

# La Programmation en langage C

Révision

**La cible : SMI S3**

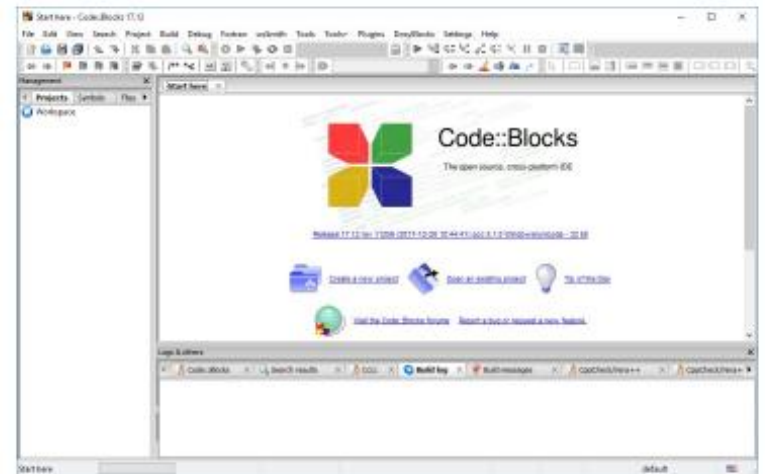
Pr S.ELFILALI

# Introduction

# Programme et Logiciel



Jeux vidéos



Editeur de code

# Ecrire le code d'un gateau



## Ingrédients:

- 4 oeufs.
- 200g de sucre.
- 200g de beurre.
- 200g de farine.

## Instructions:

1. Allumer le four pour le préchauffer à 150°C.
2. Mélanger les ingrédients dans un saladier.
3. Verser le contenu du saladier dans un moule à gâteau.
4. Quand le four est chaud, enfourner le gâteau.
5. Attendre 40 minutes de cuisson.
6. Sortir le gâteau du four.
7. Laisser refroidir le gâteau.
8. Démouler le gâteau.
9. Miam !

# Du code au programme

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int vie = 3;
    int game_over = 0;

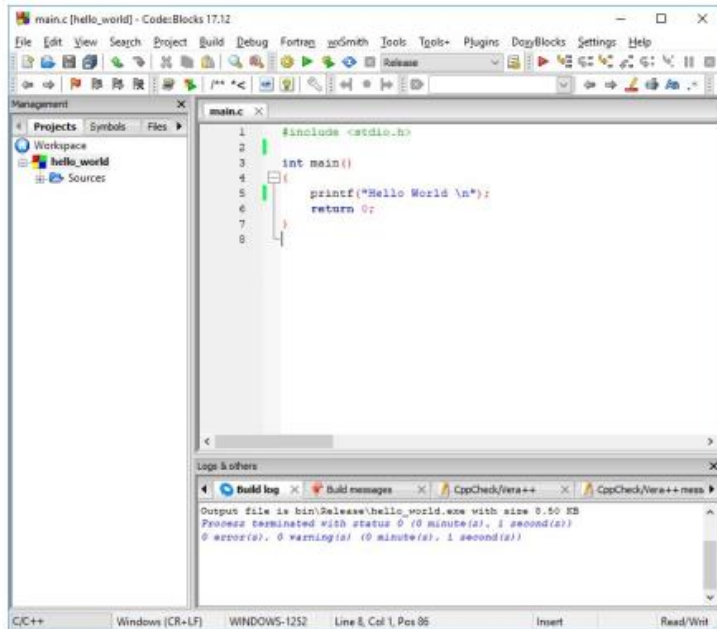
    InitialiserLeJeu();
    AfficherMenuDemarage();
    ...
}
```

Code source



Programme exécutable

# Hello world



The screenshot shows the Code::Blocks IDE with a project named 'hello\_world'. The main.c file is open, displaying the following C code:

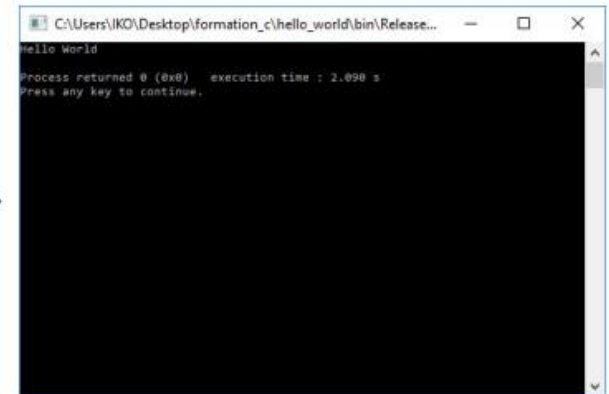
```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World \n");
6     return 0;
7 }
8
```

The bottom panel shows the 'Build log' with the following output:

```
Output file is bin\Release\hello_world.exe with size 0.50 KB
Process terminated with status 0 (0 minute(s), 1 second(s))
0 error(s), 0 warning(s) (0 minute(s), 1 second(s))
```

Code source

compilateur

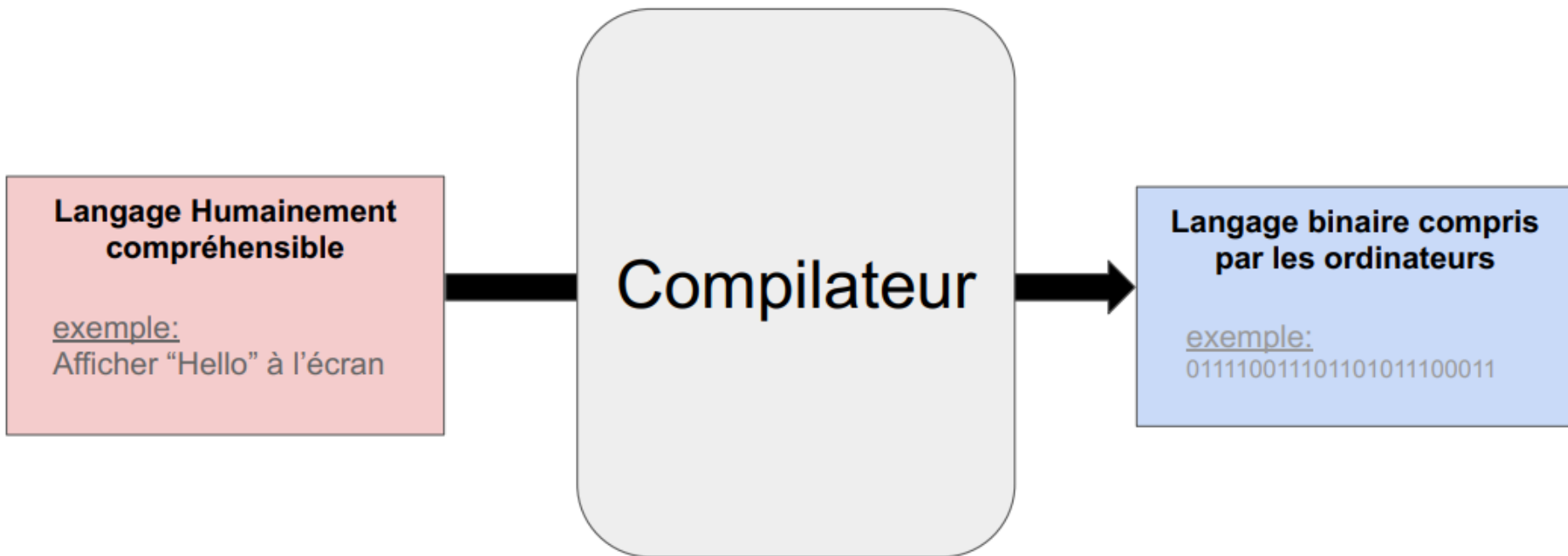


The screenshot shows a command prompt window with the following output:

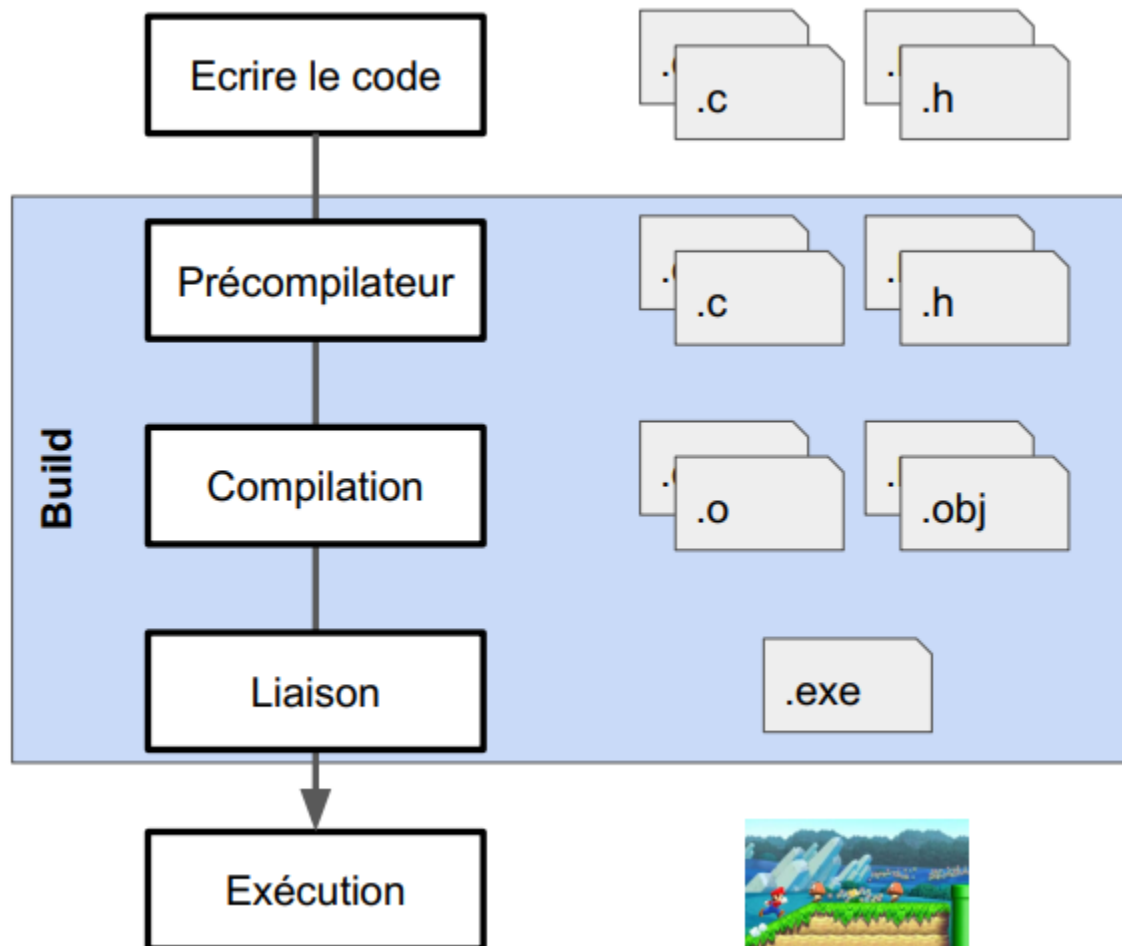
```
C:\Users\JKO\Desktop\formation_c\hello_world\bin\Release...
Hello World
Process returned 0 (0x0)   execution time : 2.098 s
Press any key to continue.
```

Programme exécutable

# Le compilateur

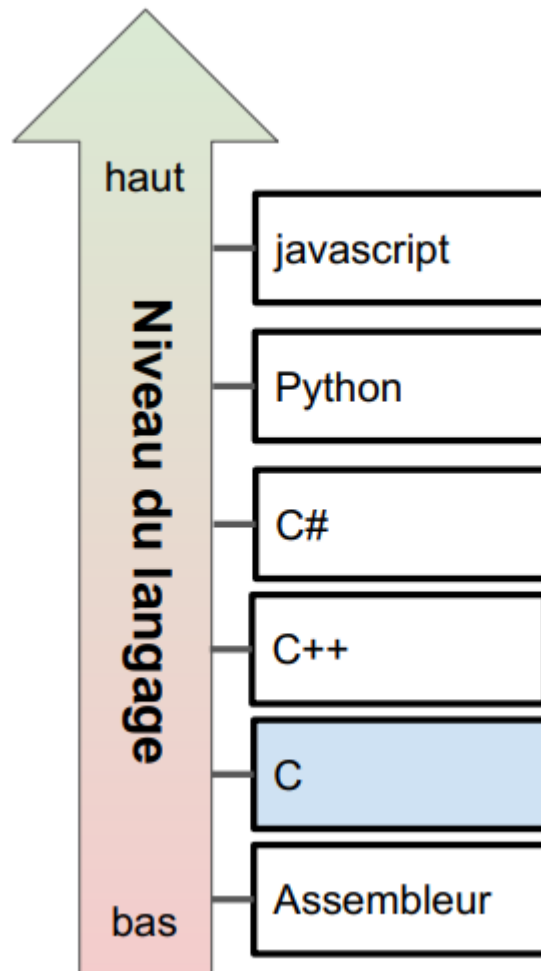


# Création d'un logiciel



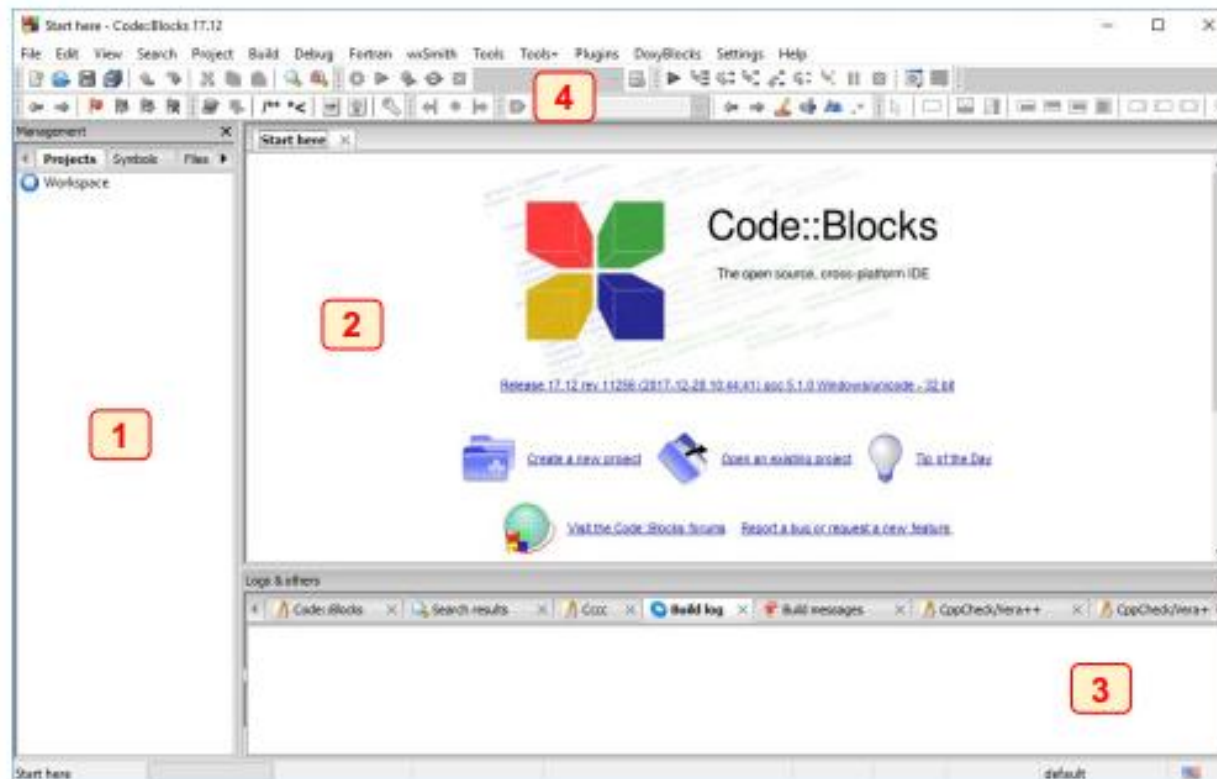


# Niveau des langages



# La mise en place

# Code::blocks



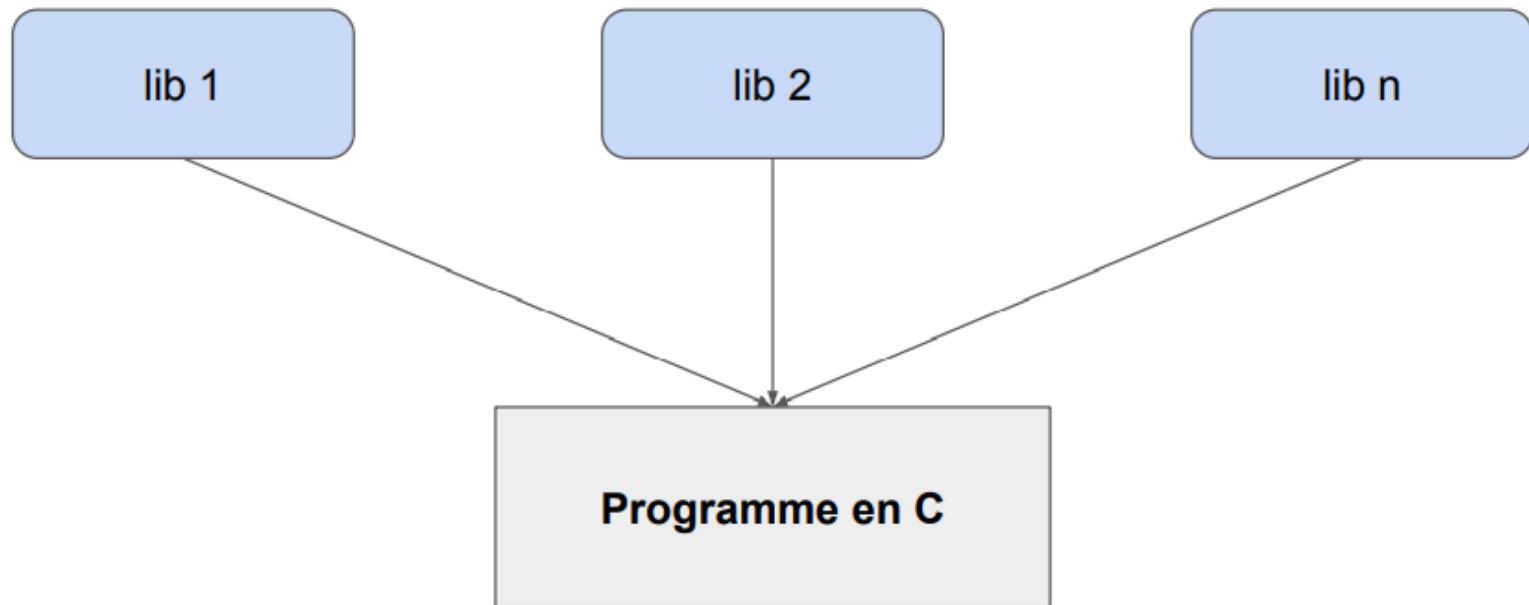
Téléchargement: <http://www.codeblocks.org/downloads/binaries>

# Compiler le code

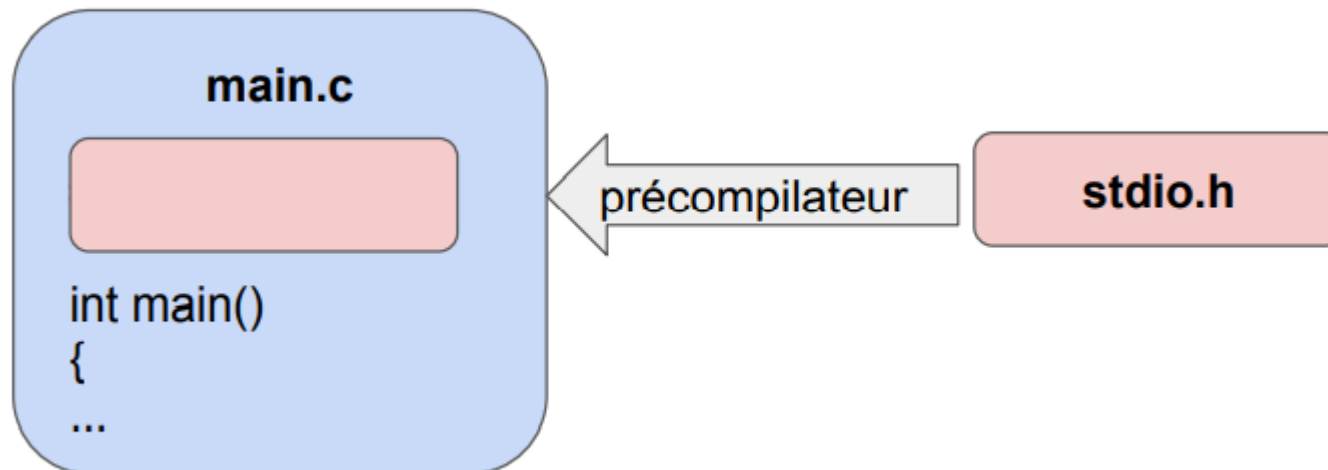


# Syntaxes et structures

# Le C et les bibliothèques



# Les INCLUDES



# Le Contenu de la STL

## STL (Standard Template Library)

stdio.h

stdlib.h

math.h

...

Documentation de la STL: <https://devdocs.io/c>



# La fonction main

