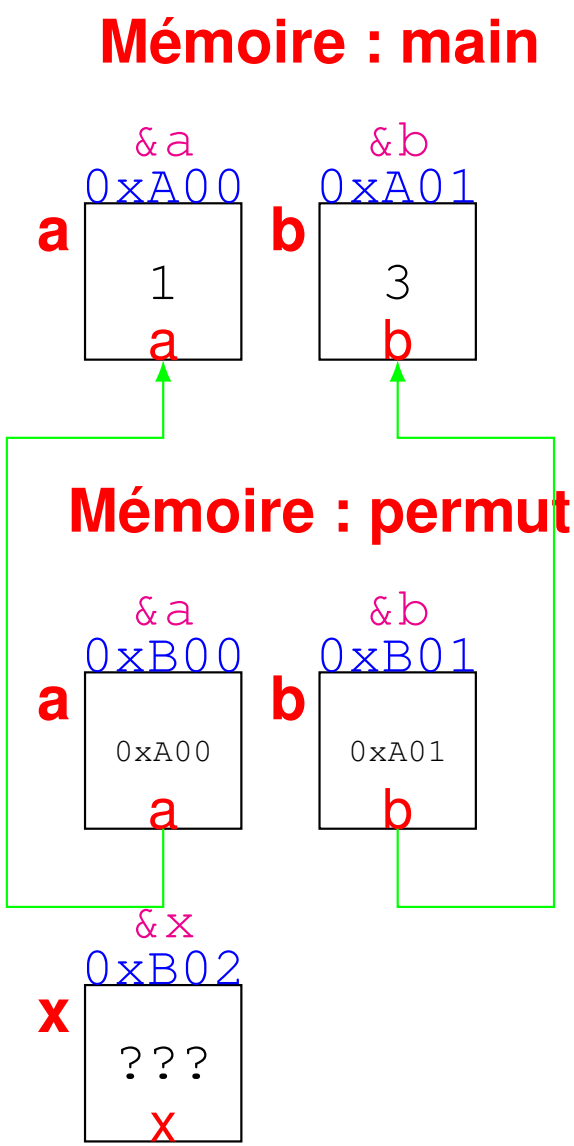
```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3   int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```



int x;

Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3   int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

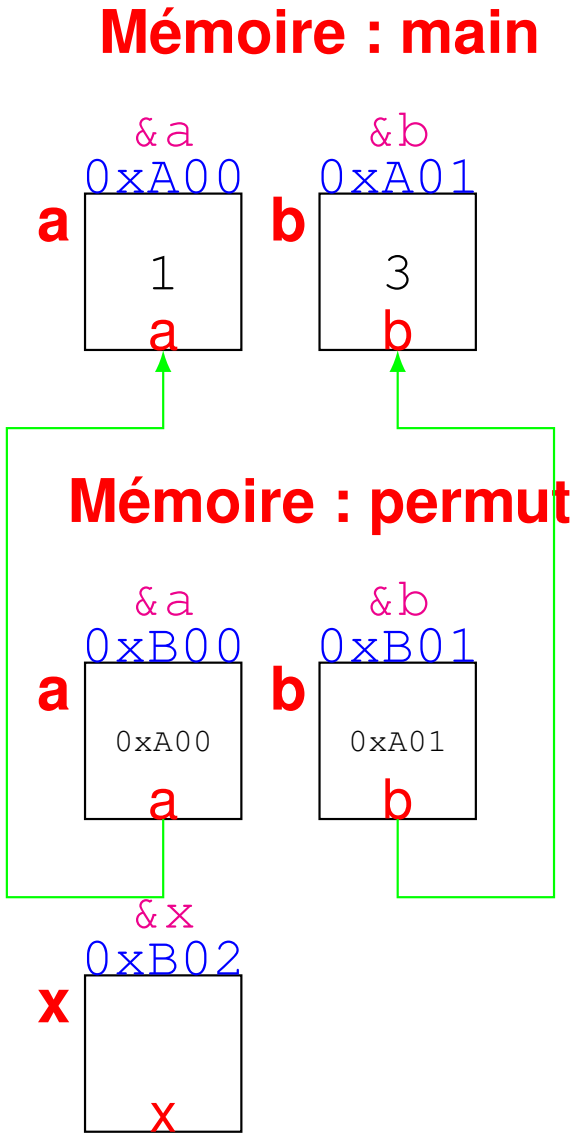


int x;

Déclaration de x (non initialisé)

Listing 4 – Exemple2.c

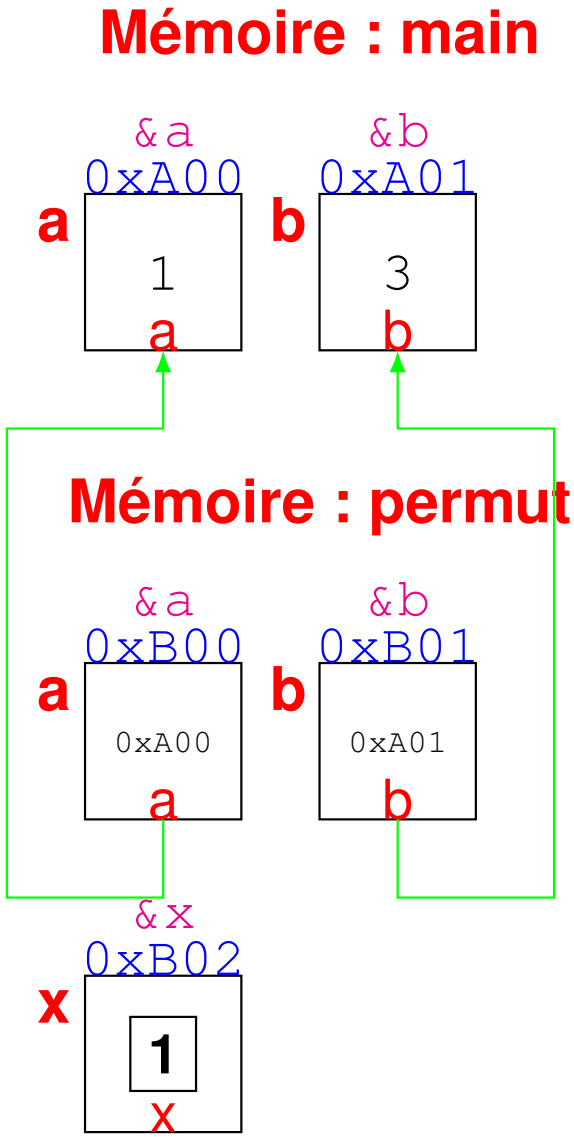
```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4  ▶  x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ■  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```



