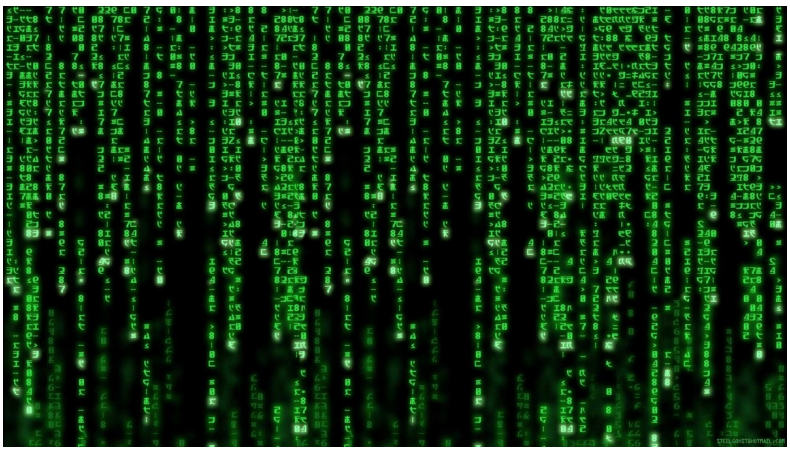
UVSQ université PARIS-SACLAY | IUT de Vélizy-Rambouillet CAMPUS DE VÉLIZY-VILLACOUBLAY

Chapitre 2 : Programmation modulaire (fonctions)



GEII 1ère année

IUT Vélizy

2022 – 2023

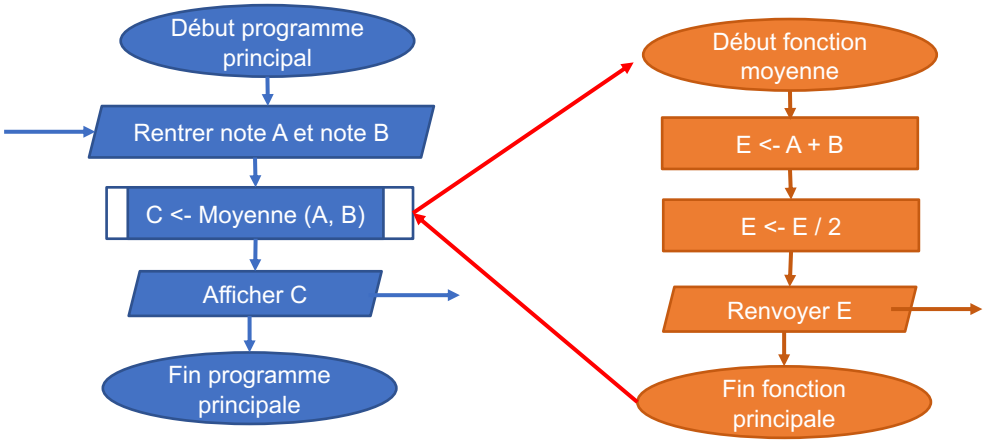
Julien Gabiot

1

Principe de la programmation modulaire

UVSQ université PARIS-SACLAY | IUT de Vélizy-Rambouillet CAMPUS DE VÉLIZY-VILLACOUBLAY

Exemple: calcule de la moyenne de deux notes par une fonction



2

Principes de la programmation modulaire

UVSQ université PARIS-SACLAY — IUT de Vélizy-Rambouillet, CAMPUS DE VÉLIZY-VILLACOUBLAY

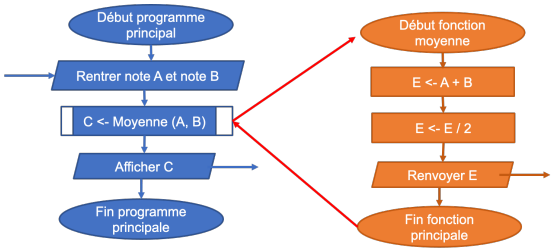
Permet d'organiser notre code en fonctions. Il s'agit en fait de découper nos programmes en petits morceaux. Chaque petit morceau sera ce qu'on appelle une fonction.