```
int main()
```

```
{
```

code

```
return 0;
```

```
}
```

Fonction principale et indispensable

# Les Commentaires

// commentaire sur une ligne

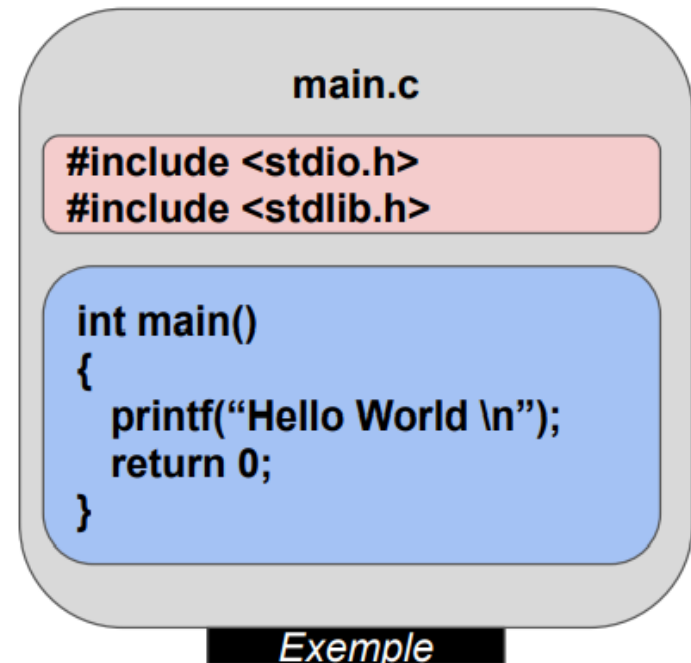
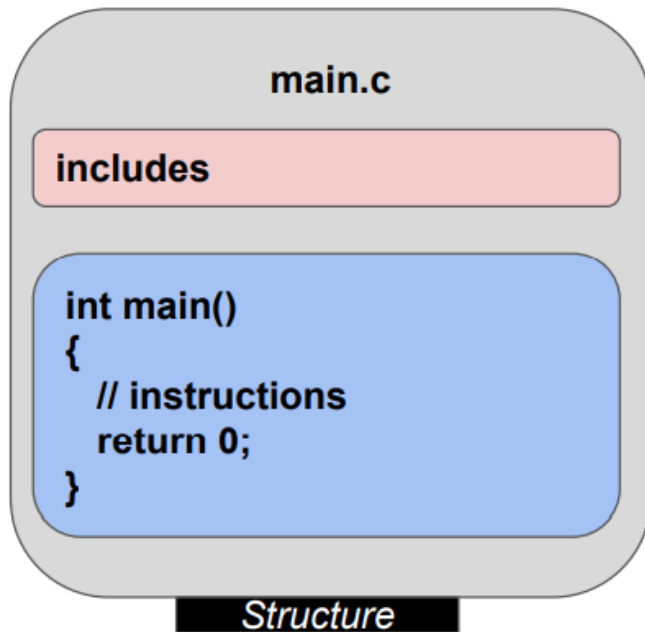
```
/*  
  bloc de  
  commentaires  
*/
```

```
// Affiche un message à l'écran  
printf("Hello world! \n");
```

Précompilateur

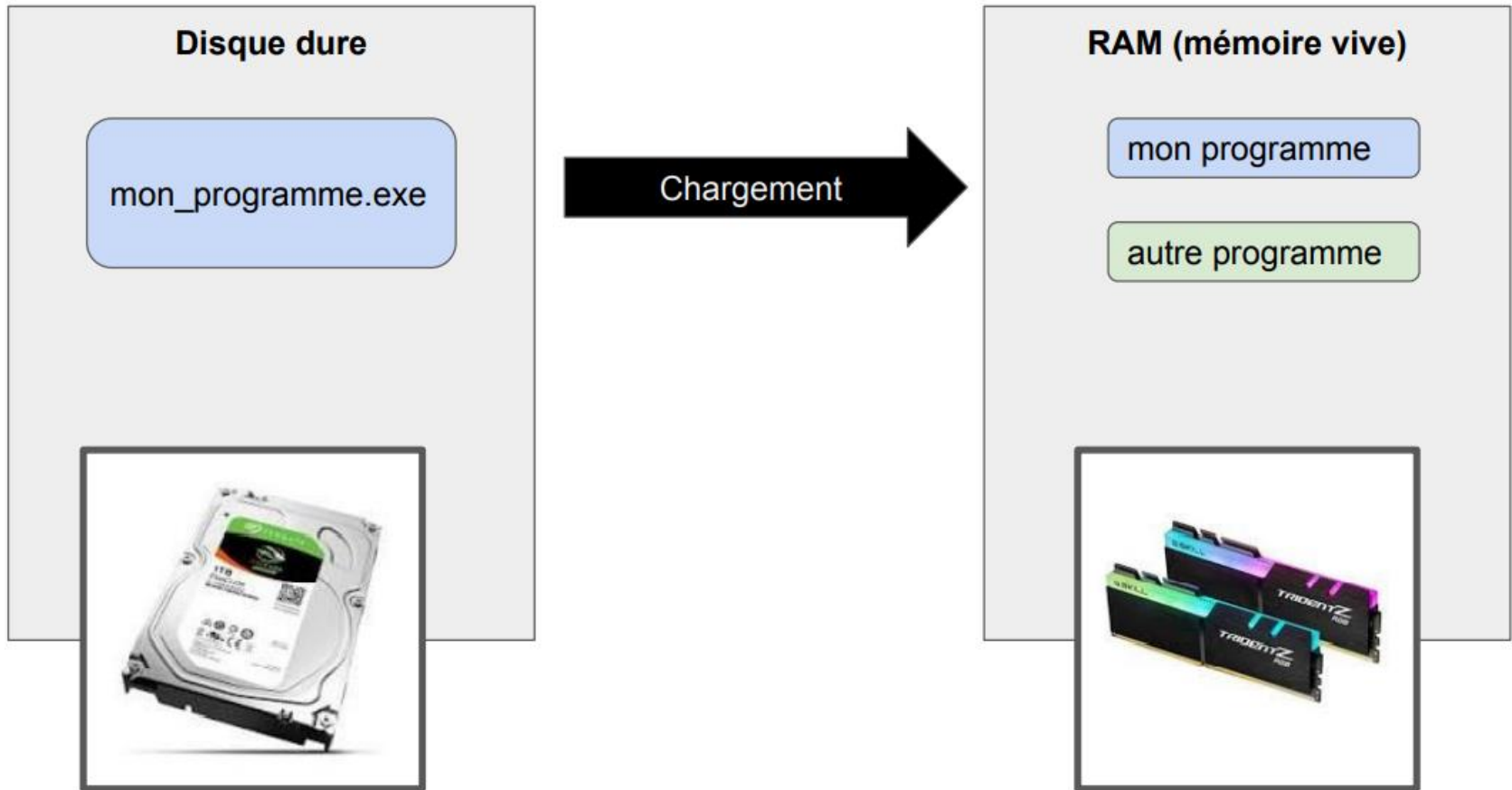
```
printf("Hello world! \n");
```

# Structure de base d'un programme en C

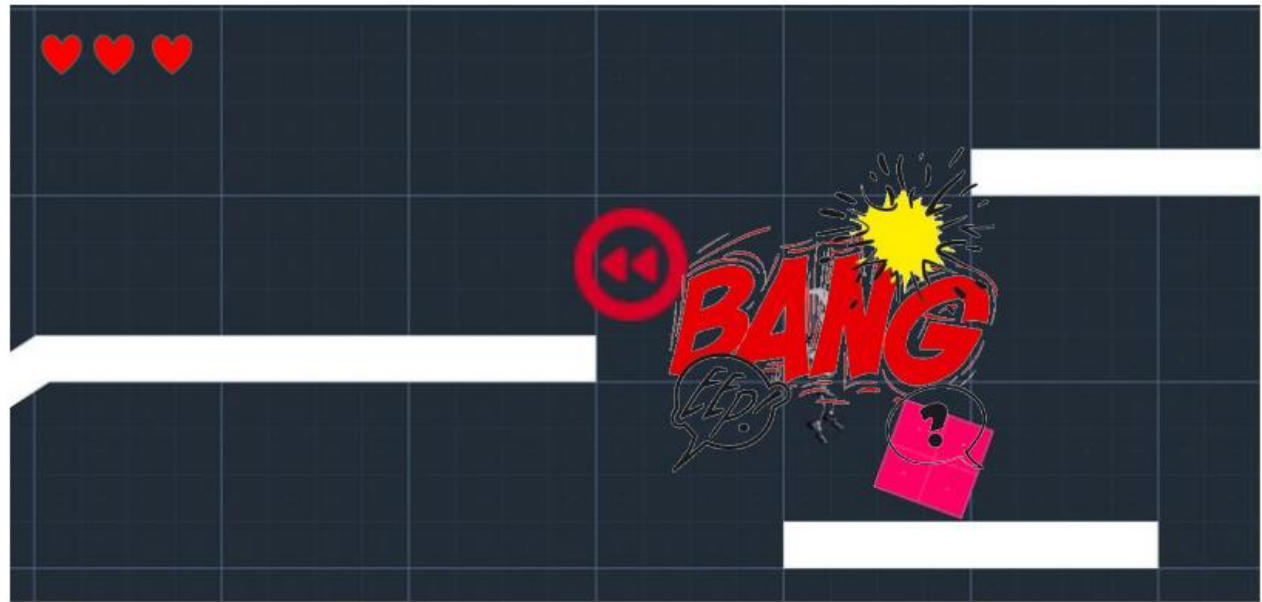
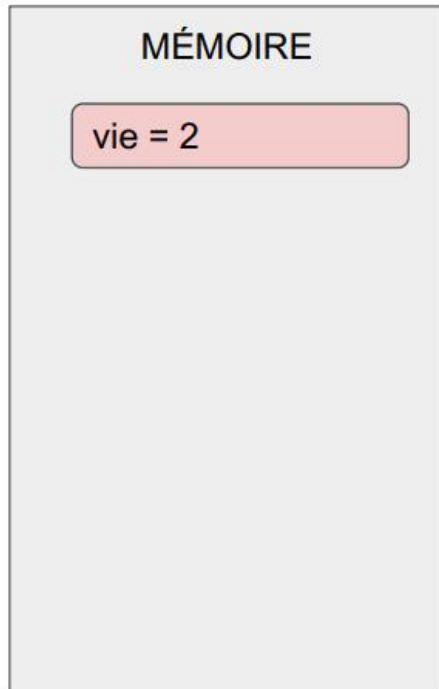


# Variables et mémoire

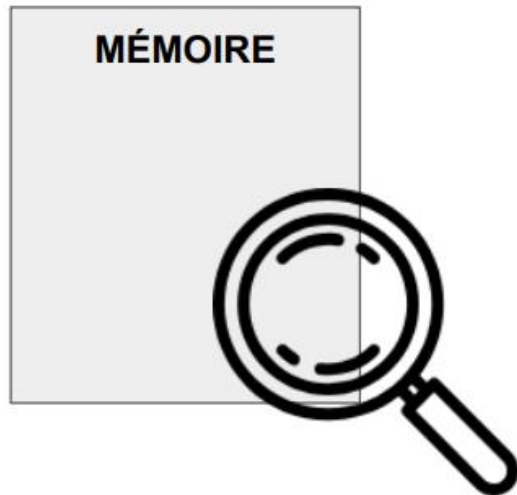
# Exécuter un programme



# Utilisation de la RAM



# Gestion de la mémoire

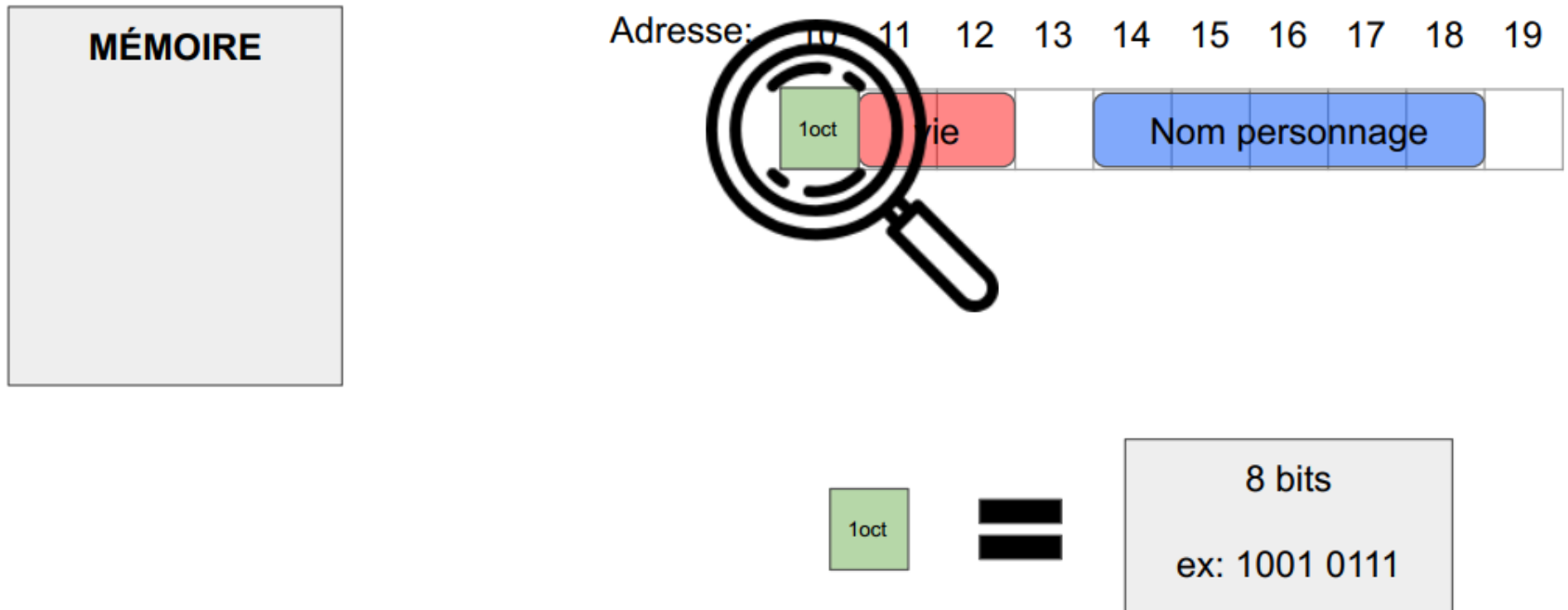


Adresse: 10 11 12 13 14 15 16 17 18 19

	vie		Nom personnage	
--	-----	--	----------------	--

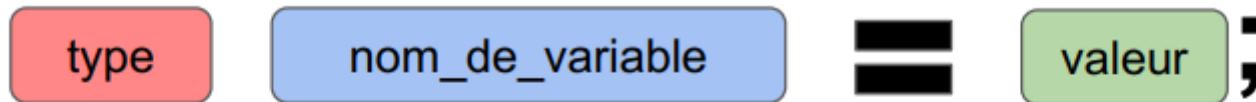


# Gestion de la mémoire





# Déclaration de variable



## Exemple:

