

COURS 6

Programmation impérative

Tableaux - Pointeurs - Fonctions

- Lien Tableaux - Pointeurs
- Tableaux de Pointeurs
- Fonctions / Pointeurs

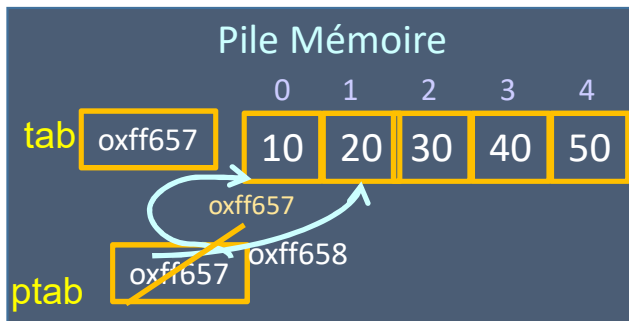
SOMMAIRE

- Informations pratiques
- Introduction
- Éléments de base
 - Programmer en Langage C – Compilation
 - Structure d'un programme / Règles d'écritures
 - Types de base
 - Constantes/Variables
 - Opérateurs
 - Instructions de contrôle
 - Pointeurs
 - Tableaux
- Fonctions
- Chaînes de caractères
- Tableaux- Pointeurs—Fonctions (TPF)
- Types Construits
- Entrées – Sorties sur Fichiers
- Compilation séparée
- Implémentation de Types Abstraits de Données

Pointeurs/ Tableaux/Fonctions

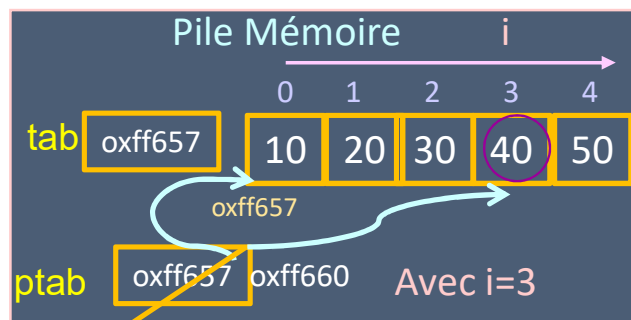
- En C, les pointeurs et les tableaux sont très fortement liés.
 - Toute opération effectuée par indexation dans un tableau peut être réalisée à l'aide de pointeurs.

Instruction	
<code>int tab[5]={10,20,30,40,50};</code> <code>int *ptab;</code>	fait pointer ptab sur l'élément d'indice 0 de tab \Leftrightarrow ptab contient l'adresse de tab[0] \Leftrightarrow ptab = &tab[0]
<code>ptab = tab;</code>	
<code>ptab ++;</code>	ptab pointe maintenant sur la case d'indice 1 de tab *ptab \Leftrightarrow tab[1]



TPF / lien pointeurs tableaux

Instruction	
<code>ptab + i ; //avec $0 \leq i < 5$</code>	fait pointer ptab sur l'élément d'indice i de tab $\Leftrightarrow \&\text{tab}[i]$
<code>*(ptab + i); //avec $0 \leq i < 5$</code>	Accès au contenu de la case dont l'adresse est ptab+i $\Leftrightarrow *(&\text{tab}[i])$ $\Leftrightarrow \text{tab}[i]$



TPF / Tableaux de pointeurs

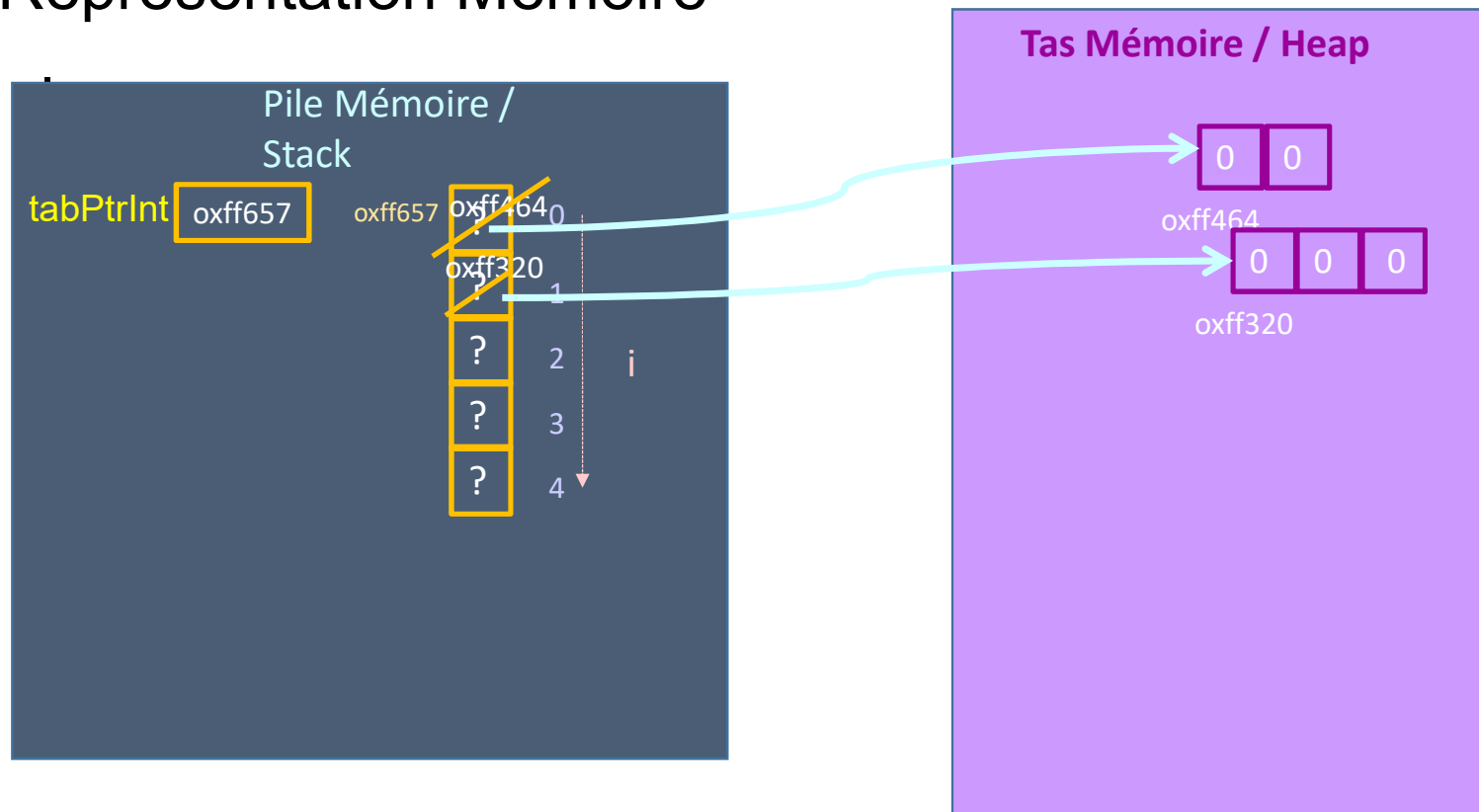
- Les pointeurs sont des variables
 - Ils peuvent être stockés dans des tableaux
 - Moyen de déclarer des tableaux bidimensionnels avec une taille de la deuxième dimension variable

