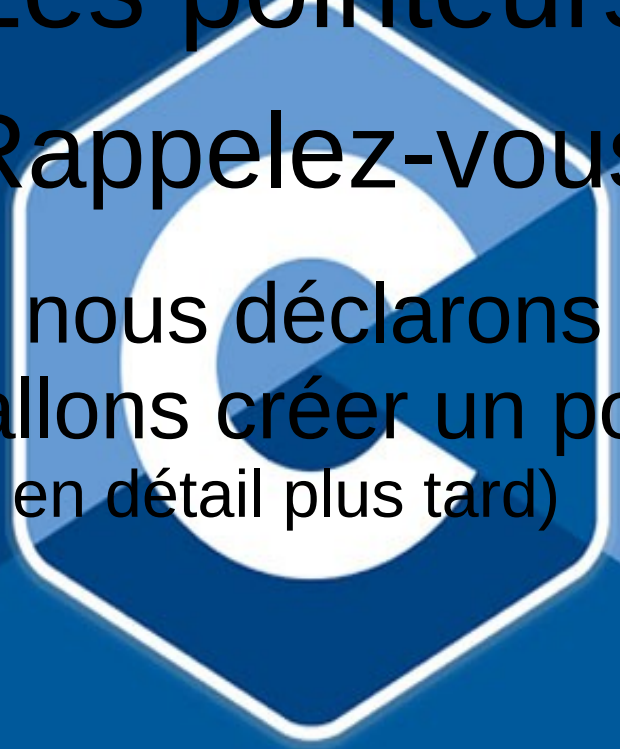
Introduction en C

Les pointeurs

Introduction en C (suite)
Les pointeurs

Rappelez-vous...

En réalité, quand nous déclarons une chaîne de caractère, nous allons créer un pointeur sur char
(nous verrons ça plus en détail plus tard)



Introduction en C (suite)
Les pointeurs

Rappelez-vous...

En réalité, quand nous déclarons une chaîne de caractère, nous allons créer un pointeur sur char
(nous verrons ça plus en détail plus tard)

C’est le moment !

Introduction en C (suite)
Les pointeurs

Les pointeurs sont un concept totalement absents de Python et de la plupart des langages modernes.

Introduction en C (suite)
Les pointeurs

Les pointeurs sont un concept totalement absents de Python et de la plupart des langages modernes.

Nous allons d’abord voir les pointeurs d’un point de vue des chaînes de caractères. Mais sachez que chaque type peut être un pointeur.

Introduction en C (suite)
Les pointeurs

Les pointeurs, comme nous l’avons vu se déclarent toujours de la même manière :

type_de_variable *nom_de_variable

Introduction en C (suite)
Les pointeurs

A quoi cela peut servir ?

Introduction en C (suite)

Les pointeurs

L’une des limitations que l’on a en C, c’est l’impossibilité de retourner plusieurs valeurs dans un programme.

Les pointeurs peuvent, palier à ce problème

Introduction en C (suite)
Les pointeurs

Par exemple, nous pouvons imaginer une fonction qui prendrait un int en paramètre, et voudrait vous dire combien il y a d'heures et combien de minutes il y a dans cet int.

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs
En Python, rien de plus simple :

```python
def convert_h_m(minutes) :
    nb_h = minutes / 60
    nb_m = minutes % 60
    return(nb_h, nb_m)
```

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs
En C, en revanche… :

```c
int h_m(int hours, int minutes) {
    hours = minutes / 60;
    minutes = minutes % 60;
}
```

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs
En C, en revanche… :

```c
int h_m(int hours, int minutes) {
    hours = minutes / 60;
    minutes = minutes % 60;
}
```

Comment pouvons-nous retourner nos heures et nos minutes nouvellement assignées ?

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs
En C, en revanche… :

