Cours langage C

Les pointeurs

Objetctifs du cours

1. Pourquoi les pointeurs?
2. Qu’est un qu’un pointeur?
3. Comment utiliser un pointeur en C?
4. Opérations avec les pointeurs.

Pourquoi les pointeurs?

Pourquoi les pointeurs

- On reprend l'exercice de la dernière fois:
- Exercice : Ecrire un programme conversion qui convertit les minutes en heures.

  Exemple 1 : pour 90 min on obtient : 1h30min

  Exemple 2 : pour 60 min on obtient : 1h0min

  Exemple 3 : pour 45min on obtient : 0h45min.
- **Question :** comment organiser ce programme sous forme de fonction?

Exercice

- Réécrire le programme précédent conversion.c pour l'organiser sous forme de fonction : conversion (int minute) et qui retourne les minutes et les heures.

Solution de l'exercice

Version sans fonction

```
#include <stdio.h>
Void Conversion(int heures, int minutes)
int main(){

    int minutes;
    printf("veuillez saisir les minutes\n");
    scanf("%d", &minutes);
    int heures = minutes/60;
    int minutes = minutes%60;
    printf("%d h %d min", heures, minutes);
}
Void Conversion(int heures, int minutes)
```

Version avec fonction => Problème

- Problème dans la définition de la fonction:
  - **Cette fonction doit retourner deux valeurs : les heures et les minutes**.
  - En C impossible de retourner deux valeurs.
- Possible solution : passer par les paramètres. Essayons => Slide suivant

1er essai : Conversion en minutes, heures avec fonction

```
#include <stdio.h>
void conversion(int minutes, int heures);
int main(){
    int minutes= 0;
    int heures = 0;
    printf("veuillez saisir les minutes\n");
    scanf("%d", &minutes);
    conversion(minutes, heures);
    printf("%d h %d min", heures,
    minutes);
}
void conversion(int minutes, int  heures){
    heures = minutes/60;
    minutes %= 60;
}
```

- Si on saisit 90 min le résultat de ce programme est 0h90min.

Deuxième essai : utilisation des pointeurs (à étudier plus tard)

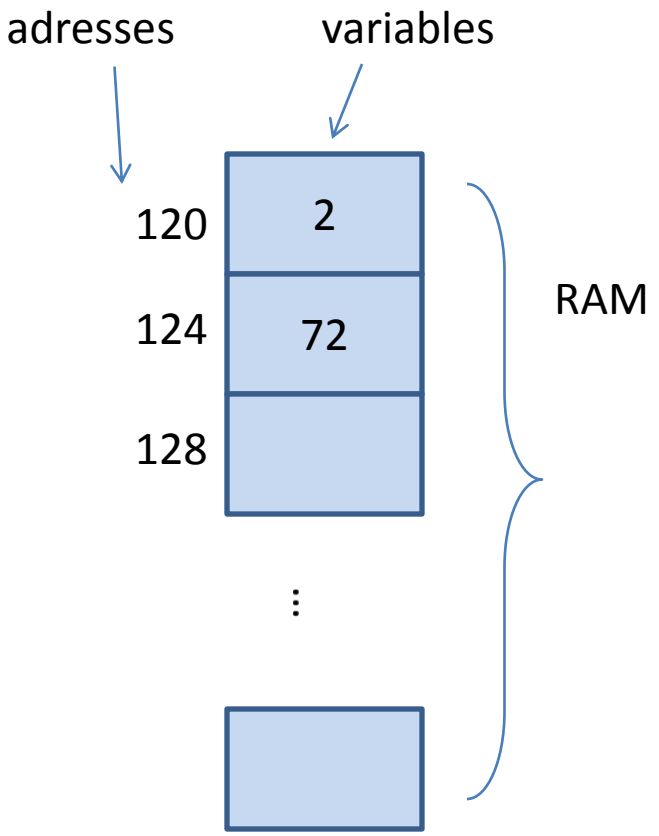
```
#include <stdio.h>
void conversion(int* minutes, int* heures);
int main(){
    int minutes= 0;
    int heures = 0;
    int* p_minutes = &minutes;
    int* p_heures = &heures;
    printf("veuillez saisir les minutes\n");
    scanf("%d", p_minutes);
    conversion(p_minutes, p_heures);
    printf("%d h %d min", *p_heures,
    *p_minutes);
}
void conversion(int* minutes, int* heures){
    *heures = *minutes/60;
    *minutes %= 60;
}
```

- Ce programme affiche bien 1h30 si on lui donne 90min.

