



PROGRAMMATION EN LANGUAGE C

Pr.: B. CHERKAOUI

WHAT IS C?

“C est un langage de programmation polyvalent qui se caractérise par une économie d’expression, un flux de contrôle et des structures de données modernes, et un riche ensemble d’opérateurs. Le C n’est pas un langage de « très haut niveau », ni un « grand » langage, et n’est pas spécialisé à un domaine d’application particulier. “

Kernighan & Richie, 1978

PLAN

- I. Généralités
- II. Éléments de base de C
- III. Entrées / Sorties
- IV. Structures de Contrôle

HISTORIQUE RAPIDE...

- Le Langage C a été conçu en 1972 par Dennis Richie et Ken Thompson
 - Deux chercheurs au Bell Labs USA
 - Objectif : développer un système d'exploitation UNIX
- En 1978, Brian Kernighan et Dennis Richie publient la définition classique du C
 - Première édition du livre "The C programming language"
- En 1983, l'ANSI décida de normaliser le langage
 - 1989: la définition de la norme ANSI C (ou C89)
 - Deuxième édition "The C programming language"
- L'ISO a repris la même norme en 1990 (ou C90)

CARACTÉRISTIQUES DU C

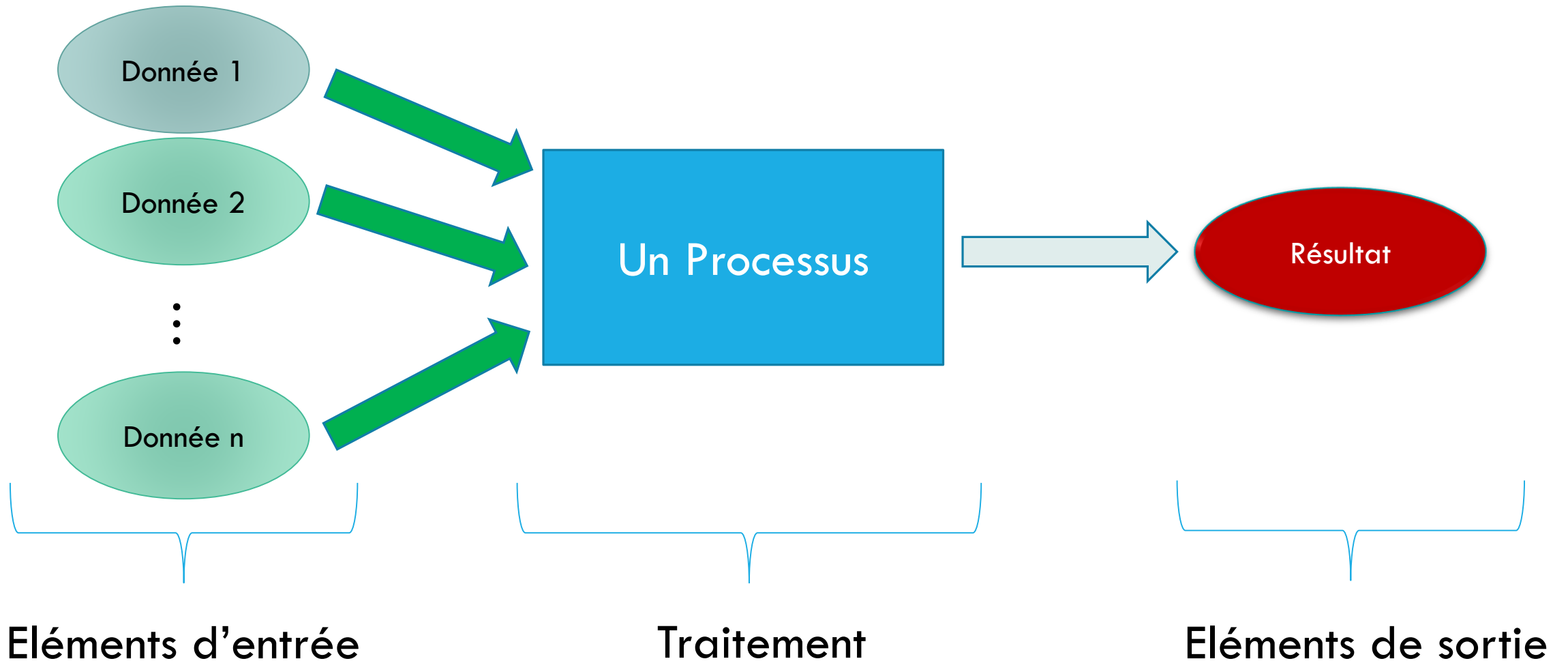
- **Universel** : n'est pas orienté vers un domaine d'application particulier (applications scientifiques, de gestion, ...)
- **Près de la machine** : offre des opérateurs qui sont proches de ceux du langage machine (manipulations de bits, d'adresses, ...) ➡ efficace
- **Modulaire** : peut être découpé en modules qui peuvent être compilés séparément
- **Portable** : en respectant le standard ANSI-C, il est possible d'utiliser le même programme sur plusieurs systèmes (hardware, système d'exploitation)

