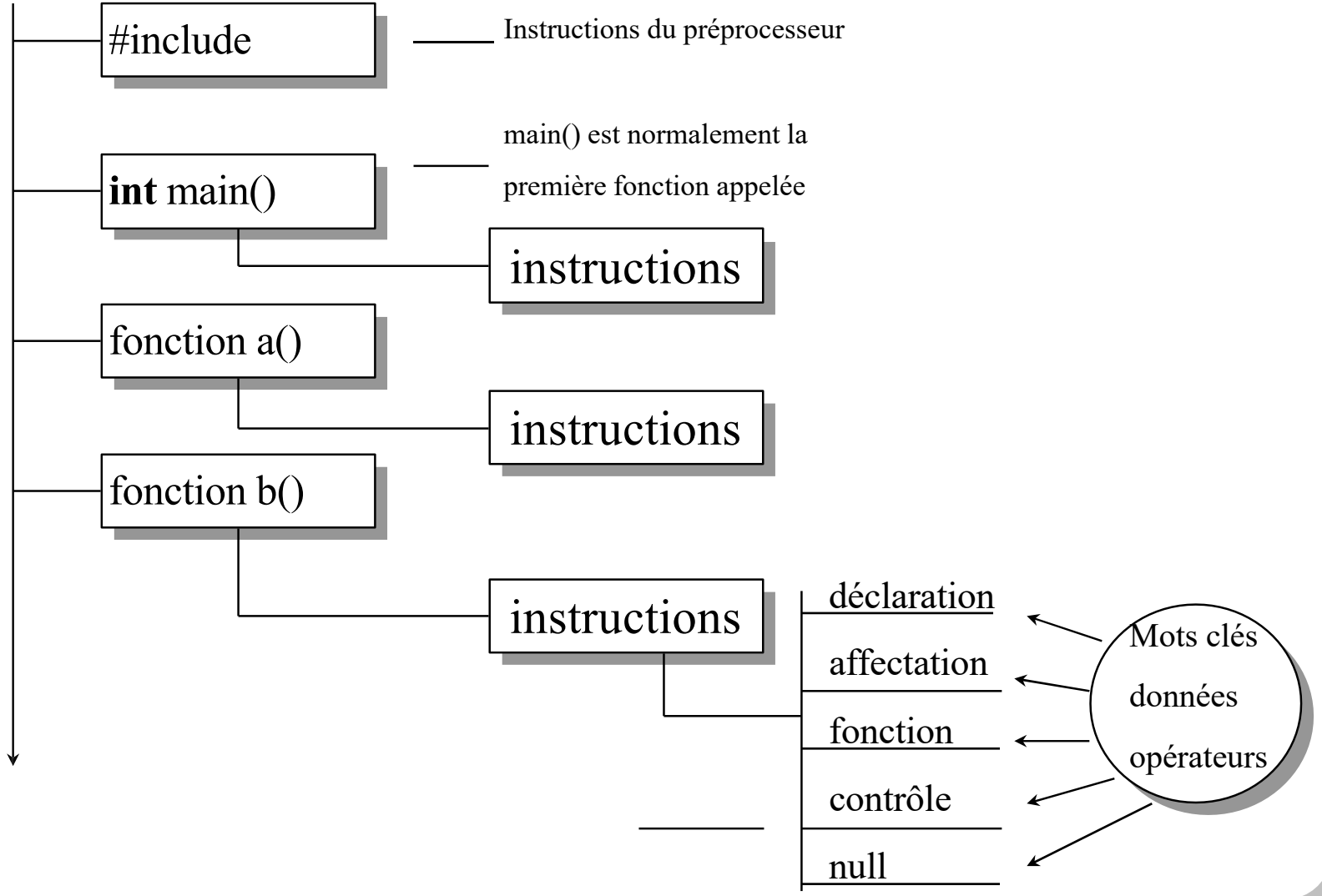
Structure d'un programme



Identificateur

Un identificateur est le nom d'une entité. Le nom d'une variable, d'une constante ou d'une fonction correspondent à des identificateurs.

Le nom est composé uniquement de caractères alphanumériques et du caractère de soulignement, ne débutant pas par un nombre.

Les lettres minuscules et majuscules sont différenciées. Ainsi TP1, Tp1, tP1, tp1 sont quatre identificateurs distincts.

