

# COURS 5

---

Programmation impérative

**Les Chaînes de caractères**

- Définition
- Représentation en mémoire
- Fonctions de la librairie standard

# 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
- Pointeurs- Tableaux-Fonctions
- Types Construits
- Entrées – Sorties sur Fichiers
- Compilation séparée
- Implémentation de Types Abstraits de Données

# Chaînes de caractères

Définition	
<p>Suite de caractères caractérisée par</p> <ul style="list-style-type: none"><li>• une adresse de début</li><li>• un caractère de marquage de fin de chaîne <code>'\0'</code></li></ul>	<p><code>'a'</code> est un caractère</p> <pre>char tabChar[ ]={'B', 'o', 'n', 'j', 'o', 'u', 'r'};</pre> <p><code>tabChar</code> est un tableau de caractères</p> <pre>char chaine[ ]={'B', 'o', 'n', 'j', 'o', 'u', 'r', '\0'};</pre> <p><code>chaine</code> est une chaîne de caractères</p>

# Chaînes de caractères

## Représentation en Mémoire

```
char c='a';
```

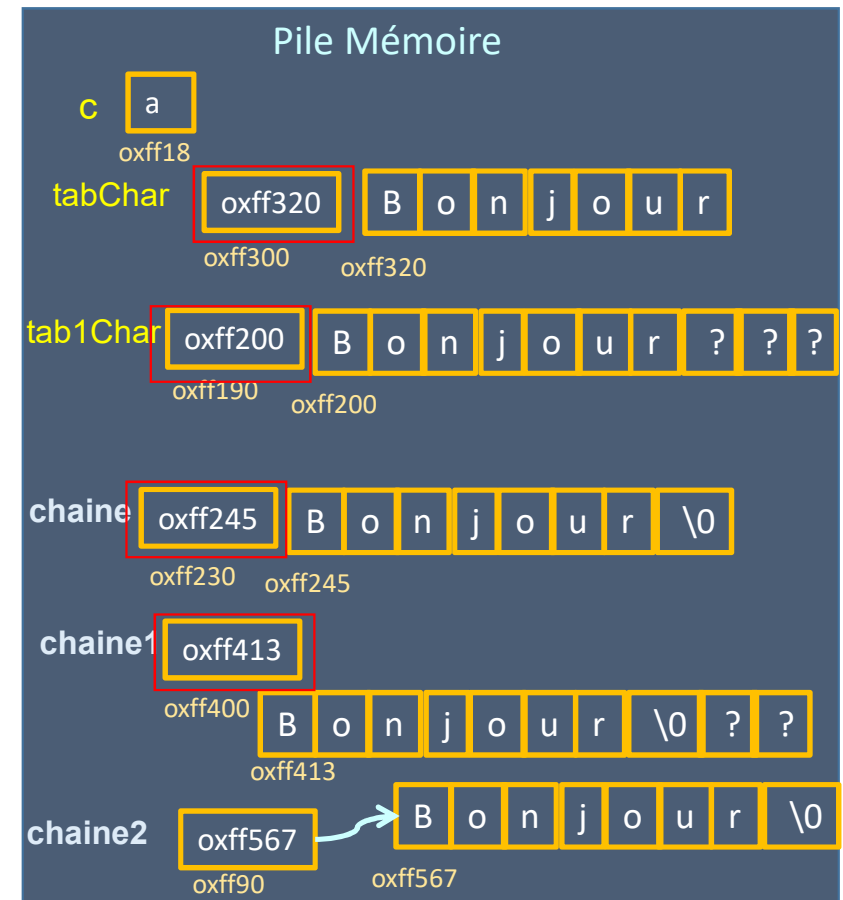
```
char tabChar[ ]={'B', 'o', 'n', 'j', 'o', 'u', 'r'};
```

```
char tab1Char[ 10]={'B', 'o', 'n', 'j', 'o', 'u', 'r'};
```

```
char chaine[ ]={'B', 'o', 'n', 'j', 'o', 'u', 'r', '\0'};
```

```
char chaine1[10]={'B', 'o', 'n', 'j', 'o', 'u', 'r', '\0'};
```

```
char *chaine2 = "Bonjour";
```



