

Programmation pour le jeu vidéo

CARLOS CRISPIM-JUNIOR, DR. ENG.

GAMAGORA 2020-2021

1. Premier programme
2. Variables
3. Opérateur
4. Fonctions/méthodes
5. Structures conditionnelles
6. Structures de contrôle
7. Conventions et syntaxe

Premier programme

```
1  using System;
2
3  namespace MonPremierProgramme
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Hello World!");
10             Console.Read();
11         }
12     }
13 }
14
15
```

Différences :

- ❖ Les accolades { }
- ❖ Le caractère de terminaison de ligne (;)
- ❖ Le mot-clé "class"
- ❖ Le mot-clé "namespace"
- ❖ La méthode "static void Main(...)"
- ❖ Using System;
- ❖ L'indentation

- Variables = cases (ou blocs) mémoires utilisées pour stocker des informations, comme des résultats intermédiaires des calculs.

- Déclaration sans initialisation :

```
type nomVariable ;
```

- Déclaration avec initialisation

```
type nomVariable = valeurParDefault;
```

Variables

■ Types variables :

Type	Description
byte	Entier de 0 à 255
short	Entier de -32768 à 32767
int	Entier de -2147483648 à 2147483647
long	Entier de -9223372036854775808 à 9223372036854775807
float	Nombre simple précision de -3,402823e38 à 3,402823e38
double	Nombre double précision de -1,79769313486232e308 à 1,79769313486232e308
decimal	Nombre décimal convenant particulièrement aux calculs financiers (en raison de ses nom significatifs après la virgule)
char	Représente un caractère
string	Une chaîne de caractère
bool	Une valeur booléenne (vrai ou faux)

<https://docs.microsoft.com/fr-fr/dotnet/csharp/programming-guide/types/index>

Exemples :

```
int age = 30;           // opération ?
```

```
age = 20;               // opération ?
```

```
Console.WriteLine(age); // affiche ?
```

```
age = 50;               // opération ?
```

```
Console.WriteLine(age); // affiche ?
```

Exemples :

```
int age = 30;           // initialisation
```

```
age = 20;               // affectation
```

```
Console.WriteLine(age); // affiche 20
```

```
age = 50;               //affectation
```

```
Console.WriteLine(age); // affiche 50
```

Définition : bloc de code avec un but spécifique .

```
type nomFonction (paramètres) {
    ...
    return « résultat »
}
```

- Instruction « return » : renvoie le résultat calculé par la fonction
- **type** : type du résultat renvoyé par la fonction
- paramètres : liste des arguments/ entrées de la fonction.

Opérateurs arithmétiques

Description	Opérateur
Somme	+
Soustraction	-
Multiplication	*
Division	/
Modulo	%
Incrémenter par 1	++
Décrémenter par 1	--
Incrémenter par i	+= i
Décrémenter par 1	-= i

- Exemple 1 : un argument, un résultat

```
public static int carreNombre(int valeur) {  
  
    return valeur*valeur;  
  
}
```

- Exemple 2 : un argument, pas de résultat

```
public static void carreNombre(int valeur) {  
  
    Console.WriteLine (valeur*valeur);  
  
}
```

- Exemple 3 : plusieurs arguments, un résultat

```
public static float moyenne_trois_nb(float a,  
    float b, float c) {  
  
    return (a + b + c) / 3;  
  
}
```

Fonctions - Portée d'une variable

Définition : la portée (ou « scope ») d'une variable est le contexte dans lequel elle peut être utilisée.