Les déclarations

- constantes
- variables
- définitions de types
- prototypes de sous-programmes
- macros

```
const int MAX = 15;
int age = 10;
typedef float Reel;
struct Point { Reel x,y; };
int sommer(int,int);
#define IFDEBUG(x) x
```

Instruction simple

Chaque ligne, terminée par un point-virgule :

```
int nbJours = 30;

cout << "Donner deux valeurs entières ";

int nombre1, nombre2;

cin >> nombre1 >> nombre2;

cout << nbJours << nombre1 << nombre2;
```

Instruction composée

Bloc d'instructions simples entre accolades.
Les variables d'un bloc n'existent plus après le bloc.

```
{
    int numero = 12345;
    string alliage = "aluminium";
    double enStock = 3.45e12;
    cout << numero << alliage << enStock;
}
// numero, alliage et enStock n'existent plus.
{
    cout << "Donner votre numéro de code";
    int numCode;
    cin  >> numCode;
    cout << numCode;
}
// numCode n'existe plus.
```

VARIABLE

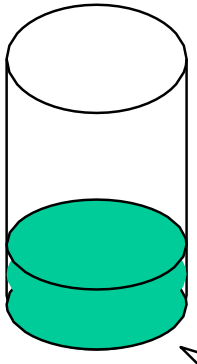
Il s'agit d'une donnée située à un emplacement mémoire dont la valeur peut être modifiée.

TYPE

Le type correspond à la sorte de variable;
Le type précise pour la variable:

- l'espace mémoire occupé;
- la représentation mémoire: la façon dont elle est mémorisée;
- les opérations admissibles.

Une variable peut être comparée à ce verre. On peut ajouter et retirer du liquide selon sa bonne volonté.



Analogie:

Verre	Variable
- Contenant	- Adresse mémoire
- Contenu	- Valeur

Le type indique :
- la grandeur du verre,
- le liquide admis,
- la façon que ce liquide doit être manipulé.

LES TYPES ENTIERS

Type	Étendue	Espace
int	-2 147 483 648 ... 2 147 483 647	4 octets
long (int)	-2 147 483 648 ... 2 147483 647	4 octets
long long (int)	-9 223 372 036 854 775 808 … 9 223 372 036 854 775 807	8 octets
short (int)	-32 768 ... 32 767	2 octets
unsigned short	0 ... 65 535	2 octets
unsigned long	0 ... 4 294 967 295	4 octets
unsigned long long	0 ... 18 446 744 073 709 551 615	8 octets

Note: ces valeurs peuvent varier selon l'ordinateur ou le compilateur.

Opérateurs arithmétiques
+, -, *, /, %

Opérateurs / et % `(a/b)*b + a%b` est égal à `a`

Opérateur	Opération	Expression	Résultat
/	Quotient (division)	`11 / 4`	2
	(l'arrondi est vers zéro)	`-11 / 4`	-2 (en C++11)
%	Reste (« modulo »)	`11 % 4`	3
	(même signe que le numérateur)	`-11 % 4`	-3 (en C++11)

Opérateurs relationnels
`==, <, >, <=, >=, !=`

LES TYPES RÉELS

Type	Étendue	Espace
float	1.17549×10^{-38}...3.40282×10^{38}	4 octets
double	$2.225\,074 \times 10^{-308}$...$1.797\,694 \times 10^{308}$	8 octets
long double	$3.362\,10 \times 10^{-4932}$...$1.189\,73 \times 10^{4932}$	10 octets

Note: ces valeurs peuvent varier selon l'ordinateur ou le compilateur. Particulièrement le « long double ».

Opérateurs arithmétiques

`+, -, *, /`

La division entre des opérandes réels donne un résultat réel

➔ `5.0 / 2.0` ou `5 / 2.0` ou `5.0 / 2` donne 2.5

Opérateurs relationnels

`==, <, >, <=, >=, !=`

Fonctions sur les entiers et réels

Fonction	Description	Exemple
double cos(double)	Le cosinus de l'argument en radian	cos(-5.2)=0.468516671
double sin(double)	Le sinus de l'argument en radian	sin(0.5)=0.479425538
double exp(double)	e^x e=2,718 281…	exp(4.0)=54.59815003
double pow(double,double)	Met un argument à une puissance donnée	pow(4.0, 2.0)=16
double log(double)	Logarithme naturel (ln x)	log(12.2) = 2.5014359

⇨ Définies dans le fichier **cmath**

Fonctions sur les entiers et les réels

FONCTIONS	DESCRIPTION	EXEMPLE
double log10 (double)	Le logarithme en base 10 de l'argument	log10(12.2)=1.08635 ..
double sqrt (double)	La racine carrée de l'argument	sqrt(16)=4.0
int abs (int)	La valeur absolue de l'argument entier	abs(-2)=2
double fabs (double)	La valeur absolue de l'argument réel	fabs(-12.3)=12.3
double ceil (double)	Arrondie au plus petit entier ≥ argument	ceil(9.2)=10.0
double floor (double)	Arrondie au plus grand entier ≤ argument	floor(9.2)=9.0

Le type booléen en C

- Il n'existe pas de type booléen proprement dit en langage C. Cependant ce langage considère FAUSSE toute variable, constante ou expression qui vaut 0 et VRAIE toute variable, constante ou expression différente de 0.