x=*a;

Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4  ▶  x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ■  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

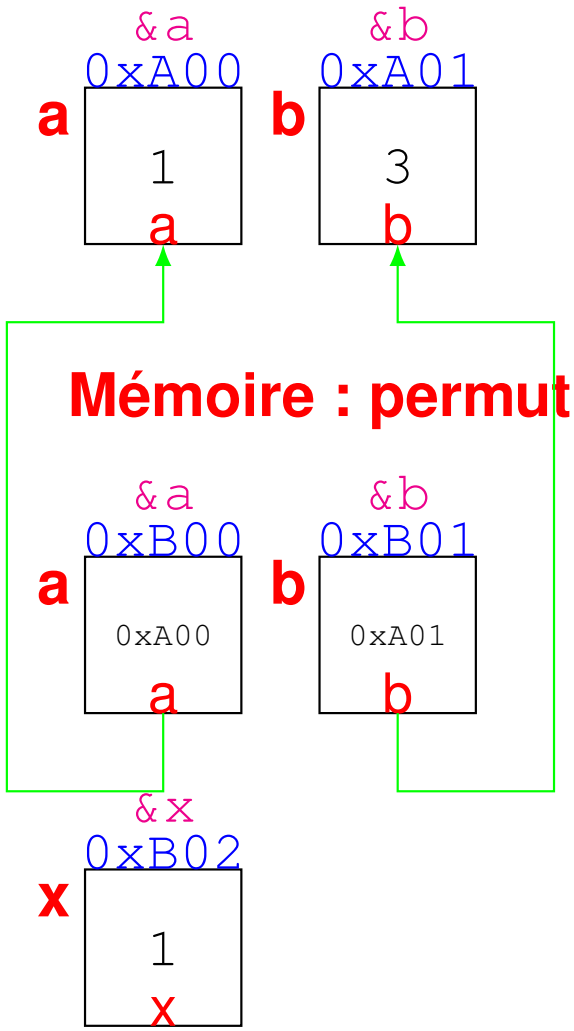


x=*a;