```
int ma_variable = 0;  
int vie = 3;
```



# Réglage de nommage des variables

## Règles

Pas de nombre en début de nom

Pas de caractères spéciaux à l'exception de \_

Pas de mots réservés en c (ex type de variable)

Pas d'accent (à, é, è , ...)

Pas d'espace

Nom explicite (très important pour la lisibilité)

tout en minuscule avec \_ pour séparer les mots

## Exemples

6variable

nom-de-variable

int

début

nom de variable

NomDeVariable

nom\_de\_variable

# Mots réservés en C

auto	break	case	char
const	default	do	double
else	enum	extern	float
for	goto	if	int
long	register	return	short
signed	sizeof	static	struct
switch	typedef	union	unsigned
void	volatile	while	

# Les types de base

Type	Taille (octets)	Valeur min	Valeur max	Affichage
int	2 / 4	-32768 / -2147483648	32767 / 2147483647	%d
char	1	-128	127	%c
float	4	-3.4e38	3.4e38	%f
double	8	-1.7e308	1.7e308	%f

# Les types signed et unsigned

Type	Taille (octets)	Valeur min	Valeur max	Affichage
unsigned int	2 / 4	0 / 0	65535/ 4294967295	%d
signed int	2 / 4	-32768 / -2147483648	32767 / 2147483647	%d
int	2 / 4	-32768 / -2147483648	32767 / 2147483647	%d
unsigned char	1	0	255	%c
signed char	1	-128	127	%c
char	1	-128	127	%c
float	4	-3.4e38	3.4e38	%f
double	8	-1.7e308	1.7e308	%f

# Les types signed et unsigned

Type	Taille (octets)	Valeur min	Valeur max	Affichage
unsigned short int / unsigned short	2	0	65535	%d
short int / short	2	-32768	32767	%d
unsigned long int / unsigned long	4 / 8	0 / 0	4294967295 / 1844674407 3709551615	%ld
long int / long	4 / 8	-2147483648 / -9223372036 854775808	2147483647 / 9223372036 854775807	%ld
unsigned int	2 / 4	0 / 0	65535/ 4294967295	%d
int	2 / 4	-32768 / -2147483648	32767 / 2147483647	%d
unsigned char	1	0	255	%c
signed char	1	-128	127	%c
char	1	-128 / 0	127 / 255	%c
float	4	-3.4e38	3.4e38	%f
double	8	-1.7e308	1.7e308	%f

# Opérations et variables

# Les Opérateurs

Symbole	Signification	Exemple
=	Assignment	ma_variable = 12
+	Addition	ma_variable + 3
-	Soustraction	ma_variable - 1
*	Multiplication	ma_variable * 2
/	Division	ma_variable / 100
%	Modulo	ma_variable % 2

$5 \% 2 = 1 \quad \Rightarrow \quad 2 * 2 + 1$



# Les opérateurs spéciaux

Symbole	Exemple	Équivalent
<b>+=</b>	ma_variable += 5	ma_variable = ma_variable + 5
<b>-=</b>	ma_variable -= 5	ma_variable = ma_variable - 5
<b>*=</b>	ma_variable *= 5	ma_variable = ma_variable * 5
<b>/=</b>	ma_variable /= 5	ma_variable = ma_variable / 5
<b>%=</b>	ma_variable %= 5	ma_variable = ma_variable % 5

# Les incréments et décréments

Symbole	Exemple	Équivalent
<b>++</b>	ma_variable ++	ma_variable = ma_variable + 1
<b>--</b>	ma_variable --	ma_variable = ma_variable - 1

# Préfixe et suffixe

```
int main()
{
    int point_vie = 10;
    int potion = 5;
    printf("Point de vie = %d\n", point_vie);

    point_vie = ++potion;

    printf("Point de vie = %d\n", point_vie);
    printf("Potion = %d\n", potion);

    return 0;
}
```

Préfixe

Point de vie = 10

Point de vie = 6

Potion = 6

```
int main()
{
    int point_vie = 10;
    int potion = 5;
    printf("Point de vie = %d\n", point_vie);

    point_vie = potion++;

    printf("Point de vie = %d\n", point_vie);
    printf("Potion = %d\n", potion);

    return 0;
}
```

Suffixe

Point de vie = 10

Point de vie = 5

Potion = 6

# Lecture depuis le clavier

Scanf ()

# Complément sur les variables

# Les constantes

**const**   type   NOM\_DE\_CONSTANTE   =   valeur ;

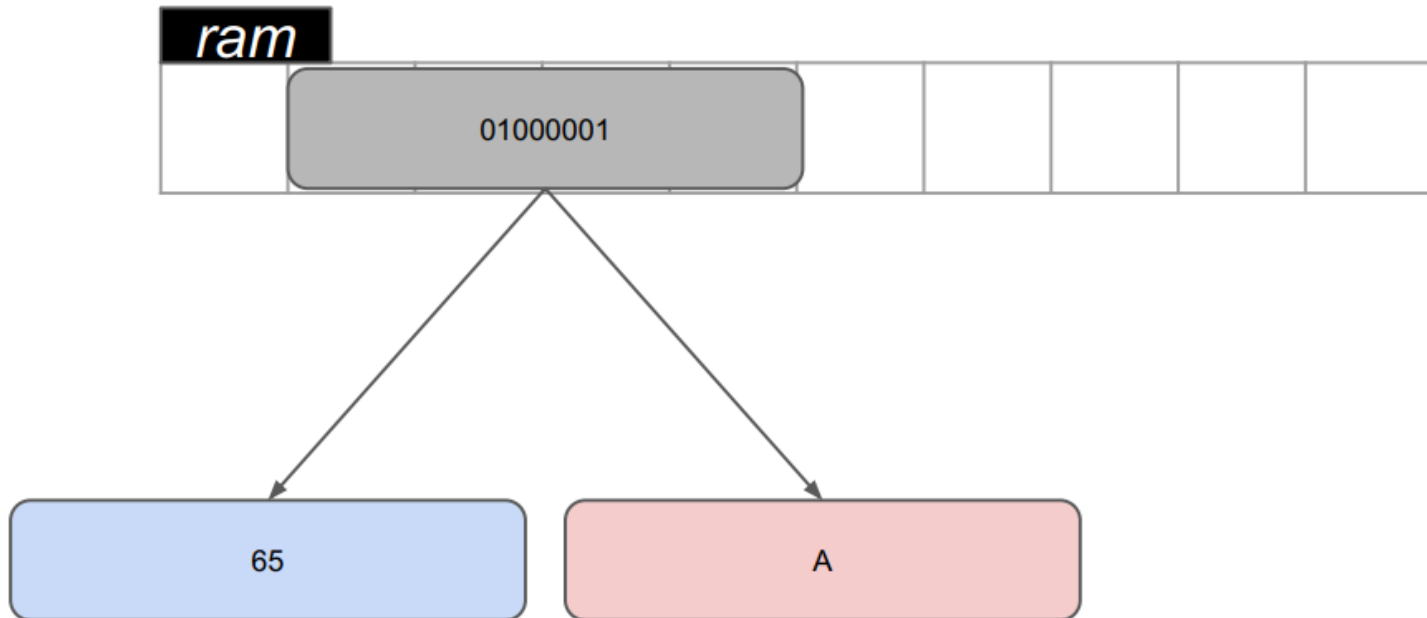
**Exemple:**

```
const int TVA = 20;  
const int MAX_VIES = 3;
```



Règles de nommage identique au variable mais tout en majuscule

# Typage



# Code ascii

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



# Overflow

	Valeur min (dec)	Valeur min (bin)	Valeur max (dec)	Valeur max (bin)
unsigned short (2 octets)	0	0000 0000 0000 0000	65535	1111 1111 1111 1111



```
unsigned short ma_variable = 0;  
ma_variable --;  
  
printf("ma_variable = %d", ma_variable);
```

*Exemple*

ma\_variable = 65535

# Les conditions

# Structure conditionnelle

Condition

```
{  
  Bloc d'instructions  
}
```

Lecture login  
Lecture password

Si login et password correcte alors

```
{  
  Afficher le message secret  
}
```

Lecture age

Si age  $\geq$  18 alors

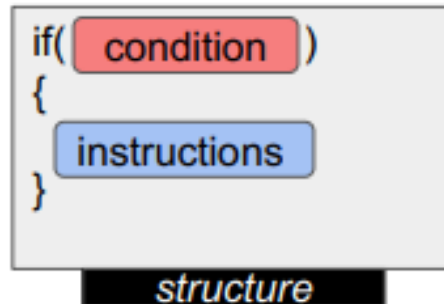
```
{  
  verser un verre de ...  
}
```

# Les conditions

Symbole	Signification	Exemple
<code>==</code>	est égale à	<code>(password == 1852)</code>
<code>!=</code>	est différent de	<code>(password != 1852)</code>
<code>&lt;</code>	est inférieur à	<code>(age_utilisateur &lt; 18)</code>
<code>&gt;</code>	est supérieur à	<code>(age_utilisateur &gt; 18)</code>
<code>&lt;=</code>	est inférieur ou égale à	<code>(age_utilisateur &lt;= 0)</code>
<code>&gt;=</code>	est supérieur ou égale à	<code>(age_utilisateur &gt;= 0)</code>

retourne 1(vrai) si la condition est respectée, sinon retourne 0(faux)

# Structure if



```
int age = 0;
printf("Quel age a tu?\n");
scanf("%d", &age);
```

```
if( age >= 18 )
{
    printf("Tu es donc majeur \n");
}
```

*Exemple*

Quelle age a tu?  
25  
Tu es donc majeur

Quelle age a tu?  
16

# Structure if else

```
if( condition )  
{  
    instructions  
}  
else  
{  
    instructions  
}
```

*Structure*

```
int age = 0;  
printf("Quel age a tu?\n");  
scanf("%d", &age);
```

```
if( age >= 18 )  
{  
    printf("Tu es donc majeur \n");  
}  
else  
{  
    printf("Tu es donc mineur \n");  
}
```

Quelle age a tu?  
25  
Tu es donc majeur

Quelle age a tu?  
16  
Tu es donc mineur

*Exemple*

# Opérateur AND &&

condition1 && condition2

condition 1	condition 2	condition1 && condition2
1(vrai)	1(vrai)	1(vrai)
1(vrai)	0(faux)	0(faux)
0(faux)	1(vrai)	0(faux)
0(faux)	0(faux)	0(faux)

```
if( age >= 18 && argent >= 5)
{
    printf("Voici votre biere \n");
}
```

**Exemple**

# Opérateur OU ||

condition1

||

condition2

condition 1	condition 2	condition1    condition2
1(vrai)	1(vrai)	1(vrai)
1(vrai)	0(faux)	1(vrai)
0(faux)	1(vrai)	1(vrai)
0(faux)	0(faux)	0(faux)

```
if( password==1852 || age>=18)
{
    printf("Acces autorises \n");
}
```

**Exemple**



# Opérateur NOT !

! condition

condition	! condition
1(vrai)	0(faux)
0(faux)	1(vrai)

```
if( !(age < 18) )  
{  
    printf("Acces autorises \n");  
}
```

*Exemple*

# Les Opérateurs logiques

Symbole	Signification	Exemple
&&	and (et)	(age>=0 && age<100)
	or (ou)	(password==1852    age>=18)
!	not (non)	!(age_utilisateur < 18)

# Priorité des Opérateurs logiques

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main()
6  {
7      int age_utilisateur = 17;
8      float taille_utilisateur = 1.70;
9      int accompagne = 1;
10
11     if( (age_utilisateur >= 18) && (taille_utilisateur > 1.65) || (accompagne) )
12     {
13         printf("c'est ok\n");
14     }
15     else
16     {
17         printf("ce n'est pas possible !\n");
18     }
19
20     return 0;
21 }
22
```

Faux && (vrai || vrai)

Faux && vrai

faux

Faux && vrai || vrai

Faux || vrai

vrai

# Exercice

- 1- Demander à l'utilisateur de rentrer un nombre entier
- 2- Afficher si le nombre est pair ou impair

```
Donnez un nombre entier: 8
8 est un nombre pair
```

1- Il faut utiliser la fonction scanf. Attention, cette fonction utilise les adresses des variables "&ma\_variable". 2- Il faut utiliser l'opérateur modulo "%" pour récupérer le reste d'une division et ensuite tester sa valeur.