LA COMPILATION

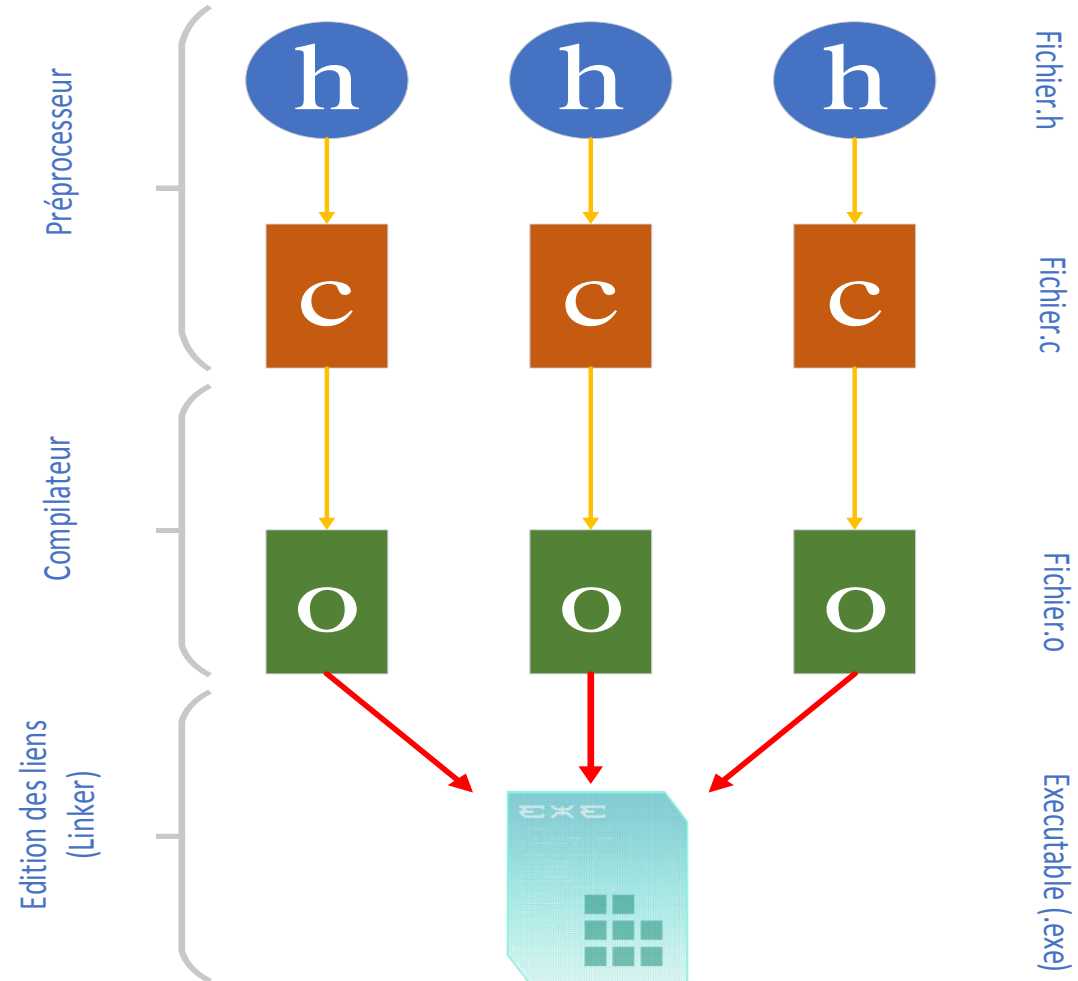
« Le fait de transformer un code source lisible par un humain en un fichier binaire exécutable par une machine »

