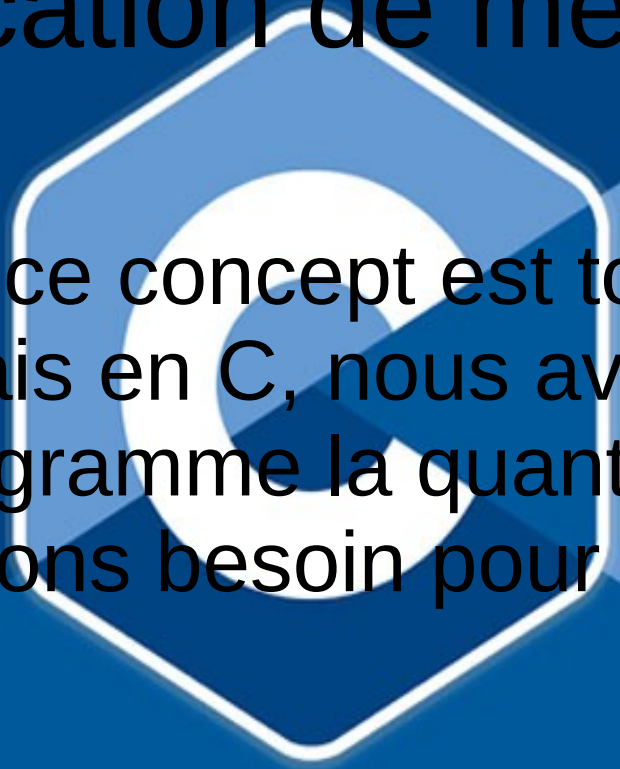
L’allocation de mémoire

Introduction en C (suite)
L’allocation de mémoire

Maintenant que l’on a commencé à manipuler la mémoire et ses adresses, nous allons parler de son allocation.

Introduction en C (suite)
L’allocation de mémoire



Encore une fois, ce concept est totalement absent de Python. Mais en C, nous avons besoin de signaler au programme la quantité de mémoire dont nous avons besoin pour une variable.

Introduction en C (suite)
L'allocation de mémoire

```
char *str;

str =  "Hello world";
```

Rappelez-vous. Ici, nous avons déclaré un pointeur sur char, et nous avons ensuite initialiser cette variable à « Hello world ».

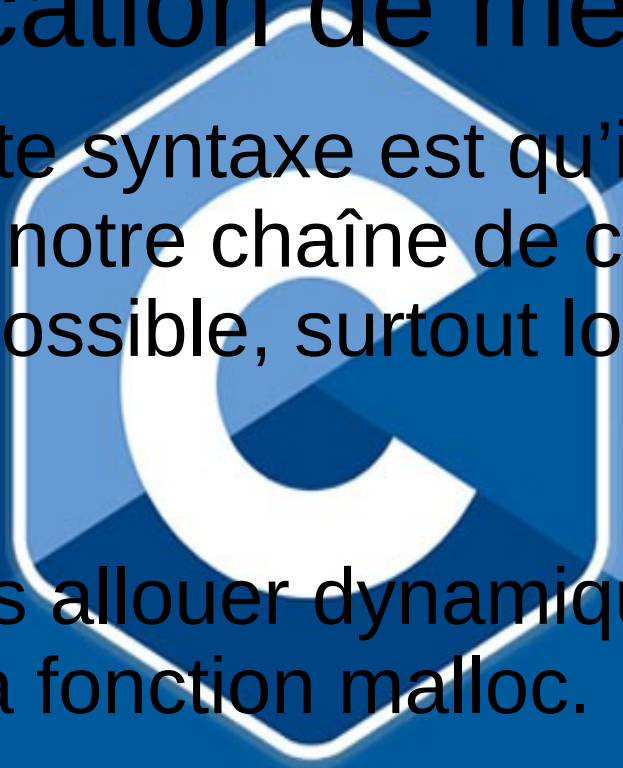
Introduction en C (suite)
L’allocation de mémoire

Maintenant, imaginons que l’on veut créer un pointeur sur char ne pouvant contenir que n caractères. Pas un de plus.

Nous avons effectivement vu la syntaxe :

```
char str[12] = "Hello world\0"
```

Introduction en C (suite)
L’allocation de mémoire



Le problème de cette syntaxe est qu’il faut connaître à l’avance la taille de notre chaîne de caractère. Ce qui n’est pas toujours possible, surtout lors de longs programmes.

En C, nous pouvons allouer dynamiquement la mémoire, grâce à la fonction malloc.

Introduction en C (suite)
L'allocation de mémoire

```
char *str;
int len_str;

len_str = 12;
str = (char*)malloc(sizeof(*str) * len_str);
```

Déclaration de deux variables. L'une étant un char *, l'autre un int

Initialisation de la variable len_str à 12

Type de retour du pointeur créé par malloc

Fonction de préprocessing sizeof permettant de récupérer le nombre d'octet du type de notre variable

Peut être remplacer par « char », car nous avons besoin de connaître le nombre d'octet d'un char

La taille de notre len. Multiplié par notre nombre d'octet, nous avons la taille totale à allouée pour notre variable

Introduction en C (suite)
L'allocation de mémoire

Une fois que l'on a plus besoin de la mémoire du pointeur, il faut penser à libérer la mémoire pour éviter toute fuite mémoire.

Pour cela, associer à la fonction malloc, il y a la fonction free, qui est justement faite pour ça.

```c
char *str = (char *)malloc(sizeof(char) * 12);

free(str);
```

Introduction en C (suite)
L’allocation de mémoire

Exercices pour comprendre le malloc :
strjoin, strdup etc...