Définition	
<pre>#define NB 5 int * tabPtrInt[NB];</pre>	<p>définition d'un tableau de NB pointeurs d'entiers</p> <p>Chaque pointeur doit être initialisé par une adresse existante ou par allocation dynamique</p>
<pre>tabPtrInt[0]= (int *) calloc(2, sizeof(int));</pre>	<p>Allocation du pointeur d'entier stocké dans la 1^{ère} case de tabPtrInt</p>
<pre>tabPtrInt[1]= (int *) calloc(3, sizeof(int));</pre>	<p>Allocation du pointeur d'entier stocké dans la 2^{ème} case de tabPtrInt</p>




TPF / Tableaux de pointeurs

- Représentation Mémoire

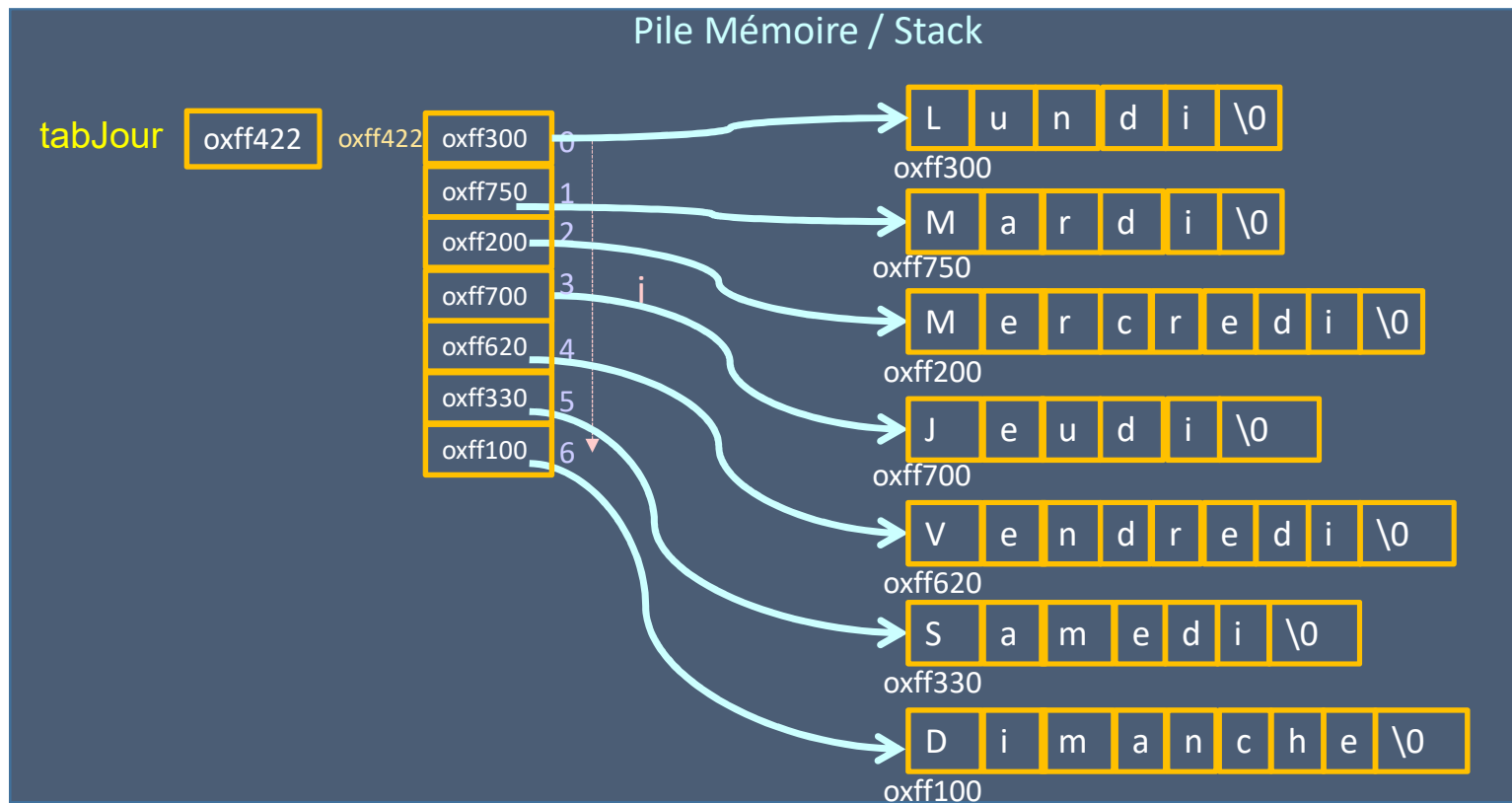


TPF / Tableaux de pointeurs

Définition	
<pre>#define NB 7</pre> <pre>char * tabPtrChar[NB];</pre>	définition d'un tableau de NB pointeurs de caractères Chaque pointeur doit être initialisé par affectation d'une adresse existante ou par allocation dynamique 