- C est un langage compilé
 - Par opposition au langages interprétés (Python, Matlab, ...)
- Un programme C, écrit dans un fichier source, est traduit en totalité en langage machine avant exécution.
- Quatre phases :
 - Traitement par le préprocesseur
 - Compilation
 - Assemblage
 - Édition de liens

LA COMPILATION



LA COMPILATION



ELÉMENTS DE BASE: PREMIER PROGRAMME

Pour réussir son programme, 3 étapes sont essentielles:

- A éditer via un éditeur de texte → Fichier source: programme.c
- Compiler le programme
- Exécuter 😊

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

ELÉMENTS DE BASE: STRUCTURE D'UN PROGRAMME

IMPORTANT:

Chaque instructions termine impérativement par un point-virgule ;

○ Partie 1: Les déclarations

Elle comporte la déclaration des fonctions des bibliothèques (bibliothèque standard ou autre) par inclusion de fichiers fournis avec le langage et peut comprendre des déclarations des variables « globales ».

○ Partie 2 : Le corps du programme

Tout programme C doit comporter une fonction principale main. Cette fonction est celle utilisée par le système pour exécuter le programme.

```
#include <stdio.h>

int main()
{
    instruction 1;
    instruction 2;
    .
    .
    .
    instruction n;
}
```

Partie 1:
Déclaration

Partie 2:
Corps du
programme

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

- Six catégories de composants élémentaires :
 1. Les identificateurs
 2. Les mots clés
 3. Les types prédéfinis
 4. Les opérateurs
 5. Les chaînes de caractères
- Les commentaires sont enlevés par le préprocesseur

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

1. Les identificateurs:

Le rôle d'un identificateur est de donner un nom à une entité du programme. Plus précisément, un identificateur peut désigner :

- Un nom de variable ou de fonction
- Un type défini par *typedef*, *struct*, *union* ou *enum*
- Une étiquette

Un identificateur est une suite de caractères parmi :

- Les lettres (minuscules ou majuscules, mais non accentuées)
- Les chiffres
- le ``blanc souligné" (_)

Ne doit pas être un des mots clés du langage.

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

2. Les mots clés:

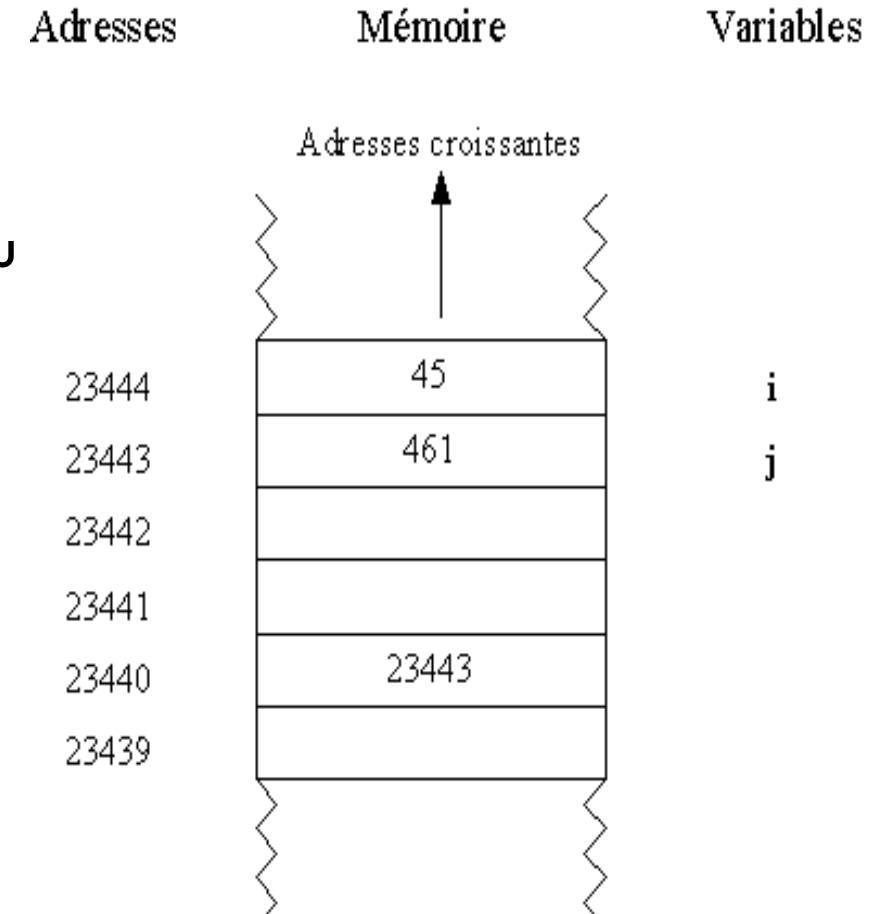
Un certain nombre de mots, appelés mots-clefs, sont réservés pour le langage lui-même et ne peuvent pas être utilisés comme identificateurs. L'ANSI C compte 32 mots clefs :

