C++ et C# - Similitudes et ressemblances

420-V31-SF – Programmation de Jeux Vidéo III

fparadis@cegep-ste-foy.qc.ca

420-V31-SF – Programmation de Jeux Vidéo III

Le C++: Variables et constantes

fparadis@cegep-ste-foy.qc.ca

Types de variables

Variables entières:

Type	Plage	Taille
char	-128 à 127	Entier 8 bits signé
unsigned char	0 à 255	Entier 8 bits non signé
short	-32768 à 32767	Entier 16 bits signé
unsigned short	0 à 65 535	Entier 16 bits non signé
int ou long	-2147483648 à 2147483647	Entier 32 bits signé
unsigned int ou unsigned long	0 à 4294967295	Entier 32 bits non signé

Types de variables

Variables en virgule flottante:

Type	Plage approximative	Taille	Précision
float	$\pm 3.4 \times 10^{-38}$ à $\pm 3.4 \times 10^{38}$	32 bits	7 chiffres
double	$\pm 1.7 \times 10^{-308}$ à $\pm 1.7 \times 10^{308}$	64 bits	15 chiffres

Valeurs littérales:

Notation hexadécimale : *précédée de 0x ou 0X*

Notation scientifique :

par exemple, x = 9.3E4; au lieu de x = 93000;

Types de variables

Variables en virgule flottante:

Type	Plage approximative	Taille	Précision
float	$\pm 3.4 \times 10^{-38}$ à $\pm 3.4 \times 10^{38}$	32 bits	7 chiffres
double	$\pm 1.7 \times 10^{-308}$ à $\pm 1.7 \times 10^{308}$	64 bits	15 chiffres

Variables logiques:

Type	Plage	Taille
bool	true ou false	8 bits

Constantes caractères

Pour initialiser une variable de type caractère, on peut utiliser le code ASCII ou les constantes suivantes :

Constante	Signification
\'	'
\"	"
\\	\
\?	?
\0	Null
\a	Bip
\b	Backspace
\f	Saut de page
\n	Saut de ligne
\r	Retour de chariot
\t	Tabulation horizontale
\v	Tabulation verticale

Déclaration de variable

Pour définir une variable, il faut spécifier le type suivi du nom de variable.

```
int x;
```

On peut initialiser une variable à la déclaration

```
int x = 25;
```

On peut initialiser une variable après la déclaration

```
int x;
x = 25;
```

On peut déclarer plusieurs variables de même type en une seule instruction

```
int x, y;
```

Bien sûr, on peut déclarer et initialiser plusieurs variables de même type en une seule instruction

```
int x=0, y=1;
```

Note : Une variable doit être initialisée avant d'être utilisée (PAS D'INITIALISATION PAR DÉFAUT: votre variable sera utilisable mais contiendra du "junk")

Déclaration de constante

Pour définir une constante avec un nom pour plus de clarté, il faut ajouter le mot clé const avant la définition

```
const int MAX_ETUDIANT = 25;
```

Il vous est possible de rencontrer des valeurs de pré-compilation dans du code (const n'existe pas en C)

```
#define MAX_ETUDIANT 25
```

Avant la compilation, le compilateur va remplacer toutes les instances de MAX_ETUDIANT par 25 (Ça reste peu recommandé)

Portée d'une variable

Comme en C#, une variable est accessible pour le bloc où elle est définie.

Un bloc est délimité par des accolades (fonction, if, for, while, do-while).

Nous y reviendrons plus en détail lors des structures algorithmiques.

Par exemple :

```
if (total > 0)
{
    int deduction;
    // suite des instructions
}
```

La variable « deduction » n'est connue que dans le « if » et sera supprimée à la fin du « if »

420-V31-SF – Programmation de Jeux Vidéo III

Structures algorithmiques en C++

fparadis@cegep-ste-foy.qc.ca

Structures algorithmiques en C++

If, Switch, While, Do While, For, Break et Continue fonctionnent exactement comme en C#

Comme en C#, la portée doit toujours être la plus restreinte possible. Puisqu'on peut déclarer les variables à mesure qu'on en a besoin, il faut en profiter (permet de limiter les erreurs).