Opération	Opérateur
Négation	!
Conjonction (ET)	&&
Disjonction inclusive (OU)	||
Disjonction exclusive (OUE)	^

Le type booléen en C

- Une variable de type entier est souvent utilisée comme variable booléenne. Le contexte d'utilisation devrait préciser la nature booléenne de cette variable entière.

- Le résultat d'une expression booléenne est soit 0 (pour faux) ou 1 (pour vrai).

Le type booléen en C++

- Le C++ offre un nouveau type qui est un type booléen.
- L'identificateur de type est *bool*
- Les deux seules valeurs que peut prendre une variable de type *bool* sont *true* ou *false*.
- Il est préférable d'utiliser ce type booléen lorsqu'il s'agit d'une variable ne manipulant que deux valeurs.
- Augmente la compréhensibilité d'un programme.
- Le manipulateur d'E/S *boolalpha* (compilateur récent) permet d'afficher une chaîne correspondante à *true* ou *false*, autrement les valeurs 0 ou 1 sont affichées.

Le type caractère

- L'identificateur de type est *char*
- Un caractère est identifié en le plaçant entre apostrophes:
  - `char lettre = 'A';`
- Un caractère, déclaré `signed char` ou `unsigned char`, peut être manipulé comme une valeur entière de l'intervalle -128..127 ou 0..255.
- Il est possible d'utiliser les opérateurs relationnels sur les caractères. (`==`, `!=`, `<`, `>`, `<=`, `>=`)
- Les fonctions sur les expressions de type *char* sont définies dans le ficher **cctype**

Fonction	Description	Exemple
int isalpha(int argument)	Teste si l'argument est compris entre 'A' et 'Z ' ou 'a' ou 'z'	isalpha('d') vaut VRAI
int isdigit(int argument)	Teste si l'argument est compris entre '0' et '9'	isdigit('2') vaut VRAI
int isalnum(int argument)	Teste si l'argument est alphanumérique	isalnum('X') vaut VRAI
int islower(int argument)	Teste si l'argument est compris entre 'a' et 'z'	islower('a') vaut VRAI
int isupper(int argument)	Teste si l'argument est compris entre 'A' et 'Z'	isupper('T') vaut VRAI
int tolower(int argument)	Convertit un caractère en minuscule	tolower('S') vaut 's'
int toupper(int argument)	Convertit un caractère en majuscule	toupper('c') vaut 'C'

Les chaînes de caractères en C++

- Déclaration: *string laChaine;*
- Une chaîne de caractères est identifiée en la plaçant entre guillemets.
  - ➢string laChaine = "Que la vie est belle! ";
- L'accès à un caractère de la chaîne s'effectue en précisant sa position:
  - ➢*laChaine[2]* correspond au troisième caractère de la chaîne
  - ➢Le premier caractère d'une chaîne est à la position 0
- Il est possible d'utiliser les opérateurs relationnels sur les chaînes de caractères en se basant sur l'ordre alphabétique. (==, !=, <, >, <=, >=)
- Les fonctions agissant sur les chaînes de caractères sont dans le fichier **string**.

Fonctions sur les chaînes de caractères

Fonction	Description	Exemple
dest = source;	Copie la chaîne *source* dans la chaîne *dest*ination. La *source* peut aussi être une chaîne C.	chaine = "bonjour"; // chaine contient "bonjour"

dest += source;	Concatène la chaîne ou caractère *source* à la chaîne *dest*ination.
	La *source* peut aussi être une chaîne C.
		// chaine contient "bonjour"
		chaine += " les ami";
		chaine += 's';
		// chaine contient "bonjour les amis"

dest = source1 + source2;	Concaténation de deux chaînes, ou une chaîne et un caractère, le résultat est affecté à *dest*. Il doit y avoir une 'string' C++ au moins d'un côté de l'addition, l'autre peut être une chaîne C ou un caractère.	chaine = "bonjour"; chaine = chaine + " les ami" + 's'; // chaine contient "bonjour les amis"

⇨ Définies dans le fichier **string**

Fonctions sur les chaînes de caractères (suite)