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

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

3. Les types prédéfinis

- Le C est un langage typé: Toute variable, constante ou fonction est d'un type précis.
- Le type définit la représentation mémoire d'un objet.
- Les types de base en C concernent:
 - Les entiers
 - Les flottants (nombres réels)
 - Les caractères



ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Les types entiers

Le type entier représente l'ensemble des entiers relatifs (positifs et négatifs) avec plusieurs sous-types:

Type	Taille	Valeurs	Intervalle
char	8 bits	caractères	$[-128, 127]$
short	16 bits	Entiers courts	$[-32768, 32767]$
int	32 bits	Entiers	$[-2^{31}, 2^{31} - 1]$
long	64 bits	Entiers long	$[-2^{63}, 2^{63} - 1]$
unsigned char	8 bits	caractères	$[0, 255]$
unsigned short	16 bits	Entiers courts non signés	$[0, 65536]$
unsigned int	32 bits	Entiers non signés	$[0, 2^{32} - 1]$
unsigned long	64 bits	Entiers long non signés	$[0, 2^{64} - 1]$

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Les types flottants

3 types correspondant à différentes précisions :

Type	Taille	Valeurs
float	32 bits	Flottants simple précision
double	64 bits	Flottants double précision
long double	128 bits	Flottants précision étendue

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Les types caractères

- Le type « **char** » représente le jeu de caractères de la machine.
- Un char en C est codé sur un octet (8 bits).
- Un caractère est encodé en utilisant un entier avant sa représentation binaire en mémoire.
- Plusieurs encodages sont utilisés:
 - ASCII (American Standard Code for Information Interchange)
 - Version originale (1960) représente 128 caractères avec les nombres de 0 à 127 sur 7 bits.
 - Version étendue (1980) représente 256 sur 8bits
 - Unicode (1990) représente 65 536 sur 16 bits

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

- L'ensemble des caractères ASCII (Version originale)

	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	SP	!	"	#	\$	%	&	`
4	()	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[\]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	DEL		

Comment lire:

R sur ligne 8 colonne 2 est encodé par 82.

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

- Les caractères non imprimables :

NOTATION EN C	CODE ASCII (hexadécimal)	ABRÉVIATION USUELLE	SIGNIFICATION
<code>\a</code>	07	BEL	cloche ou bip (alert ou audible bell)
<code>\b</code>	08	BS	Retour arrière (Backspace)
<code>\f</code>	0C	FF	Saut de page (Form Feed)
<code>\n</code>	0A	LF	Saut de ligne (Line Feed)
<code>\r</code>	0D	CR	Retour chariot (Carriage Return)
<code>\t</code>	09	HT	Tabulation horizontale (Horizontal Tab)
<code>\v</code>	0B	VT	Tabulation verticale (Vertical Tab)
<code>\\</code>	5C	<code>\</code>	
<code>\'</code>	2C	<code>'</code>	
<code>\"</code>	22	<code>"</code>	
<code>\?</code>	3F	<code>?</code>	