Portée de la classe / globale

```
public class PorteeVariable : MonoBehaviour  
{  
    private int alpha = 2;
```

Portée d'une fonction / locale

```
void monExemple (int stylo, int couleur){  
    int reponse = 0;  
    reponse = stylo * couleur * alpha;  
}
```

Fonctions - Portée d'une variable


```
namespace CM_Jeu_Video
{
    class Program
    {
        public static void carreNombre(int valeur)
        {
            int carre = valeur * valeur;
        }

        static void Main(string[] args)
        {
            carreNombre(5);
            Console.WriteLine("Le carre de 5 est : " + carre);
        }
    }
}
```

Fonctions - Portée d'une variable

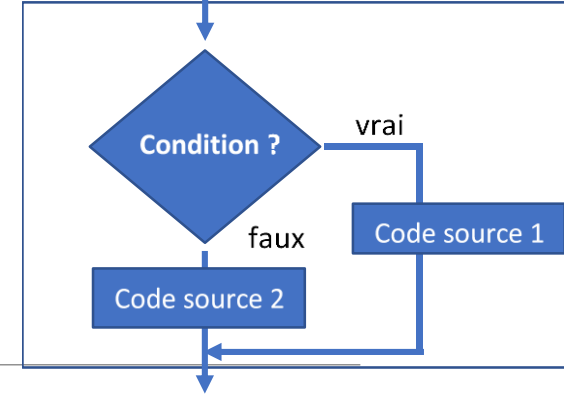
```
namespace CM_Jeu_Video
{
    class Program
    {
        public static int carreNombre(int valeur)
        {
            int carre = valeur * valeur;
            return carre;
        }

        static void Main(string[] args)
        {
            int resultat = carreNombre(5);
            Console.WriteLine("Le carre de 5 est : " + resultat);
        }
    }
}
```



Pause Visual Studio

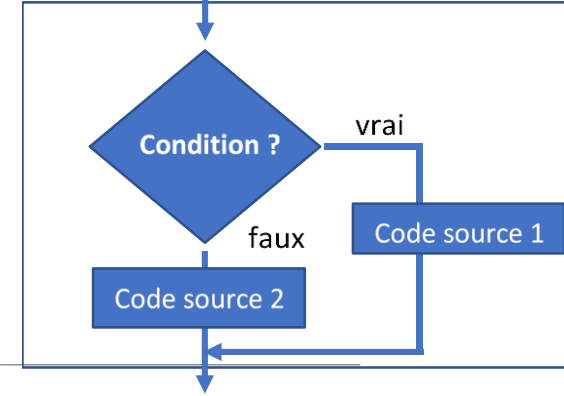
Structures conditionnelles



Définition : il s'agit de ce que l'on appelle des tests conditionnels. Elles sont évaluées lors de l'exécution d'un programme et en fonction de son résultat (vrai ou faux), nous allons exécuter une ou plusieurs instructions.

Opération	Opérateur	Exemples			
Égalité	==	2 == 2	true	vrai	1
Différence	!=	2 != 2	false	faux	0
Supérieur ou égal	>=	2 >= 3	false	faux	0
Supérieur ou égal	>	6 > 2	true	vrai	1
Inférieur ou égal	<=	5 <= 5	true	vrai	1
Inférieur	<	5 < 4	false	faux	0
Et logique	&&	true && true	true	vrai	1
		true && false	false	faux	0
		false && false	false	faux	0
Ou logique		true false	true	vrai	1
		false false	false	faux	0
Négation	!	true	false	faux	0
		false	true	vrai	1

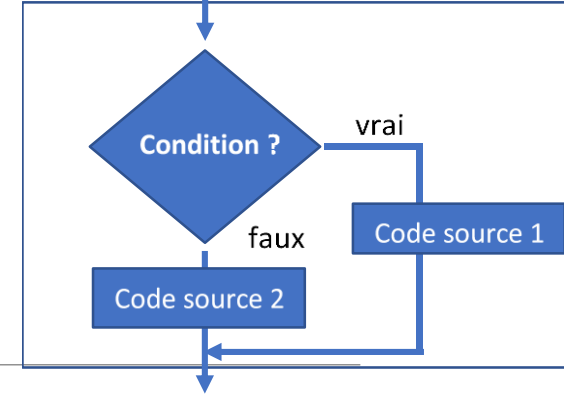
Structures conditionnelles



Exemple 1 : une condition

```
//bool personnageVivant = <expression>;  
  
if ( personnageVivant ) {  
  
    // ... mon code  
  
}
```

