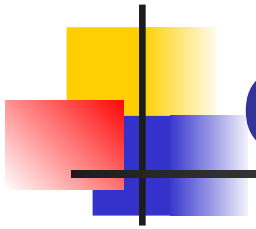
Informatique Industrielle (Informatique 1)

Enseignants: Fakhreddine Ghaffari (fakhreddine.ghaffari@u-cergy.fr)

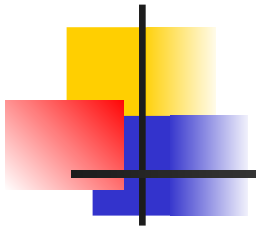
Stéphane Zuckerman (stephane.zuckerman@u-cergy.fr)

Année Universitaire 2019/2020



Objectifs et contenu du cours

- Objectifs
  - Introduction à l'informatique
  - Appréhender un langage de programmation haut niveau
- Cours/TD
  - 17 séances de 1,5h
- TP
  - 8 séances de 3h
  - utilisation de l'environnement Code::Blocks

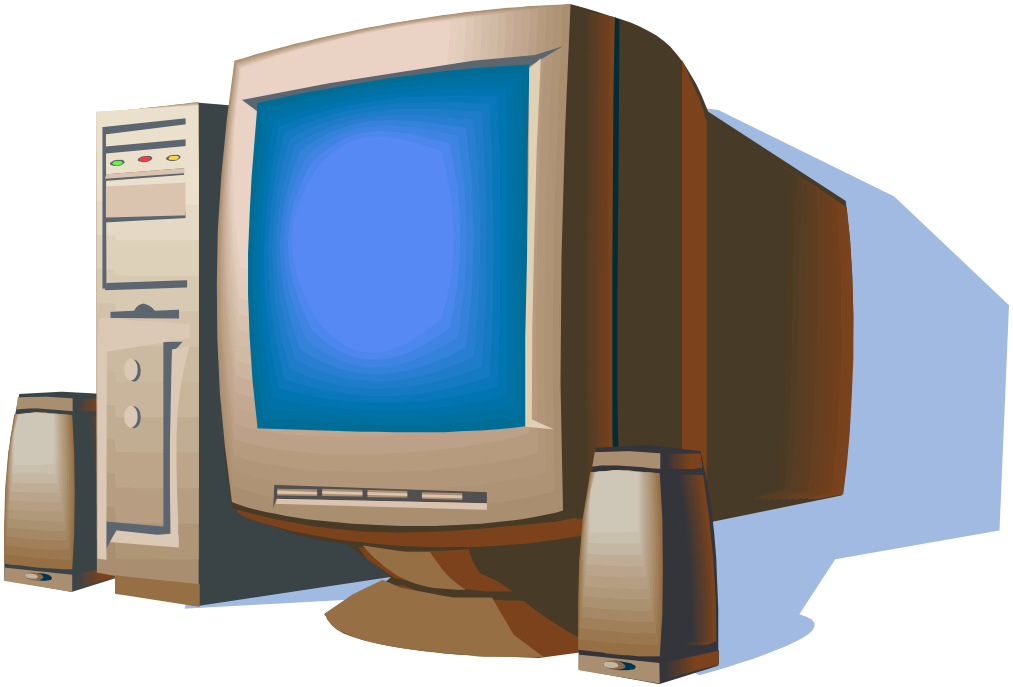


Pourquoi le langage C ?

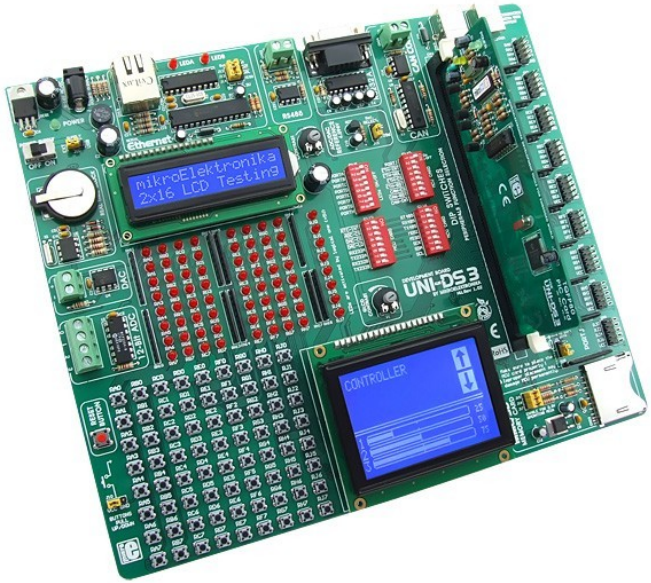
- Le langage C a été pendant longtemps le langage le plus utilisé
- Il l'est encore dans bon nombre de contextes
  - Programmation système
  - Informatique industrielle
  - Programmation de circuits
- Il constitue la base du C++
  - Pratiquement la même syntaxe
  - Philosophie objets en +