Fonction	Description	Exemple
chaine.size() ou *chaine.length()*	Retourne la longueur de la *chaine*.	chaine = "bonjour les amis"; cout << chaine.size(); // affiche 16
chaine1 op *chaine2* (où op est un des opérateurs relationnels)	Compare *chaine1* à *chaine2* selon l'ordre alphabétique. **Attention:** les minuscules et les majuscules sont distinctes.	chaine1 contient "bonjour" chaine2 contient "bonsoir" chaine1 < chaine2 vaut true chaine2 < chaine1 vaut false chaine1 != chaine2 vaut true
chaine.find(autre)	Cherche la chaîne de caractères *autre* dans *chaine* et retourne sa position. Si *autre* n'est pas dans *chaine*, la valeur de retour est *chaine.npos*.	chaine = "Bon matin"; autre = "matin"; cout << chaine.find(autre); // affiche 4

⇨ Définies dans le fichier **string**

Fonctions sur les chaînes de caractères (suite)

Fonction	Description	Exemple

chaine.erase(pos, nbre);	Enlève *nbre* caractères de la chaîne, à partir de la position *pos*.	chaine = "bonjour les amis";
		chaine.erase(7, 4);
		// chaine contient "bonjour amis"

chaine.insert(pos, ajout);	Insère *ajout* dans *chaine*, à partir de la position *pos*.	chaine = "la lo lu";
		chaine.insert(3, "le li ");
		// chaine contient "la le li lo lu"

chaine.resize(longueur);	Redimensionne la chaine à la *longueur* spécifiée.
	La chaine peut être tronquée ou augmentée.
		chaine = "bon beigne";
		chaine.resize(7);
		// chaine contient "bon bei"

⇨ Définies dans le fichier **string**

Fonctions sur les chaînes de caractères (suite)

Fonction	Description	Exemple

chaine.substr(pos, long);	Retourne la sous-chaîne de longueur *long* à partir de la position *pos*.	chaine = "bonjour les amis";
		autre = chaine.substr(8, 3);
		// autre contient "les"

chaine.replace(pos, long, autre);	Remplace *long* caractères dans *chaine* par *autre*, à partir de la position *pos*;
	chaine peut être augmentée ou diminuée.
		chaine = "milou et tintin";
		chaine.replace(6, 2, "le chien de");
		// chaine est "milou le chien de tintin"

chaine.c_str();	Retourne *chaine* sous forme de chaîne de caractères C; la chaîne C est « const » (ne peut pas être modifiée).	char strC[40]; strcpy(strC, chaine.c_str());

⇨ Définies dans le fichier **string**

Les chaînes de caractères en C

- Déclaration: *char laChaine[25];*
- Manipulation: toute manipulation d'une chaîne de caractères doit être réalisée à l'aide d'une fonction. Ces fonctions sont définies dans le fichier <**cstring**>
- L'accès à un caractère de la chaîne s'effectue en précisant sa position:
  - ➢ *laChaine[2]* correspond au troisième caractère de la chaîne
- La terminaison du contenu inscrit dans la chaîne est marquée par le caractère «\0».

Fonctions sur les chaînes de caractères

Fonction	Description	Exemple
strcpy(destination, source)	Copie la chaîne *source* dans la chaine *destination*.	strcpy(chaine, "bonjour"); chaine contient "bonjour"
strncpy(dest, source, nbcar)	Copie *nbcar* de la chaîne *source* à la chaine *dest.*	strncpy(chaine, "bonjour", 3); chaine contient "bon"
strcat(dest, source)	Concatène la chaîne *source* à la chaîne *dest.*	chaine contient "bonjour" strcat(chaine, " les amis"); chaine contient "bonjour les amis"
strncat(dest, source, nbcar)	Concaténe *nbcar* de la chaîne *source* à la chaîne *dest*.	strncat(chaine, " les amis", 4); chaine contient "bonjour les"
strcmp(chaine1, chaine2) *strncmp(chaine1, chaine2, nbcar)*	Compare *chaine1* à *chaine2* Compare *nbcar* de *chaine1* à *chaine2* Retourne: valeur<0 si *chaine1<chaine2* 0 si *chaine1=chaine2* valeur>0 si *chaine1>chaine2*	chaine1 contient "bonjour" chaine2 contient "bonsoir" strcmp(chaine1, chaine2) vaut un entier<0 strcmp(chaine2, chaine1) vaut un entier >0 strncmp(chaine1, chaine2, 3) vaut 0

Fonctions sur les chaînes de caractères (suite)

