Les fonctions

Programmation de base sur PC

Plan du cours

- Pourquoi une fonction ?
- Anatomie d’une fonction.
- Appel d’une fonction
- Créer des fonctions de base.
- DRY dans les fonctions ?
- Les paramètres par défaut.
- Le chaînage des fonctions.
- Exercices

Les fonctions

Dans un logiciel, tout est fonction...

Entrer des notes informatisées et partager son code

- Ginette aime beaucoup son nouveau programme lui permettant de rentrer 10 notes pour ses 10 étudiants.
- Elle aimerais maintenant pouvoir partager votre travail de calcul de la moyenne à ses collègue enseignants qui ont chacun un nombre différent d'étudiant.
- Elle se dit ça doit être facile, suffit de copier et coller le code et de changer le 10 par le nombre d'élève...

Partie du code copié-collé

- Le code de la moyenne pour le collègue #1 qui a 25 élèves.

```
for(int i = 0; i < 25; i++){
  sum += notes[i];
}
```

Seul le nombre d'itérations changera pour tous les copier-collers.

- Le code pour un collègue #2 de l'université qui a 225 élèves.

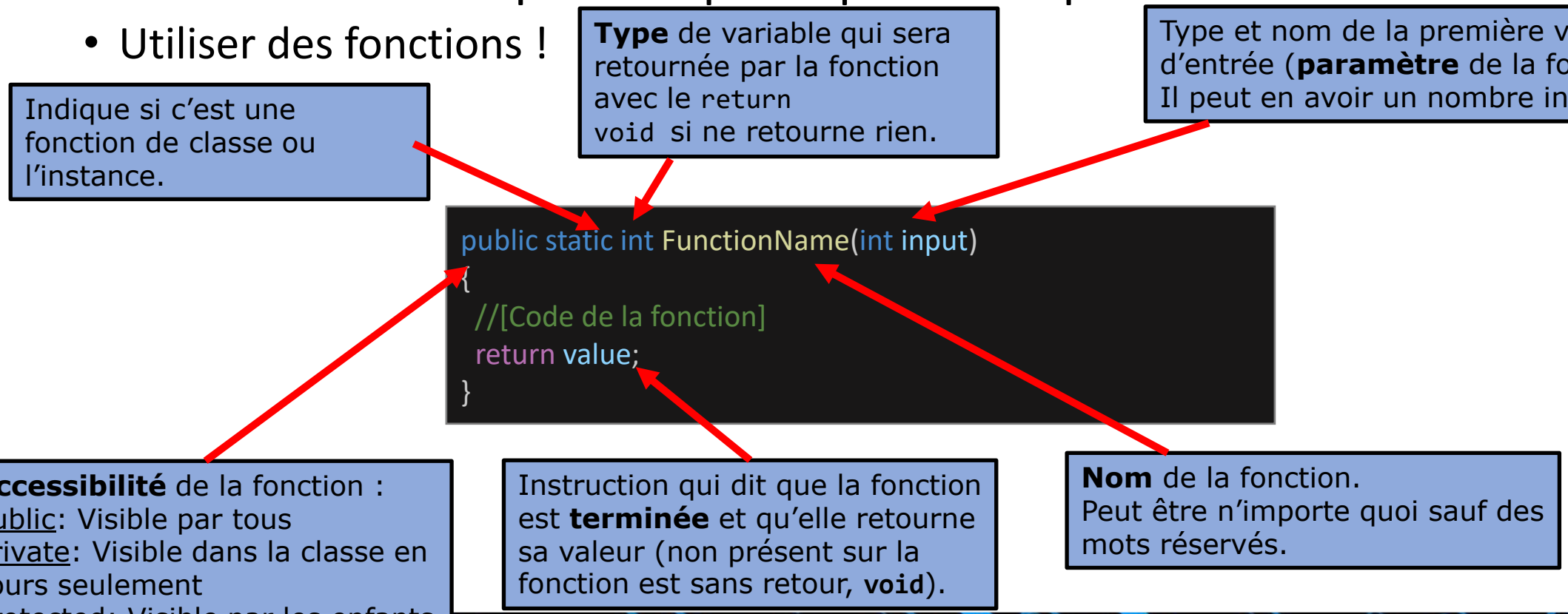
```
for(int i = 0; i < 225; i++){
  sum += notes[i];
}
```

- Le code pour un collègue #3 qui fait de l'aide en groupe.

```
for(int i = 0; i < 4; i++){
  sum += notes[i];
}
```

Anatomie d'une fonction

- Ce code n'est pas très **DRY**, on répère le même code plusieurs fois.
- Alors comment faire pour ne pas répéter des partie de code ?
  - Utiliser des fonctions !



Éviter le copier-coller

- Ginette se demande maintenant comment ces fonctions doivent être programmées afin d'éviter les copier-collers.
- Elle se dit qu'il serait probablement intéressant de pouvoir réutiliser le code de l'entrée des notes et du calcul de la moyenne à plusieurs places dans le code.
- Mais elle se demande comment faire...

Fonction pour lire des notes

- Voici le code d'une fonction servant à lire des notes

Accessible à tous.

Fonction de classe.

Ne retourne rien.