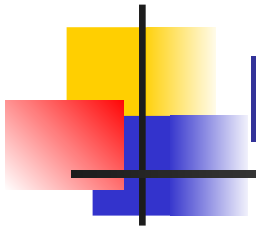
Cibles

- Compatible PC



- Carte Microcontrôleur





Introduction

- Langage crée par Dennis Ritchie en 1972 dans le cadre du développement d'UNIX (conçu par Ken Thompson)
- Langage de haut niveau mais donnant néanmoins accès aux ressources internes de l'ordinateur
- Langage compilé
- Langage fonctionnel et modulaire

Structure d'un programme en C

```
#include <…>

int main(void)
{
…
Instruction n;
Instruction n+1;
…
}
```

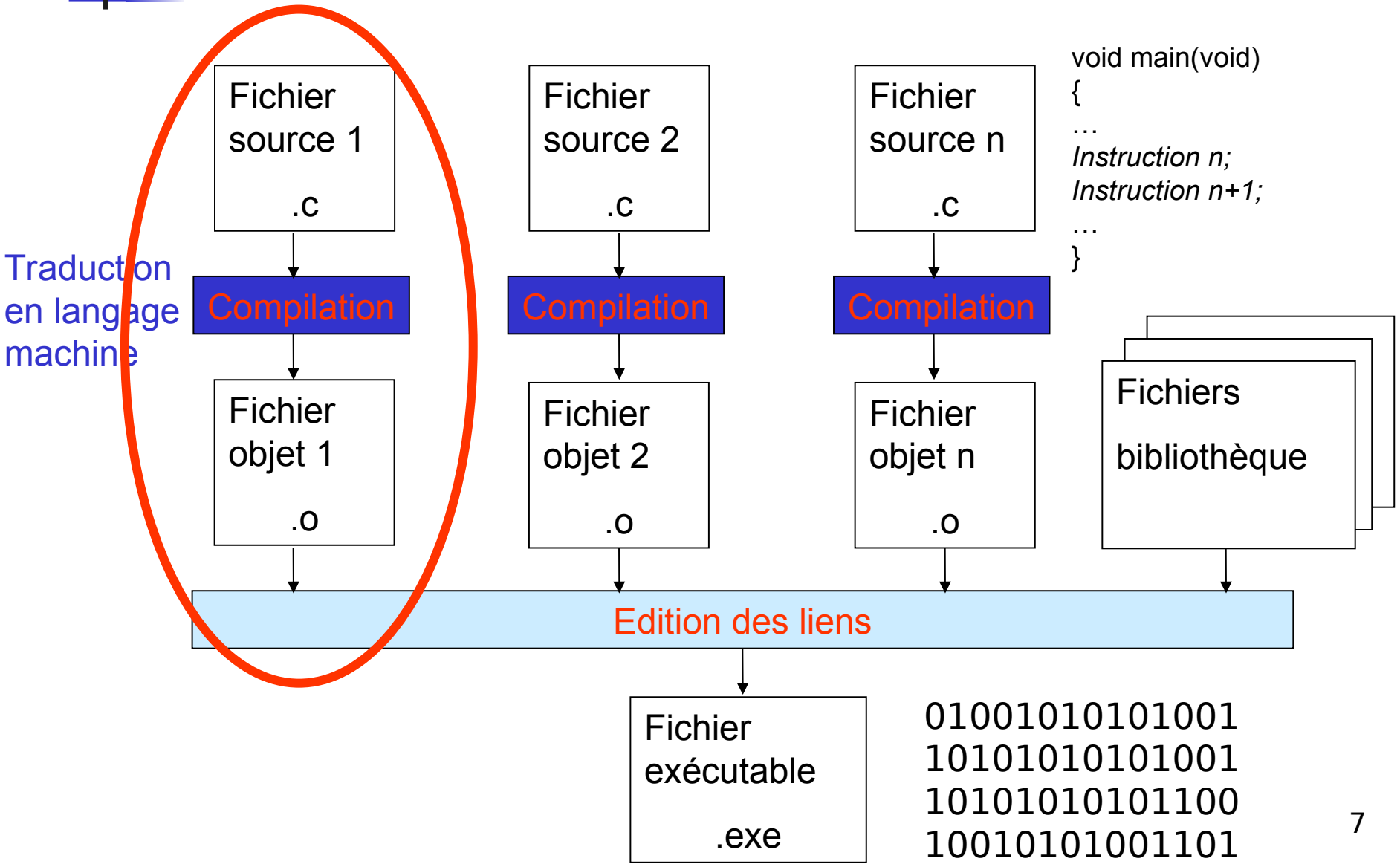
Directives de compilation

Fonction principale

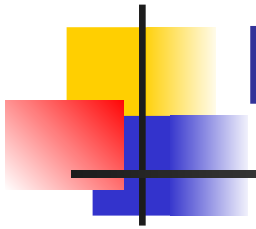
Exemple:

```
int main(void)
{   ...
    a=b+c;
    c=c+1;
    ...
}
```