Structures conditionnelles

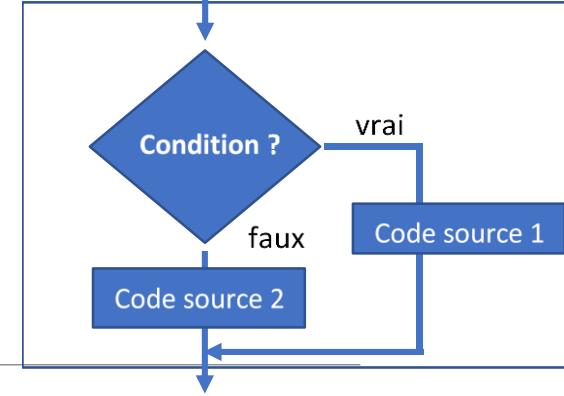


Exemple 2 : plusieurs conditions

```
//bool personnageVivant = <expression>;  
//int score = <expression>;
```

```
if ( personnageVivant && score > 0) {  
  
    // ... mon code  
  
}
```

Structures conditionnelles



Exemple 3 : instructions **if-else** (si .. sinon ..)

```
bool personnageVivant = <expression>;  
//int score = <expression>;
```

```
if ( personnageVivant && score > 0) {
```

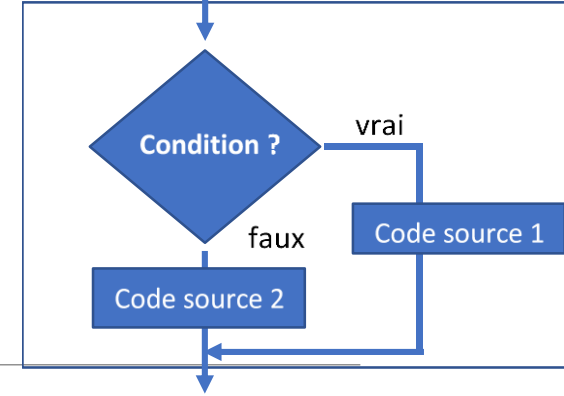
```
    // ... code source #1
```

```
} else {
```

```
    // ... code source #2
```

```
}
```

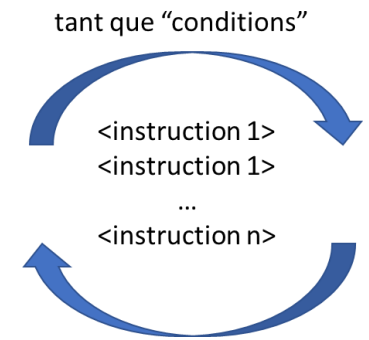
Structures conditionnelles



Exemple 4 : utilisation d'**else if** (sinon .. si ..) :

```
bool personnageVivant = <expression>;  
  
if ( personnageVivant && score > 0) {  
    // ... code source #1  
}  
else if (personnageVivant) {  
    // ... code #2  
}  
else {  
    // ... code source #3  
}
```

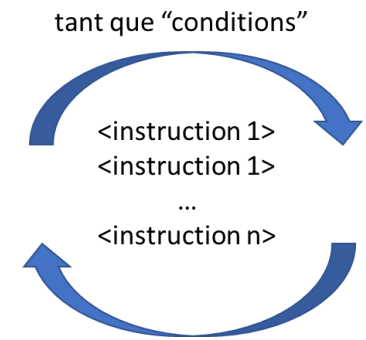
Structures de contrôle



Définition : les boucles sont des structures de contrôle de type itératif qui permettent de répéter un bloc de code tant que la condition de fin n'est pas satisfaite.