# Chaînes de caractères

## Représentation en mémoire

Par un tableau de caractères  
`char chaine[NB];`



### Adresse de début de la chaîne

⇔ adresse de la 1ère case du tableau de caractères

**Ne pas oublier le caractère `'\0'` à la fin des caractères s'il n'est pas mis implicitement**

⇔ Penser à dimensionner correctement votre tableau

**Un tableau est une adresse non réassignable**

⇔ On ne pourra pas la changer en cours d'exécution du programme

`char ch[100] = "Bonjour";`

~~`char ch[100];  
ch = "bonjour";`~~



# Chaînes de caractères

## Représentation en mémoire

### Par un pointeur de caractères

```
char *chaine;
```



### Adresse de début de la chaîne

⇔ adresse contenue dans le pointeur

### Ne pas oublier d'initialiser le pointeur

⇔ Par l'affectation d'une adresse existante

⇔ Par une allocation dynamique de la mémoire



### Ne pas oublier le caractère `'\0'` à la fin des caractères

s'il n'est pas mis implicitement




### C'est une adresse réassignable

⇔ Pourra être modifiée au cours de l'exécution du programme

⇔ Éviter le problème des fuites mémoires

```
char *ch = "Bonjour";
```

# Chaînes de caractères / Fonctions prédéfinies

Fonctions	
Entrées à partir du clavier	Définies dans la librairie <stdio.h>
scanf format %s	<p>lecture d'une chaîne de caractère sur l'entrée standard scanf s'arrête quand un espace une tabulation ou un RC est tapé</p> <div> <pre>char ch[100]; scanf("%s", ch);</pre> <div> <del> <pre>char *ch1; scanf("%s", ch1);</pre> </del> <p>ch1 : pointeur non alloué</p> </div> </div> 
gets	<p>lit une ligne complète jusqu'à ce que l'on tape RC la place dans ch, elle remplace le caractère de fin de ligne par "\0", retourne ch ou NULL si une erreur survient</p> <pre>char ch[100]; gets (ch); //on peut récupérer ou pas le retour</pre>
fgets fgets ( ch, NMAX, stdin) ;	<p>lit au maximum NMAX caractères sur le flux de l'entrée standard, les place dans ch, et met le '\0' à la suite</p> <p>⇔ La place mémoire réservée pour ch doit être de NMAX+1</p>

# Chaînes de caractères / Fonctions prédéfinies

Fonctions	
Sorties au niveau de l'écran	Définies dans la librairie <stdio.h>
printf	Affiche tous les caractères à partir de l'adresse de début jusqu'au \0
format %s	<pre>char ch[100]="Bonjour"; printf("%s", ch);</pre>
puts	<pre>char ch[100] = "Bonjour à tous"; puts (ch);</pre> <p>écrit la chaîne ch et un caractère de fin de ligne(invisible) à l'écran qui envoie le curseur sur la ligne suivante</p>



# Chaînes de caractères / Fonctions prédéfinies

Fonctions	
Entrées/Sorties sur Ch de caractères	Définies dans la librairie <stdio.h>
sscanf	extraction des caractères de ch1 ('\0' compris) et écriture dans ch2
format %s	<pre>char ch1[100]="Bonjour"; char ch2[100];  sscanf (ch1, "%s ", ch2) ;</pre> <div>ch1: Bonjour ch2: Bonjour</div>
sprintf	<pre>sprintf(char *ch, char *chaineFormate , type val) ; //type de val correspond au format spécifié dans chaineFormate Écriture de la chaine formatée dans ch char ch[100] ; int x=4; sprintf(ch, "le nombre est %d", x) ;</pre> <div>ch: le nombre est 4</div>

ch

oxff200

l

e

n

o

m

b

r

e

e

s

t

4

\0

?

...