<pre>char *tabJour[]={"Lundi", "Mardi", "Mercredi", "jeudi", "Vendredi", "Samedi", "Dimanche" };</pre>	Affectation à la première case de tabJour de l'adresse allouée par une allocation statique pour la chaîne de caractères "Lundi " Etc ...

TPF / Tableaux de pointeurs

- Représentation Mémoire



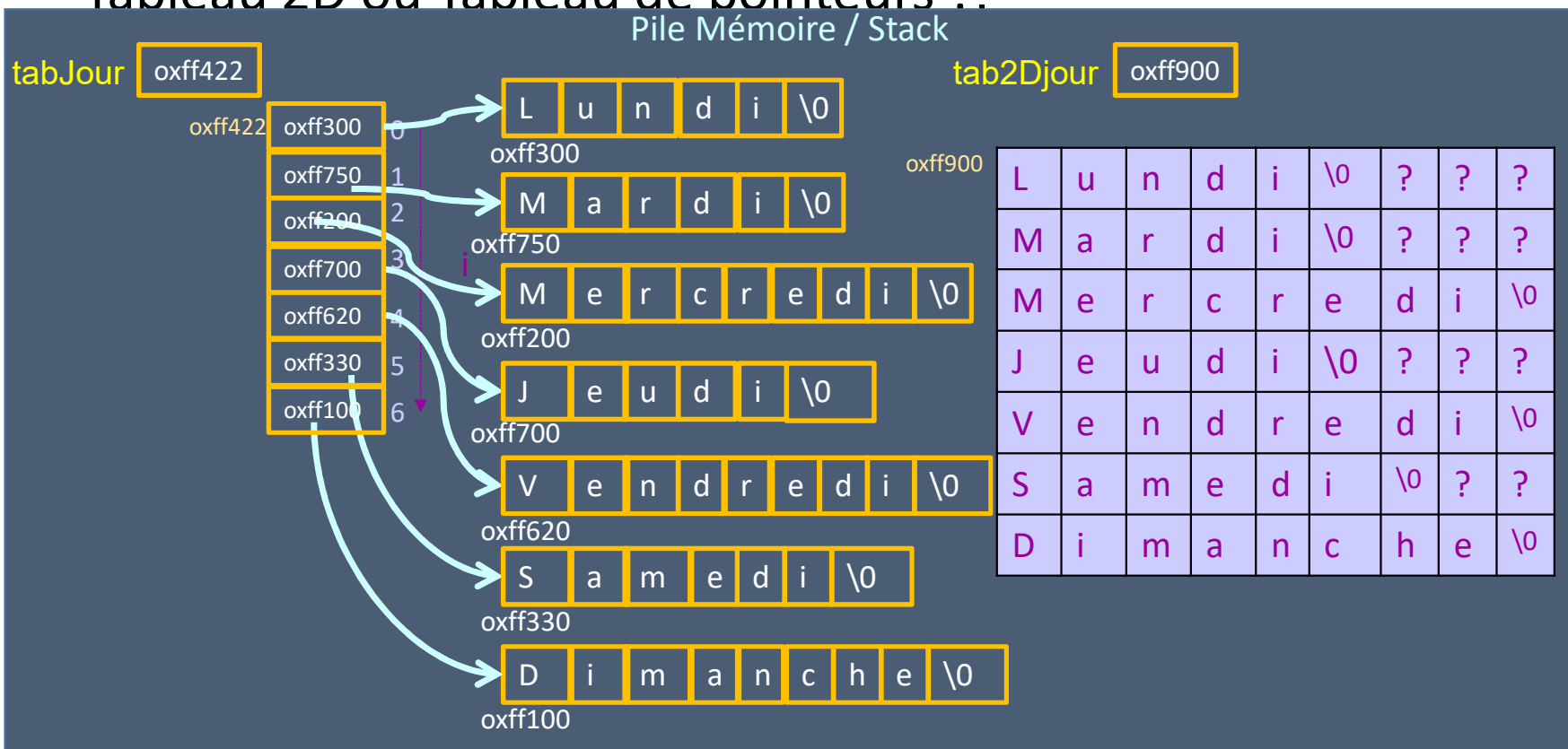
TPF / Tableaux de pointeurs

- Tableau 2D ou Tableau de pointeurs ?.

Définition	
<pre>char *tabJour[]={"Lundi", "Mardi", "Mercredi", "jeudi", "Vendredi", "Samedi", "Dimanche" };</pre>	<p>Affectation à la première case de tabJour de l'adresse allouée par une allocation statique pour la chaîne de caractères "Lundi «</p> <p>etc...</p>
<pre>char tab2Djour[7][9]={"Lundi", "Mardi", "Mercredi", "jeudi", "Vendredi", "Samedi", "Dimanche" };</pre>	<p>Définition d'un tableau de caractères bidimensionnel, dont les cases de chaque ligne sont initialisées avec une chaîne de caractères</p>

TPF / Tableaux de pointeurs

- Tableau 2D ou Tableau de pointeurs ?