Opération	Opérateur	Utilisation	Valeur
Incrément	++	a++ ;	6
Décrément	--	a-- ;	5
Incrémenter par	+=	a+=2;	7
Décrémenter par	-=	a-=3;	4

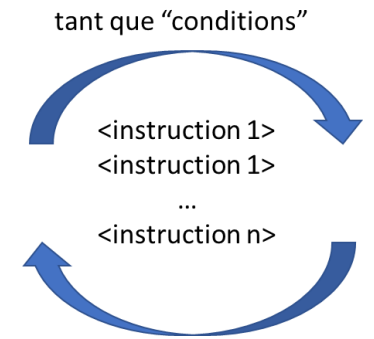
Structures de contrôle



Instruction « do » : s'exécute au moins une fois et jusqu'à ce que l'expression spécifiée soit fausse.

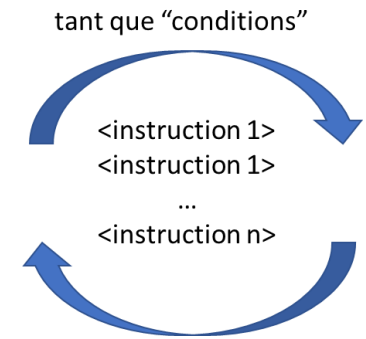
```
do {  
  
    //... code du jeu  
  
} while (vie > 0 );
```

Structures de contrôle



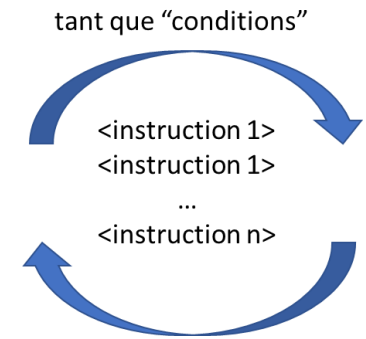
Instruction « **while** » : s'exécute tant que l'expression spécifiées est vraie.

```
while (vie > 0 ) {  
  
    //... code du jeu  
  
}
```



Instruction « for » : utile pour traiter des tableaux et d'autres structures pour lesquelles vous savez à l'avance combien d'itérations vous souhaitez effectuer.

```
for (int i = 1; i <= nombreArbre; i++)  
{  
    mesArbres[i] = creerArbre(i);  
}
```

Instruction « foreach » : répète un groupe d'instructions pour chaque élément d'une collection d'objets :

```
int[] fibarray = new int[] { 0, 1, 1, 2, 3, 5, 8, 13 };

foreach (int element in fibarray)
{
    Debug.Log(element);
}
```

Variables et fonctions :

- Déclarer les variables au début de la fonction
- Toujours initialiser les variables
- Attention, C# est sensible à la casse!
 - MonCompteurBoucle != monCompteurBoucle
 - ageDuvisteur != ageDuVisiteur
- Nommer les variables et fonctions avec des noms pertinents à leur usage
- Suivre des conventions de nommage (e.g., Camelcase)

Commentaires :

- Sur une ligne :

```
//Cette boucle itère sur tous les éléments de la liste  
foreach (int i in collections){}
```

- Sur plusieurs lignes :

```
int aire_triangle(x,y){  
    /*  
    La fonction aire(x, y) calcule l'aire d'un rectangle par  
    rapport a sa longueur x et largeur y.  
    */  
}
```

Fonctions - Debug.Log

■ Exemple 3 :

```
void carreNombre(int valeur) {  
  
    Debug.Log(valeur*valeur);  
  
}
```

-
- *Guide de programmation C Sharp*. Disponible: <https://docs.microsoft.com/fr-fr/dotnet/csharp/programming-guide/>
 - A. Cardinale. *Créez des Jeux de A à Z avec Unity : Bases et jeux mobiles*. D-Booker. 2016, 2eme edition.
 - *Apprendre avec Unity*. Disponible: <https://unity3d.com/fr/learn>.

Programmation pour le jeu vidéo

CARLOS CRISPIM-JUNIOR, DR. ENG.

GAMAGORA 2020-2021 - UNIVERSITÉ LUMIÈRE LYON 2

carlos.crispimjunior@univ-lyon2.fr