GMP – 2015 – Maxime Guériau

Initiation à la programmation

Intervenant :
maxime.gueriau@univ-lyon1.fr
liris.cnrs.fr/maxime.gueriau

Supports de cours de Baudouin Dafflon
baudouin.dafflon@univ-lyon1.fr
cv.bdafflon.eu

Plan

- Plan du cours :
  - Introduction
    - Introduction aux algorithmes
    - Découverte de la programmation avec Logo
  - Programmation
    - Le langage C
    - Structure d'un programme
    - Variables
    - Premier programme
    - Entrées / sorties
    - Structures de contrôle
      - Conditions
      - Boucles
    - Architecture
      - Fonctions
      - Bibliothèques
    - Structures de données (tableaux)

Plan

- Plan du cours :
  - Introduction
    - Introduction aux algorithmes
    - Découverte de la programmation avec Logo
  - Programmation
    - Le langage C
    - Structure d'un programme
    - Variables
    - Premier programme
    - Entrées / sorties
    - Structures de contrôle
      - Conditions
      - Boucles
    - Architecture
      - Fonctions
      - Bibliothèques
    - Structures de données (tableaux)

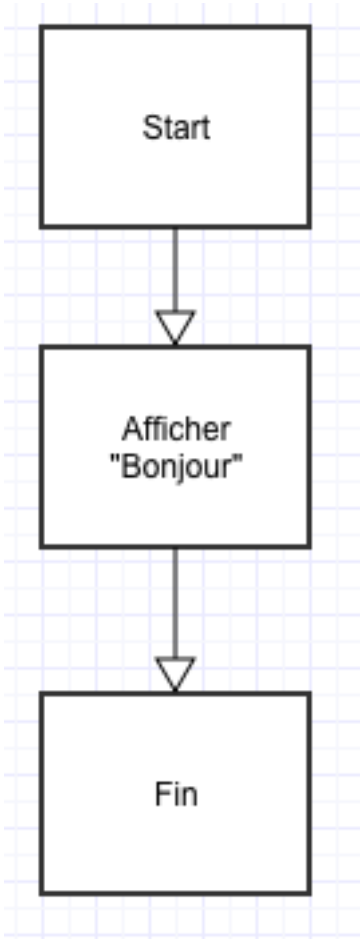
Introduction : algorithmes

- Qu'est ce qu'un algorithme ?
  - Un algorithme est un « ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé au moyen d'un nombre fini d'opérations »
  - « Un algorithme peut être traduit, grâce à un langage de programmation, en un programme exécutable par un ordinateur. »

Source: www.larousse.fr

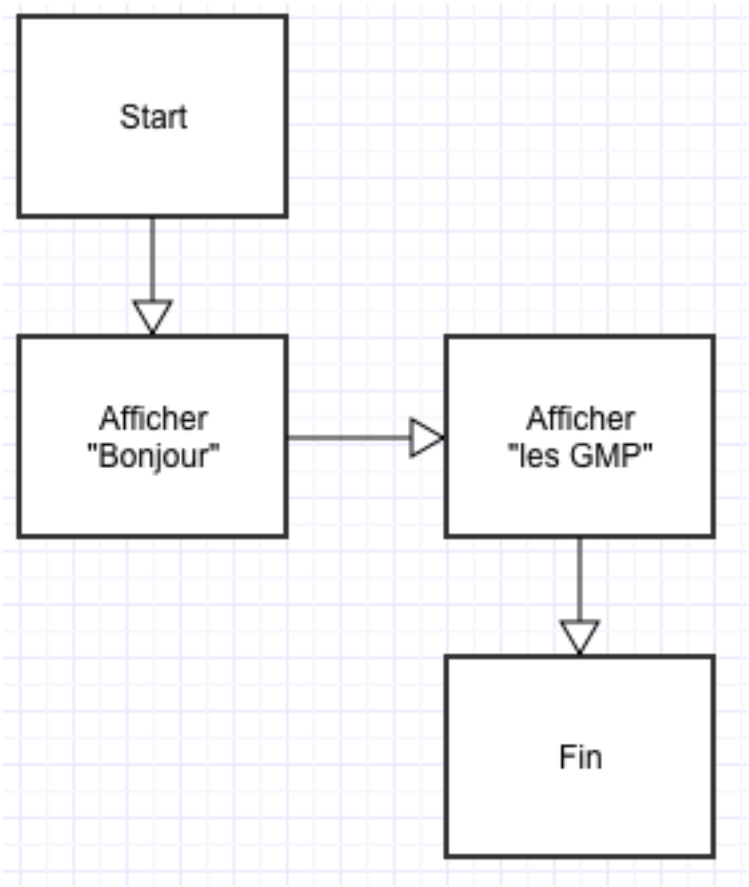
Introduction : algorithmes

- Exemples d'algorithmes :
  - Dire « Bonjour » :