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main()
{
    int nombre = 0;

    // Lecture de la saisie utilisateur
    printf("Donnez une nombre entier: ");
    scanf("%d", &nombre);

    // Test sur l'entier pour savoir si il est pair ou non
    if( (nombre%2) == 0)
        printf("\t%d est un nombre pair\n", nombre);
    else
        printf("\t%d est un nombre impair\n", nombre);

    return 0;
}
```

# L'imbrication des conditions : If else if else

```
if( condition )
{
    instructions
}
else if ( condition )
{
    instructions
}
...
else
{
    instructions
}
```

*Structure*

```
if(note == 0)
{
    printf("Null! \n");
}
else if(note < 10)
{
    printf("Peut mieux faire \n");
}
else if(note < 15)
{
    printf("Bien\n");
}
else
{
    printf("Tres bien! \n");
}
```

note = 3  
Peut Mieux faire

note = 0  
Null!

note = 15  
Tres bien !

*exemple*

# Structure switch

```
switch( variable )
{
    case valeur :
        instructions
    break;

    ...

    default :
        instructions
    break;
}
```

**structure**

```
int numero_mois = 9;
switch(numero_mois)
{
    case 1:
        printf("Janvier\n");
        break;
    ...

    case 12:
        printf("Decembre\n");
        break;

    default:
        printf("numero de mois invalide\n");
        break;
}
```

**exemple**

Septembre

# Opérateur conditionnel

condition ? valeur si vrai : valeur si faux ;

```
int nb_vie = 5;  
int points = (nb_vie >= 3) ? 300 : 0;  
printf("Vous avez %d points\n", points);
```

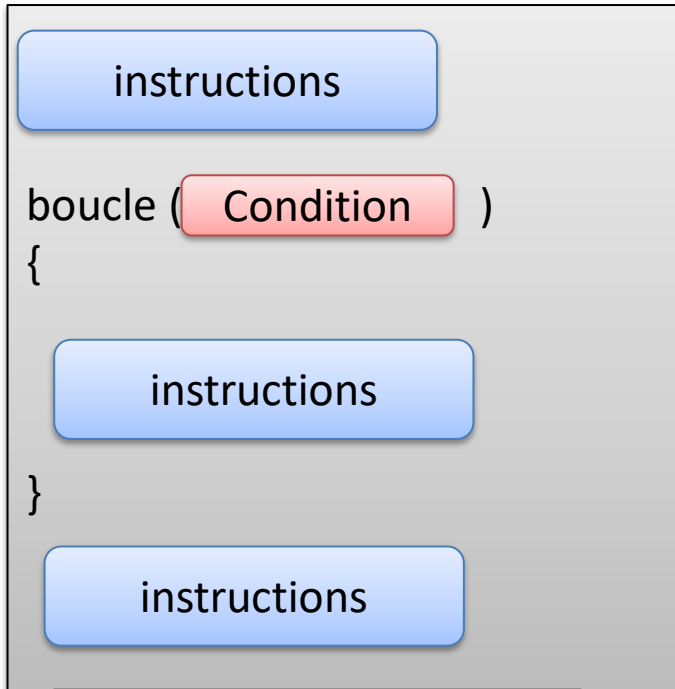
*exemple*

Vous avez 300 points

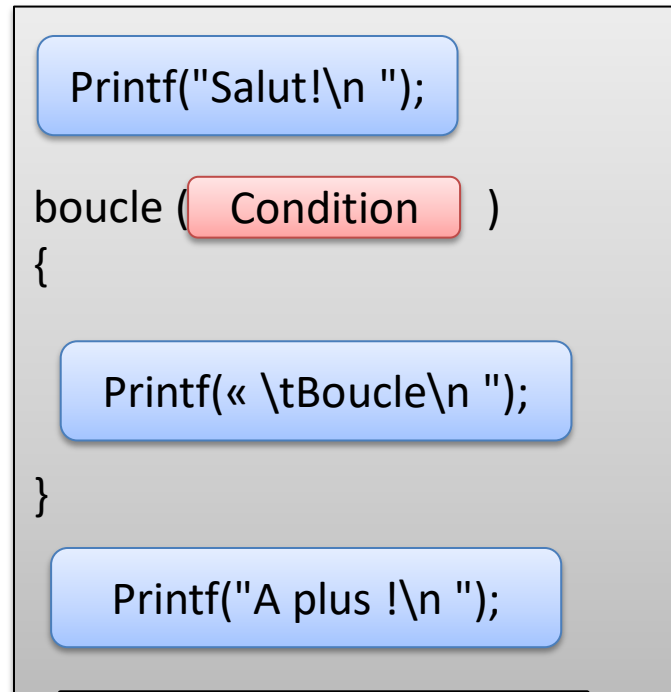


# Les Boucles

# Les boucles



Structure



Exemple

Salut !

Boucle  
Boucle  
Boucle  
Boucle  
Boucle

A plus !

Boucles en C : while, do while et for

# Les boucles

C'est quoi une boucle ?

Boucle c'est un ensemble d'instructions qui est exécuté un certain nombre de fois tant que la condition liée est vraie

# Les boucles

## La boucle while

```
While ( Condition )  
{  
    instructions  
}
```

Structure

```
Int var =0;  
While ( var < 5 )  
{  
    Printf("%d\n" , var);  
    Var++;  
}
```

Exemple

0  
1  
2  
3  
4

# Les boucles

## La boucle while

Exemple : compte à rebours

Passer sur codeblocks et faire un programme sans boucle et avec la boucle

Utiliser les fonctions sleep et générer le bip par `printf("\a");`

*la fonction `Sleep("temps en millisecondes");`, permet de faire attendre le programme pendant un certains nombre de millisecondes avant de continuer à exécuter les instructions.*

## La boucle while

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int compte=10;
    printf("\a");
    getch();
    printf("explosion dans 10 s");

    printf("\ncompte à rebours ..... !\n");

    while (compte>0)
    {
        printf("\n%d",compte);
        compte--;
        sleep(1);
    }

    printf("\a");

    printf("\nBOOOOOO MMM");
    return 0;
}
```

# Les boucles

## La boucle do while

```
do  
{  
    instructions  
} While ( Condition );
```

Structure

```
Int var =6;  
do  
{  
    Printf("%d\n" , var);  
    Var++;  
} While ( var < 5 );
```

Exemple

6

# Les boucles

## La boucle do while

Exemple : contrôle de saisie

Passer sur codeblocks et faire un programme qui demande à l'utilisateur de saisir un nombre entre 1 et 10



# Les boucles

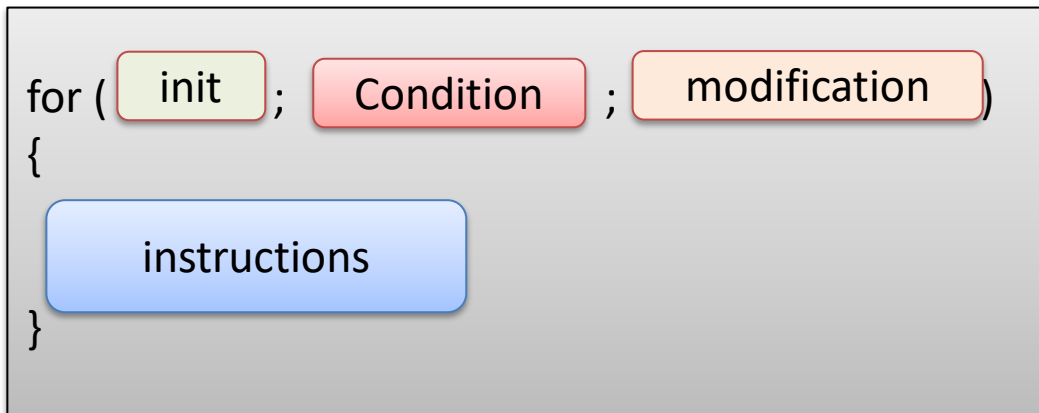
## La boucle do while

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int saisie_utilisateur = 0;
    do
    {
        printf("donner un nombre entre 1 et 10 : ");
        scanf("%d",&saisie_utilisateur);
    }
    while (saisie_utilisateur <1 || saisie_utilisateur >10);
    return 0;
}
```

# Les boucles

## La boucle for



Structure

```
for (int var =0 ; var<5 ; var++ )  
{  
    printf("%d\n",var);  
}
```

Exemple

0  
1  
2  
3  
4

```
int var =0 ;  
while(var<5)  
{  
    printf("%d\n",var);  
    var ++ ;  
}
```

Equivalent

0  
1  
2  
3  
4

# Les boucles

## La boucle for

Exemple : compte à rebours

Passer sur codeblocks et faire un programme avec la boucle for

Utiliser les fonctions sleep et générer le bip par `printf("\a");`

*la fonction `Sleep("temps en millisecondes");`, permet de faire attendre le programme pendant un certains nombre de millisecondes avant de continuer à exécuter les instructions.*

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int compteur = 10;
    printf("Explosion dans  ");
    for( compteur = 10 ; compteur>0 ; compteur--)
    {
        printf("\n%d", compteur);
    }
    printf("\n\na BOOM");
    return 0;
}
```

# Règles d'or des boucles

Comment choisir la boucle qu'on va utiliser ?

Si on connaît le nombre d'itération du bloc d'instruction alors on utilise la boucle for

Si on connaît pas le nombre d'itération du bloc alors nous avons deux choix :

Si on veut exécuter au moins une fois le bloc d'instruction alors c'est la boucle « do .. While »

Et dans tous les autres cas , on utilise la boucle while

# Règles d'or des boucles

## Boucle infinie

C'est une boucle qui va s'exécuter à l'infini!

Cad que sa condition ne va jamais passer à faux!!  
Le programme va se bloquer !!

Le seul moyen pour le débloquent : **Ctrl+C** ou **Alt F4**

### Exemple

Apportez les modification suivantes au programme et commentez :

1. Compteur ++
2. Modifier le type en short et compteur ++
3. Modifier la condition : >-99999 et compteur --

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int compteur = 10;
    printf("Explosion dans ");
    for( compteur = 10 ; compteur>0 ; compteur --)
    {
        printf("\n%d", compteur);
    }
    printf("\n\na BOOM");
    return 0;
}
```

# Règles d'or des boucles

## Boucle infinie

Apportez les modifications suivantes au programme et commentez :

1. **Compteur ++** : boucle infinie, on n'arrive pas à voir le max de ce type, on doit l'arrêter par Ctrl+C ... la condition peut être fautive au max de ce type
2. **Modifier le type en short et compteur ++** : il s'arrête au max du short qui est 32767 et puis il passe à la première valeur de type int avec une incrémentation et dans ce cas la condition n'est pas vérifiée et la boucle s'arrête !
3. **Modifier la condition : >-99999 et compteur --** : la condition est toujours vérifiée et il ne va jamais s'arrêter !!!!!

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int compteur = 10;
    printf("Explosion dans ");
    for( compteur = 10 ; compteur>0 ; compteur --)
    {
        printf("\n%d", compteur);
    }
    printf("\n\n BOOM");
    return 0;
}
```

# Les boucles

## Exercice : code PIN

- 1- Créer un programme qui demande à l'utilisateur de rentrer le code PIN à 4 chiffres. Si il trouve le bon code, afficher le message : téléphone déverrouillé sinon afficher un message d'erreur et redemander d'entrer le code.
- 2- Gérer un nombre maximum de tentatives: Exemple 3 tentatives

```
Code PIN: 1234
    Erreur, il vous reste 2 tentatives.
Code PIN: 0000
    Telephone deverrouille
```

- 1- On ne connaît pas le nombre d'itération mais on souhaite entrer dans la boucle au moins une fois. On utilise alors une boucle "Do While"
- 2- Créer une constante pour le nombre maximum de tentatives.



# Les boucles

## Solution : code PIN

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    const int CODE_PIN = 0000;
    const int MAX_TENTATIVE = 3;
    int saisie;
    int nbre_Tentative= MAX_
TENTATIVE;
    // lecture de la saisie

    do {
        printf("Code Pin : ");
        scanf("%d",&saisie);

        if(saisie==CODE_PIN)
        {
            printf("\tTelephone deverrouille\n");
        }
        else
        {
            nbre_Tentative --;
            printf("\tErreur , il vous reste %d tentatives\n", nbre_Tentative);
        }
    }
    while(saisie!=CODE_PIN && nbre_Tentative > 0 );
    return 0;
}
```

Code PIN: 1234  
Erreur, il vous reste 2 tentatives.  
Code PIN: 0000  
Telephone deverrouille

# Les boucles

## Break et Continue

### break : sortir de la boucle courante

```
int main()
{
    int var ;
    printf("debut \n");
    for (var =0 ; var<5 ; var++ )
    {
        if(var ==2)
        {
            break;
        }
        printf("%d\n",var);
    }
    printf("fin \n");
    return 0;
}
```

debut  
0  
1  
fin

break

### continue : sauter a la fin de la boucle

```
int main()
{
    int var ;
    printf("debut \n");
    for (var =0 ; var<5 ; var++ )
    {
        if(var ==2)
        {
            continue;
        }
        printf("%d\n",var);
    }
    printf("fin \n");
    return 0;
}
```

debut  
0  
1  
3  
4  
fin

continue

# Les boucles

## Break et Continue : exemple

```
int main()
{
    const int CODE_PIN = 0000;
    const int MAX_TENTATIVE = 3;
    int saisie_utilisateur;
    int nbre_Tentative= 0;
    int compteur;
    for( nbre_Tentative= 1 ;nbre_Tentative<=MAX_TENTATIVE ; nbre_Tentative++)
    {
        printf("Code Pin : ");
        scanf("%d",&saisie_utilisateur);
        if(saisie_utilisateur<0)
        {
            printf("code utilisateur doit etre superieur a 0 \n");
            continue;
        }
        if(saisie_utilisateur==CODE_PIN)
        {
            printf("\t Bienvenue\n");
            break;
        }
        else
            printf("\t Code incorrect  \n");
    }
    return 0;
}
```

# Les boucles

## Les boucles imbriquées

**Exercice : afficher toutes les tables de multiplication de 1 à 10**

**Aide : commencer par afficher une seule table de multiplication**

<b>Table de 1</b> 1 x 1 = 1 1 x 2 = 2 1 x 3 = 3 1 x 4 = 4 1 x 5 = 5 1 x 6 = 6 1 x 7 = 7 1 x 8 = 8 1 x 9 = 9 1 x 10 = 10	<b>Table de 2</b> 2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 2 x 4 = 8 2 x 5 = 10 2 x 6 = 12 2 x 7 = 14 2 x 8 = 16 2 x 9 = 18 2 x 10 = 20	<b>Table de 3</b> 3 x 1 = 3 3 x 2 = 6 3 x 3 = 9 3 x 4 = 12 3 x 5 = 15 3 x 6 = 18 3 x 7 = 21 3 x 8 = 24 3 x 9 = 27 3 x 10 = 30	<b>Table de 4</b> 4 x 1 = 4 4 x 2 = 8 4 x 3 = 12 4 x 4 = 16 4 x 5 = 20 4 x 6 = 24 4 x 7 = 28 4 x 8 = 32 4 x 9 = 36 4 x 10 = 40	<b>Table de 5</b> 5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50
<b>Table de 6</b> 6 x 1 = 6 6 x 2 = 12 6 x 3 = 18 6 x 4 = 24 6 x 5 = 30 6 x 6 = 36 6 x 7 = 42 6 x 8 = 48 6 x 9 = 54 6 x 10 = 60	<b>Table de 7</b> 7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 7 x 4 = 28 7 x 5 = 35 7 x 6 = 42 7 x 7 = 49 7 x 8 = 56 7 x 9 = 63 7 x 10 = 70	<b>Table de 8</b> 8 x 1 = 8 8 x 2 = 16 8 x 3 = 24 8 x 4 = 32 8 x 5 = 40 8 x 6 = 48 8 x 7 = 56 8 x 8 = 64 8 x 9 = 72 8 x 10 = 80	<b>Table de 9</b> 9 x 1 = 9 9 x 2 = 18 9 x 3 = 27 9 x 4 = 36 9 x 5 = 45 9 x 6 = 54 9 x 7 = 63 9 x 8 = 72 9 x 9 = 81 9 x 10 = 90	<b>Table de 10</b> 10 x 1 = 10 10 x 2 = 20 10 x 3 = 30 10 x 4 = 40 10 x 5 = 50 10 x 6 = 60 10 x 7 = 70 10 x 8 = 80 10 x 9 = 90 10 x 10 = 100

# Les boucles

## Les boucles imbriquées

**Exercice : afficher toutes les tables de multiplication de 1 à 10**

**Aide : commencer par afficher une seule table de multiplication**