Flot de compilation



Les types en C

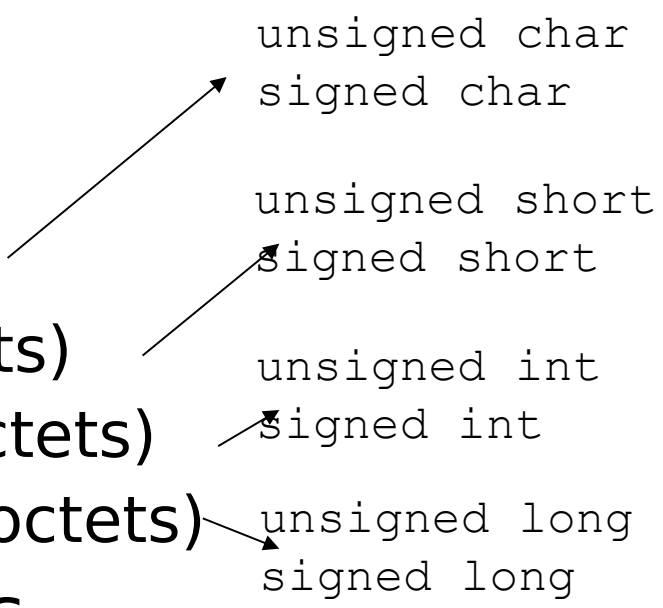


Rôle

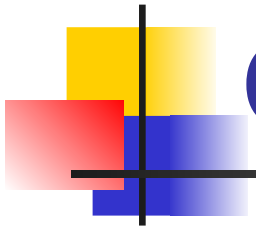
- Ils permettent de déterminer l'ensemble des valeurs que peut prendre un objet :
  - entiers, caractères
  - réels
  - autres types définis par le programmeur

Les types en C

- Types entiers
  - Qualificatifs
    - Signés
    - Non signés
  - `char` (1 octet)
  - `short` (2 octets)
  - `int` (2 ou 4 octets)
  - `long int` (4 octets)



- Types flottants
  - `float` (4 octets) simple précision
  - `double` (8 octets) double précision
  - `long double` (10 octets)

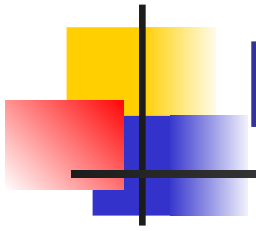


Gamme de représentation

Type entier	signed	unsigned
`char`	-128 à 127 (-2^7 à 2^7-1)	0 à 255 (0 à 2^8-1)
`short`	-32768 à 32767 (-2^{16} à $2^{16}-1$)	0 à 65535 (0 à $2^{16}-1$)
`int`	-2147483648 à 2147483647 (-2^{31} à $2^{31}-1$)	0 à 4294967295 (0 à $2^{32}-1$)
`long int`	idem	idem

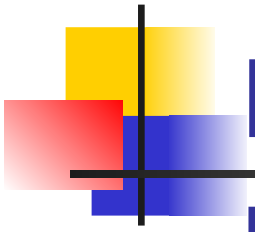
Gamme de représentation

Type flottant	
`float`	10^{-38} à 10^{38}
`double`	10^{-308} à 10^{308}
`long double`	10^{-4932} à 10^{4932}



Le type char

- Le type char est codé sur 1 octet.
- Il sert à stocker
  - des nombres entiers signés ou non-signés
    - `signed char`
    - `unsigned char`
  - des caractères



Nombres entiers non signés représentables (sur 8 bits)

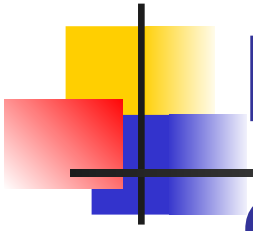
00000000	0
00000001	1
00000010	2
00000011	3
…	…
01111111	127
10000000	128
10000001	129
…	…
11111111	255

type `unsigned char`

Nombres entiers signés représentables (sur 8 bits)

00000000	0
00000001	1
00000010	2
00000011	3
…	…
01111111	127
10000000	-128
10000001	-127
…	…
11111111	-1

type `signed char`



La représentation des caractères

- Dans un ordinateur, un caractère est représenté par une valeur numérique.
  - C’est l’interprétation de cette valeur qui va déterminer ce qui sera affiché à l’écran
  - L’un des standards les plus courants est le format ASCII
    - 128 symboles, codés sur 7 bits
    - Les 32 premiers caractères sont non imprimables
    - Le format ASCII étendu permet de rajouter 128 symboles supplémentaires (format 8 bits).
  - Exemple d’autres formats : UTF-8, EBCDIC, …