?

oxff200

oxff299

# Chaînes de caractères / Fonctions prédéfinies

Fonctions	
Manipulations de chaînes de caractères	Définies dans la librairie <string.h>
<b>strcpy</b>	<b>strcpy(ch1, ch2) ;</b> copie les caractères de ch2 ('\0' compris) dans ch1
Copie des caractères d'une chaîne dans une autre chaîne	<pre>char ch[10]; strcpy(ch, "Bonjour !");</pre> <div>ch: Bonjour !</div> <p>copie de la chaîne de caractères <b>Bonjour !</b> dans la chaîne de caractères <b>ch</b></p>
<b>strcat</b>	<b>strcat(ch1, ch2) ;</b> concatène la chaîne ch2 à la suite de ch1
Concaténation de 2 chaînes de caractères	<pre>char ch1[100]; strcpy(ch1, "Bonjour"); strcat(ch1, " à vous"); printf("ch1: %s", ch1);</pre> <div>ch1: Bonjour à vous</div>

# Chaînes de caractères / Fonctions prédéfinies

Fonctions	
<b>Manipulations de chaînes de caractères</b>	<b>Définies dans la librairie &lt;string.h&gt;</b>
<b>strcmp</b>  Comparaison de 2 chaînes de caractères dans l'ordre lexicographique	<pre>int strcmp(char *ch1, char *ch2);</pre> <p>compare caractère par caractère selon le code ascii ch1 et ch2 et s'arrête à la première différence</p> <p>Renvoie</p> <ul style="list-style-type: none"><li>• 0 si les 2 chaînes sont identiques</li><li>• une valeur négative si ch1 &lt; ch2</li><li>• une valeur positive si ch1 &gt; ch2</li></ul>

```
char ch1[100] ;  
char ch2[100] ;  
int diff;  
strcpy(ch1, "Bonjour") ;  
strcpy(ch2, "Bonsoir") ;  
diff=strcmp(ch1,ch2) ;  
printf("diff: %d", diff);
```

diff: -1

```
char ch1[100] ;  
char ch2[100] ;  
int diff;  
strcpy(ch1, "Bonjour") ;  
strcpy(ch2, "Bonjour") ;  
diff=strcmp(ch1,ch2) ;  
printf("diff: %d", diff);
```

diff: 0


```
char ch1[100] ;  
char ch2[100] ;  
int diff;  
strcpy(ch1, "Bonsoir") ;  
strcpy(ch2, "Bonjour") ;  
diff=strcmp(ch1,ch2) ;  
printf("diff: %d\n", diff);
```

diff: 1

# Chaînes de caractères / Fonctions prédéfinies

Fonctions	
Manipulations de chaînes de caractères	Définies dans la librairie <string.h>
<b>strlen</b>  Renvoie la longueur d'une chaîne de caractères	<b>strlen(ch)</b> ; retourne la longueur de ch (sans compter '\0')  <pre>char ch[100] ; int longueur;  strcpy(ch, "Bonjour") ; longueur=strlen(ch);  printf("longueur de la chaîne ch: %d", longueur) ;</pre> <div>longueur de la chaîne ch: 7</div>

# Chaînes de caractères / Fonctions prédéfinies

Stockage mémoire	Il faut penser à
Chaîne de dimension fixe allouée de façon automatique dans la pile	sur-dimensionner le tableau pour être sûr de pouvoir stocker toute la chaîne de caractères  non utilisation de certaines cases mémoires réservées
Chaîne de longueur variable allouée dans le tas	ne retourner que l'adresse d'objets alloués de façon dynamique en mémoire (dans le tas)  A éviter les fuites mémoires ⇔ Autant de free que d'allocations dynamiques