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Les constantes en C

- Une constante est une valeur qui apparaît littéralement dans le code source d'un programme. Exemple: 123, 'A', "Hello", 1.5, ...
- La manière avec laquelle on écrit une constante détermine implicitement son type
- Définition des constantes symboliques à l'aide de la directive :

`#define NOM Valeur`

- Demande au préprocesseur de remplacer NOM par Valeur dans la suite du fichier source. Exemples:

`#define PI 3.14`

`#define N 100`

`#define MIN 0`

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Variables

- Les variables servent à stocker les valeurs des données utilisées pendant l'exécution d'un programme.
- Les variables doivent être déclarées avant d'être utilisées, elles doivent être caractérisées par :



ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Déclaration des variables

- Les déclarations introduisent les variables qui seront utilisées, fixent leur type et parfois aussi leur valeur de départ (initialisation)
- Syntaxe de déclaration en C:

`<Type> <NomVar1>,<NomVar2>,...,<NomVarN>;`

- Exemples:

`int i, j,k;`

`float x, y ;`

`double z=1.5;`

`short compteur;`

`char c=`A`;`

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Le choix d'un identificateur (nom d'une variable ou d'une fonction) est soumis à quelques règles :

- doit être constitué uniquement de lettres, de chiffres et du caractère souligné _ (Eviter les caractères de ponctuation et les espaces)

Correcte: PRIX_HT, prixHT

Incorrecte: PRIX-HT, prix HT, prix.HT

- doit commencer par une lettre (y compris le caractère souligné)

Correcte : A1, A1

Incorrecte: 1A

- doit être différent des mots clés réservés du langage.

Remarque: C distingue entre les majuscules et les minuscules. **NOMBRE** et **nombre** sont deux identificateurs différents.

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

4. Les opérateurs:

Le langage C est riche en opérateurs. Outre les opérateurs standards, il comporte des opérateurs originaux d'affectation, d'incrémentation et de manipulation de bits.

Catégorie	Opérateurs	Syntaxe et remarques
Affectation	=	<code>variable = expression;</code> • Ne pas confondre avec ==
arithmétiques	+ - * / %	• Division entière et réelle : si les deux opérandes sont entières, / produira une division entière (quotient de la division). Exemple : float x; x = 3/2; → x = 1.0 x = 3.0/2; → x = 1.5 • Pas d'opérateur de puissance en C. on utilise la fonction pow(x,y) de math.h
Comparaison	< <= > >= == !=	<code>expression1 op expression2</code> • Le résultat est de type int (pas de type booléen en C): 1 si vrai, et 0 sinon.

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Catégorie	Opérateurs	Syntaxe et remarques
Logiques booléens	<code>&&</code> (le ET) <code> </code> (le OU) <code>!</code> (le NON)	<ul style="list-style-type: none">• Le résultat est de type <code>int</code>: 1 si vrai, et 0 sinon.• L'évaluation d'une expression se fait de gauche à droite et s'arrête dès que le résultat final est déterminé. <p>Exemple:</p> <pre>int i, j, n; if (i!=j && i<n && j<n) i<n ne sera évaluée que si i!=j est vraie j<n ne sera évaluée que si i!=j et i<n vraies</pre>

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Catégorie	Opérateurs	Syntaxe et remarques																																				
Logiques bit à bit	& ^	Exemple et signification: unsigned char a = 103, b = 41; //sur 8 bits																																				
(bitwise operators)	- << >>	<table><tr><th>expr</th><th>binaire</th><th>déc</th><th>signification</th></tr><tr><td>a</td><td>0110 0111</td><td>103</td><td>valeur de a</td></tr><tr><td>b</td><td>0010 1001</td><td>41</td><td>valeur de b</td></tr><tr><td>a & b</td><td>0010 0001</td><td>33</td><td>et bit à bit</td></tr><tr><td>a b</td><td>0110 1111</td><td>111</td><td>ou bit à bit</td></tr><tr><td>a ^ b</td><td>0100 1110</td><td>78</td><td>ou exclusif</td></tr><tr><td>~a</td><td>1001 1000</td><td>152</td><td>complément à 1</td></tr><tr><td>a >>2</td><td>0001 1001</td><td>25</td><td>décalage à droite</td></tr><tr><td>a <<3</td><td>0011 1000</td><td>56</td><td>décalage à gauche</td></tr></table>	expr	binaire	déc	signification	a	0110 0111	103	valeur de a	b	0010 1001	41	valeur de b	a & b	0010 0001	33	et bit à bit	a b	0110 1111	111	ou bit à bit	a ^ b	0100 1110	78	ou exclusif	~a	1001 1000	152	complément à 1	a >>2	0001 1001	25	décalage à droite	a <<3	0011 1000	56	décalage à gauche
expr	binaire	déc	signification																																			
a	0110 0111	103	valeur de a																																			
b	0010 1001	41	valeur de b																																			
a & b	0010 0001	33	et bit à bit																																			
a b	0110 1111	111	ou bit à bit																																			
a ^ b	0100 1110	78	ou exclusif																																			
~a	1001 1000	152	complément à 1																																			
a >>2	0001 1001	25	décalage à droite																																			
a <<3	0011 1000	56	décalage à gauche																																			