```
int main()
{
    int ligne = 0;
    int id_table = 1;

    for(id_table=1; id_table<=10; id_table++)
    {

        printf("table multiplication %d\n", id_table);
        printf("*****\n");
        for(ligne=1; ligne<=10; ligne++)
        {
            printf("\t%d x %d = %d\n", ligne, id_table, ligne*id_table);
        }
        printf("\n") ;
    }
    return 0;
}
```

# Les boucles

Jeux kahoot

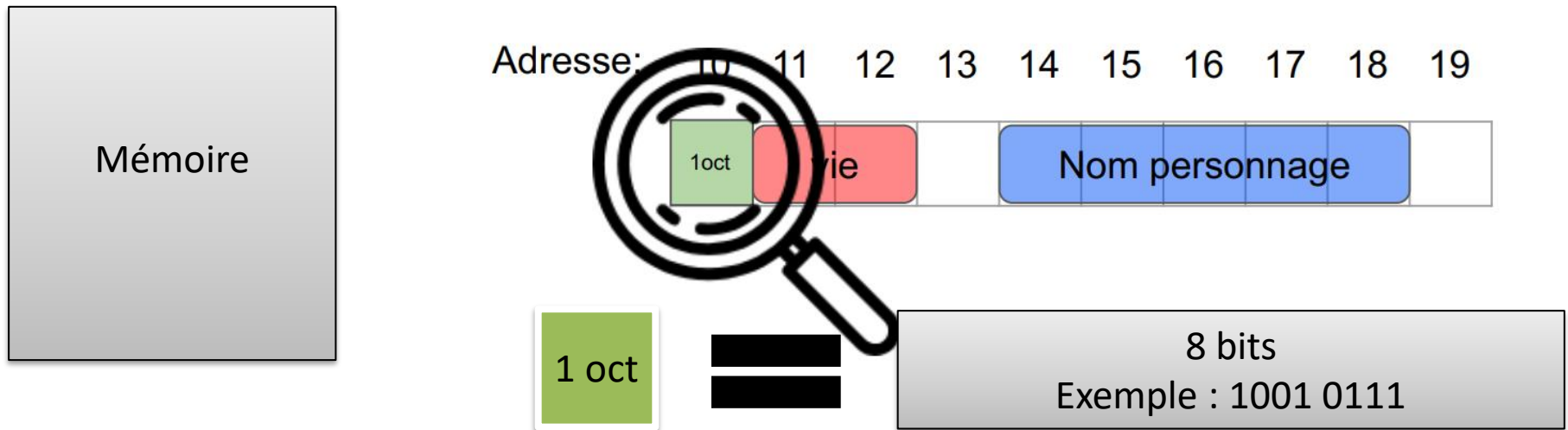
<https://create.kahoot.it/details/boucles/db371927-a176-4c1f-8811-2b3b73759cdb>

Kahoot!

# Les pointeurs

# Rappel :

## Gestion de la mémoire et variable



- Le programme est exécuté depuis la RAM
- La mémoire est composée de cases continues et chacune de ces cases est représentée par un octet
- Un octet c'est 8 bits
- Ces cases sont identifiées de manières uniques à l'aide de leurs adresses
- Une variable est une boîte qui occupe un nombre entier de ces cases et qui permet de stocker les données en binaire
- La taille et l'interprétation dépend de leur type de la variable
- On peut accéder au contenu de la variable à l'aide de son nom qui sert comme label pour accéder à la variable



# Notion de pointeur

- Un *pointeur* est une variable contenant l'adresse d'une autre variable d'un type donné.
- Il permet donc d'accéder **indirectement** à une variable.



Adresse: 10 11 12 13 14 15 16 17 18 19



# Adresses et variables

Adresse: 10 11 12 13 14 15 16 17 18 19



```
int age_utilisateur = 25;
```

```
printf("Age utilisateur: %d\n", age_utilisateur);  
printf("Adresse: %d\n", &age_utilisateur);  
printf("Adresse: %p\n", &age_utilisateur);
```

Age utilisateur: 25  
Adresse: 10  
Adresse: 0x0A

# Adresses et variables

Exemple sur codeblocks :

Afficher le contenu et l'adresse d'une variable en décimale et en hexadécimale

```
#include <stdio.h>
#include <stdlib.h>

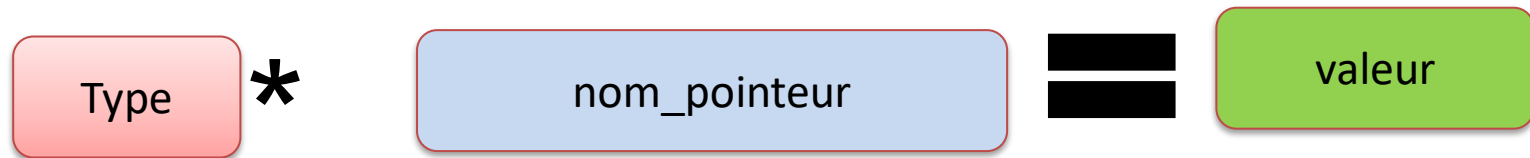
int main()
{
    int mon_int=10;
    printf("\nmon int = %d",mon_int );
    printf("\nAdresse de mon_int = %d",&mon_int);
    printf("\nAdresse de mon_int = %p",&mon_int);
    return 0;
}
```

```
mon int = 10
Adresse de mon_int = 1703736
Adresse de mon_int = 0019FF38
```

# Représentation hexadécimale

Représentation	Alphabet	Exemple
Décimale (10)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	12
Binaire (2)	0, 1	1100
Hexadécimale (16)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	C

# Déclaration d'un pointeur



## Exemple :

Char\* p\_char = 0;

Int\* p\_int = NULL;



# Déclaration d'un pointeur

- Sur codeblocks
- Reprendre l'exemple précédant et déclarer des pointeurs sur les variables déjà déclarées

# Utilisation des pointeurs

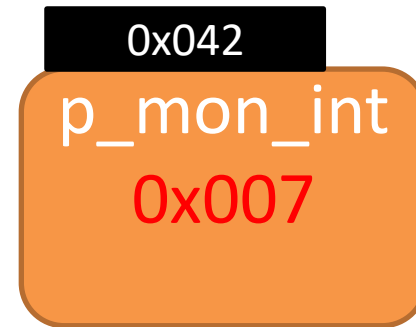


**`mon_int` (contenue de `mon_int`)**

-> 12

**`&mon_int` (adresse de `mon_int`)**

-> 0x007



**`&p_mon_int` (adresse de `p_mon_int`)**

-> 0x042

**`p_mon_int` (contenue de `p_mon_int`)**

-> 0x007

**`*p_mon_int` (contenue de 0x007)**

-> 12

# Dangers des pointeurs

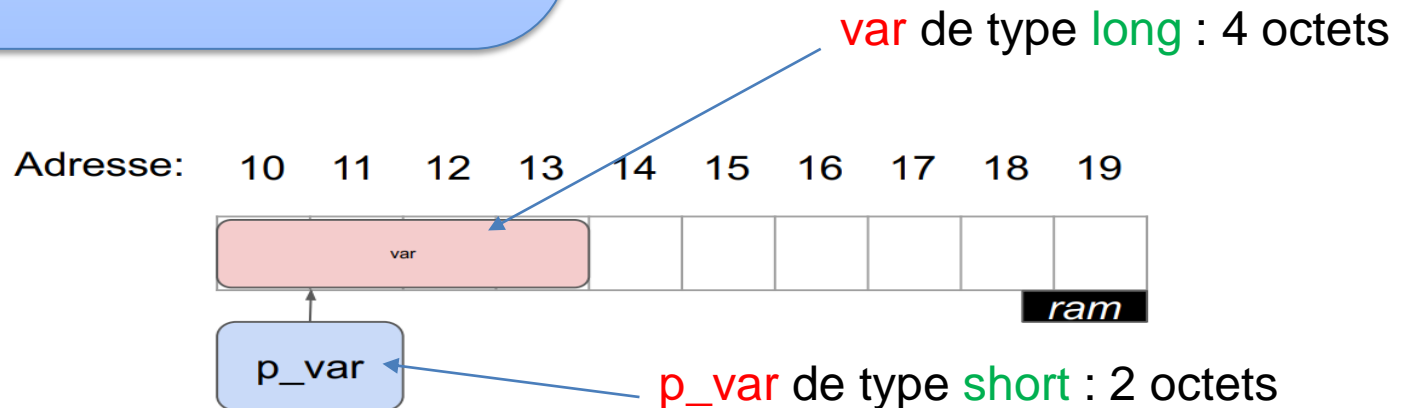
## 1. Problème de typage :

Les pointeurs doivent correspondre aux types que l'on souhaite pointer

```
int main()
{
    long var = 65321;
    short* p_var = &var;

    printf("ma variable = %ld \n", var);
    printf("ma variable = %ld \n", *p_var);
}
```

ma variable = 65321  
ma variable = -215





# Dangers des pointeurs

## 2. Pointeur NULL:

On essaie d'afficher le contenu des variables pointées par NULL

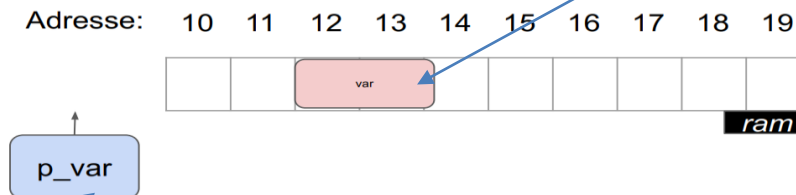
```
int main()
{
    short var = 123;
    short* p_var = NULL; // short* p_var = 0;

    printf("ma variable = %ld \n", var);
    printf("ma variable = %ld \n", *p_var);
}
```

ma variable = 123

Crash !

var de type short : 2 octets



p\_var de type short : ne point null part

# Dangers des pointeurs

## 2. Pointeur NULL:

On essaie d'afficher le contenu des variables pointées par NULL

Pour que le programme ne crashe pas :

```
int main()
{
    short var =123;
    short* p_var =NULL; // short* p_var = 0;
    printf("ma variable = %ld \n",var);
    if(p_var ==NULL)
    {
        printf("\n pointeur NULL !!!\n ");
    }
    else
    {
        printf("ma variable = %ld \n",*p_var);
    }
    return 0;
}
```

ma variable = 123

pointeur NULL !!!

# Dangers des pointeurs

## 2. Pointeur NULL:

On essaie d'afficher le contenu des variables pointées par NULL

Pour que le programme ne crashe pas :

```
int main()
{
    short var =123;
    short* p_var =&var;
    printf("ma variable = %ld \n",var);
    if(p_var ==NULL)
    {
        printf("\n pointeur NULL !!!\n ");
    }
    else
    {
        printf("ma variable = %ld \n",*p_var);
    }
    return 0;
}
```

ma variable = 123  
ma variable = 123

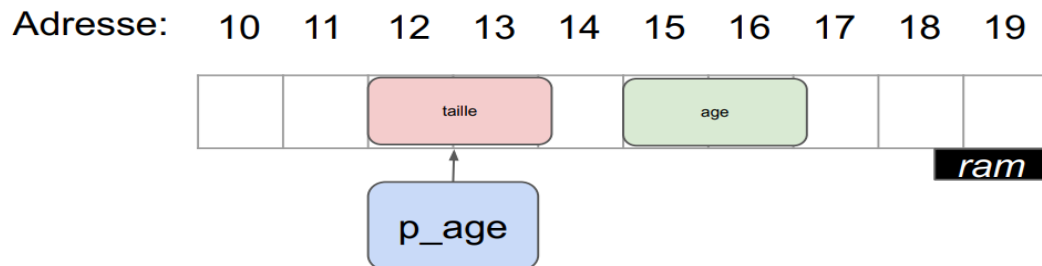
# Dangers des pointeurs

## 3. Jardinage :

On essaie de pointer sur une autre variable valide , problème difficile à trouver !!!

```
int main()
{
    int age=25;
    int taille =165;
    int* p_age=0;
    p_age=&taille;
    printf("\n donner votre age : ");
    scanf("%d",p_age);
    printf("votre age est : %d ans ",age);
    return 0;
}
```

donner votre age : 30  
votre age est : 25 ans



# Tester nos connaissances

- Jeux kahoot
- <https://create.kahoot.it/details/pointeur/823599f7-6485-4658-825d-523c3bec364c>

# EXERCICE

1. Créer une variable de type char
2. Afficher le contenu de la variable, sa taille en mémoire et son adresse

```
Informations sur ma variable:  
type: char  
taille: 1 octet  
contenu: A  
adresse: 0060FF0F
```

Aide : pour afficher une adresse il faut utiliser

**%p**

# EXERCICE

- `int main()`
- `{`
- `char lettre = 'A';`
- `printf("\n Informations sur ma vaiabale : \n ");`
- `printf("\t type : char \n ");`
- `printf("\t taille : %d \n ", sizeof(lettre));`
- `printf("\t contenu : %c \n ",lettre);`
- `printf("\t adresse : %p \n",&lettre );`
- `return 0;`
- `}`

```
Informations sur ma vaiabale :
    type       : char
    taille     : 1
    contenu    : A
    adresse    : 0019FF3B

Process returned 0 (0x0)   execution time : 0.206 s
Press any key to continue.
```

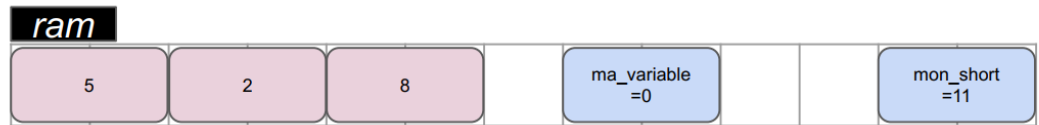
# Les Tableaux



# Les tableaux

tableau de 3 cases:

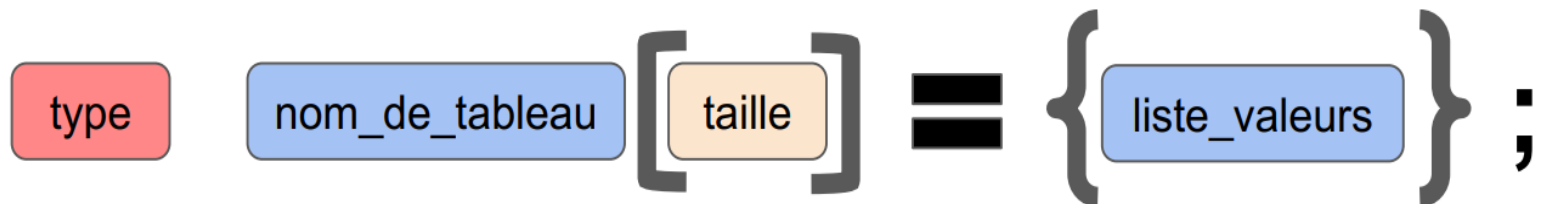
5	2	8
---	---	---



**Toutes les cases d'un tableau sont contiguës en mémoire**

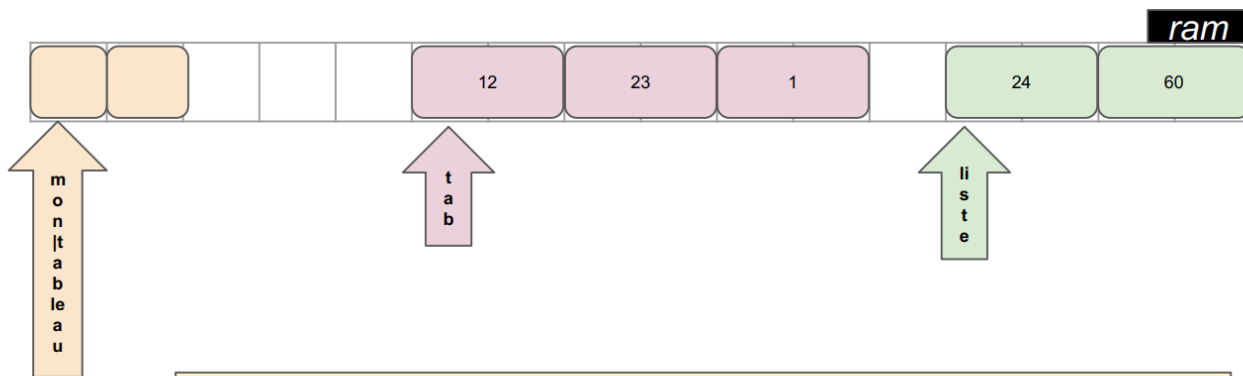
**Toutes les cases d'un tableau sont du même type**

# Déclaration d'un tableau



## Exemple:

```
char mon_tableau[2];  
short tab[3] = {12, 23, 1};  
short liste[] = {24, 60}
```



Taille du tableau fixé à la déclaration et ne peut pas varier

# Déclaration d'un tableau

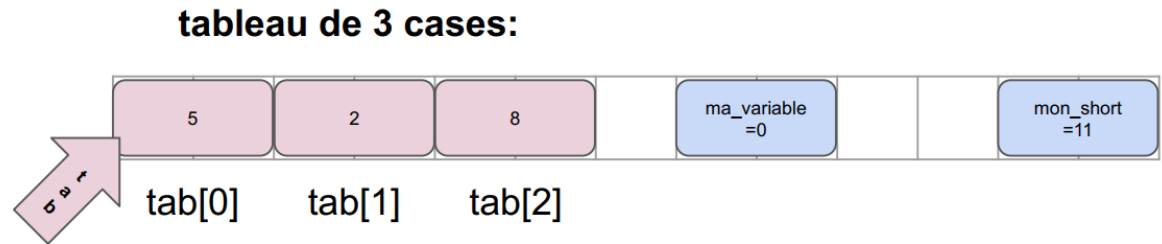
Exemple sur codeblocks