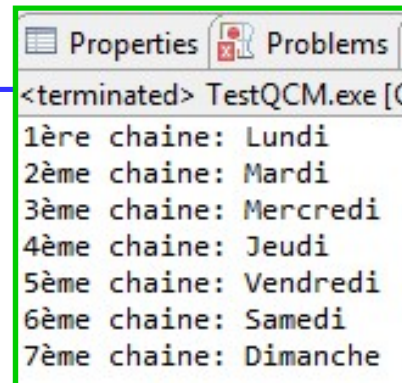
TPF / Tableaux de pointeurs

Accès aux éléments	
Soit char *tabJour[]={"Lundi", "Mardi", "Mercredi", "jeudi", "Vendredi", "Samedi", "Dimanche" };	
2ème chaîne stockée dans tabJour	tabJour[1] printf(" La 2 ^{ème} chaîne de tabJour est %s\n", tabJour[1]);
5ème caractère de la 4ème chaîne stockée dans tabJour	*(tabJour[3] +4) printf(" 5 ^{ème} caract de la 4 ^{ème} ch de tabJour: %c\n", *(tabJour[3]+4));

TPF / Tableaux de pointeurs

- Illustration

```
#include<stdio.h>
#include<stdlib.h>
void affichageCh(char *tabCh[], int n){
    int i;
    for(i=1;i<=n;i++){
        printf("%d%s chaine: %s\n", i,(i==1)?"ère":"ème",tabCh[i-1] );
    }
}
int main(){
    char *tabJour[]={"Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"};
    affichageCh(tabJour,7);
    return EXIT_SUCCESS;
}
```

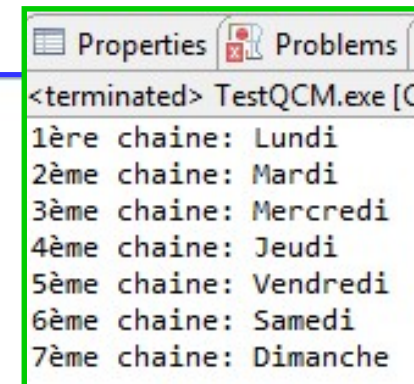


```
<terminated> TestQCM.exe [C...
1ère chaine: Lundi
2ème chaine: Mardi
3ème chaine: Mercredi
4ème chaine: Jeudi
5ème chaine: Vendredi
6ème chaine: Samedi
7ème chaine: Dimanche
```

TPF / Tableaux de pointeurs

- Illustration / Accès caractère

```
#include<stdio.h>
#include<stdlib.h>
void affichageCh(char *tabCh[], int n){
    int i, j;
    for(i=1;i<=n;i++){
        printf("%d%s chaine: ", i,(i==1)?"ère":"ème" );
        for(j=0;*(tabCh[i-1]+j)!='\0';j++)
            printf("%c", *(tabCh[i-1]+j));
        printf("\n");
    }
}
int main(){
    char *tabJour[]={"Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"};
    affichageCh(tabJour,7);
    return EXIT_SUCCESS;
}
```



<terminated> TestQCM.exe [C...
1ère chaine: Lundi
2ème chaine: Mardi
3ème chaine: Mercredi
4ème chaine: Jeudi
5ème chaine: Vendredi
6ème chaine: Samedi
7ème chaine: Dimanche

TPF / Pointeurs et Fonctions

- Passage de pointeur par valeur

```
#include <stdio.h>
#include <stdlib.h>

void permutPointeurs(int *a, int *b){
    int *aux;
    printf("Au début de permutation ==== a: %p b:%p\n", a,b);
    aux=a;
    a=b;
    b=aux;
    printf("A la fin de permutation ==== a: %p b:%p\n", a,b);
}

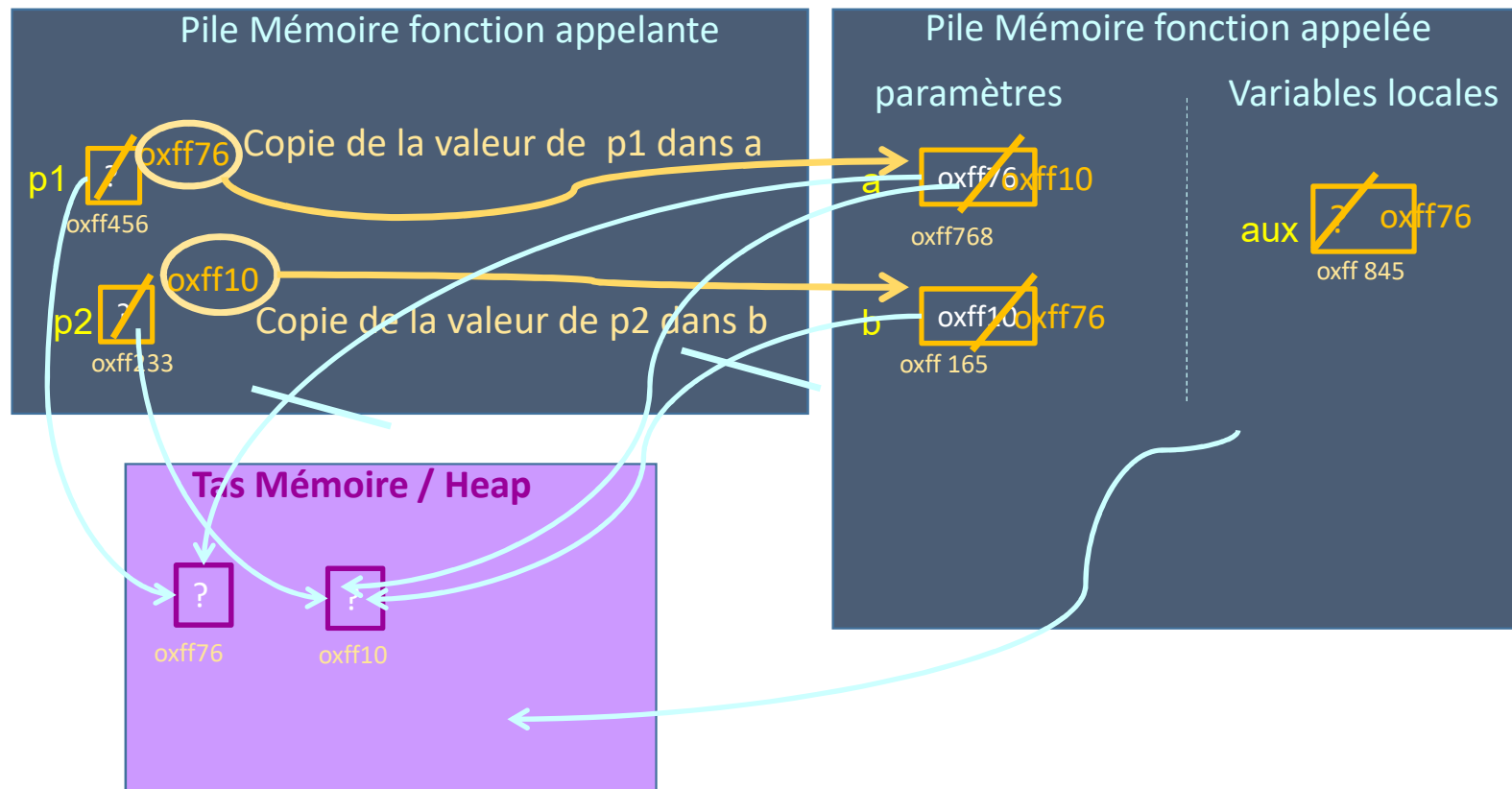
int main(){
    int *p1,*p2;

    p1 = (int *) malloc(sizeof(int));
    p2 = (int *) malloc(sizeof(int));
    if(p1!=NULL && p2!=NULL){
        printf("Avant permutation ==== p1: %p p2:%p\n", p1,p2);
        permutPointeurs(p1,p2);
        printf("Après permutation ==== p1: %p p2:%p\n", p1,p2);
    }
    else
        printf("pb allocation mémoire pour p1 et/ou p2");
    return EXIT_SUCCESS;
}
```