# Chaînes de caractères / Saisie

## Programme type de saisie de chaîne de caractères en dimension fixe – style impératif

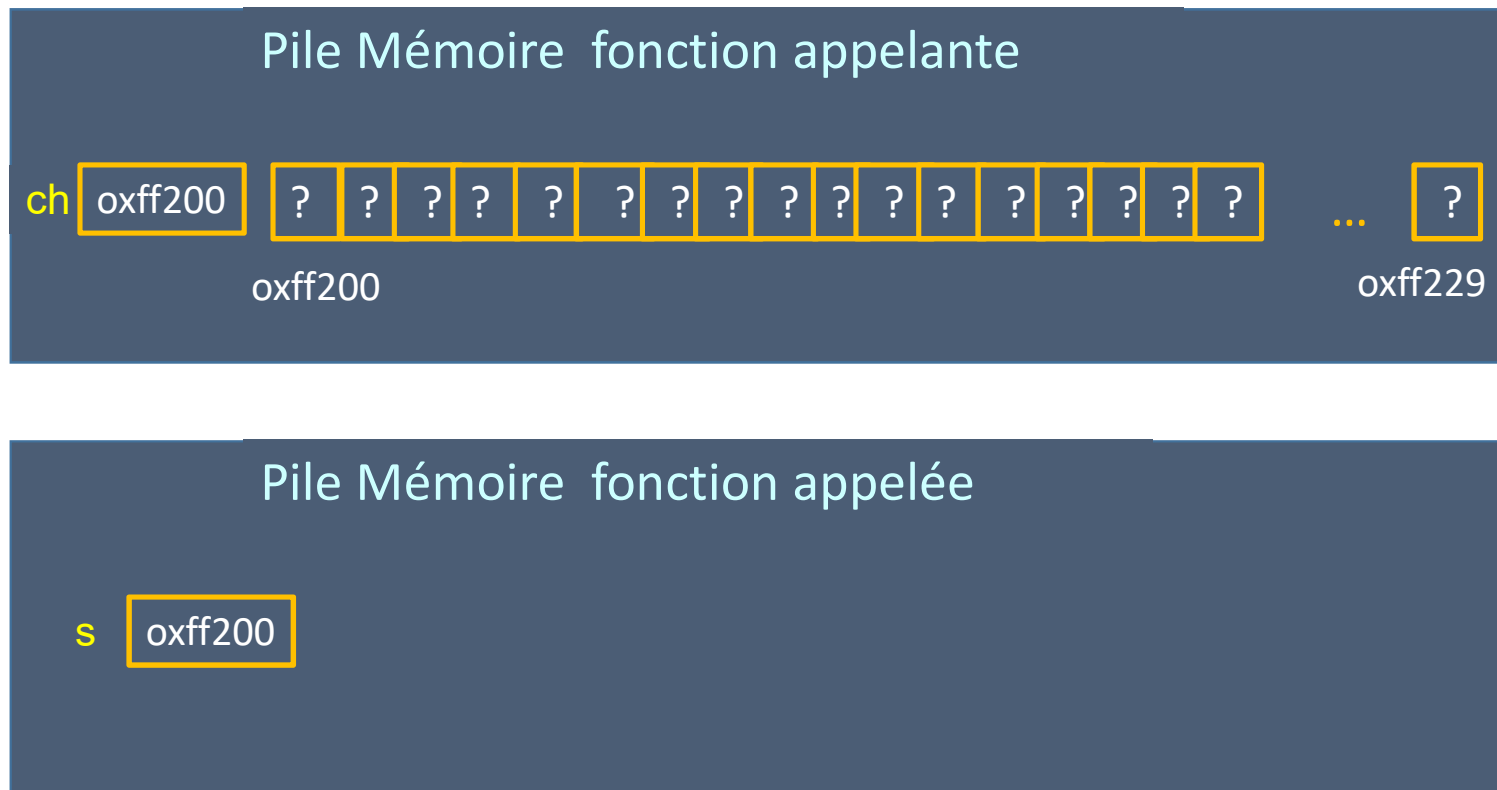
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NMAX 30