On garde la même standardisation de nomenclature qu'en C#

420-V31-SF – Programmation de Jeux Vidéo III

Opérateurs et priorités en C++

fparadis@cegep-ste-foy.qc.ca

Cardinalité des opérateurs

Unaire :

x = -3 ;

Binaire :

x = y – 3 ;

Ternaire :

// x prend la valeur 25 si a > b ou 50 sinon

x = (a > b ? 25 : 50) ;

Opérateurs arithmétiques

Opérateur	Signification
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo (reste de division entière)

Note 1 Si une division s'effectue sur deux nombres entiers, le résultat est l'entier tronqué (par exemple, 17 / 3 donne 5)

Note 2 Il n'existe pas d'opérateur pour l'exposant. Il faut utiliser la fonction pow dans la librairie <cmath>.

Opérateurs relationnels

Opérateur	Signification
= =	Égal
!=	Différent
<	Plus petit
<=	Plus petit ou égal
>	Plus grand
>=	Plus grand ou égal

Note 1 Le résultat d'une opération relationnelle retourne un booléen

Note 2 Le double égal (==) sert à tester l'égalité tandis que le symbole = sert pour l'affectation

Opérateurs logiques

Opérateur	Signification
&&	ET
||	OU
!	NON

Les tables de vérité suivantes s'appliquent pour ces opérateurs

x	y	x || y	x && y
False	False	False	False
False	True	True	False
True	False	True	False
True	True	True	True

x	!x
False	True
True	False

Note : Une évaluation arrête lorsque le résultat est trouvé. Par exemple, si un premier terme est vrai, l'opération || se terminera, car le résultat sera nécessairement vrai.

Opérateurs sur les bits

Opérateur	Signification
&	ET
|	OU
~	NON
^	OU exclusif
<<	Décalage à gauche
>>	Décalage à droite

Note : La table de vérité des opérateurs &, | et ~ est la même que les opérateurs &&, || et ! respectivement, mais ils s'appliquent sur les bits.

Opérateurs sur les bits

La table de vérité du OU exclusif (^) est la suivante :

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

Par exemple:

```
x = 3;
y = 5;
z = x | y; // z prend la valeur 7
z = x & y; // z prend la valeur 1
z = x ^ y; // z prend la valeur 6
z = x << 1; // z prend la valeur 6 (comme une multiplication par 2)
z = y >> 1; // z prend la valeur 2 (comme une division entière par 2)
```

Affectations

En plus de l'affectation standard (=), on peut effectuer une opération en affectant automatiquement le résultat. Les opérateurs suivants sont disponibles :
+=, -=, *=, /=, %=, <<=, >>=, &=, ^=, |=

Par exemple :

```
x += 3 ; // remplace x = x + 3 ;
```

Affectations

De plus, l'opérateur ++ remplace l'incrémentation de 1 et l'opérateur – remplace la décrémentation de 1.

Par exemple :

```
x++ ; // remplace x = x + 1 ;
y-- ;  // remplace y = y - 1;
```

Il existe 2 types :

PRÉ : effectue d'abord l'opération

POST : termine l'énoncé par l'opération

Par exemple :

```
x = 1 ;
y = ++x;
// Ici, x est incrémenté à 2 et affecté à y alors que,
y = x++;
// x est d'abord affecté à y (valeur 1) puis incrémenté à 2
```

Expressions

À l'exception des opérateurs d'assignation, tous les opérateurs binaires sont associatifs à gauche, (effectuées de gauche à droite).
x + y + z, est évalué en tant que (x + y) + z.

Les opérateurs d'assignation et les opérateurs conditionnels (?:) sont associatifs à droite (effectuées de droite à gauche).
x = y = z, est évalué en tant que x = (y = z).

La priorité et l'associativité peuvent être contrôlées par des parenthèses.
x + y * z vs. (x + y) * z