Qu’est ce qu’un pointeur

Variables et adresses

- La RAM de l'ordinateur est une succession de cases où on stocke les variables.
- Chaque case a une adresse.
- Ne pas confondre adresse et valeur.



Exemple

Exemple

```
#include <stdio.h>

int main(){
    int a = 2;
    //ici on affiche la valeur de a
    printf("la valeur de a est %d\n", a);

    //ici on affiche l'adresse de a
    printf("l'adresse de a est %d\n", &a);
}
```

Commentaires

- Ce programme affiche :

la valeur de a est 2

l'adresse de a est 2293324

- Pour récupérer l'adresse d'une variable on utilise l'opérateur &
- L'adresse affichée sera **différente** sur un autre ordinateur.
- On peut aussi utiliser le spécificateur de format %p pour une adresse en hexadécimale.

Pointeurs

Définition

- Un pointeur est une variable qui contient l'adresse d'une autre variable.
- Exemple :

```
//on déclare une variable var
int var = 10;
//déclaration d'un pointeur
int *p;
//on fait pointer p sur var
p = &var;
```

On peut aussi fusionner les deux dernières étapes

```
int *p;
p = &var;
```

en une seule en écrivant :

```
int *p = &var;
```

Vocabulaire :

- On dit que le pointeur p pointe sur la variable var.
- La variable var est la variable pointée.

Autres exemples de déclarations :

- char *p_char;
- double *p_double;

Comment utiliser les pointeurs

L’intérêt des pointeurs

Intérêt

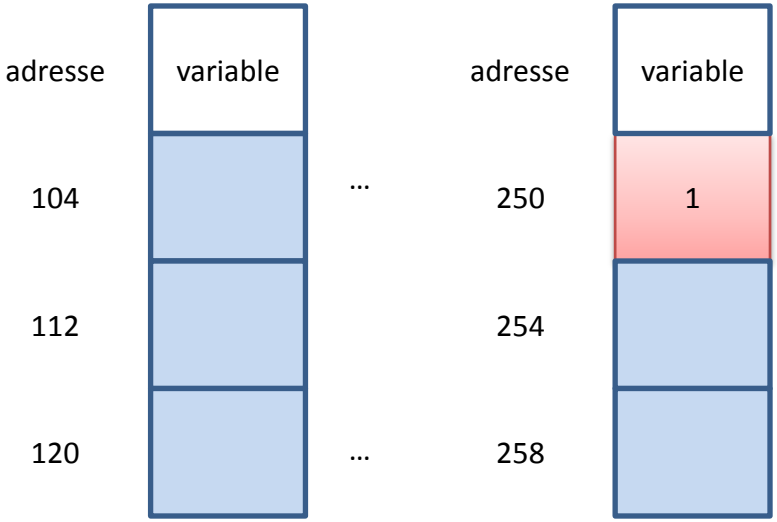
- L’intérêt des pointeurs : on peut modifier le contenu des variables pointées via un pointeur.
- Mais ceci on aurait pu le faire sans pointeur.
- On verra dans la suite du cours en quoi cela est important.