Programmation modulaire :

- ➔ décomposition du programme en modules (ou sous-programmes) qui exécutent chacun un traitement complexe
- ➔ modules appelés par le programme principal
- ➔ Les modules **communiquent entre eux par l'intermédiaire de leurs entrées/sorties**
- ➔ **chaque module écrit indépendamment** du programme principal et des autres modules
- ➔ Le traitement interne à chaque module est invisible du reste du programme

Objectifs :

- ➔ simplifier la lecture
- ➔ simplifier l'écriture
- ➔ améliorer la maintenance
- ➔ favoriser la portabilité

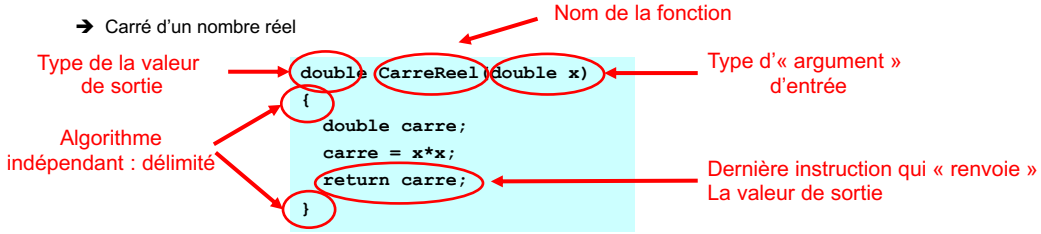


3

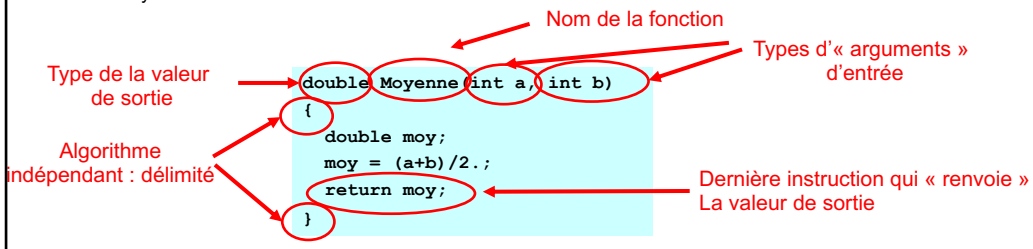
Définition d'une fonction

UVSQ université PARIS-SACLAY — IUT de Vélizy-Rambouillet, CAMPUS DE VÉLIZY-VILLACOUBLAY

➔ Carré d'un nombre réel



➔ Moyenne de 2 nombres entiers



4

Principes de la programmation modulaire

UVSQ université PARIS-SACLAY — IUT de Vélizy-Rambouillet, CAMPUS DE VÉLIZY-VILLACOUBLAY

Caractéristiques d'un module

- Nom : mêmes règles que les noms de variables
- Entrée(s) (ou argument(s) d'entrée):
  - ➔ Informations nécessaires pour le bon fonctionnement du module
  - ➔ Fourni par le module « appelant »
  - ➔ Autres termes : Données, arguments, paramètres
  - ➔ Nombre et types des entrées
- Sortie- (ou argument de sortie):
  - ➔ Résultat du traitement interne du module
  - ➔ Utilisé par les autres modules
  - ➔ Autres terme : résultat, valeur renvoyée
  - ➔ Nombre et type de la sortie

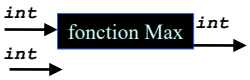
5

Argument(s) d'entrée(s) / valeur(-) de sortie

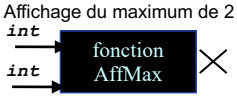
UVSQ université PARIS-SACLAY — IUT de Vélizy-Rambouillet, CAMPUS DE VÉLIZY-VILLACOUBLAY

Exemples :

