Fonctions

Vincent Archambault-Bouffard
IFT 2035 - Université de Montréal

Pour obtenir le code source de ce document

- https://github.com/archambaultv/IFT2035-UdeM
- vincent.archambault-bouffard@umontreal.ca

Plan du cours

- Comment encapsuler du code ?
- Passage d’arguments
- Fonction d’ordre supérieur
- Récursion
- Fermeture

Comment encapsuler du code ?

Fonctions = Programmes paramétrés

Réutilisation du code

Permet d’écrire des librairies

Décomposition du problème en sous-problèmes

Abstraction du code

```
salutation nom = “Hello ” ++ nom

x = salutation “Vincent”
y = salutation “Paul”
z = salutation “Julie”
```

Réutilisation du code

Passage d’arguments

Peut-on modifier les paramètres ?

```
int x = 5;

int foo(int p){
  p = 3;
  return p;
}

int y = foo x;
```

Quelle est la valeur de x maintenant ?

Paramètres formel et actuel

paramètre actuel Lors d'un appel de fonction, les paramètres fournis à la fonction sont les paramètres actuels

paramètre formel Représentant des paramètres actuels dans le corps de la fonction

Paramètres formel et actuel

```
int x = 5;

int foo(int p){
  p = 3;
  return p;
}

int y = foo x;
```

p est le paramètre formel

x est le paramètre actuel

Passage par valeur

- Le plus utilisé dans les langages de programmation (ex : C)
- Le paramètre actuel est une valeur
- Le paramètre formel est une nouvelle variable, initialisée avec une copie de la valeur du paramètre actuel
- Les modifications faites sur le paramètre n'affectent pas l'appelant

```
int x = 5;

int foo(int p){
  p = 3;
  return p;
}

int y = foo x;
```

x vaut encore 5

Le passage par valeur n'affecte pas l'appelant

Passage par référence

- Utilisé en C++
- Le paramètre actuel est un emplacement mémoire
- Le paramètre formel désigne le même emplacement mémoire
- Les modifications faites sur le paramètre affectent l'appelant

```
int x = 5;

int foo(int &p){
  p = 3;
  return;
}

foo x;
```

x vaut 3

Passage par référence en C++

Passage par référence

- Il est possible de le simuler en C avec le passage par valeur d'un pointeur

```
int x = 5;

int foo(int* p){
  *p = 3;
  return;
}

foo (&x);
```

x vaut 3

Passage par valeur d'un pointeur en C pour simuler un passage par référence

Passage par nom

- Utilisé en Haskell
- Le paramètre actuel est une expression non évaluée
- Le paramètre formel fait référence à l'expression
- Chaque utilisation du paramètre formel évalue l'expression
- Les modifications faites sur le paramètre n'affectent pas l'appelant

```
int x = 5;

int foo(int p){
  p = 3;
  return p;
}

int y = foo x;
```

x vaut encore 5

Le passage par nom n'affecte pas l'appelant

Passage par nom

- Il est possible de dupliquer les effets de bords

```
int foo(char p){
  char c = p;
  char c2 = p;
  return;
}

foo( getchar() );
```

Si C avait un passage par nom, deux caractères seraient lus par getchar

Passage par valeur-résultat

- Utilisé en Ada
- Le paramètre actuel est un emplacement mémoire
- Le paramètre formel est une nouvelle variable initialisée avec la valeur du paramètre formel
- Lorsque la fonction termine, la valeur du paramètre formel est recopiée dans l'emplacement mémoire du paramètre actuel
- Les modifications faites sur le paramètre affectent l'appelant

```
int x = 5;

int foo(int p){
  p = 3;
  toto = x;
  return p;
}

int y = foo x;
```

x vaut 3

Le passage par nom n'affecte pas l'appelant

Passage par valeur-résultat

- Possible de le simuler avec des pointeurs

```
int foo(int *p){
  p2 = *p;
  …
  *p = p2;
  return;
}
```

Le passage par nom n'affecte pas l'appelant