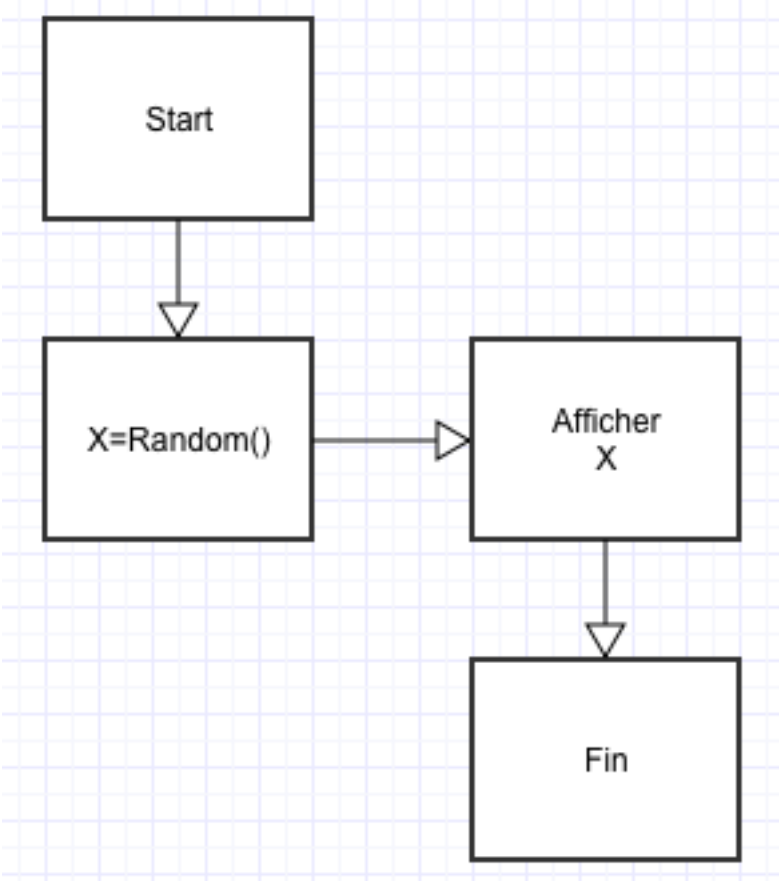
Introduction : algorithmes

- Dire « bonjour les GMP » :