Le code ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

Le code ASCII

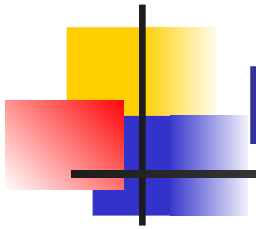
Le contenu de ma mémoire

1	2	3	4	5	6
7	8 **53**	9 **41**	10 **4C**	11 **55**	12 **54**
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	…

Le code ASCII étendu

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	|	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	⊕	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	▉	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

Source: www.asciitable.com



Manipulation de caractères

- Un caractère imprimable est accessible en l'entourant d'apostrophes (simples quotes)
  - Exemple
    - 'A'
    - '1'
    - 'f'
- Un caractère non imprimable doit être précédé du caractère anti-slash
  - Exemple
    - '\n' saut de ligne
    - '\t' tabulation horizontale
- Il est également possible d'utiliser directement le code ASCII du caractère en le précédent du caractère anti-slash
  - Exemple
    - '\65' -> caractère de code ASCII 65 soit la lettre A
    - '\49' -> caractère de code ASCII 49 soit la lettre 1

Codage des nombres réels

- Nombre flottant N en binaire :
  - Un bit de signe s
  - Un exposant e
  - Une mantisse m

$$N = (-1)^s \times m . 2^e$$

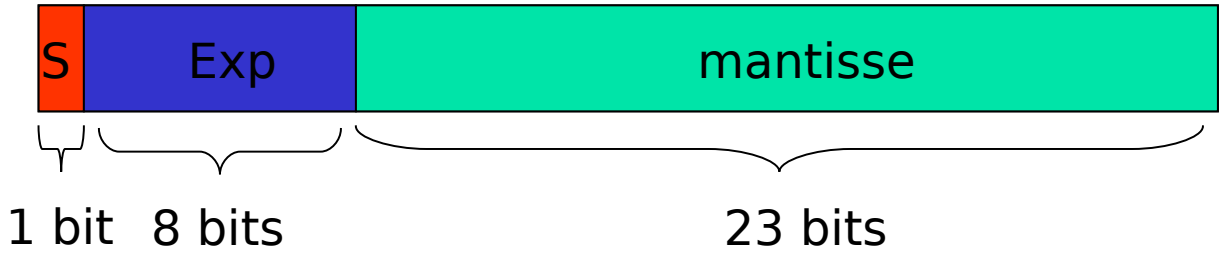
- Exemple

$$-1.01011001010101000111010 11 . 2^{-5}$$

bit de signe — mantisse — exposant

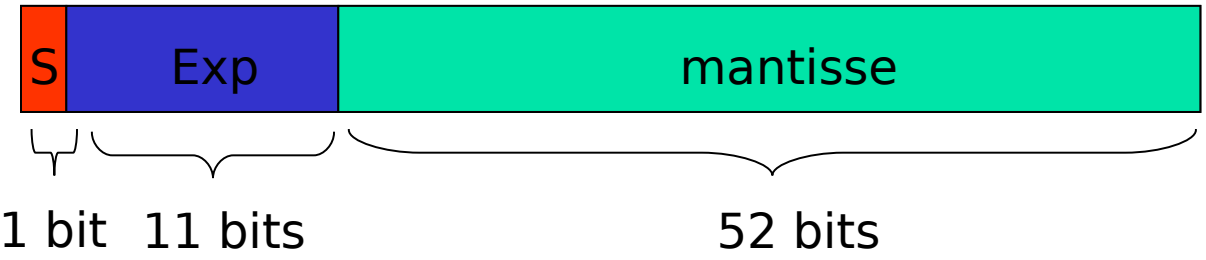
Type float

Précision simple sur 32 bits (10^{-38} à 10^{38})

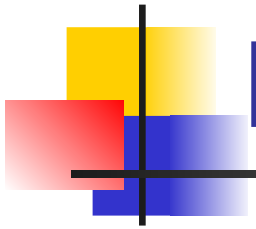


Type double

Précision double sur 64 bits (10^{-308} à 10^{308})

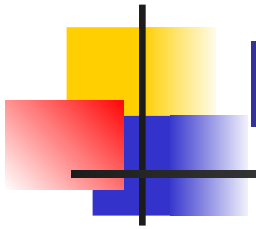


Les variables



Les variables

- Une variable est une entité permettant de stocker l'information
- Elle possède
  - Un nom ou identifiant (donné par le programmeur)
  - Une valeur
  - Un type qui caractérise l'ensemble des valeurs que peut prendre la variable
    - entier, réelle, complexe, etc.
- Elle est stockée dans la mémoire de l'ordinateur, à une adresse précise
- Une variable doit être initialisée, c'est à dire qu'on doit lui affecter une valeur.