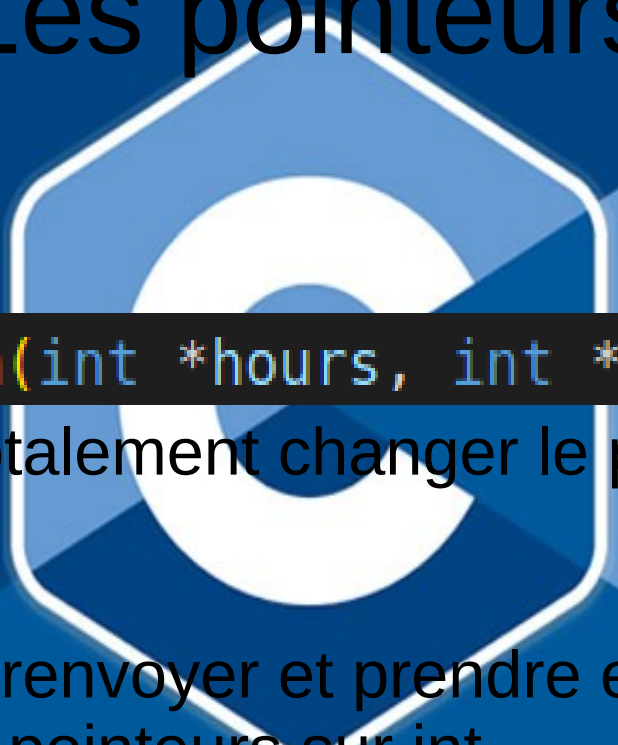
```c
int h_m(int hours, int minutes) {
    hours = minutes / 60;
    minutes = minutes % 60;
}
```

Comment pouvons-nous retourner nos heures et nos minutes nouvellement assignées ?

C’est simple. Nous ne pouvons pas. En C, nous ne pouvons retourner que les heures, ou que les minutes.

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs



```
void h_m(int *hours, int *minutes)
```

Nous allons alors totalement changer le prototype de notre fonction.

Elle ne va plus rien renvoyer et prendre en paramètre deux « int * », donc deux pointeurs sur int

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs

```c
int main(void) {
    int hours, minutes;

    hours = 0;
    minutes = 90;

    h_m(&hours, &minutes);
```

Ce nouveau prototype, implique une nouvelle utilisation, ici, dans notre main.

Nous déclarons toujours des int, mais nous ne les transmettons pas directement à notre fonction. A présent, elle prend des « int * », donc des pointeurs.

Pour transmettre un pointeur, nous allons utiliser le caractère « & », qui nous permet de transmettre l'adresse mémoire de notre variable

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs

```c
int h_m(int *hours, int *minutes)
    *hours = *minutes / 60;
    *minutes = *minutes % 60;
}
```

Notre fonction aussi a changé. Ce nouveau prototype change tout.

Pour rappel, dans le main, nous avons été forcé de transmettre l'adresse mémoire de nos variables.

Maintenant, pour faire des opérations, il nous faut manipuler non pas l'adresse, mais la valeur de nos variables. Pour cela, nous allons à nouveau utiliser caractère * pour cela.

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs

```c
int h_m(int *hours, int *minutes)
{
    *hours = *minutes / 60;
    *minutes = *minutes % 60;
}
```

Donc, si on résume.

Un pointeur, c'est une adresse mémoire. Un pointeur, pointe une adresse mémoire.

Quand l'on veut transmettre une variable comme un pointeur, on transmet donc son adresse, grâce au caractère « & ».

Si nous avons un pointeur, par exemple sur int, et que l'on veut récupérer sa valeur, il faut donc la récupérer avec le caractère « * »

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs

```c
#include <stdio.h>

int h_m(int *hours, int *minutes) {
    *hours = *minutes / 60;
    *minutes = *minutes % 60;
}

int main(void) {
    int hours, minutes;

    hours = 0;
    minutes = 90;

    h_m(&hours, &minutes);

    printf("%d h et %d m", hours, minutes);
}
```

Et ensuite ?

Voilà le programme au complet.

Dans le main, nous transmettons par adresse (et donc par pointeur) nos variables hours et minutes.

Dans la fonction h_m, nous manipulons ses valeurs directement dans l'adresse (donc le pointeur)

A la fin, nous récupérons donc bien hours = 1 et minutes = 30, car nous avons modifié les valeurs grâce aux pointeurs

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs

Introduction en C (suite)
Les pointeurs

Sur une variable, comme la variable `age` :

- `age` signifie : "Je veux la valeur de la variable `age` " ;
- `&age` signifie : "Je veux l'adresse à laquelle se trouve la variable `age` ".

Sur un pointeur, comme `pointeurSurAge` :

- `pointeurSurAge` signifie : "Je veux la valeur de `pointeurSurAge` " (cette valeur étant une adresse) ;
- `*pointeurSurAge` signifie : "Je veux la valeur de la variable qui se trouve à l'adresse contenue dans `pointeurSurAge` ".

Exemple pris : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7672176-creez-et-initialisez-des-pointeurs