Passage par valeur de pointeurs

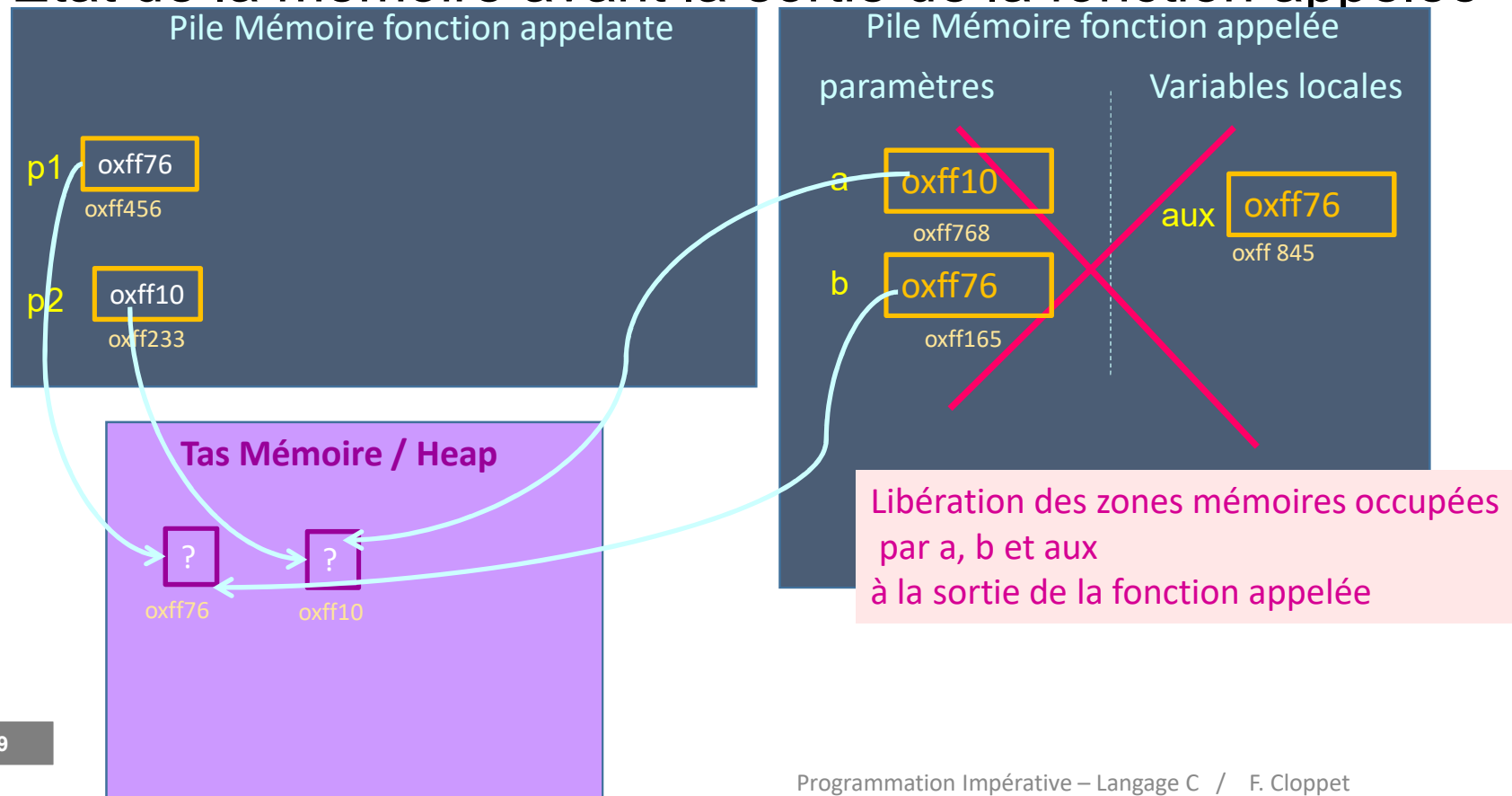
TPF / Pointeurs et Fonctions

- Passage de pointeur par valeur



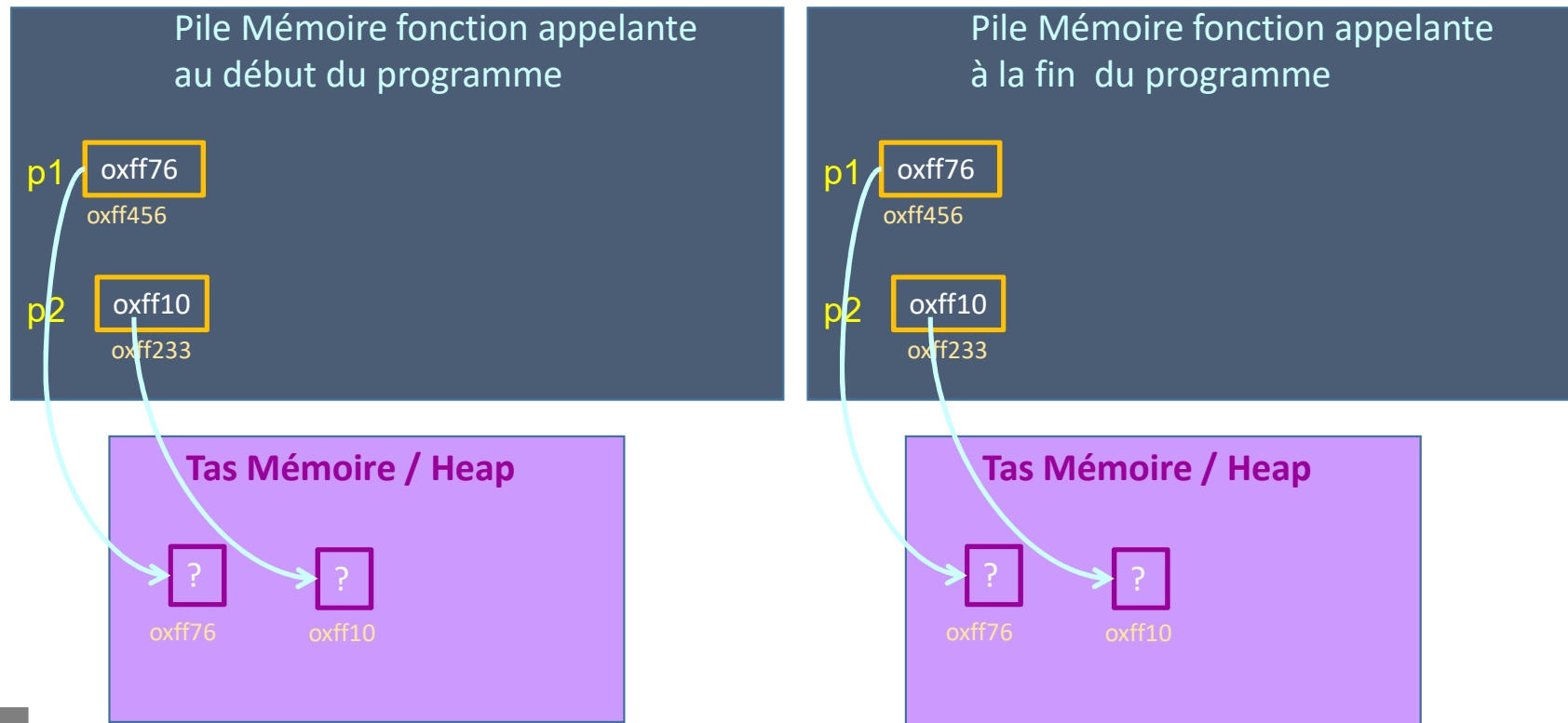
TPF / Pointeurs et Fonctions

- État de la mémoire avant la sortie de la fonction appelée



TPF / Pointeurs et Fonctions

- État de la mémoire avant et après l'appel de `permutPointeurs`



TPF / Pointeurs et Fonctions

- Passage de pointeur par adresse

```
#include <stdio.h>
#include <stdlib.h>

void permutPointeurs(int **a, int **b){
    int *aux;
    printf("Au début de permutation ==== a: %p b:%p\n", *a,*b);