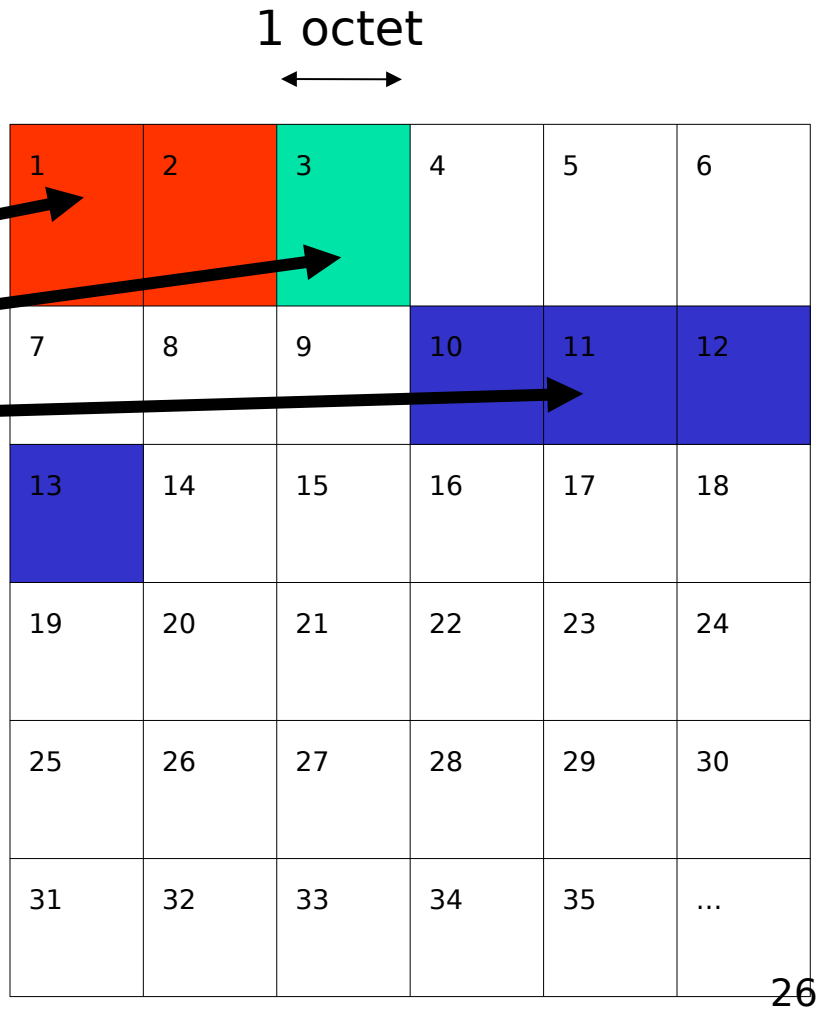
Identifiant d'une variable

- Il représente le nom donné à une variable par le programmeur
  - Exemples :

  indice
  taxe_ajoutee
  TaxeValeurAjoutee
  _chaine1
  nombre_de_lignes
- (RESTRICTION) Un identifiant ne doit pas
  - commencer par un chiffre : `3chaines`
  - contenir des accents : `pépé`
  - commencer par un '_' suivi d'un autre '_' ou d'une majuscule : `__test`, `_Animal`
- Conseil
  - un identifiant devrait porter un nom explicite relatif à l'information stockée
    - éviter les a,b,c…i,j,k, ii, jj, kk

Déclaration de variables

- Nécessaire pour réserver un emplacement mémoire pour la variable
- Syntaxe
  - TYPE identifiant;
- Exemple
  - `short nombre1;`
  - `char caractere;`
  - `float nombre2;`
- Mémoire
  - analogie avec une armoire de stockage
  - Chaque case mémoire est repérée par une adresse
  - Chaque case mémoire contient 1 octet
  - Par défaut le contenu de la mémoire n'est pas initialisé
  - L'ordinateur stocke la variable dans un emplacement libre



