Nicolas BODIN
bodin@esiea.fr

esiea

Les Pointeurs

Partie 1

Variables (passage par valeur)

```
/***************** incremente() ***************
Ajoute 1 a l'argument
Retourne la valeur de l'argument + 1
ARGS: int i: variable a modifier
**********************************************/
int incremente (int i)
{
   i = i+1;
   return i;
}
```

```
/***************** main() *******************
Point d'entrée du programme
**********************************************/
int main()
{
   int n=0, i;
   i = incremente(n);
   return 0;
}
```

Définition (passage par valeur)

L'argument est une copie de la variable.

La variable conserve sa valeur d'origine après l'appel de la fonction

A la fin du main(),
i vaut 1 et n vaut toujours 0

Ce qu'il se passe en mémoire

Adresse	Valeur	Nom
@0	10	a
@1	~~7~~ 11	b
@2	~~10~~ 11	~~i~~
@3		
@4		

```
int incremente (int i)
{
   i = i+1;
   return i;
}

    .
    .
    .

int main ()
{
  int a = 10;
  int b;

  b = incremente(a);

  return 0;
}
```

Définition

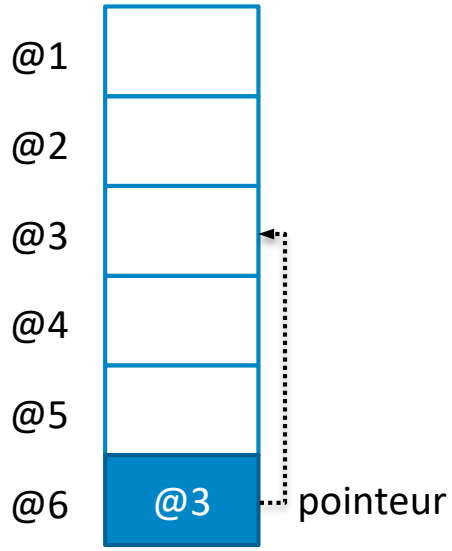
Définition (pointeur)

Un pointeur est une variable qui contient une adresse.

Référence à un *espace mémoire*

Analogie avec l'adresse d'une personne:

- Une personne = 1 adresse
- Impossible de changer l'adresse de la personne sur demande (il faut déménager)
- La facteur se réfère à l'adresse pour donner le courrier



En pratique

Déclaration et utilisation

1 pointeur = 1 type de données.

Déclaration: type + * + nom.

Possibilité d'initialiser à NULL.

Récupération de l'adresse d'une variable avec &.

```
// Déclaration
type * nom_pointeur;


// Affectation d'un pointeur
nom_pointeur = &nom_variable;
```

```
int i=10, j=5;          // variable de type entier
int *p_i = NULL;        // pointeur sur une variable de type entier

char car = 'r';         // variable de type char
char *p_car = &car;     // pointeur sur une variable de type char
```

Ce qu'il se passe en mémoire

Adresse	Valeur	Nom
@0	~~10~~ 5	varA
@1	-5	varB
@2	114	car
@3	~~NULL~~ @0	p1
@4	@1	p2

```
int varA = 10;

int varB = -5;

char car = 'r';

int *p1 = NULL;

int *p2 = &varB;

p1 = &varA;

*p1 = 5;

p1 = 10;

&varA = 23;

*varB = 8;
```

Nicolas BODIN
bodin@esiea.fr

esiea

Les Pointeurs

Partie 2

Passages par valeur / par adresse

Passage par valeur = copie de variable

Passage par adresse = modification du contenu du pointeur

```c
void permute ( int i, int j )
{
   int tmp = i;
   i = j;
   j = tmp;
}

int main()
{
   int a = 0;
   int b = 10;

   permute (a, b);

   printf ("a=%d, b=%d\n", a, b);

   return 0;
}
```

```c
void permute ( int *i, int *j )
{
   int tmp = *i;
   *i = *j;
   *j = tmp;
}

int main()
{
   int a = 0;
   int b = 10;

   permute (&a, &b);

   printf ("a=%d, b=%d\n", a, b);

   return 0;
}
```

Passages par valeur : la mémoire

Passage par valeur = copie de variable

Passage par adresse = modification du contenu du pointeur

```c
void permute ( int i, int j )
{
    int tmp = i;
    i = j;
    j = tmp;
}

int main()
{
    int a = 0;
    int b = 10;

    permute (a, b);

    printf ("a=%d, b=%d\n", a, b);

    return 0;
}
```

Adresse	Valeur	Nom
@0	0	a
@1	10	b
@2	~~0~~ 10	~~i~~
@3	~~10~~ 0	~~j~~
@4	0	~~tmp~~

Passages par adresse : la mémoire

Passage par valeur = copie de variable

Passage par adresse = modification du contenu du pointeur

```c
void permute ( int *i, int *j )
{
    int tmp = *i;
    *i = *j;
    *j = tmp;
}

int main()
{
    int a = 0;
    int b = 10;

    permute (&a, &b);

    printf ("a=%d, b=%d\n", a, b);

    return 0;
}
```

Adresse	Valeur	Nom
@0	~~0~~ 10	a
@1	~~10~~ 0	b
@2	@0	~~i~~
@3	@1	~~j~~
@4	0	~~tmp~~

Nicolas BODIN
bodin@esiea.fr

esiea

Les Pointeurs

Partie 3 : un exemple qui déchire

Passages par adresse : la mémoire

```c
int main()
{
    int a      = 0;
    short s    = 1337;
    char c1    = 'R';
    char c2    = 'S';
    int *p_a   = &a;
    char *p_c  = &c1;
    char *p_c_bis = &c2;

    *p_a++;
    c2 = *p_c + 2;
    *p_c = *p_c_bis;
    p_c = p_c_bis;
    s+=*p_a;
    s=s**p_a;

    return 0;
}
```

Adresse	Nom	Valeur
0x24FC10 0x24FC11 0x24FC12 0x24FC13	a	~~0~~ 1
0x24FC14 0x24FC15	s	~~1337~~ 1338
0x24FC16	c1	~~82~~ 84
0x24FC17	c2	~~83~~ 84
0x24FC18 0x24FC19 0x24FC1A 0x24FC1B	p_a	0x24FC10
0x24FC1C 0x24FC1D 0x24FC1E 0x24FC1F	p_c	~~0x24FC16~~ 0x24FC17
0x24FC20 0x24FC21 0x24FC22 0x24FC23	p_c_bis	0x24FC17