Fonction	Description	Exemple
strlen(chaine)	Retourne la longueur de la chaîne	strlen("bonjour les amis") retourne 16
strlwr(chaine)	Convertit une chaîne de caractères en caractères minuscules	chaine contient "Bon MaTin" strlwr(chaine); chaine contient "bon matin"
strupr(chaine)	Convertit une chaîne de caractères en caractères majuscules	strupr(chaine); chaine contient "BON MATIN"
strset(chaine, carac)	Initialise tous les caractères d'une chaîne à *carac*	chaine contient "carte postale" strset(chaine, 'x'); chaine contient "xxxxxxxxxxxxx"
strnset(chaine, carac, nbcar)	Initialise *nbcar* caractères d'une chaînes de caractères à *carac*	strnset(chaine, 'x', 3); chaine contient "xxxte postale"

⇨ Définies dans le fichier **cstring**

Le type pointeur

- Une variable pointeur correspond à une variable qui contient l'adresse mémoire d'une donnée.
- Déclaration: `int* ptrEntier;`
- L'opérateur & «adresse de» est utilisé pour connaître l'adresse d'une donnée.

```
int entier;
int* ptrEntier = &entier;
```

- L'identificateur d'une chaîne de caractères C contient l'adresse du premier caractère de la chaîne, il correspond donc à un pointeur.
- Attention aux multiples significations de * et &, selon qu'ils sont utilisés comme opérateur binaire, unaire ou comme type.

Le type pointeur

```
int main()
{
    int      ageAlex = 7, ageSandie = 27;
    double   x = 1.2345, y = 32.14;

    int* ptrEntier;
    ptrEntier = &ageAlex;
    *ptrEntier += ageSandie;
    cout << "ageAlex est " << ageAlex << endl;

    double* ptrReel;
    ptrReel = &x;
    y += 5 * (*ptrReel);
    cout << "y contient la valeur " << y << endl;
}
```

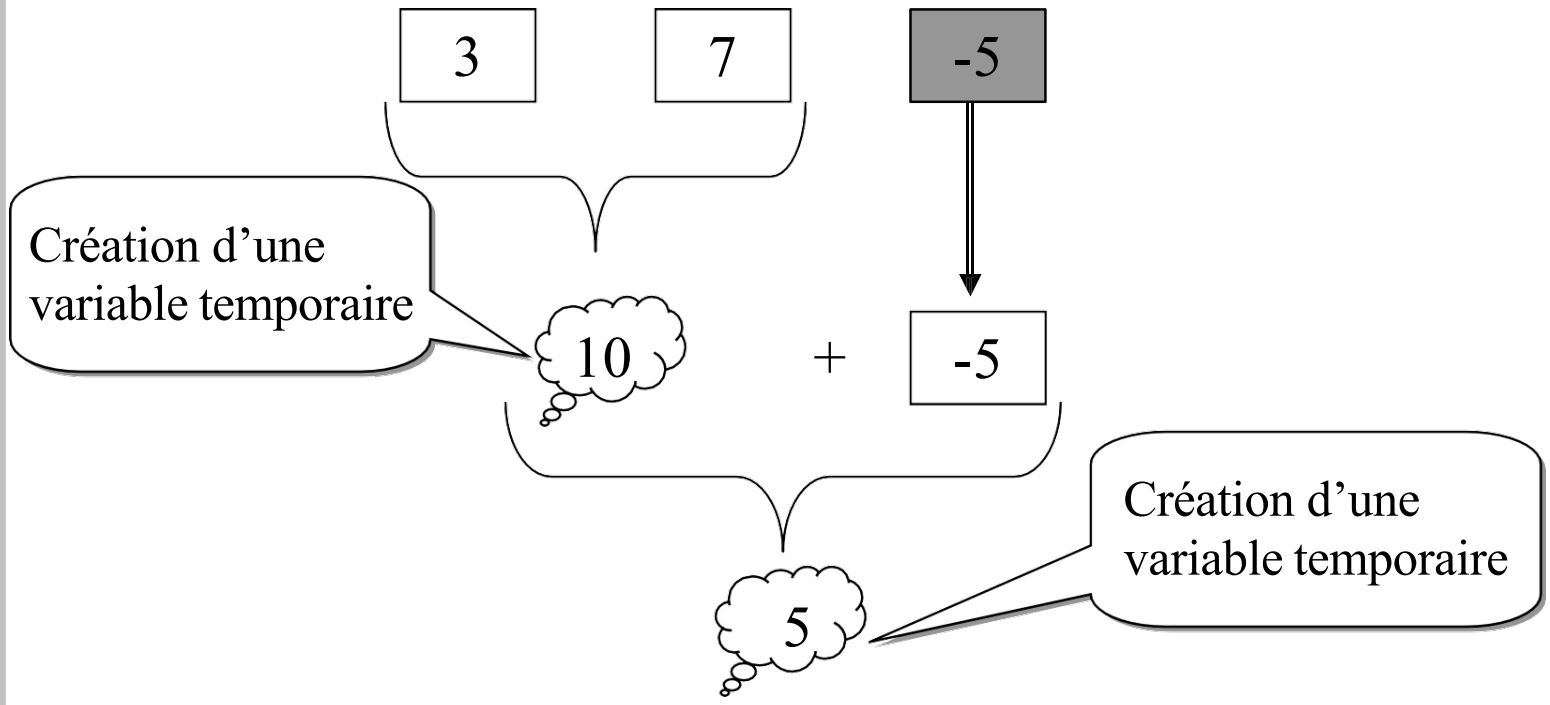
```
Résultat de l'exécution:
ageAlex est 34
y contient la valeur 38.3125
```