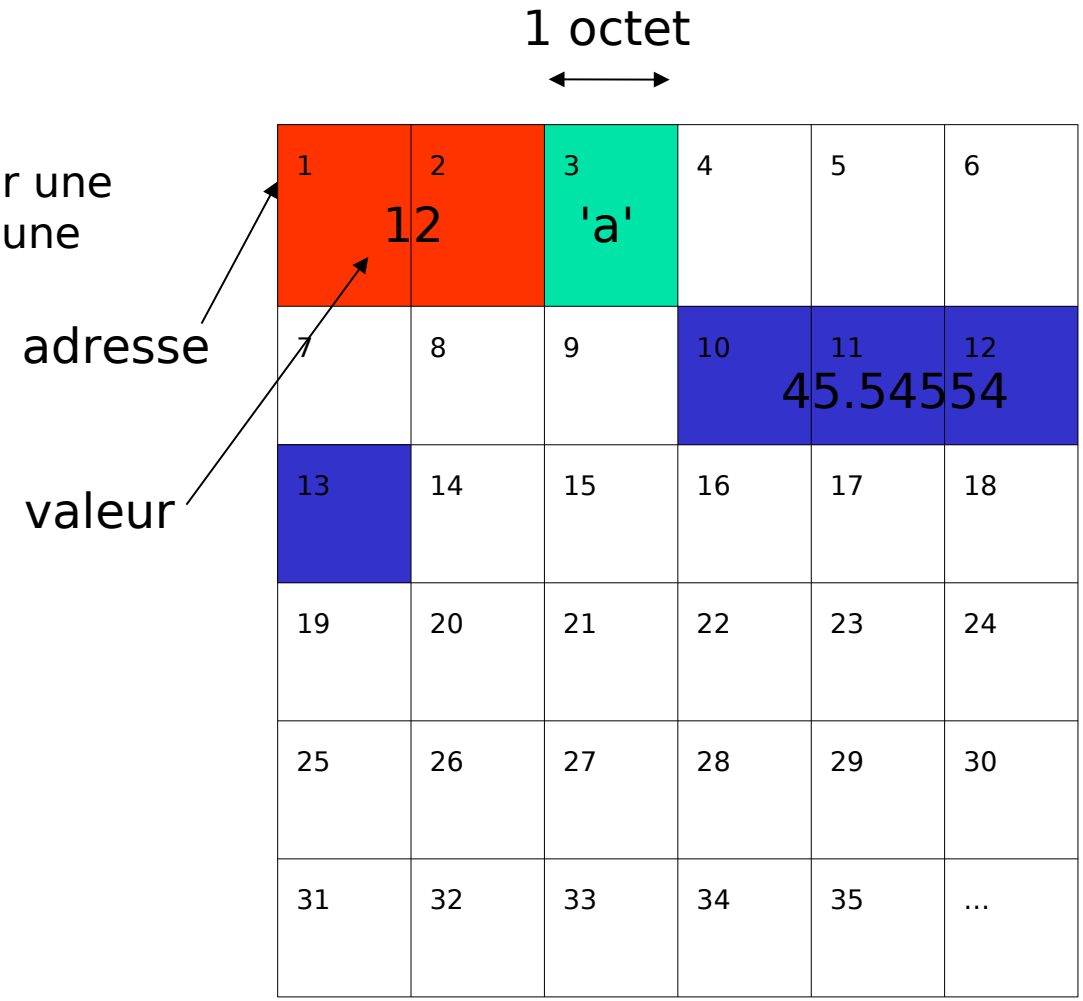
Initialisation de variables

- Il est nécessaire d'initialiser une variable afin de lui donner une valeur
- Exemple:

nombre1=12;

caractere='a';

nombre2=45.54554;



Initialisation lors de la déclaration

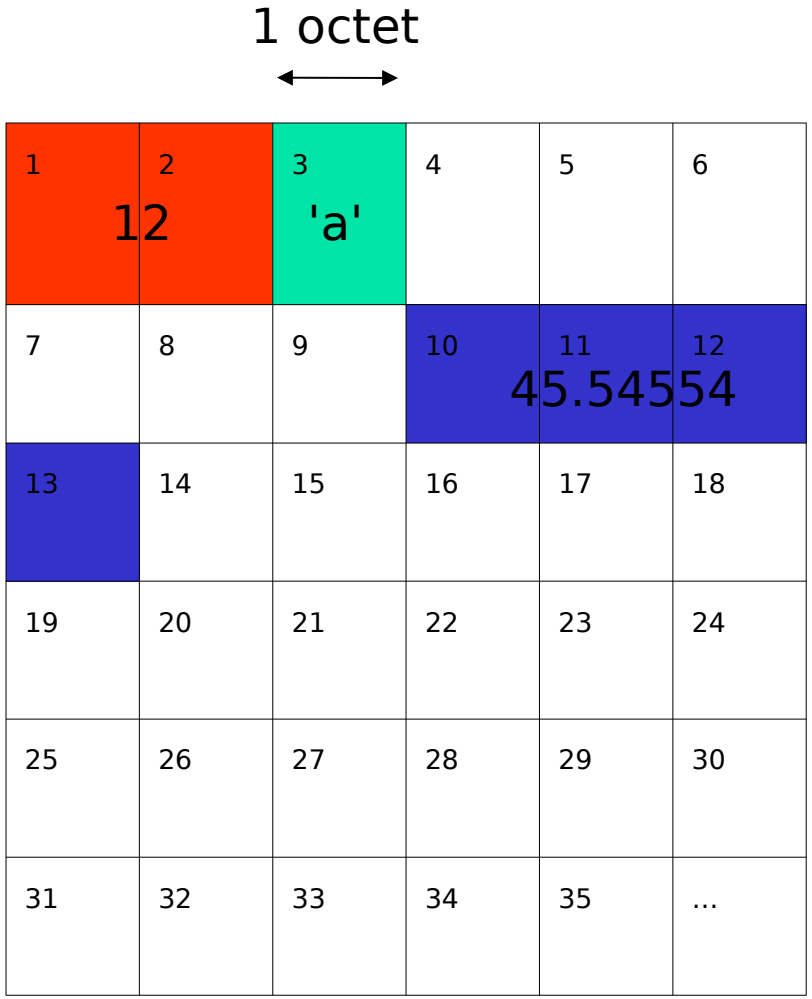
- En C, il est également possible de fournir une valeur directement lors de la déclaration d'une variable

- Exemple

```
short nombre1=12;
char caractere='a';
float nombre2=45.54554;
```

- L'opérateur = est appelé opérateur d'affectation.