    aux=*a;
    *a=*b;
    *b=aux;
    printf("A la fin de permutation ==== a: %p b:%p\n", *a,*b);
}

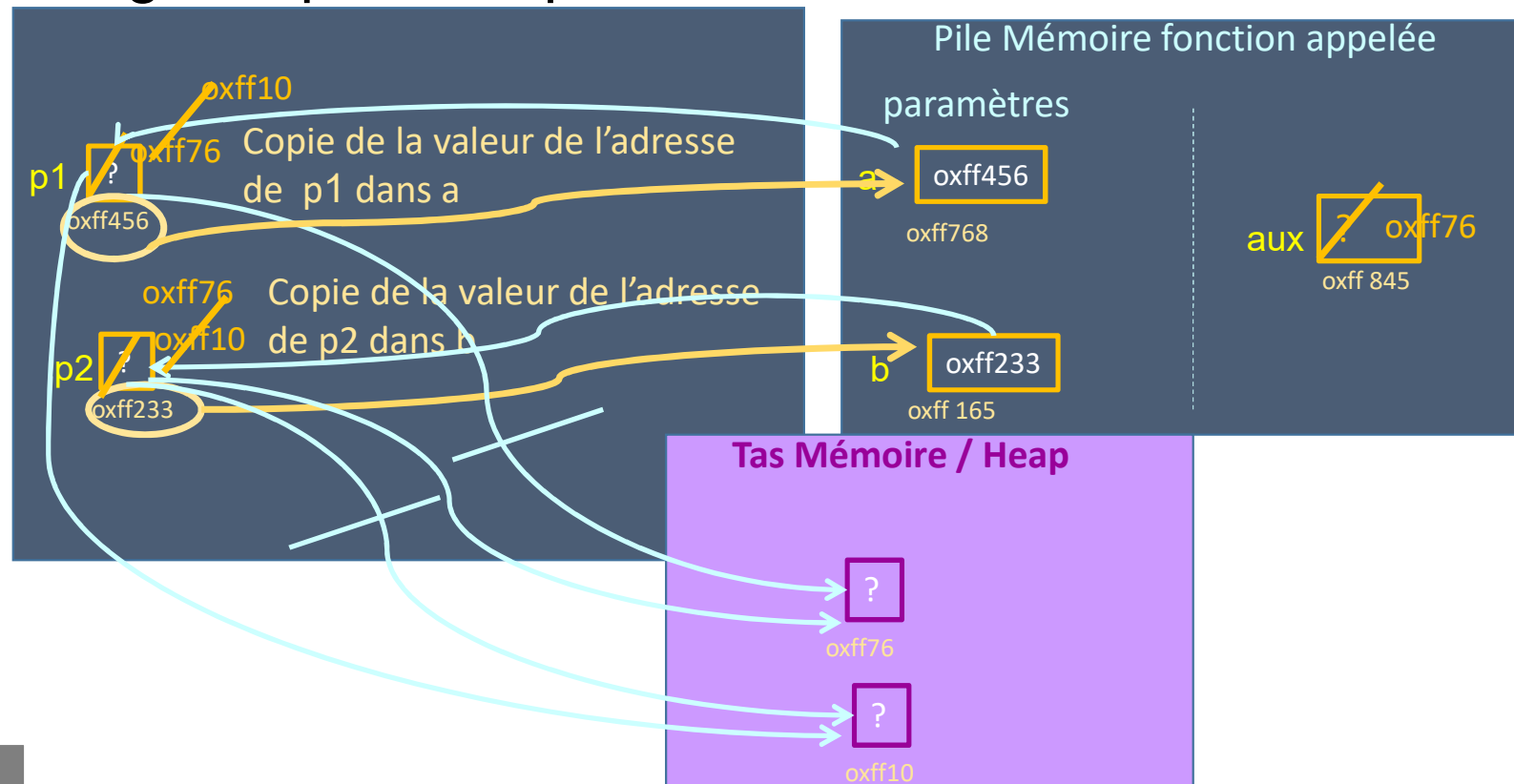
int main(){
    int *p1,*p2;

    p1 = (int *) malloc(sizeof(int));
    p2 = (int *) malloc(sizeof(int));
    if(p1!=NULL && p2!=NULL){
        printf("Avant permutation ==== p1: %p p2:%p\n", p1,p2);
        permutPointeurs(&p1,&p2);
        printf("Après permutation ==== p1: %p p2:%p\n", p1,p2);
    }
    else
        printf("pb allocation mémoire pour p1 et/ou p2");
    return EXIT_SUCCESS;
}
```

