# Pointeurs

Immaculée Conception
ENSEIGNEMENT PRIVÉ
Mathieu DOMER
BTS CIEL-IR

# Qu’est-ce que c’est ?

Un pointeur est une **variable**

qui contient une **adresse** en mémoire.

```c
#include <stdio.h>

#define TODAY_YEAR 2023

int main() {
    int year, month, day;
    const int today_year = 2023, today_month = 9, today_day = 26;

    printf("Annee de naissance : ");
    scanf("%d", &year);
    printf("Mois de naissance : ");
    scanf("%d", &month);
    printf("Jour de naissance : ");
    scanf("%d", &day);

    if (day == today_day && month == today_month) {
        printf("Joyeux anniversaire !");
    }
    else {
        printf("Joyeux non-anniversaire !");
    }

    return 0;
}

```

# Déclaration

```
type * identificateur;
```

type de la donnée **contenue** dans la zone mémoire pointée

```
// Exemple :
int * ptr;              // pointeur sur un entier
char * pChar;           //    ...          un caractère
float * ptr_rayon;      //    ...          un float
```


KEEPOUT

# Initialisation / Affectation

```
int maVariable = 12;

int * pInt = &maVariable;    // Initialisation

int un_autre_entier = 4567;

pInt = &un_autre_entier;     // Affectation
```

# NULL

```
// Pour éviter de pointer n'importe où en mémoire...
int * ptr = NULL;
```

# Opérateurs

## &

Opérateur d'**adresse**

(unaire)

renvoie l'**adresse** d'une variable

## *

Opérateur d'**indirection**

(unaire)

renvoie la valeur stockée à l'adresse pointée par le pointeur

# Affichage

```
int maVariable = 12;
int * pInt = &maVariable;

printf("%p\n", pInt);
printf("%d\n", *pInt);
```

# Affichage

```c
float un_float = 3.1416;
float * pFloat = &un_float;
printf("%f\n", *pFloat);

char unChar = 'z';
float * p_char = &unChar;
printf("%c\n", *p_char);
```

# Taille

```
int * pInt = NULL;
printf("%d\n", sizeof(pInt));
```

 8

| Processeur | Taille des pointeurs |
|---|---|
| 64 bits | 8 octets |
| 32 | 4 |
| 16 | 2 |

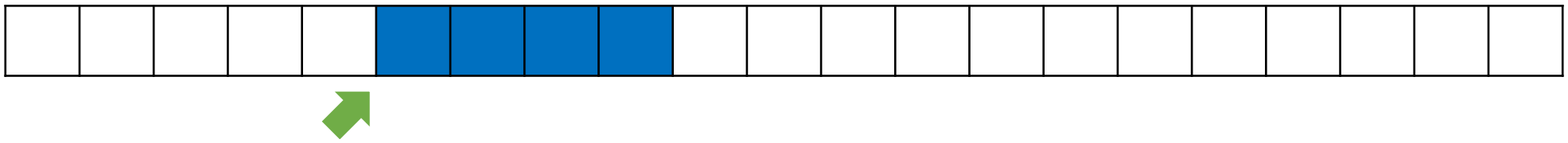
# Déclaration

```
type * identificateur;
```

type de la donnée **contenue** dans la zone mémoire pointée

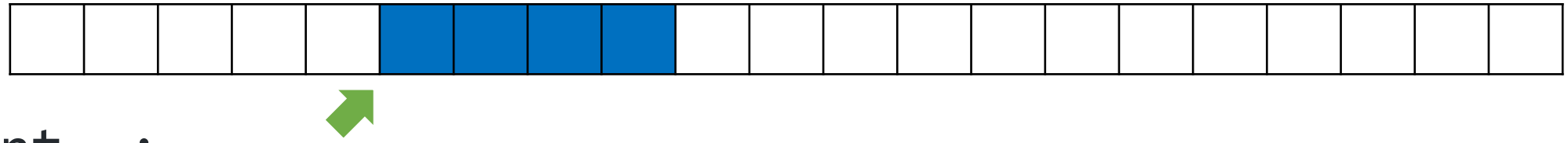
Opérateurs : `+` `-` `++` `--` `+=` `-=`

```
int maVariable = 12;
int * pInt = &maVariable;
```

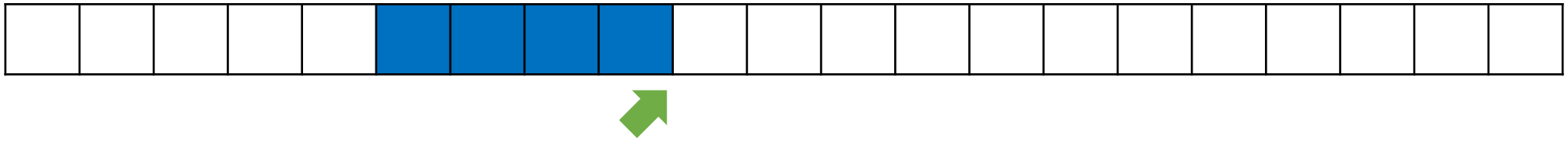


# Décalage

```
pInt++;
```

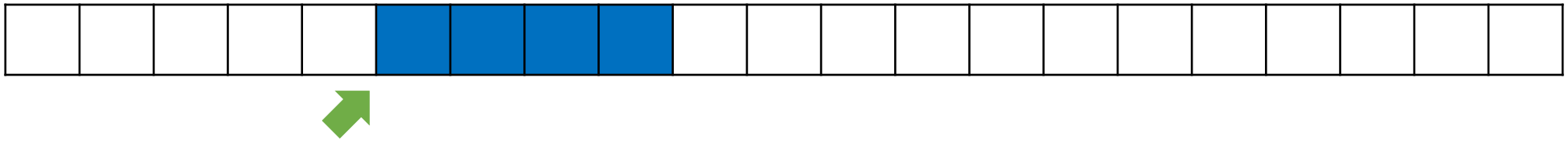


```
pInt--;
```



```
pInt += 3;

// i.e. pInt = pInt + 3;
```