  Il permet de fournir une valeur à une variable.



Exemple de programme en C

Déclaration des variables

```
int main(void)
{
   short nombre1;
   char caractere;
   float nombre2;
nombre1=12;
caractere='a';
nombre2=45.54554;
}
```

Déclaration des variables + Initialisation

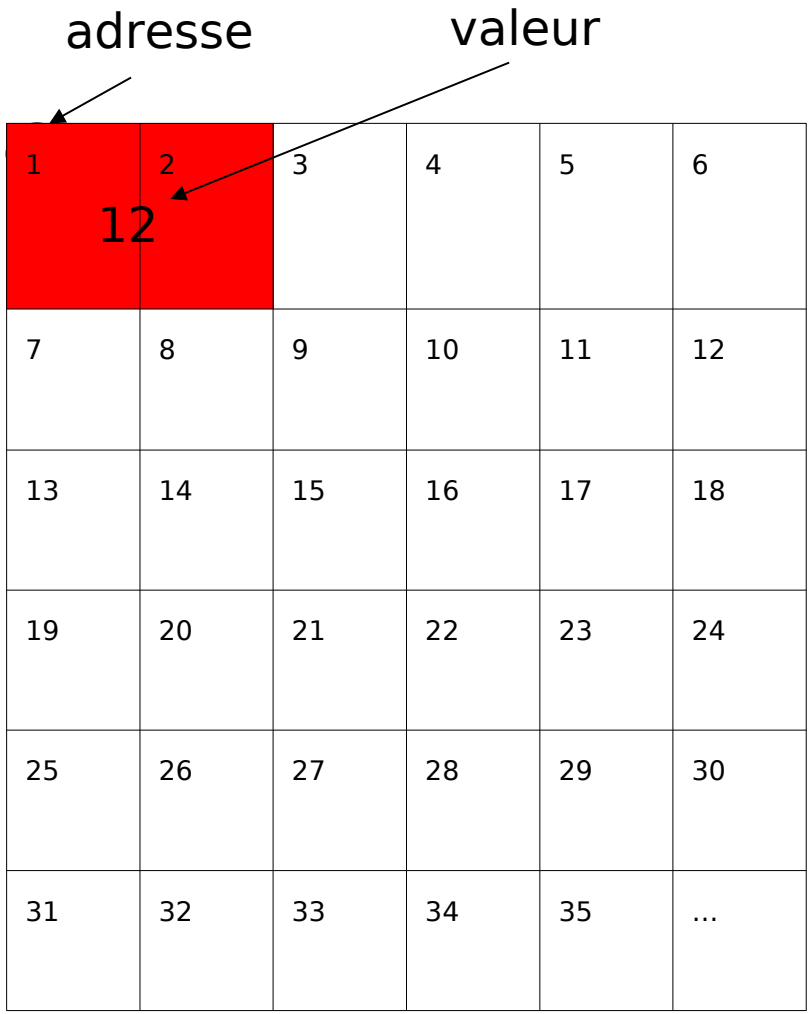
```
int main(void)
{
   short nombre1=12;
   char caractere='a';
   float
   nombre2=45.54554;
}
```

Adresse et valeur d'une variable

- En C, il est possible d'avoir accès à l'adresse de la variable via l'opérateur &
- Exemple
  - `short nombre1=12;`

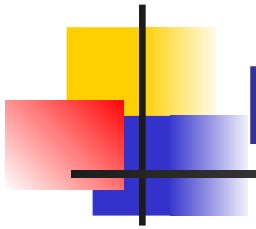
  nombre1 : 12
  &nombre1: 1

- Remarque
  - Il n'est pas possible de fixer une adresse
    - `&nombre1=1` EST IMPOSSIBLE
- Attention à ne pas confondre valeur et adresse



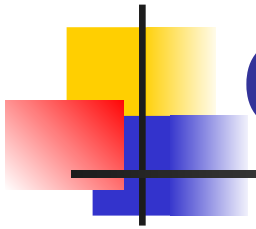
1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	...