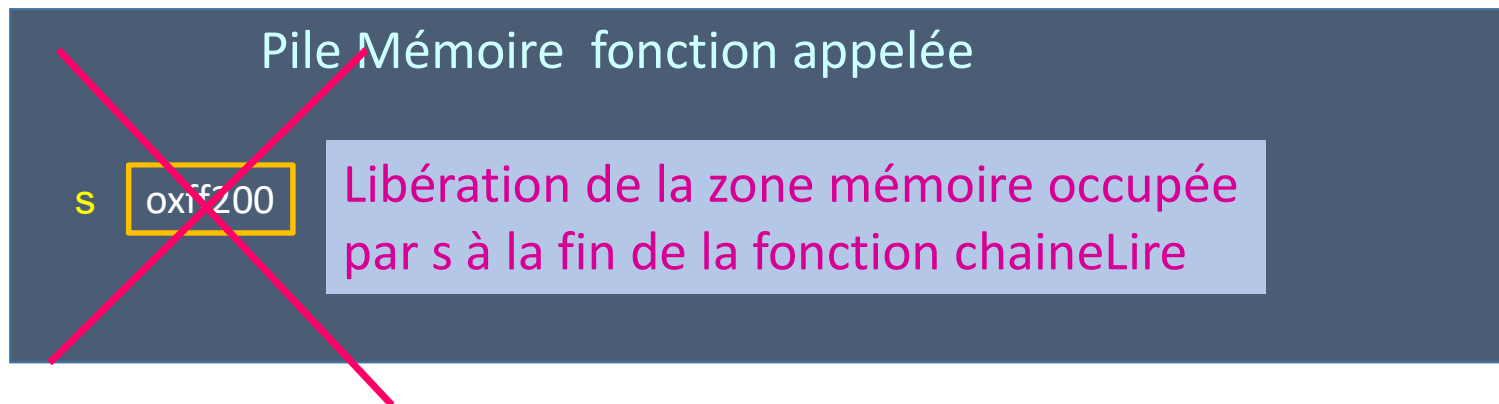
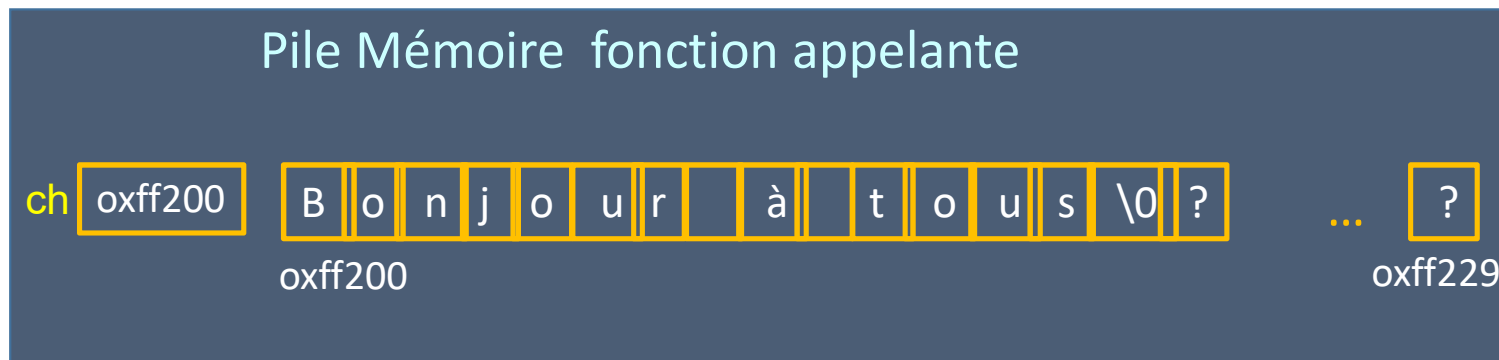
void chaineLire (char * s) {
    printf ("entrez une chaîne d'au plus %d caractères :\n", NMAX-1);
    fflush(stdout);
    fgets ( s, NMAX-1 , stdin) ;
}
OU void chaineLire (char s[])
int main () {
    char ch[NMAX] ;
    chaineLire (ch) ;
    printf ("voilà la belle chaîne : %s", ch) ;
    return EXIT_SUCCESS;
}
```

entrez une chaîne d'au plus 29 caractères :  
Bonjour à tous  
voilà la belle chaîne : Bonjour à tous

# Chaînes de caractères / Saisie



# Chaînes de caractères / Saisie





# Chaînes de caractères / Saisie

## Programme type de saisie de chaîne de caractères de longueur variable allouée dynamiquement - style fonctionnel

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

#define NMAX 30

char * ChaineSaisie () {
    char * s ;
    char buffer [NMAX] ;
    printf ("entrez une chaîne d'au plus %d caractères :\n", NMAX-1);
    fflush(stdout);
    fgets ( buffer, NMAX-1, stdin) ;
    s = (char *) malloc ( (strlen(buffer)+1) *sizeof(char) ) ;
    if (!s) {
        printf("problème d'allocation mémoire");
        exit (0);
    }
    strcpy(s, buffer);
    return s ;
}
```

# Chaînes de caractères / Saisie

Programme type de saisie de chaîne de caractères  
de longueur variable allouée dynamiquement - style fonctionnel

```
int main () {  
    char *ch ;  
    ch = ChaineSaisie () ;  
    printf ("voilà la belle chaîne : %s", ch) ;  
    free(ch);  
    return EXIT_SUCCESS ;  
}
```

```
entrez une chaîne d'au plus 29 caractères :  
Bonjour à tous  
voilà la belle chaîne : Bonjour à tous
```

# Chaînes de caractères / Saisie

Pile Mémoire fonction appelante

ch ?