Tableau de `decimal` vide en entrée.

```
public static void EnterNotes(decimal[] notes) {

 for(int i = 0; i < notes.Length; i++){
  Console.Write($"Entrez la note #{i + 1} : ");
  notes[i] = decimal.Parse(Console.ReadLine());
 }

}
```

Le nombre d'itération est relatif au nombre d'éléments dans le tableau.

Aucun `return` car cette fonction ne retourne rien.

Fonction pour la moyenne

- Voici le code d'une fonction servant à calculer la moyenne

Accessible à tous

Fonction de classe

Retour la moyenne en `decimal`

Tableau de `decimal` vide en entrée

```
public static decimal GetAverage(decimal[] notes){
  decimal sum = 0, average;
  for(int i = 0; i < notes.Length; i++){
    sum += notes[i];
  }
  average = sum / notes.Length;
  return average;
}
```

Le nombre d'itération est relatif au nombre d'éléments dans le tableau

Selon son rôle, retourne un decimal contenant la moyenne des notes entrées

Appel d'une fonction

- Pour appeler une fonction nous devons utiliser son nom et paramètres.

Nombre de notes à entrer.

Création du tableau de « count » notes.

```
static void Main(string[] args) {
  int count = 10;
  decimal[] notes = new decimal[count];
  decimal average;

  EnterNotes(notes);
  average = GetAverage(notes);
  Console.WriteLine($"La moyenne des {count} notes est de { average }\n");    }
```

Appel de la fonction `EnterNotes()` qui a comme rôle de remplir un tableau passé paramètre.

Appel de la fonction `GetAverage()` en lui passant les notes entrées en paramètre. Son rôle étant de retourner la moyenne de ces notes.

Éviter le copier-coller

- Ginette pense commencer à comprendre toute la puissance des fonctions.
- Elle a un flash...
- « Serait-il possible d'utiliser ces fonctions avec mes élèves pour leur apprendre les opérateurs de base (+, - * et /) ? »

Fonction des opérateurs de base: Un exemple

- Création des fonctions de opérateurs de base.

```
public static double Sum( double x1, double x2 ) {
  return x1 + x2;
}

public static double Difference( double x1, double x2 ) {
  return x1 - x2;
}

public static double Product( double x1, double x2 ) {
  return x1 * x2;
}

public static double Quotient( double x1, double x2 ) {
  return x1 / x2;
}
```

Il peut y avoir autant de paramètres d'entrées que l'on souhaite.

Retourne directement le résultat du calcul effectué sans besoin de passer par une variable.

Appel des fonctions de base

- Pour appeler une fonction nous devons utiliser son nom et paramètres.

```
static void Main(string[] args)
{
  double x1, x2;
  Console.WriteLine( "Entrez deux nombres à multiplier : " );
  x1 = double.Parse( Console.ReadLine());
  x2 = double.Parse( Console.ReadLine());

  Console.WriteLine($"Le produit de {x1} et {x2} est { Product(x1, x2) }\n");    }
```

On sépare les paramètres à entrer par des virgules.

On appel directement la fonction `Product()` sans l'obligation de passer par une variable. Le résultat de la multiplication sera envoyé à la fonction `WriteLine()`.

Éviter le copier-coller

- Ginette passe pour une femme hyper-techno à l'école où elle travaille car les élèves apprennent leur opération de base tout en s'amusant avec C#.
- Elle a un autre flash...
- « Serait-il possible d'utiliser une seule fonction pour leur apprendre les opérateurs de base (+, - * et /) nommée `Calculate()`? »

Appel des fonctions de base dans UNE fonction

- Même les fonctions peuvent être **DRY**-ifiée ☺.

On passe en paramètre l'opération à effectuer.

```
public static double Calculate(double x1, double x2, char operation = '+'){
  switch(operation) {
    case '+': return x1 + x2;
    case '-': return x1 - x2;
    case '*': return x1 * x2;
    case '/': return x1 / x2;
    default: return 0;
  }
}
```

Valeur par défaut si rien n'est entré.
Exemple: `Calculate(10,15)` retournera `25`.

Pas besoin de `break;` car le `return;` a pour effet d'arrêter complètement l'exécution de la fonction.

Appeler des fonctions en chaîne

- Un élève de la classe de Ginette nommé Julien a « mélangé » son code et demandé à Ginette ce qu'il avait bien pu faire pour que 15 / 10 donne 3 ?
- Ginette ne sais quoi répondre.
- Elle a besoin de votre aide.
- Voici son code :

Appel chaîné de fonctions

- Code de Julien

```
static void Main(string[] args)
{
  double x1, x2, result;
  char operation;

  Console.Write("Entrez une opération à effectuer [+, -, *, /] : ");
  operation = Console.ReadLine()[0];

  Console.WriteLine("Entrez deux nombres : ");
  x1 = double.Parse(Console.ReadLine());
  x2 = double.Parse(Console.ReadLine());

  result = Calculate( Calculate(x1, x2, operation), Calculate(x1, x2, operation) );

  Console.WriteLine($"Le résultat est { result }\n");
}
```

On prend seulement le premier caractère entré. `[0]` représente le premier élément d'un tableau.

Le `Calculate()` parent effectue l'addition des deux enfants car il n'a aucun paramètre entré pour l'opération et que le « + » est par défaut.

Les résultats de fonctions sont chaînés. `Calculate(x1, x2, operation)` retourne son résultat au `Calculate()` parent.

Exercices