Exemple

```
#include <stdio.h>

int main(){
    // initialisation d'une variable à 1
    int a = 1;
    // on crée un pointeur sur cette variable
    int *p = &a;
    // on modifie le contenu de la variable a via le pointeur p
    *p = 2;
    // maintenant a vaut 2
    printf("a=%d", a);
}
```

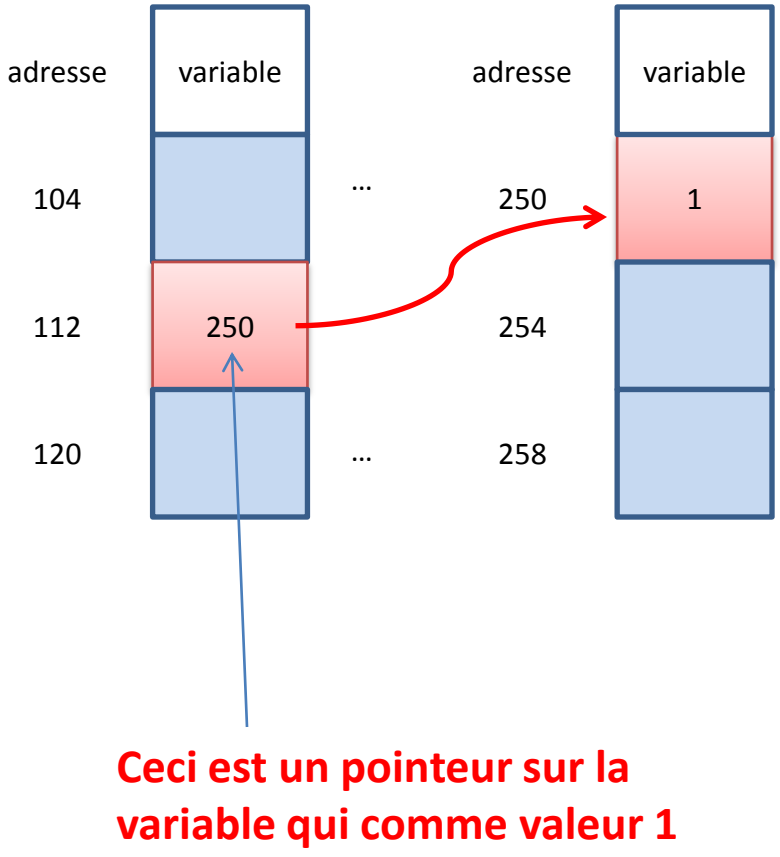
Ce qu'il se passe en mémoire (1/3)

```
#include <stdio.h>

int main(){
    // initialisation d'une variable à zéro
    int a = 1;
    // on crée un pointeur sur cette variable
    int *p = &a;
    // on modifie le contenu de la variable a via le pointeur p
    *p = 2;
    // maintenant a vaut 2
    printf("a=%d", a);
}
```



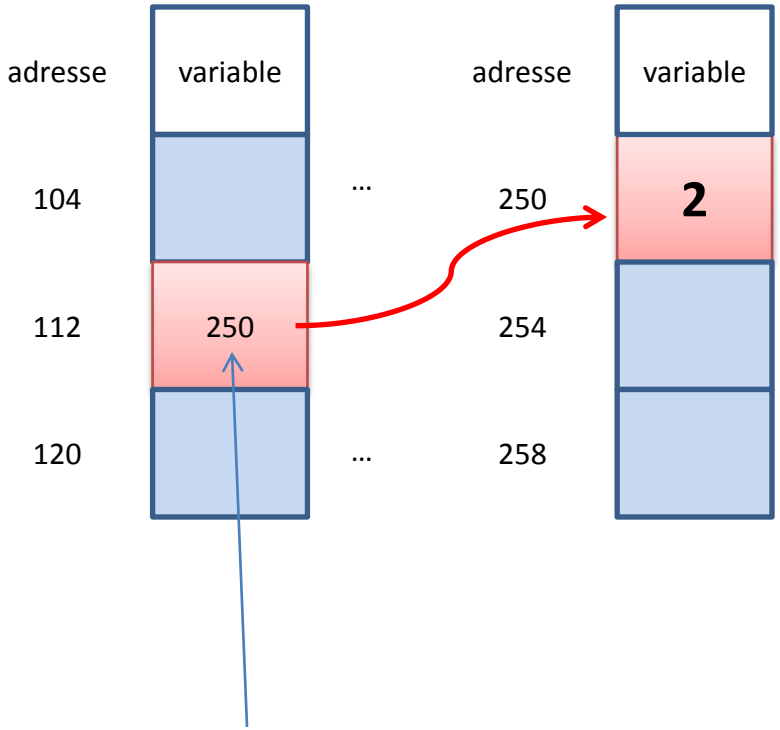
Ce qu'il se passe en mémoire (2/3)

```
#include <stdio.h>

int main(){
    // initialisation d'une variable à zéro
    int a = 1;
    // on crée un pointeur sur cette variable
    int *p =  &a;
    // on modifie le contenu de la variable a via le pointeur p
    *p = 2;
    // maintenant a vaut 2
    printf("a=%d", a);
}
```