```
int main()
{
int tirage_loto_boules[5] = {1.2.3.4.5};
int tirage_loto_etoiles[] = {6.7};
char initiales_gagnant[2];
return 0;

}
```

# Parcourir un tableau

ID:	0	1	2
	5	2	8



**1ere case d'une tableau a pour ID 0**

**Toutes les cases d'un tableau sont du même type**

**Toutes les cases d'un tableau sont contiguës en mémoire**

# Parcourir un tableau

Exemple sur codeblocks

Initialisation du  
tableau sans boucle

```
# define Taille_Tableau 3
int main()
{
//declarer le tableau
    int i;
    int mon_tableau[Taille_Tableau];
// init du tableau
    mon_tableau [0]= 0;
    mon_tableau [1]= 1;
    mon_tableau [2]= 2;

// affichage du tableau
    for(i=0;i<Taille_Tableau;i++)
    {
        printf("mon_tableau[%d] =%d \n",i, mon_tableau[i] );
    }
    return 0;
}
```

Initialisation du  
tableau avec la boucle

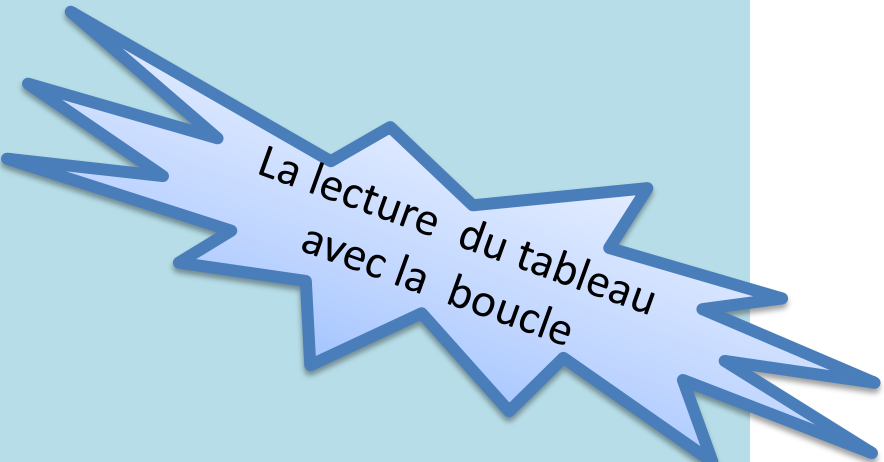
```
#include <stdlib.h>

# define Taille_Tableau 3
int main()
{
//declarer le tableau
    int i;
    int mon_tableau[Taille_Tableau];
// init du tableau
    for(i=0;i<Taille_Tableau;i++)
    {
        mon_tableau[i]=i;
    }
// affichage du tableau
    for(i=0;i<Taille_Tableau;i++)
    {
        printf("mon_tableau[%d] =%d \n",i, mon_tableau[i] );
    }
    return 0;
}
```

# Parcourir un tableau

Exemple sur codeblocks

```
# define Taille_Tableau 3
int main()
{
//declarer le tableau
    int i;
    int mon_tableau[Taille_Tableau];
// lecture du tableau : la saisie
    for(i=0;i<Taille_Tableau;i++)
    {
        printf("> ");
        scanf("%d",&mon_tableau[i]);
    }
// affichage du tableau
    for(i=0;i<Taille_Tableau;i++)
    {
        printf("mon_tableau[%d] =%d \n",i, mon_tableau[i] );
    }
    return 0;
}
```

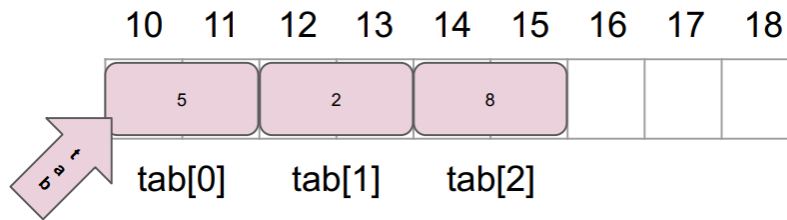


La lecture du tableau  
avec la boucle

```
> 12
> 56
> 25
mon_tableau[0] =12
mon_tableau[1] =56
mon_tableau[2] =25

Process returned 0 (0x0)   execution time : 8.344 s
Press any key to continue.
```

# Tableaux et pointeurs



`tab` (adresse du tableau)

`*tab` (contenue de la première case du tableau)

`*(tab+n-1)` (contenue de la la case `n` du tableau)

## Exemple:

`tab[0] -> 5     // *(tab+0)->5`

`tab[1] -> 2     // *(tab+1)->2`

`tab[2] -> 8     // *(tab+2)->8`

**`*tab + 1 => 5 + 1 => 6`**

# Tableaux et pointeurs

Exemple sur codeblocks

```
# define Taille_Tableau 3
int main()
{
//declarer le tableau
    int i;
    int mon_tableau[Taille_Tableau];
// lecture du tableau : la saisie
    for(i=0;i<Taille_Tableau;i++)
    {
        printf("> ");
        scanf("%d", (mon_tableau+i)); }
// affichage du tableau
    for(i=0;i<Taille_Tableau;i++)
    {
        printf("mon_tableau[%d] =%d \n", i, *(mon_tableau+i));
    }
    return 0;
}
```

```
> 25
> 14
> 19
mon_tableau[0] =25
mon_tableau[1] =14
mon_tableau[2] =19
```



# Tableaux et adresses : Exercice

- 1- Créez un tableau de 5 caractères et initialisez le
- 2- Pour chaque cases, afficher l'index, le contenu et l'adresse
- 3- Faire la même chose mais en utilisant la

Aide :

- 1- `type nom[taille] = {liste_valeurs};`
- 2- Il faut utiliser `%p` pour afficher une adresse mémoire dans un `printf`.
- 3- `tab[i] = *(tab+i)`

```
tab[0] = C (0060FEF9)
tab[1] = O (0060FEFA)
tab[2] = D (0060FEFB)
tab[3] = E (0060FEFC)
tab[4] = R (0060FEFD)
```

# Tableaux et adresses : Exercice

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char tab[] = {'C','O','D','E','R'};
    int i;
    // annotation simple

    for(i=0;i<5;i++)
    {
        printf("tab[%d] = %c (%p)\n", i, tab[i], &tab[i]);
    }
    return 0;
}
```

```
tab[0] = C (0019FF34)
tab[1] = O (0019FF35)
tab[2] = D (0019FF36)
tab[3] = E (0019FF37)
tab[4] = R (0019FF38)
```

```
printf("tab[%d] = %c (%p)\n", i, *(tab+i), tab+i);
```

# Tableaux et adresses : Exercice

## Annotation simple

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char tab[] = {'C','O','D','E','R'};
    int i;
    // annotation simple

    for(i=0;i<5;i++)
    {
        printf("tab[%d] = %c (%p)\n", i, tab[i], &tab[i]);
    }
    return 0;
}
```

```
tab[0] = C (0019FF34)
tab[1] = O (0019FF35)
tab[2] = D (0019FF36)
tab[3] = E (0019FF37)
tab[4] = R (0019FF38)
```

# Tableaux et adresses : Exercice

## Annotation pointeur

```
#include <stdio.h>
#include <stdlib.h>

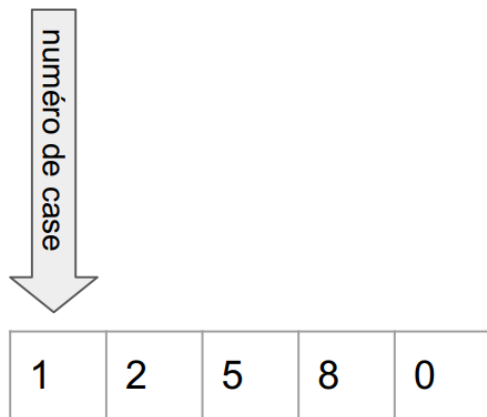
int main()
{
    char tab[] = {'C','O','D','E','R'};
    int i;

    // Annotation Pointeur
    for(i=0;i<5;i++)
    {
        printf("(tab + %d) = %c (%p)\n", i,*(tab+i),(tab+i));
    }

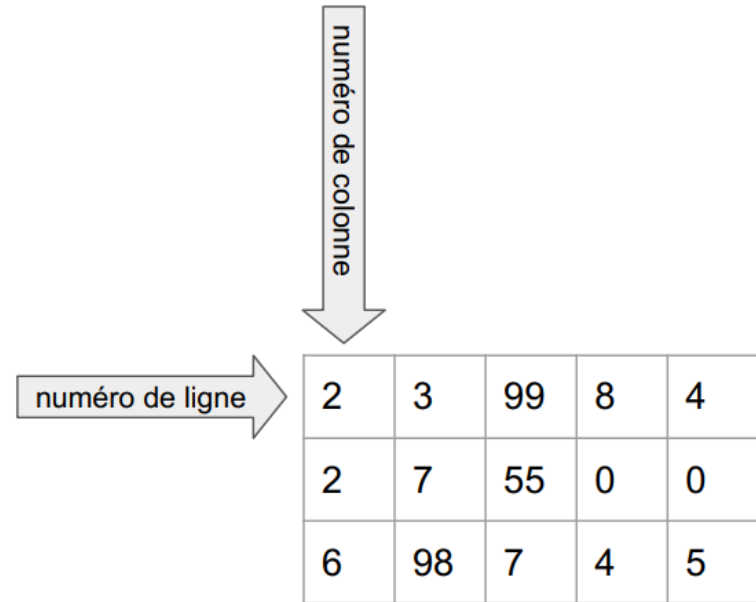
    return 0;
}
```

```
*(tab + 0) = C (0019FF34)
*(tab + 1) = O (0019FF35)
*(tab + 2) = D (0019FF36)
*(tab + 3) = E (0019FF37)
*(tab + 4) = R (0019FF38)
```

# Les tableaux multidimensionnels

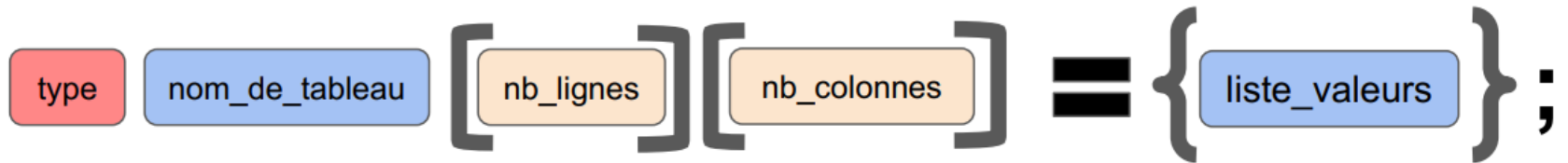


1D



2D

# Déclaration tableaux 2D

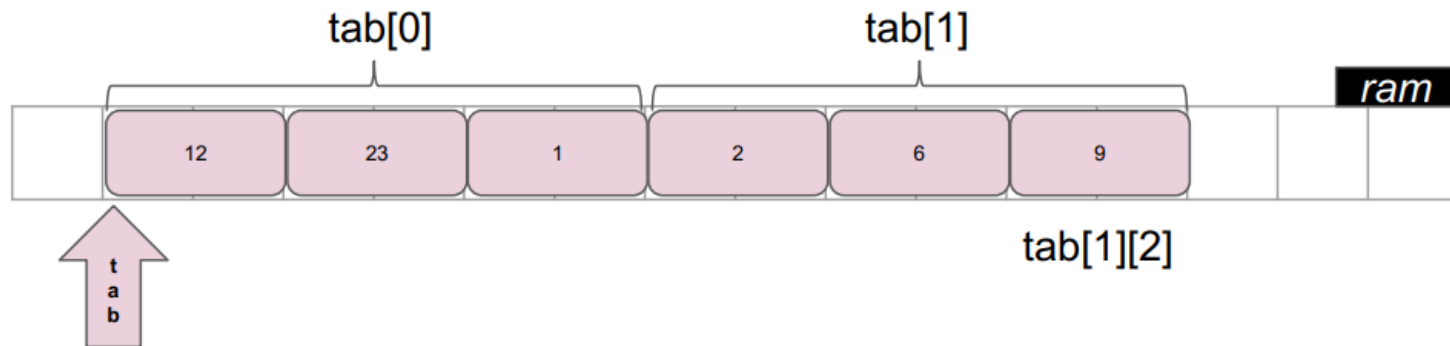


## Exemple:

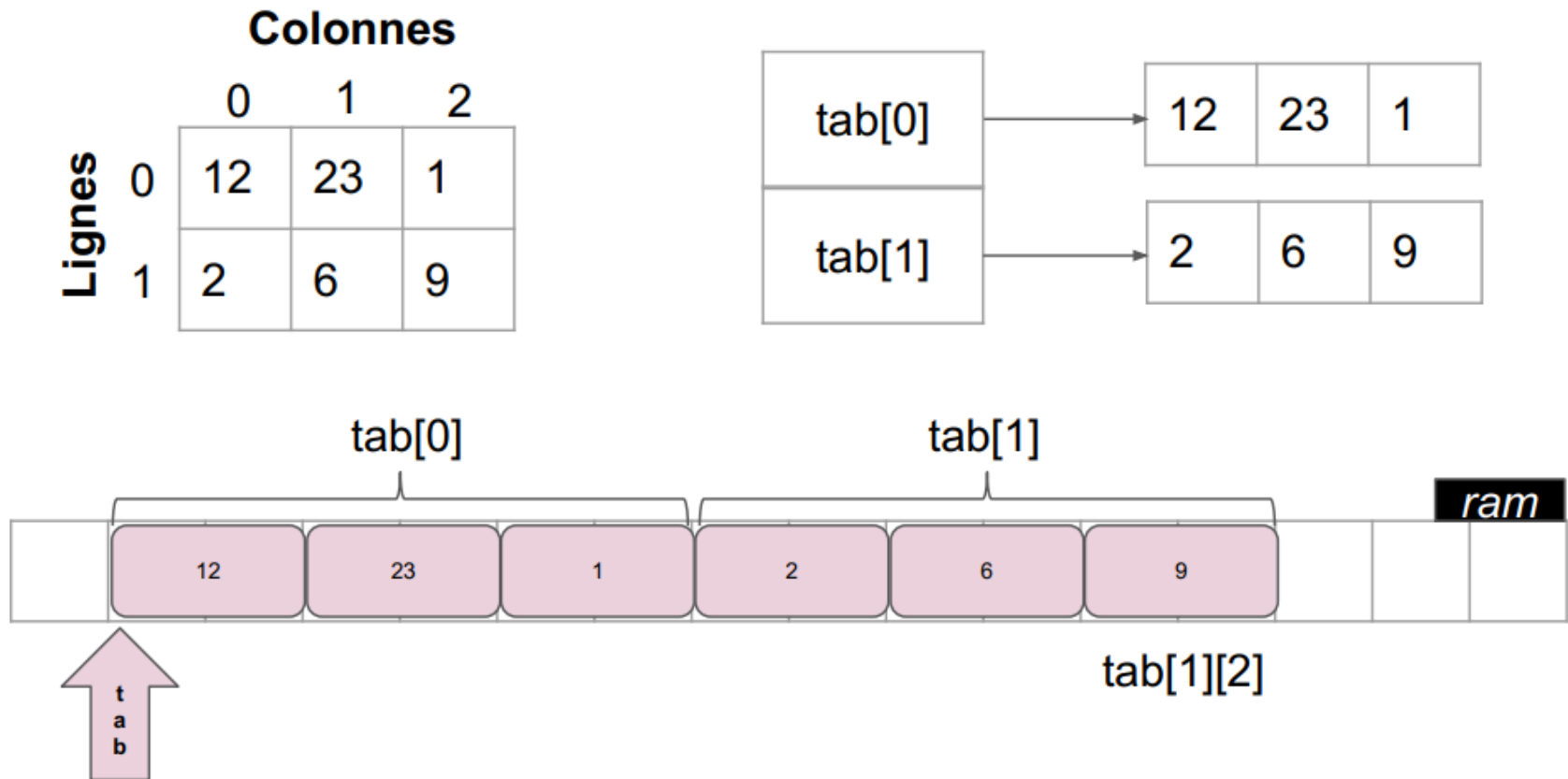
```
short tab[2][3] = { {12, 23, 1}, {2, 6, 9} };
```