Opérateurs ++ et --

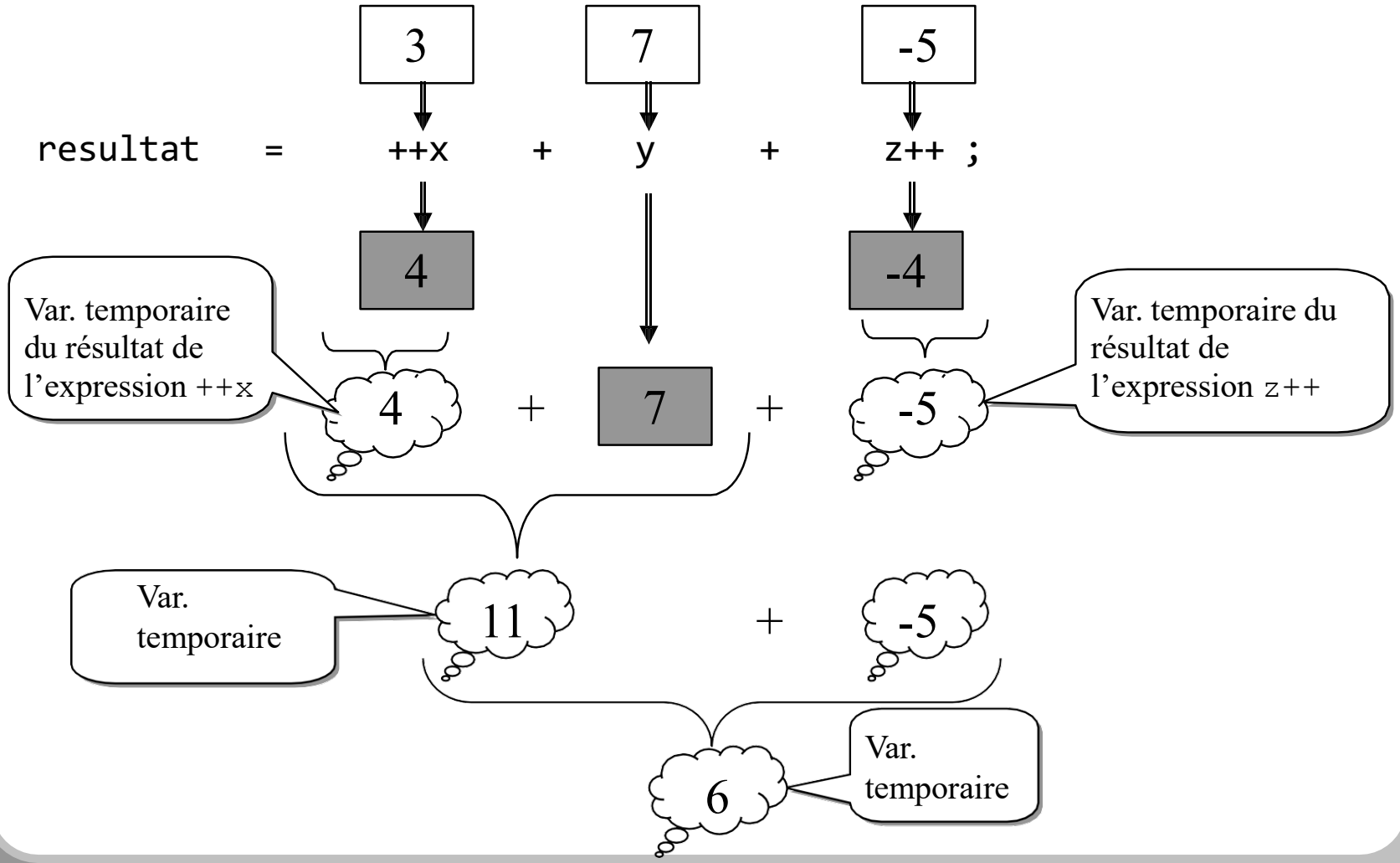
Opérateur	Opération	Exemple	Instruction équivalente
++	Ajouter 1 à la variable entière.	`++variable;`	`variable = variable+1;`
--	Enlever 1 à la variable entière	`--variable;`	`variable = variable-1;`