Ce qu'il se passe en mémoire (3/3)

```
#include <stdio.h>

int main(){
    // initialisation d'une variable à zéro
    int a = 1;
    // on crée un pointeur sur cette variable
    int *p = &a;
    // on modifie le contenu de la variable a via le pointeur p
    *p = 2;
    // maintenant a vaut 2
    printf("a=%d", a);
}
```



La nouvelle valeur de la variable a est maintenant 2

Exercice d’entraînement

1. Ecrire un programme qui déclare et initialise une variable de type float.
2. Ajouter un pointeur qui pointe sur cette variable créée.
3. Modifier la valeur initiale de la variable en passant par le pointeur créé.
4. Afficher la nouvelle valeur de la variable.

Opérateurs & et *

- & est appelé un **opérateur de référencement**.
- Une seule utilisation dans le contexte des pointeurs : pour **spécifier l'adresse d'une variable**.
- Exemple : int *p = &var;

- L'opérateur * est appelé un **opérateur de dé-référencement.**
- Il a deux utilisations dans le contexte des pointeurs :
  - Pour déclarer un pointeur
    exemple : int *p = &var;
  - Pour modifier la variable pointée par le pointeur p
    exemple : *p = 5;
    La valeur pointée par p vaut maintenant 5
- Le symbole * est collé au nom du pointeur
- On peut déclarer plusieurs pointeurs dans la même ligne :

Int *ptr1, *ptr2, *ptr3;

ATTENTION : DANGER

- Pour utiliser les pointeurs **il faut les faire pointer vers quelque chose**.
- Typiquement le code suivant bien qu'il compile, génère une erreur à l'exécution:

```
#include <stdio.h>
int main(){
    int var;
    int *p;
    printf("%d", *p);
}
```

ATTENTION : DANGER

- Pour utiliser les pointeurs **il faut les faire pointer vers quelque chose**.
- Le code suivant a maintenant un sens, compile et s'exécute correctement.

```
#include <stdio.h>
int main(){
    int var;
    int *p = &var;
    printf("%d", *p);
}
```

Résumé

- Sur une variable var :
  - var veut dire “on s’intéresse à la valeur de var”.
  - &var veut dire “on s’intéresse à l’adresse en mémoire de la variable var”.
- Sur un pointeur par exemple p_var:
  - p_var veut dire “on s’intéresse à la valeur de p_var”. Cette valeur est une adresse.
  - *p_var veut dire “on s’intéresse à la variable qui se trouve à l’adresse contenue dans

Exercice d'application 1

Le but de cet exercice est d'ajouter deux nombres en **utilisant les pointeurs**. Pour cela :

1. Ecrire un programme qui demande à l'utilisateur de saisir deux nombres a et b.
2. Créer des pointeurs sur a et b
3. Effectuer la somme des deux nombres en utilisant les pointeurs.
4. Afficher le résultat obtenu.

Exercice d’application 2

1. Revoir la solution conversion minutes en heures (version pointeurs) du slide 8.
2. Compiler et exécuter le programme.
3. Quelle remarque pourriez vous faire?

Exercice d’application 3

- Question 1 (sans pointeurs) : Ecrire un programme qui échange les valeurs de deux variables entières a et b.
- Question 2 (avec pointeurs): Modifier le programme précédent en rajoutant une fonction echange qui prend en entrée deux pointeurs sur deux variables entières a et b et qui permet d’échanger les contenus des variables a et b.