x ← *a=1

Listing 4 – Exemple2.c

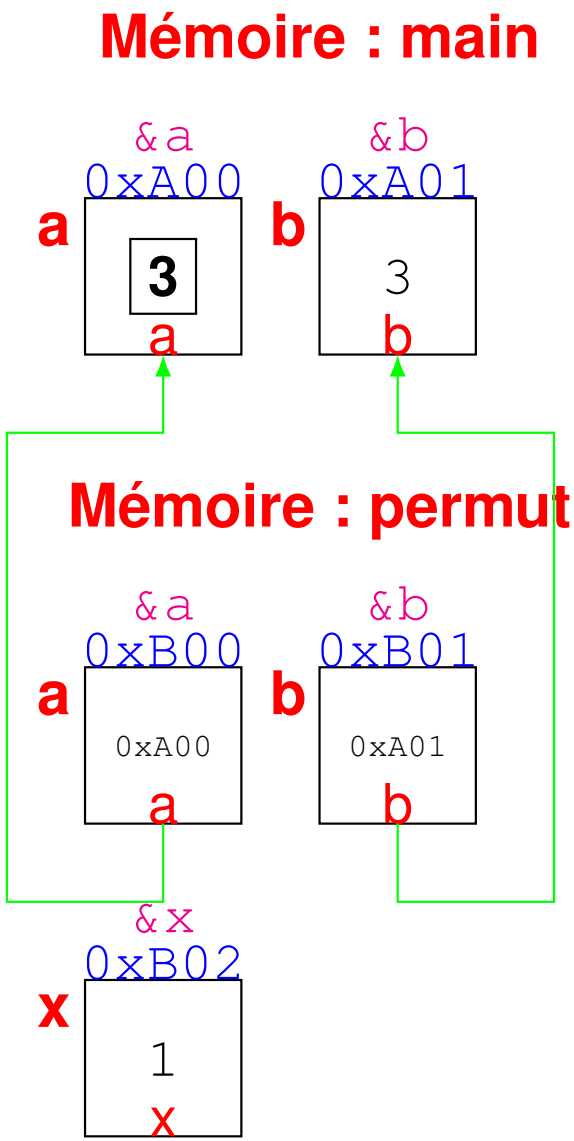
```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5  ▶  *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ■  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```



*a=*b;

Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5  ▶  *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ■  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