```
tab[1][2] //retourne la valeur 9
```

		Colonnes		
		0	1	2
Lignes	0	12	23	1
	1	2	6	9



# Tableaux 2D = tableau de tableaux



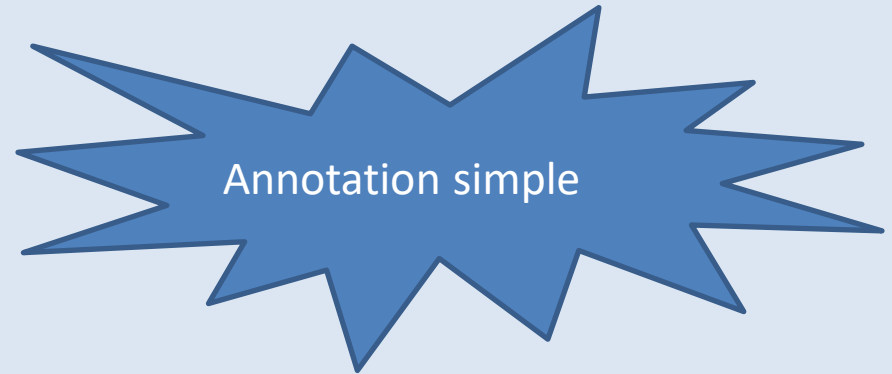
# Les tableaux multidimensionnels

## Exercices

Exemple sur codeblocks : créer un tableau à deux dimensions

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int ligne;
    int colonne;
    int tab[2][3]={{1,2,3},{4,5,6}};
    //Annotation simple
    for(ligne=0;ligne<2;ligne++)
    {
        for (colonne=0;colonne<3;colonne++)
        {
            printf("tab[%d][%d] = %d\n",ligne,colonne,tab[ligne][colonne]);
        }
    }
    return 0;
}
```





# Les tableaux multidimensionnels

## Exercices

Exemple sur codeblocks : créer un tableau à deux dimensions

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main()
```

```
{
```

```
    int ligne;
```

```
    int colonne;
```

```
    int* p_ligne;
```

```
    int tab[2][3]={1,2,3},{4,5,6}};
```

```
    for(ligne=0;ligne<2;ligne++)
```

```
    {
```

```
        p_ligne = *(tab+ligne);
```

```
        for (colonne=0;colonne<3;colonne++)
```

```
        {
```

```
            printf("tab[%d][%d] = %d\n",ligne,colonne,*(p_ligne +colonne));
```

```
//printf("tab[%d][%d] = %d\n",ligne,colonne,*(*(tab+ligne)+colonne))
```

```
        }
```

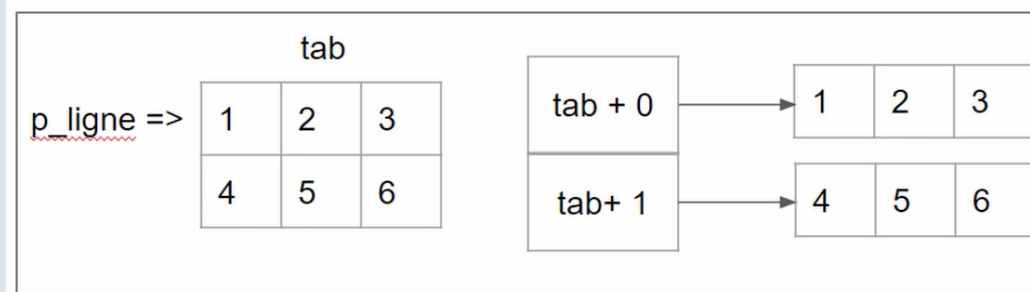
```
    }
```

```
    return 0;
```

```
}
```

Annotation pointeur

```
tab[0][0] = 1
tab[0][1] = 2
tab[0][2] = 3
tab[1][0] = 4
tab[1][1] = 5
tab[1][2] = 6
```



`*(*(tab+ligne)) = *(p_ligne)`

# Les Chaines de caractères

# les chaines de caractères

## Que signifie une chaine de caractères ?

Une suite logique de caractères et qui vont constituer des mots ou des phrases

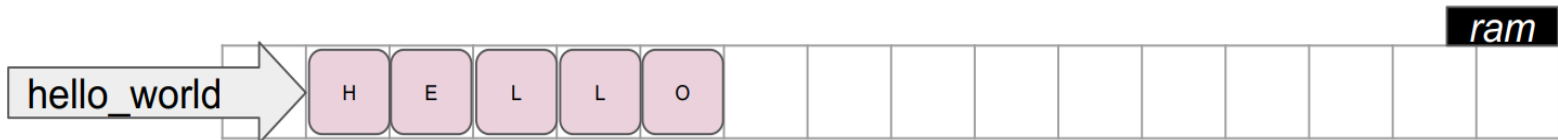
NB : En général en programmation il y a le type string qui permet de stocker des chaines de caractères mais en langage C , ce type de variable n'est pas disponible ! Mais on peut manipuler ou utiliser le string en C

# Les tableaux de char

**Exemple:**

```
char hello_world[5] = { 'H', 'E', 'L', 'L', 'O' };
```

H	E	L	L	O
---	---	---	---	---



# Les tableaux de char

Illustration sur CodeBlocks

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i;
    char salut[]={'S','A','L','U','T'};
    for(i=0;i<5;i++)
    {
        printf("%c",salut[i]);
    }

    return 0;
}
```

```
SALUT
Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.
```

# Les strings

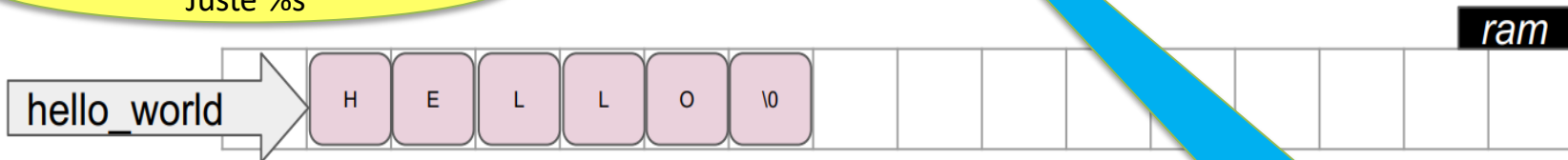
## Exemple:

```
char hello_world[6] = { 'H', 'E', 'L', 'L', 'O', '\0' };  
printf("%s", hello_world) -> HELLO
```

Pas besoin de boucle !!!  
Juste %s

Un marqueur  
Un caractère spécial

H	E	L	L	O	\0
---	---	---	---	---	----



La fin de la chaîne  
Pas la fin du tableau !

# Les strings

Illustration sur CodeBlocks

```
int main()
{
    int i;
    char salut[]={ 'S','A','L','U','T','\0' };

    printf("%s",salut);

    return 0;
}
```

```
int main()
{
    int i;
    char salut[]={ 'S','A','L','\0','U','T','\0' };

    printf("%s",salut);

    return 0;
}
```

Si on met \0 ici

```
SALUT
Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.
```

```
SAL
Process returned 0 (0x0)   execution time : 0.078 s
Press any key to continue.
```

# Les strings

Illustration sur CodeBlocks

```
int main()
{
    int i;
    char salut[] = "SALUT";
    printf("%s",salut);

    return 0;
}
```

Initialisation avec une chaîne de caractère directement

```
SALUT
Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.
```

```
int main()
{
    int i;
    char salut[]="SALUT";
    char prenom[100];
    scanf("%s", prenom);
    printf("%s, %s",salut, prenom);
    return 0;
}
```

```
Sanaa
SALUT, Sanaa
Process returned 0 (0x0)   execution time : 7.637 s
Press any key to continue.
```

Si vous mettez un prénom composé, il prend en considération que le premier mot!! (l'espace est une validation comme entrée



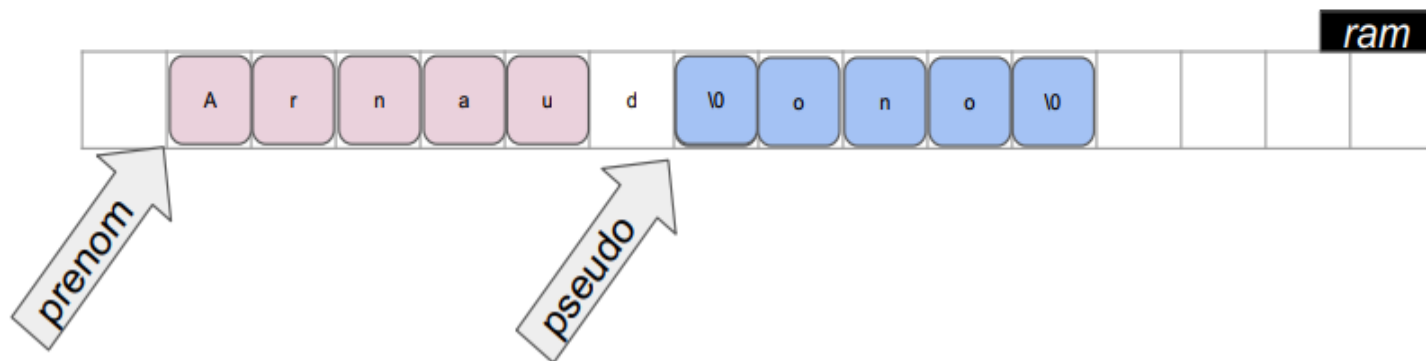
# Dépassement de mémoire

## Exemple:

```
char pseudo[5];  
pseudo = "Nono";
```

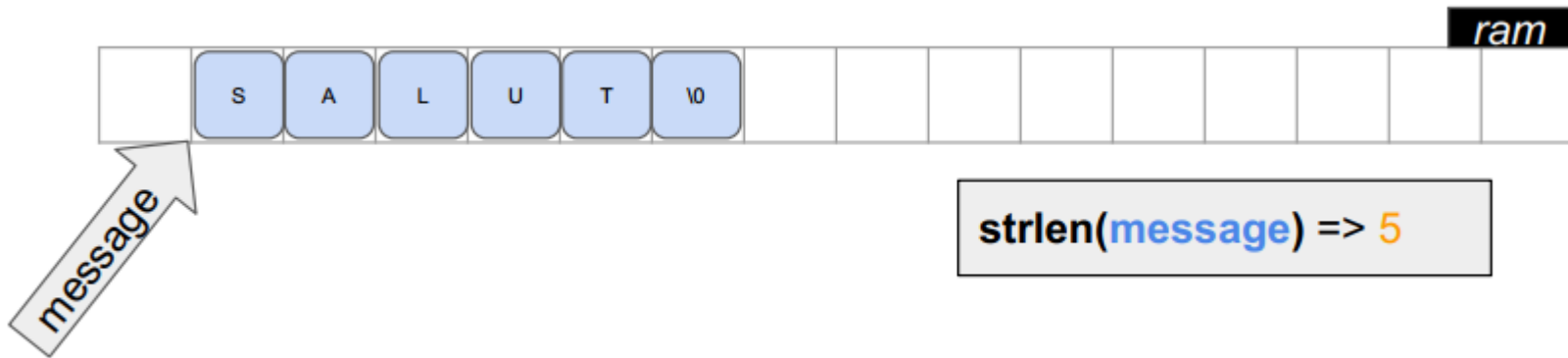
```
char prenom[5];  
prenom = "Arnaud";
```

```
printf("%s", prenom); => Arnaud  
printf("%s", pseudo); =>
```



# Strlen(string.h)

taille strlen( chaîne )



[C documentation — DevDocs](https://devdocs.io/c/) : <https://devdocs.io/c/>

**Retourne la taille de la string passée en paramètre en excluant le ' '**

# Strlen(string.h)

Illustration sur CodeBlocks

```
include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

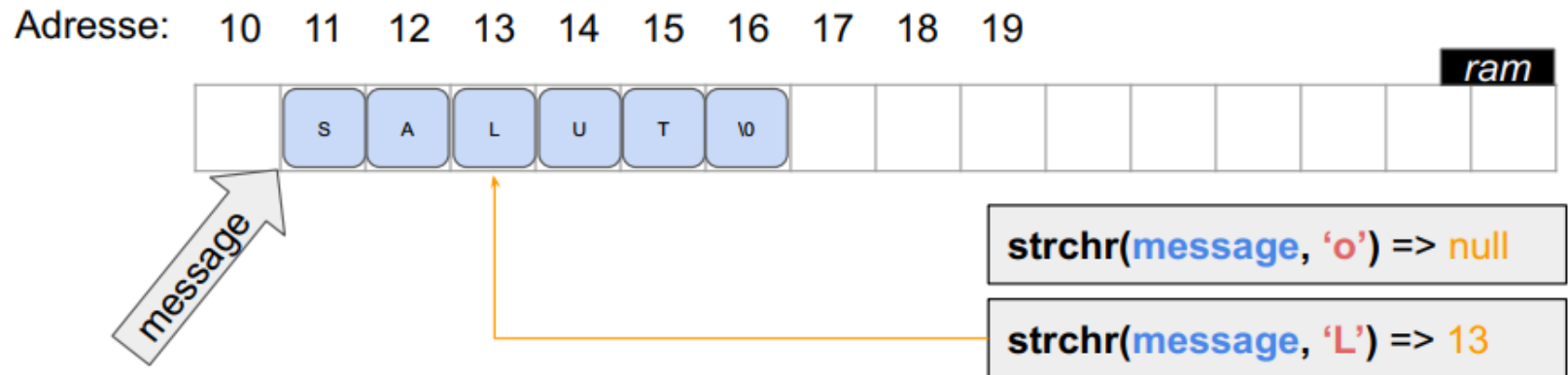
```
int main()
{
    char message[] = "hello world";
    printf("taille de message \" %s \" = %d \n",message, strlen(message));

    return 0;
}
```

taille de message " hello world " = 11

# Strchr(string.h)

pointeur strchr( chaîne , char )



Retourne la première sous chaîne qui commence par le caractère passé en argument.

Si le caractère n'est pas trouvé, elle retourne NULL.

# Strchr(string.h)

Illustration sur CodeBlocks

```
int main()
{
    char message[] = "hello world";
    char char_a_trouver = 'v';
    char* p_char = strchr(message ,char_a_trouver);

    printf("taille de message \" %s \" = %d \n",message, strlen(message));

    if (p_char ==NULL)
    {
        printf("la char \'%c\' n'est pas present dans la chaine \" %s \"",char_a_trouver,message);
    }
    else
    {
        printf("la char \'%c\'est present dans la chaine \" %s \" ",char_a_trouver,message)
    }
    return 0;
}
```

taille de message " hello world " = 11  
la char 'v' n'est pas present dans la chaine " hello world "

# Strchr(string.h)

Illustration sur CodeBlocks

```
int main()
{
    char message[] = "hello world";
    char char_a_trouver = 'e';
    char* p_char = strchr(message ,char_a_trouver);

    printf("taille de message \" %s \" = %d \n",message, strlen(message));

    if (p_char ==NULL)
    {
        printf("la char \'%c\' n'est pas present dans la chaine \" %s \"",char_a_trouver,message);
    }
    else
    {
        printf("la char \'%c\'est present dans la chaine \" %s \" ",char_a_trouver,message);
    }
    return 0;
}
```

taille de message " hello world " = 11  
la char 'e'est present dans la chaine " hello world "

# Strchr(string.h)

Illustration sur CodeBlocks

```
int main()
{
    char message[] = "hello world";
    char char_a_trouver = 'e';
    char* p_char = strchr(message + 3, char_a_trouver);

    printf("taille de message \" %s \" = %d \n", message, strlen(message));

    if (p_char == NULL)
    {
        printf("la char \" %c \" n'est pas present dans la chaine \" %s \"", char_a_trouver, message);
    }
    else
    {
        printf("la char \" %c \" est present dans la chaine \" %s \" ", char_a_trouver, message);
    }
    return 0;
}
```

taille de message " hello world " = 11  
la char 'e' n'est pas present dans la chaine " hello world "

La recherche commence à partir du troisième caractère pour faire la recherche dans un sous ensemble de chaine

# Strchr(string.h)

Illustration sur CodeBlocks