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Catégorie	Opérateurs	Syntaxe et remarques
Incrémentation Décrémentation	++ --	<ul style="list-style-type: none">• ++ ajoute 1 à son opérande• -- soustrait 1 à son opérande• S'utilisent en suffixe (<code>var++</code> et <code>var--</code>) et en préfixe (<code>++var</code> et <code>--var</code>). Exemple: <pre>int x, n; n = 5;</pre> <ul style="list-style-type: none">• <code>x = n++</code>; incrémentation après affectation → <code>x = 5</code> puis <code>n = 6</code>• <code>x = ++n</code>; incrémentation avant affectation → <code>n = 6</code> puis <code>x = 6</code>
Affectation composée	<code>+=</code> <code>-=</code> <code>*=</code> <code>/=</code> <code>%=</code>	<code>variable op= expression;</code> Équivalent à: <code>variable = variable op expression;</code>

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

Catégorie	Opérateurs	Syntaxe et remarques
Opérateur conditionnel ternaire	? :	<p><i>Condition ? Expression1 : Expression2</i></p> <ul style="list-style-type: none">• Le résultat est Expression1 si la condition est vraie et Expression2 sinon.• C'est l'équivalent d'un if – else. <p>Exemple :</p> <pre>int a, b, max, min; min = (a<=b) ? a : b; max = (a>=b) ? a : b; float x; x = (x>=0) ? x : (-x);</pre>

ELÉMENTS DE BASE: COMPOSANTS ÉLÉMENTAIRES DE C

- Vous pouvez télécharger Dev-C++ librement, par exemple sur le site:

www.bloodshed.net



ENTRÉES / SORTIES:

- Il 'agit des instructions permettant à la machine de communiquer avec l'utilisateur:
- Il s'agit des fonctions de la librairie standard **stdio.h** utilisées avec les unités classiques d'entrées / sorties: Le clavier et l'écran
- La librairie standard **stdio.h** contient un ensemble de fonctions qui assurent la lecture et l'écriture des données:
 - **printf()** écriture formatée de données
 - **scanf()** lecture formatée de données

ENTRÉES / SORTIES: FONCTION D’AFFICHAGE

- La fonction **printf**:
 - C'est une fonction d'impression formatée, les données sont converties selon le format choisi avant impression.

- Syntaxe:

`printf("Format", expression1, ..., expressionN);`

- `expression1, ...` : sont les variables et les expressions dont les valeurs sont à représenter.
- `Format` : est une chaîne de caractères qui peut contenir:
 - du texte
 - des séquences d'échappement (`'\n'`, `'\t'`, ...) ➡ Les caractères non imprimables
 - des spécificateurs de format : un ou deux caractères précédés du symbole `%`, indiquant le format d'affichage

Remarque: Le nombre de spécificateurs de format doit être égale au nombre d'expressions!