Exécution de l'évaluation d'une expression

```
resultat =  x   +   y   +   z;
```



Exécution de l'évaluation d'une expression



++ et -- en position préfixe (avant) ou postfixe (après)

Instruction	Instruction équivalente
`++i;` ou `i++;`	`i = i + 1;`
`--i;` ou `i--;`	`i = i - 1;`
`i = j++ + 4;`	`i = j + 4;` `j = j + 1;`
`i = ++j - 7;`	`j = j + 1;` `i = j - 7;`

`k = (--j) + (m++);`	`j = j - 1;`
	`k = j + m;`
	`m = m + 1;`

Traitement des bits

Opérateur	Description	Exemple	
~	Effectue le complément à 1 en inversant tous les bits	~12	~12 =~(00001100) = (11110011) = 243 ou -13
&	Réalise un ET binaire sur les bits: 1&1 donne 1 1&0 donne 0 0&1 donne 0 0&0 donne 0	7 & 2	7 00000111 & & 2 00000010 2 00000010
|	Réalise un OU binaire sur les bits: 1|1 donne 1 1|0 donne 1 0|1 donne 1 0|0 donne 0	7 | 3	7 00000111 | 3 00000011 7 00000111

Traitement des bits (suite)

Opérateur	Description	Exemple	
^	Réalise un OU-EXCLUSIF binaire sur les bits: 1^1 donne 0 1^0 donne 1 0^1 donne 1 0^0 donne 0	5 ^12	5 000000101 ^ 12 000001100 ___ _____ 9 000001001
>>	Réalise un décalage à droite du nombre de bits spécifié	7 >>2	7 >> 2 00000111 >> 2 _____ 00000001
<<	Réalise un décalage à gauche du nombre de bits spécifié	11 << 3	11 << 3 00001011 << 3 _____ 01011000

Priorité des opérateurs et leur associativité

Priorité	Opérateur	Associativité
1	`[] () -> . ++(postfixe) --(postfixe)`	gauche à droite
2	`! ~ ++(préfixe) --(préfixe) -(unaire) +(unaire) (`*type*`) *(unaire) &(unaire) sizeof`	droite à gauche
3	`* / %`	gauche à droite
4	`+ -`	gauche à droite
5	`<< >>`	gauche à droite
6	`< > <= >=`	gauche à droite
7	`== !=`	gauche à droite
8	`&`	gauche à droite
9	`^`	gauche à droite
10	`|`	gauche à droite
11	`&&`	gauche à droite
12	`||`	gauche à droite
13	`?: = += -= *= /= %= >>= <<= &= ^= |=`	droite à gauche
14	`,`	gauche à droite