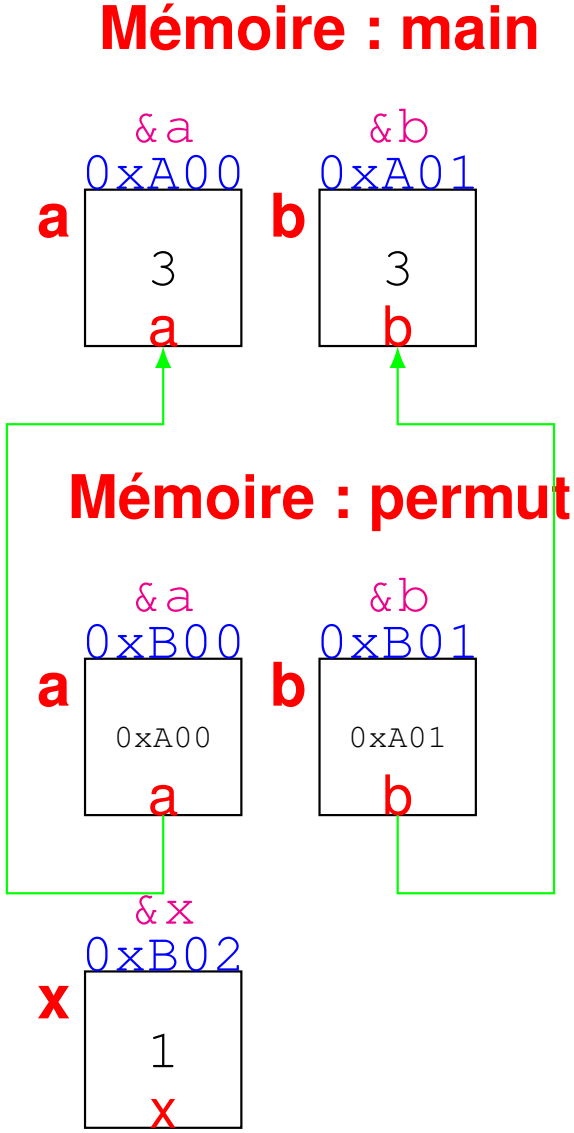


*a=*b;

La case mémoire pointée par a (i.e. a du main) reçoit la valeur contenue dans la case mémoire pointée par b (i.e. la valeur du b du main)

Listing 4 – Exemple2.c

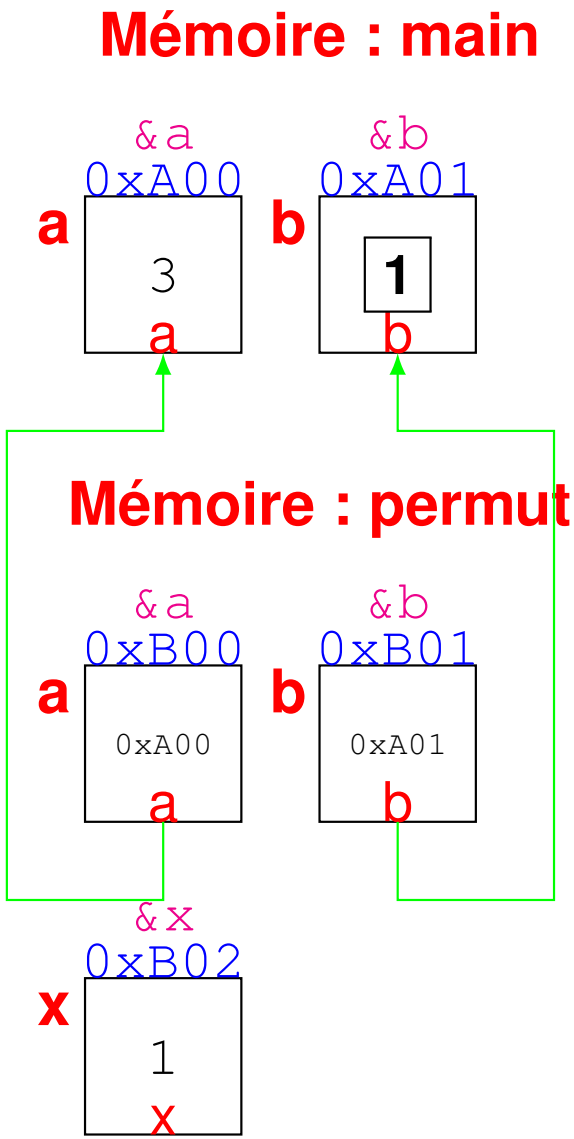
```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6  ▶  *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ■  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```



*b=x;