Pile Mémoire fonction appelée avant saisie au clavier

s ?

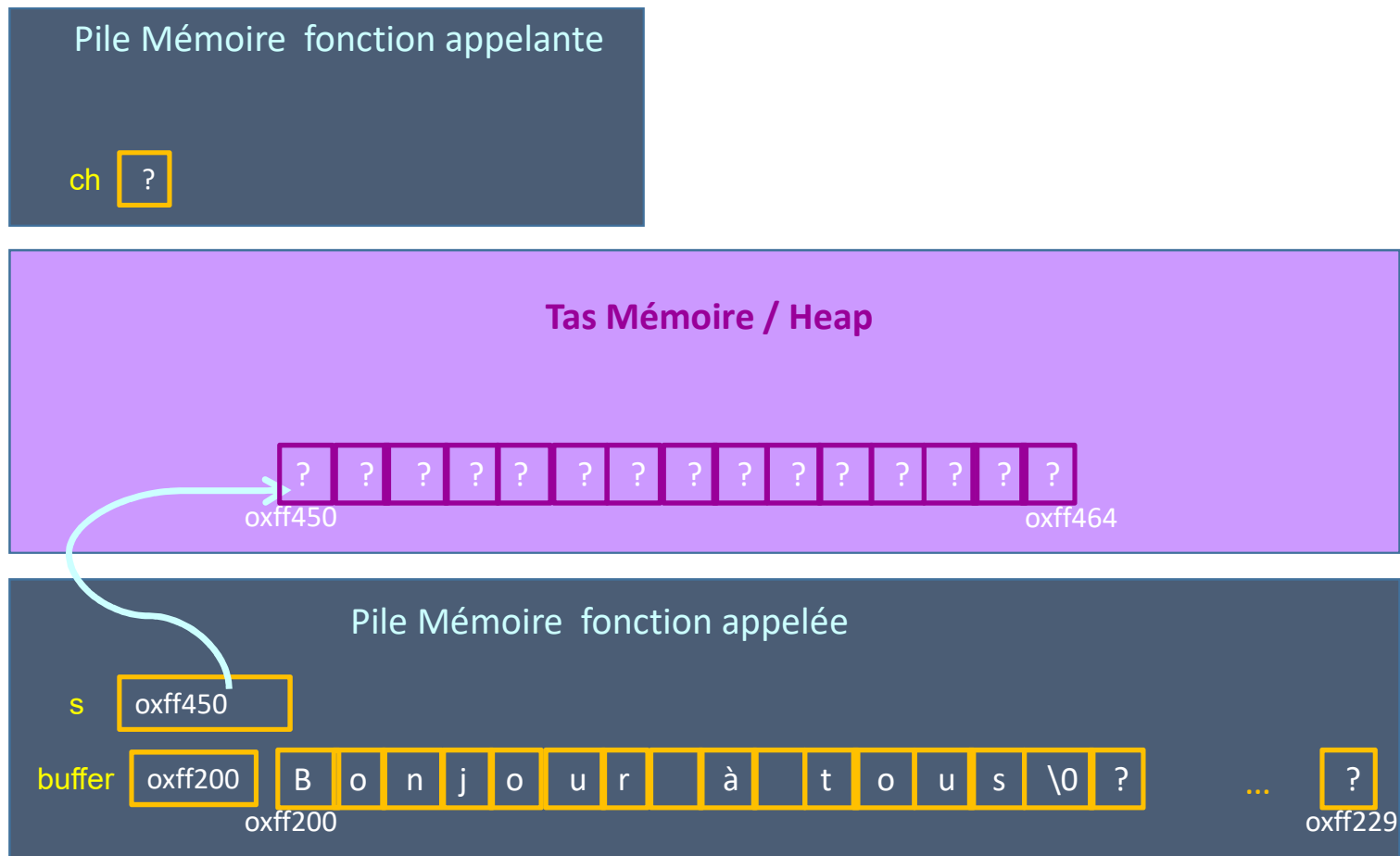
buffer 0xff200 ? ? ? ? ? ? ? ? ? ? ? ? ? ? ... ?  
oxff200 oxff229

