Chapitre 2

Identificateurs, types de base et variables

1. Identificateurs

- Les identificateurs nomment les objets C++ (variables, fonctions, classes etc.).
- Un identificateur peut-être d'au moins un caractère. Le premier caractère doit être une lettre, un digit ou bien un souligné.
- Un identificateur peut-être arbitrairement très long mais certains compilateurs ne vont distinguer que les 31 premiers caractères.
- Les identificateurs qui contiennent un double souligné ou bien qui commencent par un souligné suivi par une lettre en majuscule sont réservés pour être utilisés par le système.

Identificateurs	Commentaires
`Affiche, indice, entree, sortie`	style normal
`I216, q, zz, a231k`	Ok, mais bonjour la compréhension !
`_Sys, __PART_, __C__`	Réservés pour le système
`9parties, passe-2, 50$billet`	Incorrects
`bool, true, return, int, class`	Réservés, ne peuvent être utilisés.

2. Mots clés réservés

- Un identificateur ne peut-être un mot clé réservé du langage.

Les mots clés du C++ sont :

asm	do	If	return	typedef
auto	double	inline	short	typeid
bool	dynamic_cast	Int	signed	typename
break	else	long	sizeof	union
case	enum	mutable	static	unsigned
catch	explicit	namespace	static_cast	using
char	export	new	struct	virtual
class	extern	operator	switch	void
const	false	private	template	volatile
const_cast	float	protected	this	wchar_t
continue	for	public	throw	while
default	friend	register	true	
delete	goto	reinterpret_cast	try	

Certains de ces mots clés proviennent du langage C. D'autres ont été repris par le langage Java.

3. Types primitifs

- Les types primitifs disponibles déjà en C et Java sont utilisables en C++.
- Les données peuvent être :
    1. un nombre entier,
    2. un réel,
    3. un booléen,
    4. un caractère,
    5. une chaîne de caractères.

3.1 Nombre entier

- Décimal, octal ou hexadécimal.
- Un préfixe permet de préciser la base : 0X ou 0x pour un entier hexadécimal, 0 pour un octal, « rien » pour un décimal.
- Ce nombre peut avoir aussi un suffixe, une combinaison des lettres U (ou u) et L (ou l). U pour « Unsigned » non signé et L « Long » pour long.
- Si un nombre entier est codé sur 2 bits dont le bit le plus élevé est le bit de signe, on ne peut avoir pour cette représentation que les nombres entiers suivants : -1 (11), -0 (10), +0 (00) et 1 (01).
- Si un nombre entier est non signé, le bit le plus élevé ne sert plus le signe puisque tous les nombres sont des entiers positifs. Ce bit va permettre de représenter encore plus de nombres. Nous aurons ainsi, toujours pour un entier codé sur 2 bits, les nombres suivants : 0 (00), 1 (01), 2 (10) et 3 (11).
- La représentation « Long » signifie que nous allons coder les nombres en utilisant plus de précision (donc plus de bits).

Exemples	Commentaires
`314 0314 0x314`	Légales : décimal, octal, hexadécimal
`314u, 314L`	Légales : non signé, long
`0XABC`	Légale : un hexadécimal
`098`	Illégale : un octal prend une valeur entre 0 et 7.
`314uu`	Illégale : deux fois « u ».

3.2 Réel

- Un nombre représenté en virgule flottante et peut avoir un de ces types :
    1. Float : pour une représentation en simple précision.
    2. Double : pour une représentation en double précision.
    3. Long double : pour une représentation à précision étendue.

Exemples	Commentaires
`3.14f 3.14F`	Légales : « Float »
`3.14 .314`	Légales : « double »
`3.14L 3.14l`	Légales : « Long double »
`9. 9.0 9E1`	Légales : « double » 3.0
`900e-2 .09e-2 90e-1`	Légales : « double » 9.0

3.3 Booléen

- Une constante du type « bool » qui peut prendre les valeurs « true » pour vraie et « false » pour faux.

```
bool test = false ;

if (test) {
     cout << "Le test est vrai" << endl ;
}else{
     cout << "Le test est faux" << endl ;
}else
```

3.4 Caractère

- Un caractère est généralement écrit entre apostrophes.
- Le type utilisé pour le représenter est « char ».
- Par exemple, le caractère « a » peut être exprimé sous l'une des formes suivantes : 'a', char (97), '\0141', '\x61'.
- Quelques caractères de contrôle nécessitent l'utilisation d'une forme symbolique. Cette forme est représentée par « \ ». Par exemple '\t' signifie une tabulation, '\?' pour représenter un point d'interrogation.
- Un caractère a une valeur égale à la valeur numérique du caractère dans le code caractère de la machine. Si le code « Ascii » est utilisé, la valeur numérique de « a » est 97 alors que celle de « A » est 65.
- Un caractère est représenté sur 1 octet. Dans ce cas, sa valeur numérique sera entre 0 et 127.

- Un caractère peut-être représenté aussi d'une manière non signée. Dans ce cas, sa valeur numérique sera comprise entre 0 et 255. Ici, on parle d'une table « Ascii » étendue.

3.5 Chaîne de caractères

- Elle est constituée d'une série de caractères.
- Elle est délimitée par les guillemets « " ».
- Les constantes chaînes de caractères sont stockées en zone de mémoire permanente.
- Elles sont considérées comme un tableau dont la taille est la longueur de la chaîne plus 1. Ce « 1 » en plus, c'est le caractère nul « \0 » placé à la fin de la chaîne.

4. Déclaration et définition

- Pour pouvoir utiliser un identificateur, un compilateur doit savoir à quoi correspond cet identificateur : un type donné, un nom de variable, un nom de fonction ou toute autre chose.
- Une déclaration informe le compilateur sur la nature exacte de l'identificateur.
- Une définition est associée à un espace mémoire, une valeur, un corps ou le contenu d'une déclaration.
- La différence entre une déclaration et une définition est que la première fournit le nom et le type de l'identificateur alors que la seconde une vue matérielle de l'identificateur.

5. Variable

- Une variable possède un nom unique et doit être déclarée avant d'être utilisée.

Exemples	Commentaires
`int x ;`	Déclaration simple
`int x=2 ;`	Déclaration + initialisation
`int x = z = 2 ;`	Déclaration + initialisation en chaîne
`int x, z=2 ;`	Déclaration de x et z. Initialisation de z

- Pour une variable les notions de déclaration et de définition se confondent.
- Une variable peut être déclarée à n'importe quel emplacement du programme (c'est aussi le cas en Java mais pas en C).
- Une variable peut-être initialisée au moment de sa déclaration.
- Une variable peut-être utilisée sans être préalablement initialisée (ce serait une erreur en Java).
- On peut décaler son initialisation plus loin dans le programme.
- Une variable peut-être initialisée aussi avec la valeur d'une expression.

```
int x = 0;
int z = x+2 ;
```