Passage par adresse de pointeurs

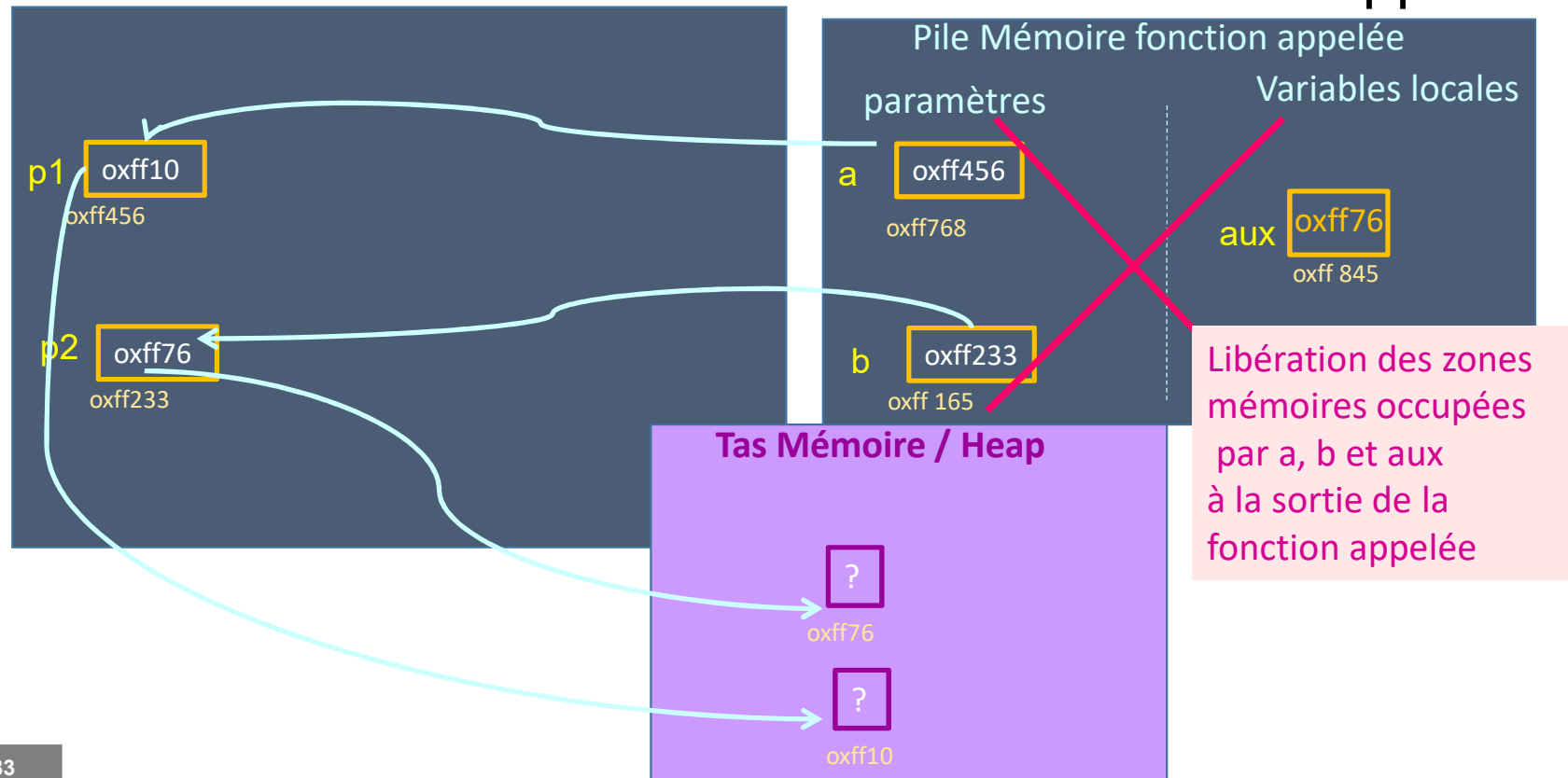
TPF / Pointeurs et Fonctions

- Passage de pointeur par adresse



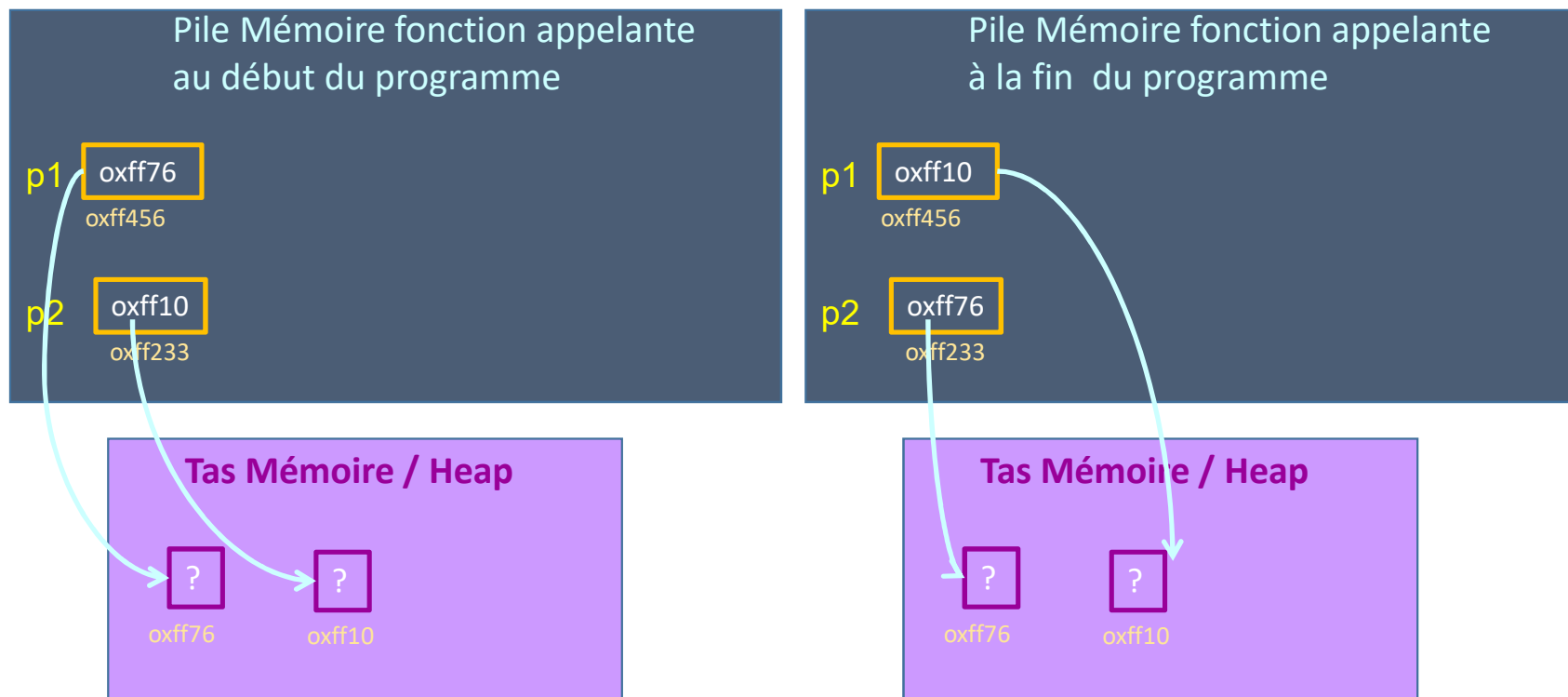
TPF / Pointeurs et Fonctions

- État de la mémoire avant la sortie de la fonction appelée



TPF / Pointeurs et Fonctions

- État de la mémoire avant et après l'appel de `permutPointeurs`



TPF / Zones mémoires

- Un programme utilise 4 zones mémoires
 - Le segment de code contenant
 - les instructions
 - Le segment de pile contenant
 - Les adresse de retour des fonctions
 - Les paramètres
 - Les variables locales
 - Le segment de données contenant
 - Les variables globales
 - Le tas ou mémoire dynamique contenant
 - Les données allouées dynamiquement au cours de l'exécution du programme