Listing 4 – Exemple2.c

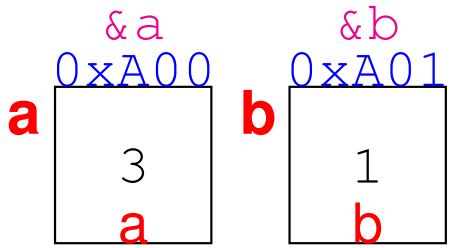
```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6  ▶  *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ■  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```



*b=x;

La case mémoire pointée par b (i.e. b du main) reçoit x (=1)

Mémoire : main



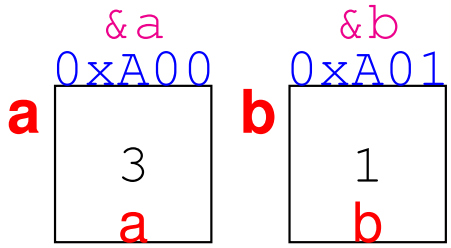
Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10 ▶  permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

permut(&a,&b);

Fin de l'appel de la fonction permut : retour au main

Mémoire : main

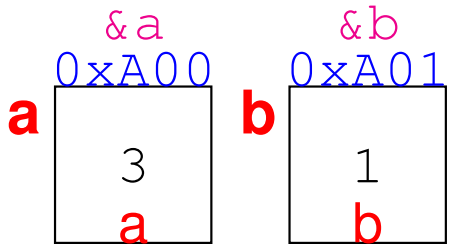


Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11 ▶  printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

```
printf ("a=%d,b=%d\n",a,b);
```

Mémoire : main



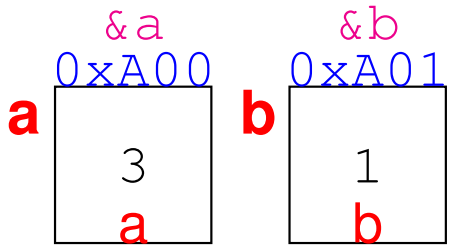
Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11 ▶  printf("a=%d,b=%d\n",a;b);
12    return 1;
13 }
```

printf ("a=%d,b=%d\n",a,b);

Affichage de : a=3,b=1

Mémoire : main

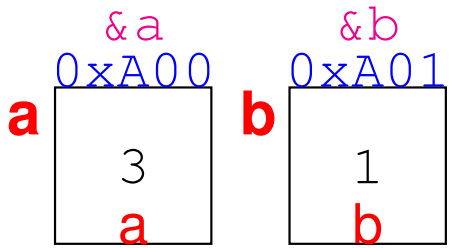


Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12 ▶  return 1;
13 }
```

return 1;

Mémoire : main



Listing 4 – Exemple2.c

```
1  #include <stdio.h>
2   void permut(int *a,int *b){
3     int x;
4     x=*a;
5     *a=*b;
6     *b=x;
7  }
8  int main(){
9     int a=1,b=3;
10    permut(&a,&b);
11    printf("a=%d,b=%d\n",a;b);
12 ▶  return 1;
13 }
```

return 1;

La fonction main est terminée et retourne 1

Extern (1/2)

Une fonction ne peut-être définie à l'intérieur d'une autre fonction : une fonction est donc toujours externe.

```
1 extern int max(int a,int b)
2 {
3   return a>b ? a : b;
4 }
5
6 int max(int a,int b)
7 {
8   return a>b ? a : b;
9 }
```

Les deux définitions sont équivalentes.

Extern (2/2)

```
1 extern int max(int a,int b);
2
3 int max(int a,int b);
```

Les deux déclarations sont équivalentes.

Confidentialité des fonctions

Une fonction peut-être rendue confidentielle, c'est à dire inutilisable en dehors du fichier où elle a été définie en utilisant le mot clef `static`.

Passage d'arguments au **main**

```c
#include <string.h>
int main(int argc, char *argv[]){
  void argument(char *s);
  char **arg=argv;
  arg++;
  while (--argc)
    argument(*arg++);
}
void argument(char *s){
  printf("argument = %s, longueur = %d\n",s,strlen(s));
}
```

- `argc` contient le nombre d'éléments du tableau pointé par `argv`.
- `argv` permet d'acceder aux différents arguments passés en paramètre au programme.

Pointeurs de fonctions

```
1  #include <string.h>
2
3  void f(int i){
4     printf("function␣f␣:␣%d\n",i);
5  }
6  void g(int i){
7     printf("function␣g␣:␣%d\n",i);
8  }
9
10 int main(void)
11 {
12    void (*p)(int);
13    p=f;
14    p(3);
15    p=g;
16    p(4);
17 }
```