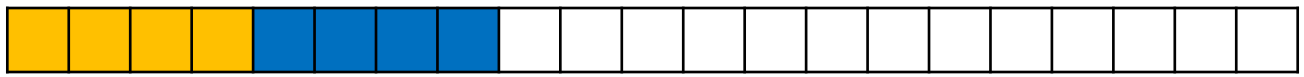
# En action !

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

# Déclaration de 2 entiers

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

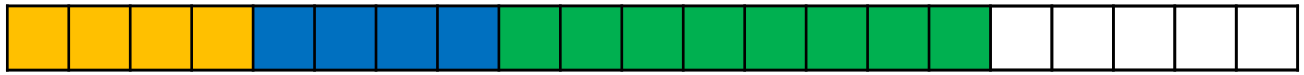
| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | | j |
| 0x104 | | i |



# Initialisation d’un pointeur

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

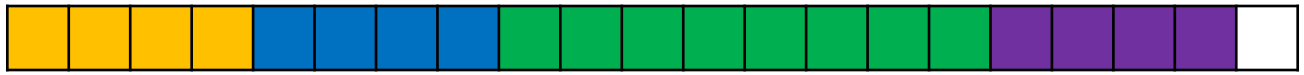
| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | | j |
| 0x104 | | i |
| 0x108 | NULL | p |



# Initialisation d’un réel

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | | j |
| 0x104 | | i |
| 0x108 | NULL | p |
| 0x110 | 0.0 | f |



# Affectation

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | | j |
| 0x104 | 5 | i |
| 0x108 | NULL | p |
| 0x10c | 0.0 | f |

# Affectation

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | | j |
| 0x104 | 5 | i |
| 0x108 | 0x104 | p |
| 0x10c | 0.0 | f |

# Affectation

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | 5 | j |
| 0x104 | 5 | i |
| 0x108 | 0x104 | p |
| 0x10c | 0.0 | f |

# Affectation

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | 5 | j |
| 0x104 | 7 | i |
| 0x108 | 0x104 | p |
| 0x10c | 0.0 | f |

# Incrémentation

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | 5 | j |
| 0x104 | 8 | i |
| 0x108 | 0x104 | p |
| 0x10c | 0.0 | f |

# Incrémentation

```
int i, j;
int *p = NULL;
float f = 0.0;
i = 5;
p = &i;
j = *p;
*p = j + 2;
i++;
p++;
```

| Adresse | Valeur | Variable |
|---|---|---|
| 0x100 | 5 | j |
| 0x104 | 8 | i |
| 0x108 | 0x108 | p |
| 0x10c | 0.0 | f |

# Pointeur de pointeur

```
int i = 42;
int * p = &i;
int ** pp = &p;
int *** ppp = &pp;

printf("%d\n", ***ppp);
```

42


# Exercices

```c
#include <stdio.h>
int main() {
    int a = 4, b = 8, c = 15, d = 16, e = 23, f = 42;
    printf("%d\n",     *&f          );              // 42
    printf("%d\n",   *(&f + 1)      );              // 23
    printf("%d\n",      c + 1       );              // 16
    printf("%d\n",   *(&a - 3)      );              // 16
    printf("%d\n",     &a - &b      );              //  1
}
```

```c
int a = 53;
int b, c;
int * p;
p = &a;
b = *p + 5;
c = &b - p;
p += c;
*p += c;
```

| Variable | Adresse |
|---|---|
| a | 000000bc06dffd98 |
| b | 000000bc06dffd9c |
| c | 000000bc06dffda0 |
| p | 000000bc06dffd90 |

```
int f = 42, e = 23, d = 16, c = 15, b = 8, a = 4;
int * p = &d, * q = &a;
*p -= 1;
++*(p - 1);
p = (int*) ((long long)p - 3 * sizeof(int));
f = a == *p ? a : b;
e = &b == p + 1 ? b : a;
*q = -1;
for (p = &c; p <= &f; p++) {
    *q += *p;
}
*(q + 1) += *(q + 3);
```

| Variable | Adresse |
|---|---|
| a | 000000bc06dffd98 |
| b | 000000bc06dffd9c |
| c | 000000bc06dffda0 |
| d | 000000bc06dffda4 |
| e | 000000bc06dffda8 |
| f | 000000bc06dffdac |
| p | 000000bc06dffd90 |
| q | 000000bc06dffd88 |

# Swap !

Créer une fonction permettant d’**échanger** les valeurs de deux variables.

# Uppercase !

Créer une fonction capable de transformer en majuscule la lettre (si c'en est une…) contenus dans une variable de type char.