```
int main()
{
    char message[] = "hello world";
    char char_a_trouver = 'o';
    char* p_char = strchr(message + 3, char_a_trouver);

    printf("taille de message \" %s \" = %d \n", message, strlen(message));

    if (p_char == NULL)
    {
        printf("la char \'%c\' n'est pas present dans la chaine \" %s \" ", char_a_trouver, message);
    }
    else
    {
        printf("la char \'%c\' est present dans la chaine \" %s \" la fin du message apres le char est %s", char_a_trouver, message, p_char);
    }
    return 0;
}
```

taille de message " hello world " = 11  
la char 'o' est present dans la chaine " hello world " la fin du message apres le char est o world

Il retourne le premier  
caractère trouvé si il y a des  
doublons



# Strcmp(string.h)

`int strcmp( chaîne 1, chaîne 2 )`

`strcmp("Tata", "Toto") => -1`

`strcmp("Toto", "Toto") => 0`

`strcmp("Toto", "Tata") => 1`

**Compare deux chaînes, retourne 0 si elles sont identiques, sinon elle retourne -1 dans le cas où la première chaîne est avant dans l'ordre alphabétique et elle retourne 1 dans l'autre cas.**

# Strcmp(string.h)

Illustration sur CodeBlocks

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char message[] = "hello world";

    if(strcmp(message,"hello world")== 0)
    {
        printf("\nles deux chaines sont identiques\n");
    }
    else
    {
        printf("\nles deux chaines sont differentes \n");
    }
    return 0;
}
```

les deux chaines sont identiques

# Strcmp(string.h)

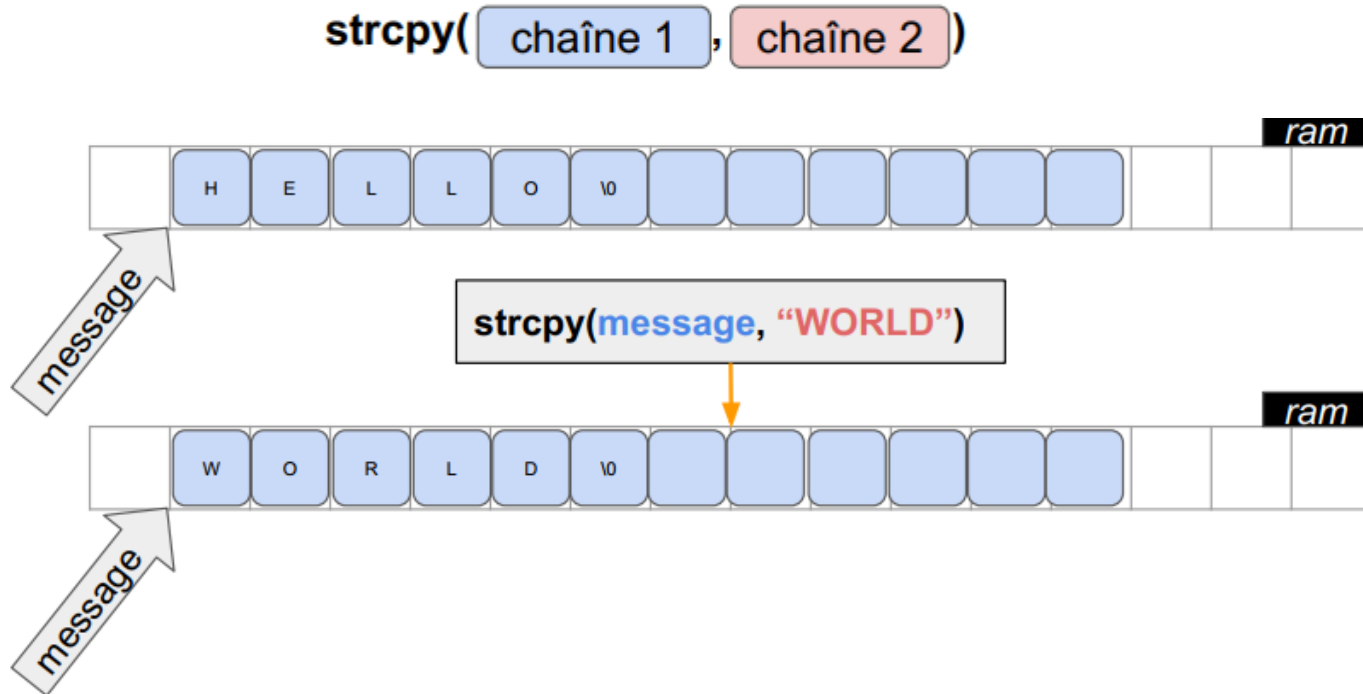
Illustration sur CodeBlocks

```
int main()
{
    char message[] = "hello world";

    if(strcmp(message,"hello SMI")== 0)
    {
        printf("\nles deux chaines sont identiques\n");
    }
    else
    {
        printf("\nles deux chaines sont differentes \n");
    }
    return 0;
}
```

les deux chaines sont differentes

# Strcpy(string.h)



**Copie le contenu de la chaîne 2 dans la chaîne 1.  
Attention à ce que le tableau de chaîne 1 soit suffisamment grand pour  
contenir chaîne 2.**

# Strcpy(string.h)

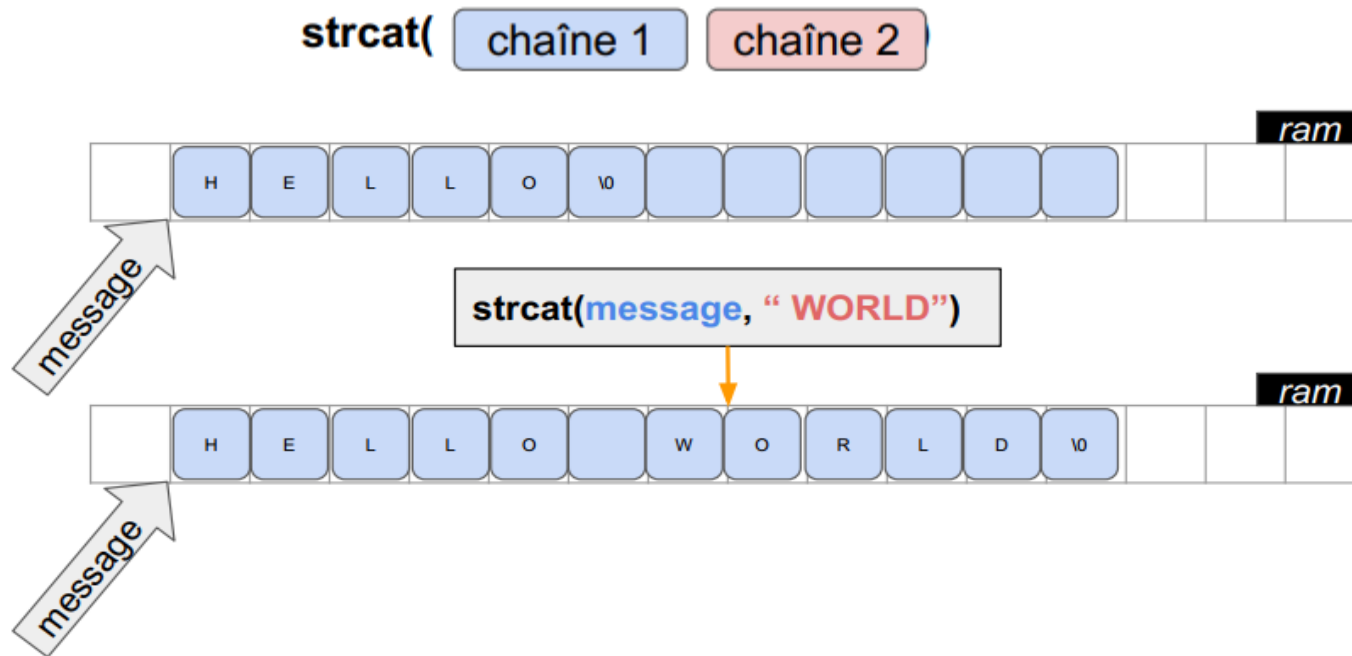
Illustration sur CodeBlocks

```
int main()
{
    char message[100] = "hello";

    printf("message = %s\n",message);
    strcpy(message,"world");
    printf("message = %s \n",message);
    return 0;
}
```

```
message = hello
message = world
```

# Strcat(string.h)



**Insère le contenu de la chaîne 2 devant la chaîne 1.  
Attention à ce que le tableau de chaîne 1 soit suffisamment grand pour  
contenir chaîne 1 et chaîne 2.**

# Strcat(string.h)

Illustration sur CodeBlocks

```
int main()
{
    char message[100] = "hello";

    strcat(message, " world");
    printf("message = %s \n", message );
    return 0;
}
```

message = hello world

# Strtol(string.h)

`long` `strtol`( `chaîne` , `pt_fin` , `base` )

`strtol("123456", NULL, 10) => 123456`

`strtol("12€", NULL, 10) => 12`

**Convertit une chaîne en valeur numérique entière qu'elle représente.**

**pt\_fin** : la fonction s'en sert pour renvoyer la position du premier caractère qu'elle a lu et qui n'était pas un nombre. On peut mettre **NULL** si on ne souhaite pas l'utiliser.

**base** : base avec laquelle le nombre est écrit dans la chaîne. Le plus souvent base 10



# Strtod(string.h)

`double strtod( chaîne , pt_fin )`

`strtod("12.3", NULL) => 12.3`

`strtod("12.2€", NULL) => 12.2`

**Convertit une chaîne en valeur numérique flottante qu'elle représente.**

**pt\_fin : la fonction s'en sert pour renvoyer la position du premier caractère qu'elle a lu et qui n'était pas un nombre. On peut mettre NULL si on ne souhaite pas l'utiliser.**

# Strtol(string.h) et Strtod(string.h)

Illustration sur CodeBlocks

```
int main()
{
    char message[100] = "hello";
```

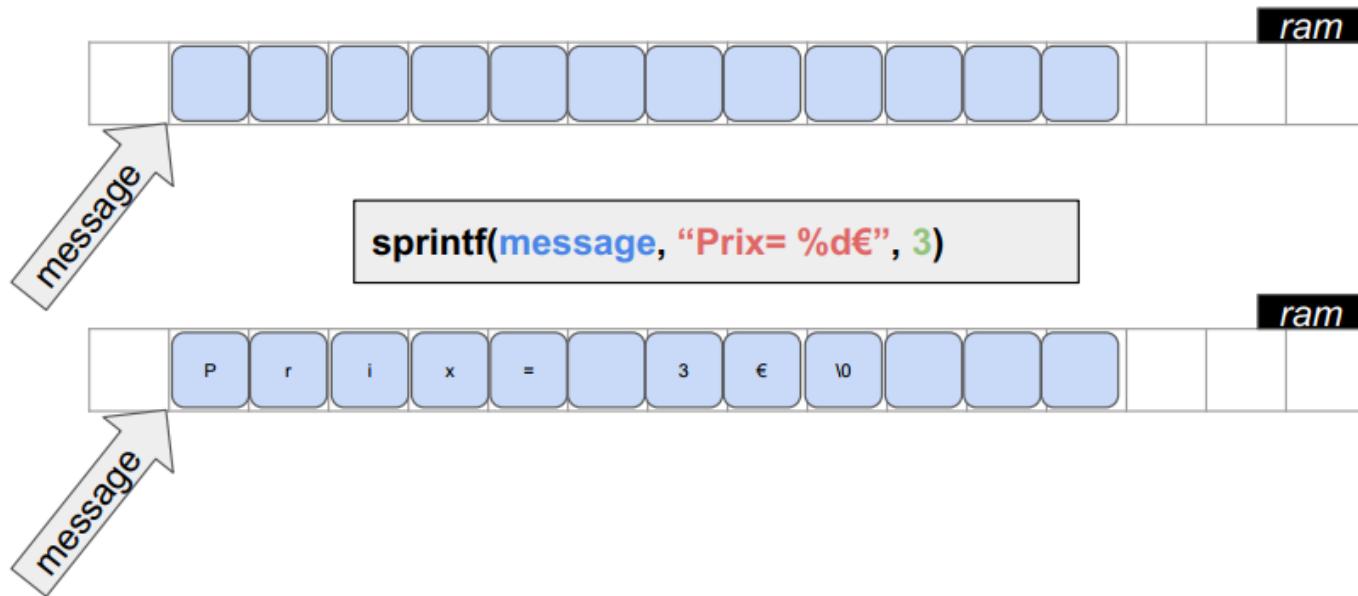
```
    printf("l entier de %s = %d \n ", "1234", strtol("1234", NULL, 10));
    printf("le float de %s = %f \n ", "123,4", strtod("123,4", NULL));
```

```
    return 0;
}
```

```
l entier de 1234 = 1234
le float de 123,4 = 123.000000
```

# Sprintf(stdio.h)

**sprintf(** chaîne destination , chaîne template , params )



Comme la fonction `printf`, `sprintf` permet de formater une chaîne de caractère mais au lieu de l'afficher dans la console, on l'écrit dans un tableau de char (string).

# Sprintf(stdio.h)

Illustration sur CodeBlocks

```
int main()
{
    char message[100] = "hello";
    char Nom[20];
    int Age;
    sprintf(message,"mon nom est %s et j ai %d ans ","SANAA",48);
    printf("%s",message);
    return 0;
}
```

mon nom est SANAA et j ai 48 ans

# Exercice : Carte d'identité

## Objectifs :

1- Créer un programme qui demande à l'utilisateur les informations suivantes et les affiche:

- Date de naissance: string au format jj/mm/aaaa
- Nom: string
- Prenom: string
- Couleur des yeux: string

donnez les informations suivantes :

Date de naissance jj/mm/aaaa :24/05/2002

Nom : KENZA

Prenom :TLEMCANI

Couleur des yeux :VERTS

vous vous appelez KENZA TLEMCANI , vous etes nee  
le 24/05/2002 et vos yeux sont VERTS

## Aide

Il faut utiliser %s dans un scanf pour lire une string. Attention, un espace dans la saisie est considéré comme un séparateur de paramètres dans scanf.

# Exercice : Carte d'identité

```
int main()
{
    char date[100];
    char nom[100];
    char prenom[100];
    char couleur[100];
    // lecture saisie utilisateur
    printf("donnez les informations suivantes : \n");
    printf("\t Date de naissance jj/mm/aaaa :");
    scanf("%s",date);
    printf("\t Nom :");
    scanf("%s",nom);
    printf("\t Prenom :");
    scanf("%s",prenom);
    printf("\t Couleur des yeux :");
    scanf("%s",couleur);
    // affichage des informations
    printf("vous vous appelez %s %s , vous etes nee le %s et vos yeux sont %s",nom,prenom,date,couleur);

    return 0;
}
```

donnez les informations suivantes :

Date de naissance jj/mm/aaaa :24/05/2002

Nom : KENZA

Prenom :TLEMCANI

Couleur des yeux :VERTS

vous vous appelez KENZA TLEMCANI , vous etes nee le 24/05/2002 et vos yeux sont VERTS

# lecture sécurisée avec fgets

retour fgets( chaîne , taille , stdin)

```
int TAILLE= 100;
char message[TAILLE];
printf("Ecrire un message de %d caractères maximum\n > ");

if( fgets(message, TAILLE, stdin) != NULL)
{
    printf("\n %s \n", message);
}
else
{
    printf("Une erreur est survenue \n");
}
```

*Exemple*

Ecrire un message de 100  
caractères maximum  
> Bonjour et bienvenue dans cette  
formation sur le C

Bonjour et bienvenue dans cette  
formation sur le C

**fgets retourne NULL si il y a une erreur lors de la lecture**

# lecture sécurisée avec fgets

Illustration sur CodeBlocks

```
int main()
{
    char nom[100];
    printf("comment tu t'appelles ? ");
    scanf("%s",nom);

    printf("tu t'appelles %s ", nom );
    return 0;
}
```

comment tu t'appelles ? sanaa  
tu t'appelles sanaa

comment tu t'appelles ? Sanaa EL FILALI  
tu t'appelles Sanaa



# lecture sécurisée avec fgets

Illustration sur CodeBlocks

```
int main()
{
    char nom[100];
    printf("comment tu t'appelles ? ");
    if(fgets(nom, 99, stdin) != NULL)
    {
        printf("tu ne sais rien %s ! ", nom);
    }
    return 0;
}
```

comment tu t'appelles ? Sanaa  
tu ne sais rien Sanaa  
!

```
int main()
{
    char nom[100];
    printf("comment tu t'appelles ? ");
    if(fgets(nom, 5, stdin) != NULL)
    {
        printf("tu ne sais rien %s ! ", nom);
    }
    return 0;
}
```

comment tu t'appelles ? SANAA  
tu ne sais rien SANA !

limiter la saisie

# lecture sécurisée avec fgets

Illustration sur CodeBlocks

```
int main()
{
    char prix[100];
    printf("donner un prix ? ");

    if(fgets(prix,99,stdin) != NULL)
    {
        printf("vous avez indique %f \n", strtod(prix,NULL));
    }
    return 0;
}
```

donner un prix ? 368.25  
vous avez indique 368.250000

# Jeux pour valider les connaissances

<https://create.kahoot.it/share/chaine-de-caractere/e8de39a6-04b4-4ef2-97d0-a39976b125d6>