Instruction et expression

- `nbJours = (mois=4)*30+(jour=14);`
  `(4) *30+ (14)`

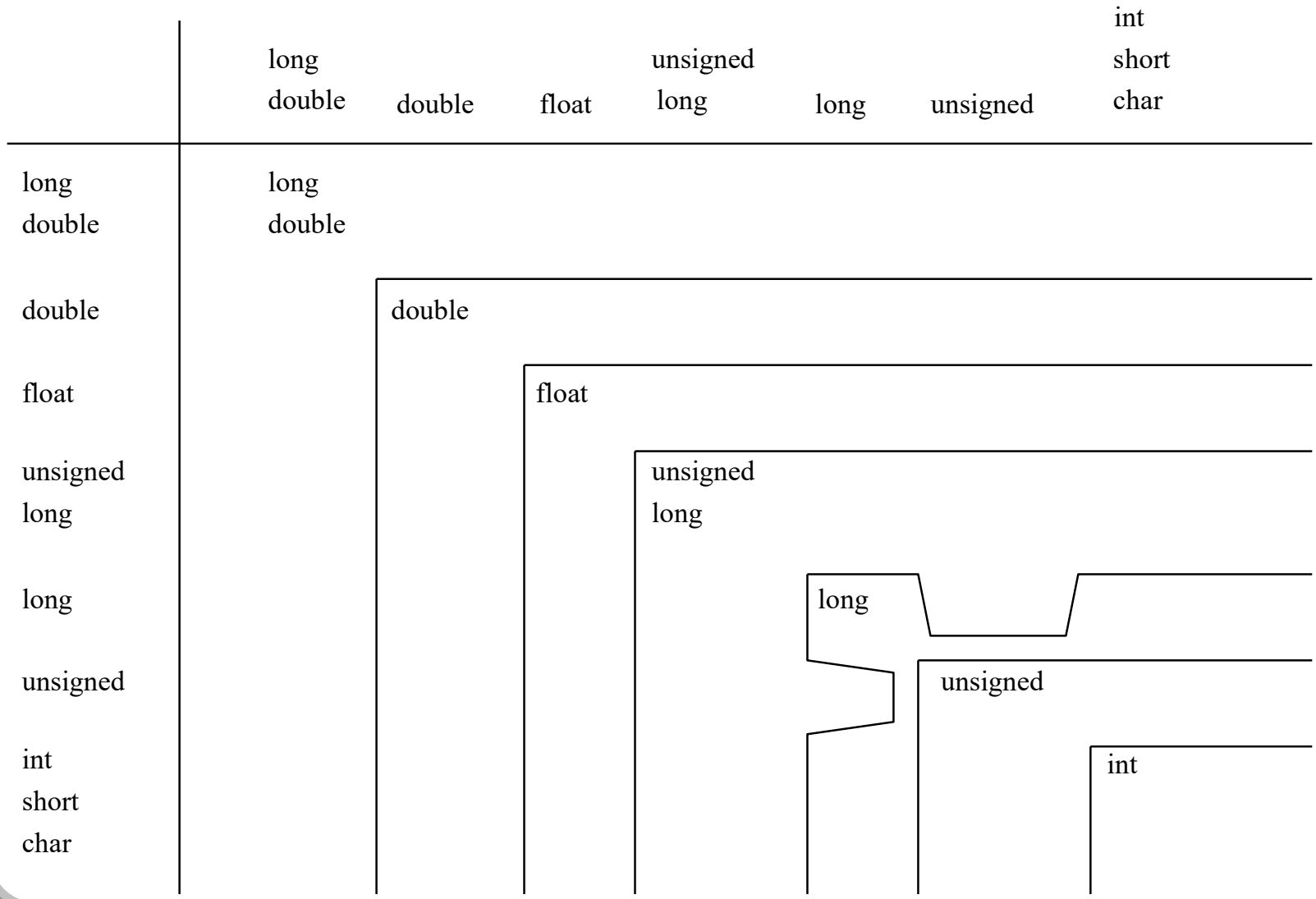
- `nbre1 = nbre2 = nbr3 = 0;`

  `nbre1 = (nbre2 = (nbre3 = 0));`

- `Valeur += 5;`

  `Valeur = Valeur + 5;`

Conversion implicite



Conversion implicite

Expression	Type	Expression	Type
c-s/i	int	u+c-i	unsigned
c-5	int	u+c-i-u_long	unsigned long
u/2.0-i	double	7*5*u_long	unsigned long
c + 1.24	double	long_d+c-24	long double
d*3	double	f+3*s-i	float
4+i/u_long	unsigned long	long_d-f*2	long double

Conversion explicite

A partir des déclarations, `float valeur;`
`char car;`

`S(long)('A' + 5.4)` L'addition donne un `float`, qui est ensuite converti en `long`.

```
Svaleur = float(int(car) + 1);
```

La variable car est convertie en `int`, l'expression `int(car) + 1` est de type `int`, elle est ensuite convertie explicitement en `float`.

Autre façon en C++

```
➢valeur = static_cast<float>(car + 1);
```

Conversion de type

```
int    a = 2;
double x = 17.1, y = 8.95;
char   c1,c2,c3,c4,c5,c6,c7,c8;
double z1,z2,z3,z4,z5,z6,z7,z8;

c1 = (char)a + (char)x;
c2 = (char)(a + (int)x);
c3 = (char)(a + x);
c4 = a + x;

z1 = (double)(int)x * (int)y;
z2 = (double)((int)x * (int)y);
z3 = (double)((int)(x * y));
z4 = x * y;

c5 = char(a) + char(x);
c6 = char(a + int(x));
c7 = char(a + x);
c8 = a + x;

z5 = double(int(x)) * int(y);
z6 = double(int(x) * int(y));
z7 = double(int(x * y));
z8 = x * y;
```

syntaxe C

une des syntaxes C++