Pile Mémoire fonction appelée après saisie au clavier

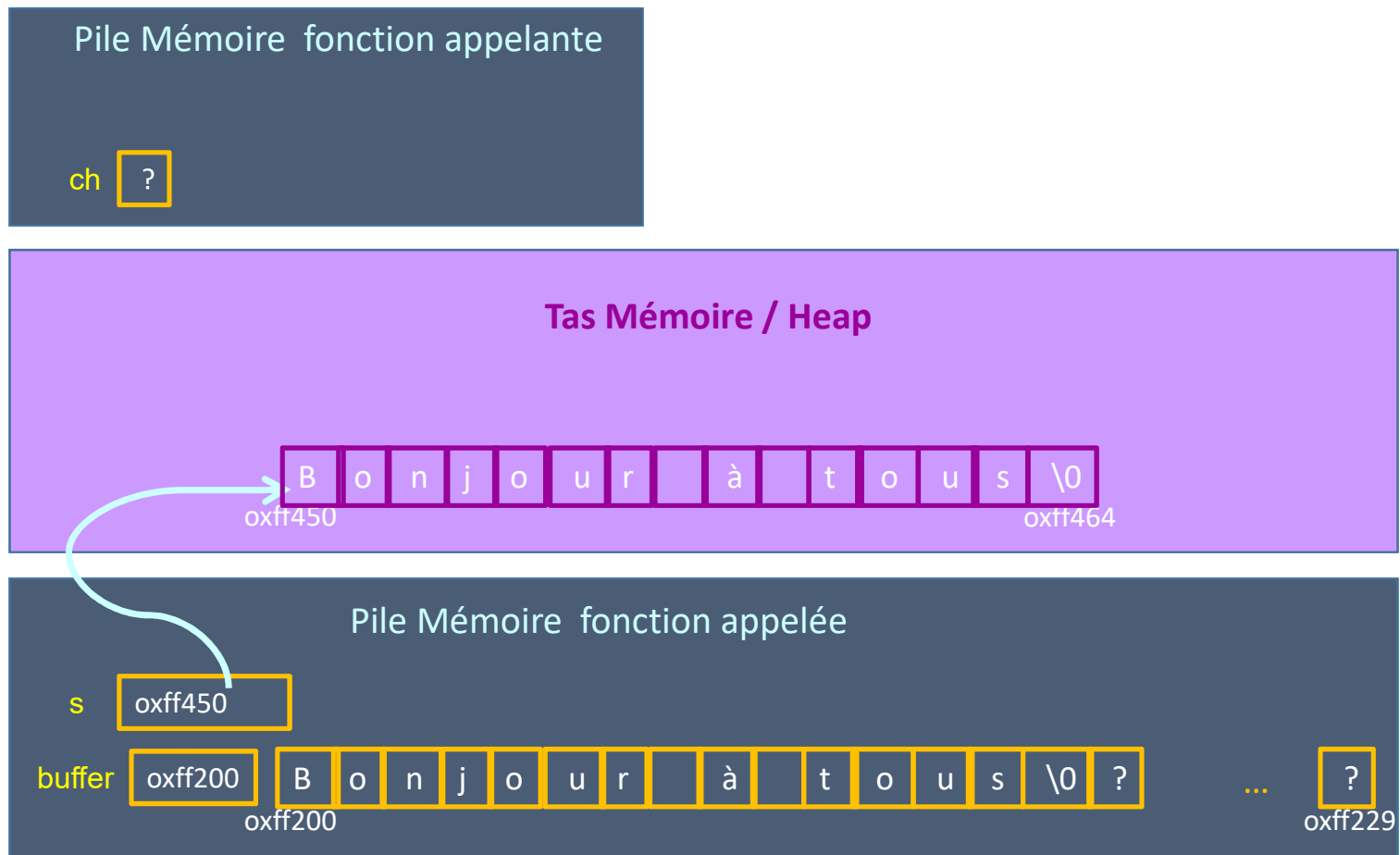
s ?