ENTRÉES / SORTIES: FONCTION D’AFFICHAGE

- Spécificateurs de format pour printf:

format	conversion en	écriture
%d	int	décimale signée
%ld	long int	décimale signée
%u	unsigned int	décimale non signée
%lu	unsigned long int	décimale non signée
%o	unsigned int	octale non signée
%lo	unsigned long int	octale non signée
%x	unsigned int	hexadécimale non signée
%lx	unsigned long int	hexadécimale non signée
%f	double	décimale virgule fixe
%lf	long double	décimale virgule fixe
%e	double	décimale notation exponentielle
%le	long double	décimale notation exponentielle
%g	double	décimale, représentation la plus courte parmi %f et %e
%lg	long double	décimale, représentation la plus courte parmi %lf et %le
%c	unsigned char	caractère
%s	char*	chaîne de caractères

ENTRÉES / SORTIES: FONCTION D’AFFICHAGE

○ 'affichage du texte peut être contrôlé à l'aide des séquences d'échappement(caractères non imprimables) :

- `\n` : nouvelle ligne
- `\t` : tabulation horizontale
- `\a` : signal sonore
- `\b` : retour arrière
- `\r` : retour chariot
- `\v` : tabulation verticale
- `\f` : saut de page
- `\\` : back slash (`\`)
- `\'` : apostrophe
- `\"` : guillemet

ENTRÉES / SORTIES: FONCTION D’AFFICHAGE

○ Exemple 1:

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

```
main()
```

```
{
```

```
    int i=1 , j=2, N=15;
```

```
    printf("la somme de %d et %d est %d \n", i, j, i+j);
```

```
    printf(" N= %0x \n" , N);
```

```
    char c='A' ;
```

```
    printf(" le code Ascii de %c est %d \n", c, c);
```

```
}
```

```
la somme de 1 et 2 est 3
N= f
le code Ascii de A est 65
Appuyez sur une touche pour continuer...
```

ENTRÉES / SORTIES: FONCTION D’AFFICHAGE

○ Exemple 2:

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

```
main()
```

```
{
```

```
    float x=10.5, y=2.5, result;
```

```
    result = x/y;
```

```
    printf("%f divisé par %f égal à %f \n", x, y, result);
```






```
    printf("%f divisé par %f égal à %f \n", x, y, x/y);
```

```
}
```

```
10.500000 divisé par 2.500000 égal à 4.200000
10.500000 divisé par 2.500000 égal à 4.200000
```







ENTRÉES / SORTIES: FONCTION D'AFFICHAGE

- Par défaut, les entiers sont affichés sans espaces avant ou après.
- Pour agir sur l'affichage, un nombre est placé après % afin de préciser le nombre de caractères **minimum à utiliser**
- Exemple: `printf("%4d" , n);`

<code>n = 25;</code>		<code>~~20</code> (~: espace)
<code>n = 75636</code>		<code>75636</code>
<code>printf("%4X", 123);</code>		<code>~~ 7B</code>
<code>printf("%4x", 123);</code>		<code>~~ 7b</code>
<code>printf("%4o", 123);</code>		<code>~173</code>

ENTRÉES / SORTIES: FONCTION D’AFFICHAGE

- Pour les réels, on peut préciser la largeur minimale de la valeur à afficher et le nombre de chiffres après le point décimal.
- La précision par défaut est fixée à six décimales. Les positions décimales sont arrondies à la valeur la plus proche. Exemples :

<code>printf("%f", 100.123);</code>		100.123000
<code>printf("%12f", 100.123);</code>		~~100.123000
<code>printf("%.2f", 100.123);</code>		100.12
<code>printf("%5.0f", 100.123);</code>		~~100
<code>printf("%10.3f", 100.123);</code>		~~~100.123
<code>printf("%.4f", 1.23456);</code>		1.2346

ENTRÉES / SORTIES: FONCTION DE LECTURE

- La fonction **scanf** permet de saisir des données au clavier les convertir selon les formats spécifiés puis les stocker en mémoire.

Syntaxe:

```
scanf("formats", adresse1, adresse2, ... , adresseN);
```

- **Formats:** le format de lecture de données, est le même que pour printf
- **Adresse1, Adresse2,..., AdresseN:** adresses des variables auxquelles les données seront attribuées. L'adresse d'une variable est indiquée par le **nom** de la variable précédé du signe **&**

ENTRÉES / SORTIES: FONCTION DE LECTURE

○ Exemple:

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

```
main()
```

```
{
```

```
    int i , j;
```

```
    scanf("%d%d", &i, &j);
```

```
    printf("i=%d et j=%d", i, j);
```

```
}
```

Ce programme permet de lire un réel simple et un autre double du clavier et les afficher à l'écran