420-V31-SF – Programmation de Jeux Vidéo III

Les tableaux

fparadis@cegep-ste-foy.qc.ca

Les tableaux unidimensionnels

Voici 2 exemples de définition de tableau.

```
// Un tableau de int
int tableau[tailleDuTableau];

// Un tableau de 5 objets de type Compte
Compte tableau2[5];
```

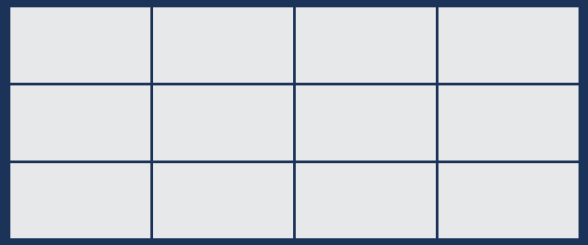
Pour accéder à un item du tableau:

```
// On assigne la valeur 18 au 3e item du tableau
tableau[2] = 18;
```

Tableaux multidimensionnels

Les tableaux multidimensionnels sont des tableaux qui contiennent des tableaux.

Par exemple un tableau bidimensionnel (3 lignes, 4 colonnes) :



équivaut en mémoire à 3 tableaux de 4 items comme suit:

Tableaux multidimensionnels

Un tableau bidimensionnel se définit de la manière suivante :

```
int nomTableau[nbLignes][nbColonnes];
int tableau[3][4];
```

On peut représenter le 2e tableau de l'exemple de la manière suivante :

Tableau[0][0]	Tableau[0][1]	Tableau[0][2]	Tableau[0][3]
Tableau[1][0]	Tableau[1][1]	Tableau[1][2]	Tableau[1][3]
Tableau[2][0]	Tableau[2][1]	Tableau[2][2]	Tableau[2][3]

Tableaux multidimensionnels

Référence à un objet du tableau :

```
tableau[0][1] = 15;
```

Initialiser un tableau :

```
int tableau[2][4] = {{0, 1, 2, 3}, {4, 5, 6, 7}};
```

Initialiser le tableau par programmation :

```
for (int ligne = 0; ligne < NB_LIGNES; ligne++)
{
  for (int colonne = 0; colonne < NB_COL; colonne++)
  {
      tableau[ligne][colonne] = 0;
  }
}
```

Passage de tableau en paramètre

```
// On ne doit laisser qu'une seule dimension inconnue…
void InitialiserTableau(int tableau[][4])
{
     // [...]
}

int main()
{
     int tableau[3][4];

     InitialiserTableau(tableau);

     return 0;
}
```

Notes : La fonction *InitialiserTableau* voit la variable comme un tableau de tableaux à 4 éléments.

On pourrait passer un tableau avec plus d'une dimension inconnue, mais nous aurons besoin des pointeurs pour cela

Structure en mémoire des tableaux

En C++, un tableau n'est nullement un objet complet, il s'agit tout simplement de variables du même type regroupées au même endroit.

Ex: créez et initialisez un tableau de 10 int et parcourez-en 12. Que ce passe-t-il?

Créez et initialisez un tableau à deux dimensions de 2x10 et faites un for i < 25 et affichez le contenu de [0][i]. Que ce passe-t-il?

420-V31-SF – Programmation de Jeux Vidéo III

Mot clé static

fparadis@cegep-ste-foy.qc.ca

Mot-clé statique

À la base exactement la même utilisation qu'en C#

- Pouvez-vous me le rappeler?
- Qu'est-ce qu'une classe "de service"
- Que faudrait-il faire avec le constructeur de ce genre de classe?

Mais en C++ il y a une utilisation supplémentaire de static…

Mot-clé static

Dans une fonction:

```
static int compteur = 0;
```

La déclaration se fera une seule fois.

Si on entre à nouveau dans la fonction, l'état de cette variable sera identique au moment où on l'a quitté.

```
int testStatique(int parametre)
{
  • static int test = 0; //Initialisé une et une seule fois au début du programme
  • test += parametre;
  • return test;  //Si au premier appel de la fonction, param vaut 3
}                 //alors au second appel de cette fonction, test vaudra 3
```

Mot-clé static

Dans les faits, une variable static à une fonction est une variable globale, mais dont la portée est limité à l'endroit où elle a été déclarée.

C'est une pratique considérée désuète. Peut-être pratique, mais ne l'utilisez pas si vous trouvez une meilleure solution

420-V31-SF – Programmation de Jeux Vidéo III

Mini-intro aux pointeurs

fparadis@cegep-ste-foy.qc.ca

Les Pointeurs

- Variable contenant une adresse
  (adresse d'une variable, fonction, objet…)
- Référence sur un élément
- Peut pointer sur un élément déjà existant OU sur un tout nouvel élément crée sur le tas (heap) avec le mot-clé "new"

Les Pointeurs

```
int x;
int *p; // déclaration d'un pointeur

p = &x ; // &x renvoie l'adresse de x
         // sur le pointeur directement, on
         //manipule des adresses

*p = 1 ; // équivalent à x = 1, avec le * derrière on
         // manipule le pointeur comme une variable normale

cout << p << *p ; // on affiche l'adresse de x, puis "1"
```

Utilité des pointeurs

- On peut se servir d'un pointeur pour créer un tableau
- Plus facile de passer le tableau en paramètre ainsi
- On peut définir par exemple

```
int *tableau;
// ou
int tableau[];
```

Utilité des pointeurs

- Lorsqu'on ne connaît pas la quantité de mémoire requise au début d'un programme

- Allocation en cours d'exécution et non à la compilation

- Création d'un bloc de mémoire : instruction new (retourne un pointeur sur le bloc)

- Le bloc est stocké dans une zone mémoire nommée HEAP (Tas en français)

Allocation dynamique

- La fonction new retourne nullptr (null) si manque de mémoire

```
int * ptr;
ptr = new int(7);

// Pour un bloc de 10 entiers
ptr = new int [10];
```

- Suppression d'un bloc : instruction delete

```
delete ptr; // type simple
delete [] ptr; // tableau
```

Attention: new

Contrairement à C# où vous pouvez faire new sur pratiquement n'importe quoi. Ce n'est pas le cas en C++

Si vous n'avez pas affaire à un pointeur, mais à une simple instance, new ne peut pas être utilisé.

Si vous avez une classe CollisionBox et que vous faites
`CollisionBox cb;`
cb a appelé son constructeur par défaut et est déjà prêt à être utilisé; new ne fonctionnera pas et nous aurez une erreur de compilation.

Il ne faut pas mettre de parenthèses si un constructeur par défaut existe et qu'on veut l'utiliser.

Par contre `CollisionBox * cb = new CollisionBox();` est légal;

nullptr

NULL en C++ est nullptr

Vous verrez parfois NULL ou 0 pour mettre un pointeur à null; c'est du relent du C. Utilisez nullptr

Bien important de faire un delete avant de sortir d'un méthode qui utilise un pointeur ou d'effacer les pointeurs utilisés par une instance de classe avant sa destruction, sinon vous provoquerez une fuite de mémoire (memory leak).

Destructeur

```
MyClass::~MyClass()
{
   delete monPointeur
   delete[] monTableau()
}
```

Tous les pointeurs alloués doivent être effacés explicitement.

420-V31-SF – Programmation de Jeux Vidéo III

Ce qu'il faut effacer lors d'une remise

fparadis@cegep-ste-foy.qc.ca

À effacer

1- Le répertoire .vs

2- Tous les répertoires Debug et Release

3- Le fichier [Votre projet].VC.bd