➔ Maximum de 2 nombres entiers



➔ Affichage du maximum de 2 nombres entiers



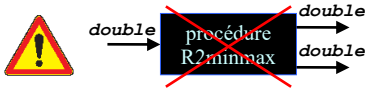
➔ Carré d'un nombre entier

➔ Carré d'un nombre réel

➔ Moyenne de 2 nombres entiers

➔ Welcome

➔ **Fonction qui renvoie plusieurs arguments : impossible**



6

Positionnement de la fonction dans le programme

IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY

```
#include <stdio.h>
```
Instructions de précompilation

```
double CarreReel(double x)
```
Déclaration des sous-programmes

Définition du programme principal
```
int main(void)
{
  double x=4.2,y;
y = CarreReel(5);
  y = CarreReel(x);
  y = CarreReel(3*x+4);
  y = CarreReel(y);
}
```

Définition des sous-programmes
```
double CarreReel(double x)
{
  double carre;
  carre = x*x;
  return carre;
}
```

Définition des sous-programmes

7

Déclaration du prototype d'une fonction

UVSQ université PARIS-SACLAY
IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY

Permet au programme principal de connaitre les caractéristiques de la fonction (nom, entrée(s), sortie)

Se place avant la définition du programme principal (juste après les instructions de précompilation)

Type `void` : pas d'entrée ou de sortie

Instruction de déclaration : se termine par `;`

- Syntaxe : **type nom (types des arguments) ;**

Type de « l'argument » de sortie → `int Max(int, int);` ← Type des « arguments » d'entrée

```
int Max(int, int);
void AffMax(int, int);
int CarreInt(int);
double CarreReel(double);
double moyenne(int, int);
```

Void = pas d'argument De sortie → `void welcome (void);` ← Void = pas d'argument D'entrée

```
double calculR2(double, double);
```

On peut donner un nom au entrées

```
double calculR2(double T, double I);
```

8

Appel d'une fonction

UVSQ université PARIS-SACLAY — IUT de Vélizy-Rambouillet, CAMPUS DE VÉLIZY-VILLACOUBLAY

Une fonction n'est exécutée que si elle est appelée :

- ➔ dans le programme principal
- ➔ dans une autre fonction elle-même appelée par le programme principal
- ➔ Lors de l'appel d'un sous-programme, le programme principal se met en veille jusqu'à la fin de l'exécution du sous-programme
- ➔ Instruction return donne la valeur récupérée par le programme principal

```
int main(void)
{
  double x=4.2,y;
  y = CarreReel(5);
  y = CarreReel(x);
  y = CarreReel(3*x+4);
  y = CarreReel(y);
}
```

```
double CarreReel(double x)
{
  double carre;
  carre = x*x;
  return carre;
}
```

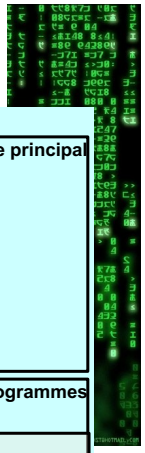
```
double CarreReel(double x)
{
  x = x*x;
}
```


Principe de « Localité » des variables

UVSQ université PARIS-SACLAY — IUT de Vélizy-Rambouillet, CAMPUS DE VÉLIZY-VILLACOUBLAY

Déclaration des variables

- ➔ variables internes : déclarées dans le module
- ➔ Paramètres formels : entrées du module, leur valeur (initiale) est fixée lors de l'appel du module
- ➔ dans tous les cas, les variables sont locales :
  - ➔ définie uniquement dans le module où elle est déclarée
  - ➔ inconnue par les autres modules
  - ➔ « apparaît » et « disparaît » en même temps que le module
  - ➔ nouvelle initialisation à chaque nouvel appel du module



```
......
int main(void)                    programme principal
{
  double x=4.2,y;
  y = carre;
  y = CarreReel(x);
  y = CarreReel(3*x+4);
  y = CarreReel(y);
}
```

```
double CarreReel(double x)        sous-programmes
{
  double carre;
  carre = x*x;
  return carre;
}
```

Variable(s) locale(s) déclarée(s) en début du corps de la fonction