Opérateurs et expressions



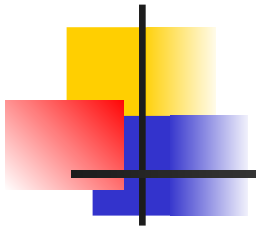
Les opérateurs et expressions

- **Opérateur**
  - Symbole d’opération qui permet d’agir sur des entités (opérandes)
- **Opérande**
  - Entité utilisée par un opérateur
- **Expression**
  - Suite d’opérandes et d’opérateurs
  - Elle possède une valeur
  - Elle possède un type



Opérateurs

- Types d'opérateurs
  - Unaire (n'admet qu'une seule opérande)
    - op A
  - Binaire (admet deux opérandes) de part et d'autre de l'opérateur
    - A op B
- `+` est un opérateur binaire
  - `a+b 5+2 45.45+23.3`
- `!` est un opérateur unaire
  - `!32 !a !nombre1`



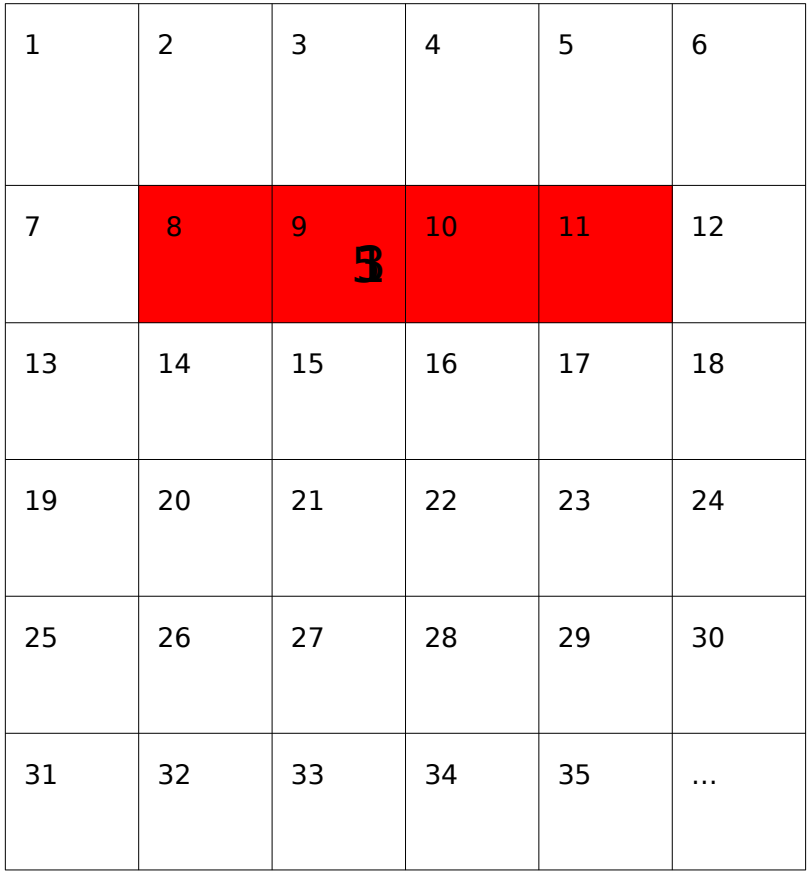
Types d’opérateurs

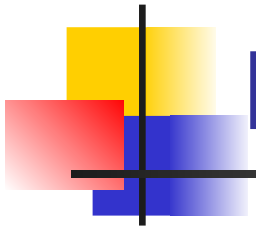
- Opérateur d’affectation
- Opérateurs arithmétiques
- Opérateurs de comparaison
- Opérateurs booléens
- Opérateurs sur chaînes de caractères
- …

L’ opérateur d’affectation

- En langage C, l’opérateur d’affectation est le =
- Exemple

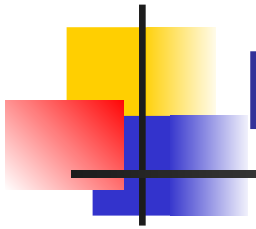
```
int main(void)
{
   int a;
   a = 3;
   a = 5;
   a = 1;
}
```





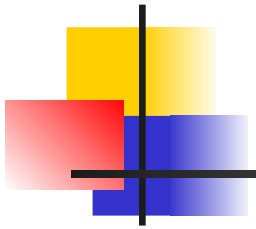
Les opérateurs arithmétiques

- Addition **+**
  - `5+4`
- Soustraction **–**
  - `7-1`
- Multiplication *****
  - `7*4.456`
- Division **/**
  - `3/2`
- Modulo **%** (reste de la division entière)
  - `11%5` vaut 1



Les expressions

- Une expression est une suite d’opérandes et d’opérateurs
  - Elle possède une valeur
  - Elle possède un type
- A l'intérieur d'une expression, les opérateurs possèdent une priorité
- Exemple
  - `a=5+2-1` possède la valeur 6
  - `6*4-3` possède la valeur 21



Exemple de programme

```
#include <stdio.h>
int main(void)
{
   int valeur1;
   int valeur2;
   int somme;


   somme=valeur1 + valeur2;

}
```