buffer 0xff200 B o n j o u r   à   t o u s \0 ? ... ?  
oxff200 oxff229

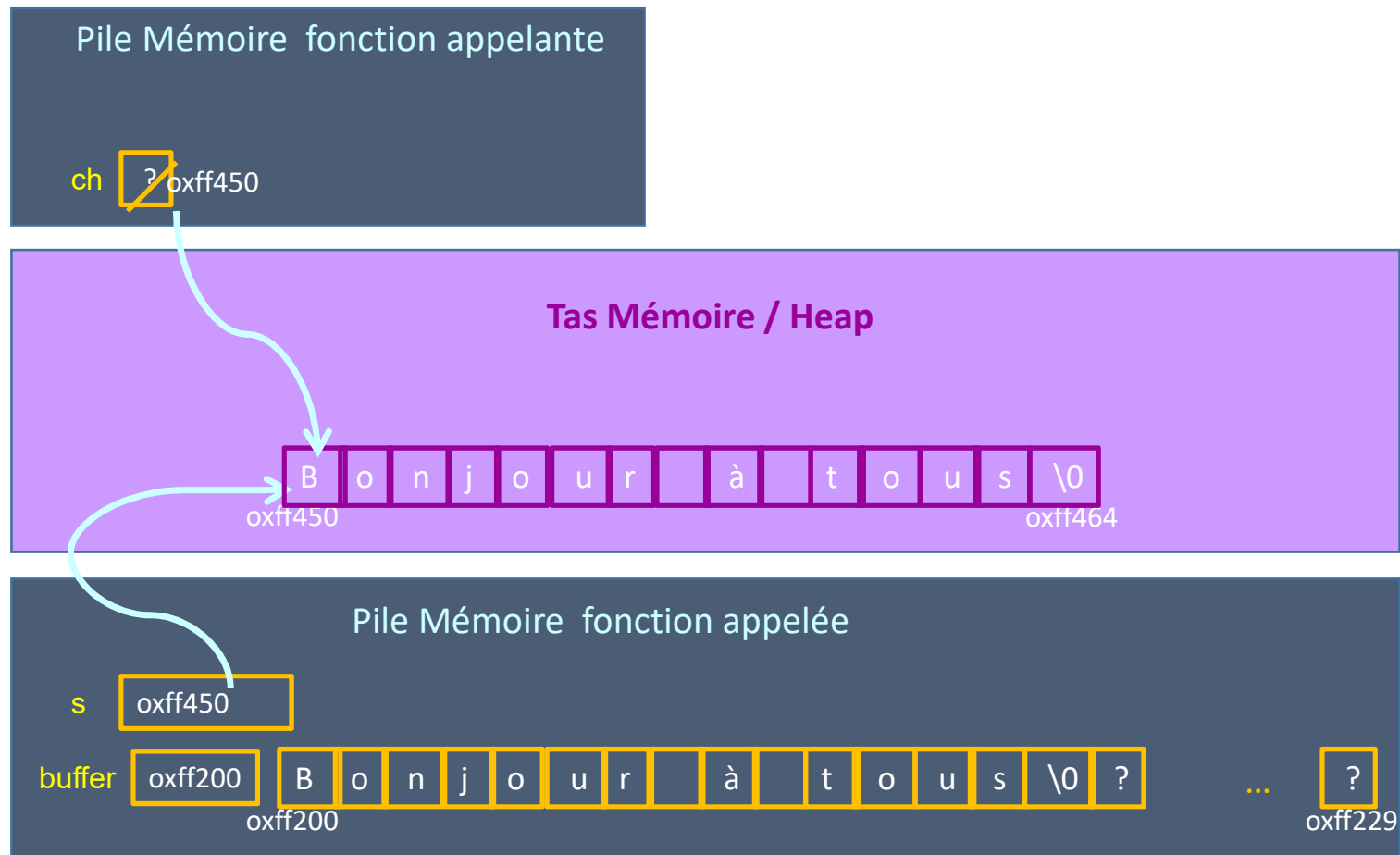
# Chaînes de caractères / Saisie



# Chaînes de caractères / Saisie



# Chaînes de caractères / Saisie



# Chaînes de caractères / Saisie

