

Types de données

PEAS

AND

BUCKW

RICE

BEANS

CHERRY

APRICOT

Immaculée Conception
ENSEIGNEMENT PRIVÉ

Mathieu DOMER
BTS CIEL-IR

MILLET

Types : Entiers

Type	Taille (octets/bits)	Limites
char	1 / 8	-128 à 127
unsigned char	1 / 8	0 à 255
short	2 / 16	-32 768 à +32 767
unsigned short	2 / 16	0 à +65 535
int	4 / 32	-2 147 483 648 à +2 147 483 647
unsigned int	4 / 32	0 à +4 294 967 295
long	4 / 32	-2 147 483 648 à +2 147 483 647
unsigned long	4 / 32	0 à +4 294 967 295
long long	8 / 64	-9 223 372 036 854 775 807 à +9 223 372 036 854 775 807
unsigned long long	8 / 64	0 à +18 446 744 073 709 551 615

Types : Entiers

```
int a; long c; short b; unsigned int x;
```

Affectation de valeur en fonction de la base :

Décimal (10) : 1, 289 , -9999

Octal (8) : **01**, **0273**, **0777**

Hexadécimal (16) : **0x7DF**, **0x7DF**

Types : Réels

Type	Taille (octets/bits)	Limites
float	4 / 32	Min : 1.175494×10^{-38} / Max : 3.402823×10^{38}
double	8 / 64	Min : $2.225074 \times 10^{-308}$ / Max : 1.797693×10^{308}
long double	16 / 128	Min : $2.225074 \times 10^{-308}$ / Max : 1.797693×10^{308}

```
float a; double c; long double b;
```

Décimal : 0.2 ou 927.308

Exposant : 2E-6 ou 0.06E+3



X FILES

AUX FRONTIÈRES DU RÉEL



⚠ Comparaison de réels ⚠

$$0.1 + 0.2 == 0.3$$

0.1 en IEEE754 donne 0.10000001490116119384765625

0.2 en IEEE754 donne 0.2000000298023223876953125

donc $0.1 + 0.2 = 0.30000004470348358154296875$

0.3 en IEEE754 donne 0.30000011920928955078125

$$0.1\text{f} + 0.2\text{f} == 0.3\text{f}$$



⚠️ Calculs avec des réels ⚠️

1 / 3.0f = 0.33333343267440795898437500

1 / 3.0 = 0.33333333333333314829616256

3 / 10.0 = 0.29999999999999988897769754

3 * 0.1 = 0.30000000000000044408920985

Formats de printf

Format	Sortie	Exemple
d ou i	Entier (4 octets) en base 10 signé	392
u	Entier en base 10 non signé	7235
o	Entier en octal (base 8) non signé	610
x ou X	Entier en hexadécimal (base 16) non signé	7fa ou 7FA
f	Réel avec virgule flottante	392.65
e	Réel en notation scientifique	3.9265e+2
c	Caractère	a
hhd	Entier (1 octet) en décimal signé	97
hd	Entier (2 octets) en décimal signé	97
ld	Entier (4 octets) en décimal signé	
lld	Entier (8 octets) en décimal signé	
s	Chaîne de caractères	exemple
%	Pourcentage	%

Pointeurs



Qu'est-ce que c'est ?

Un pointeur est une **variable**
qui contient une **adresse** en mémoire.

```
1 #include <stdio.h>
2
3 #define TODAY_YEAR 2023
4
5 int main() {
6     int year, month, day;
7     const int today_year = 2023, today_month = 9, today_day = 26;
8
9     printf("Année de naissance : ");
10    scanf("%d", &year);
11    printf("Mois de naissance : ");
12    scanf("%d", &month);
13    printf("Jour de naissance : ");
14    scanf("%d", &day);
15
16    if (day == today_day && month == today_month) {
17        printf("Joyeux anniversaire !");
18    }
19    else {
20        printf("Joyeux non-anniversaire !");
21    }
22
23    return 0;
24}
25
```



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

KEEP OUT

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); → 0000005ef91ffa14
```

```
printf("%d\n", *pInt); → 12
```

Affichage

```
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

Décalage

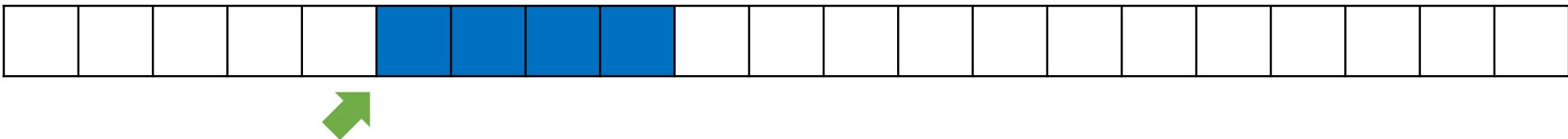
type * identificateur;

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

Opérateurs : + - ++ -- += -=

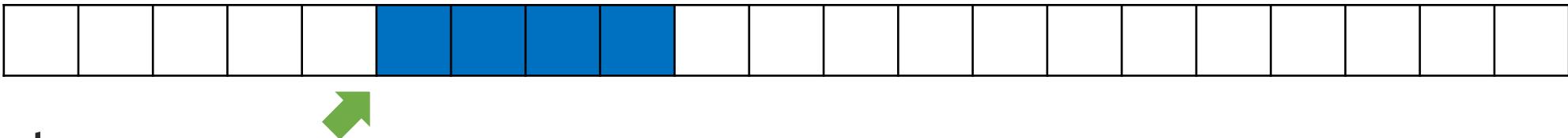
```
int maVariable = 12;
```

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

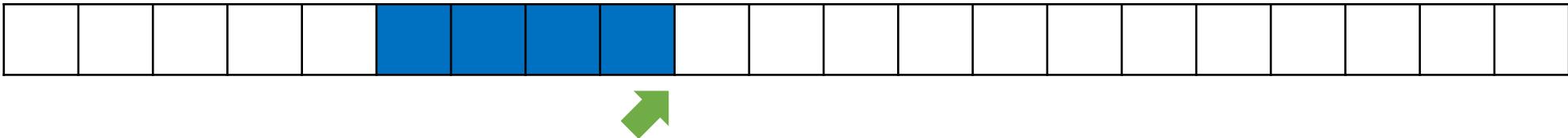


Décalage

`pInt++;`

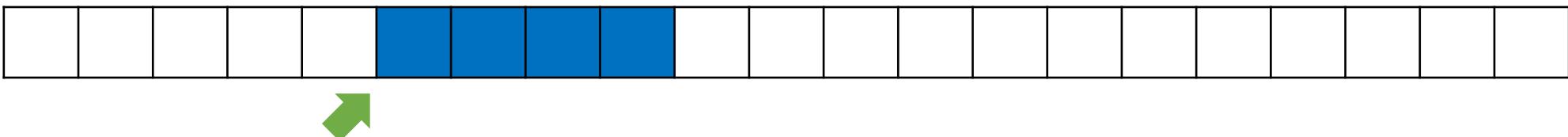


`pInt--;`



`pInt += 3;`

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



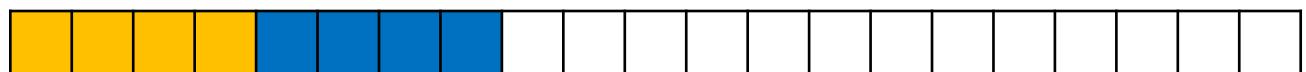
En action !

```
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 de pointeur

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

```
printf("%d\n", ***ppp); ➔ 42
```