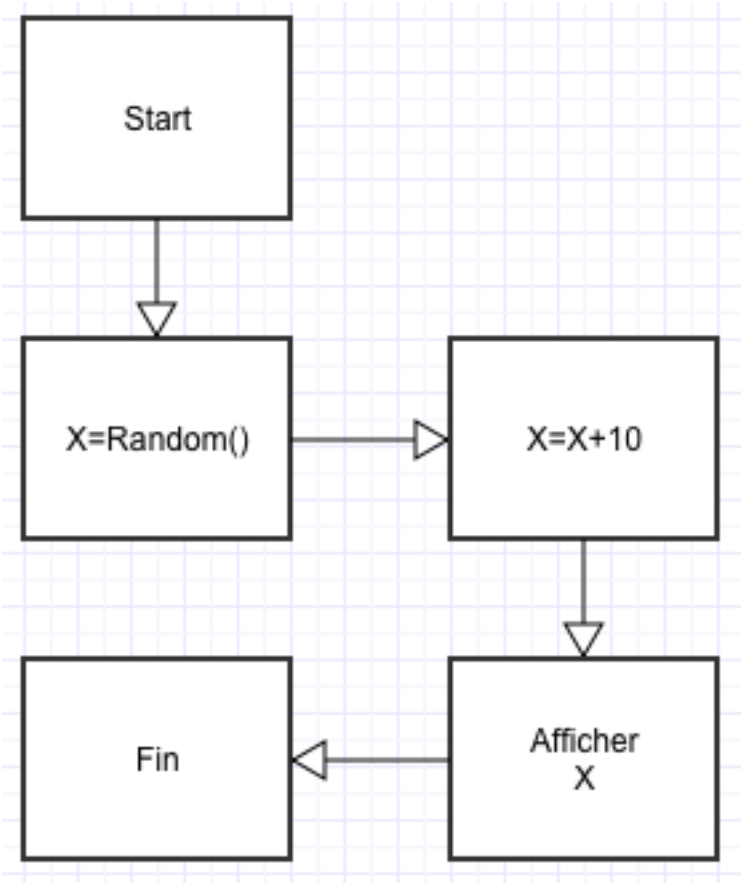
Introduction : algorithmes

- Afficher un nombre aléatoire

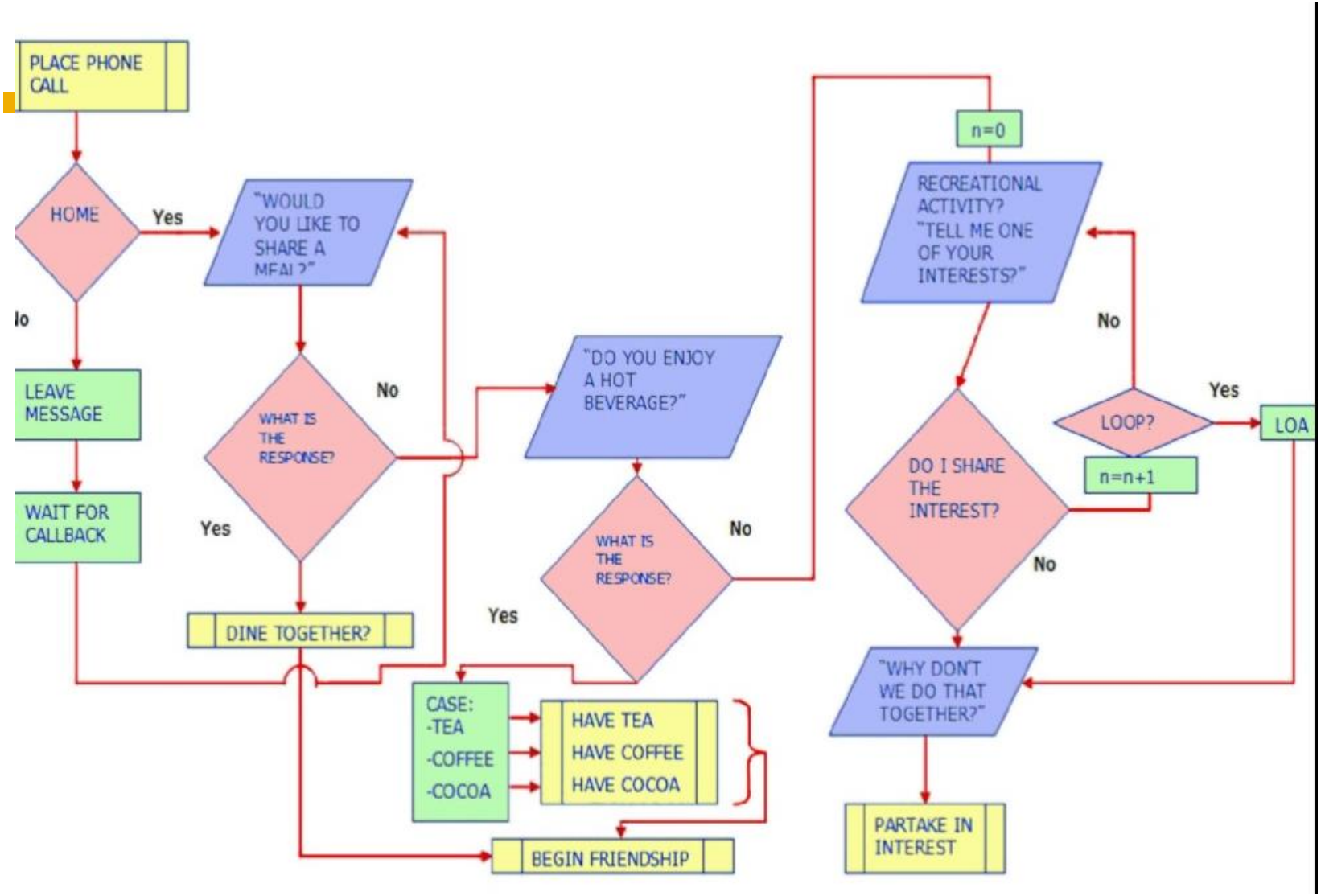


Introduction : algorithmes

- Afficher un nombre aléatoire +10



Introduction : algorithmes

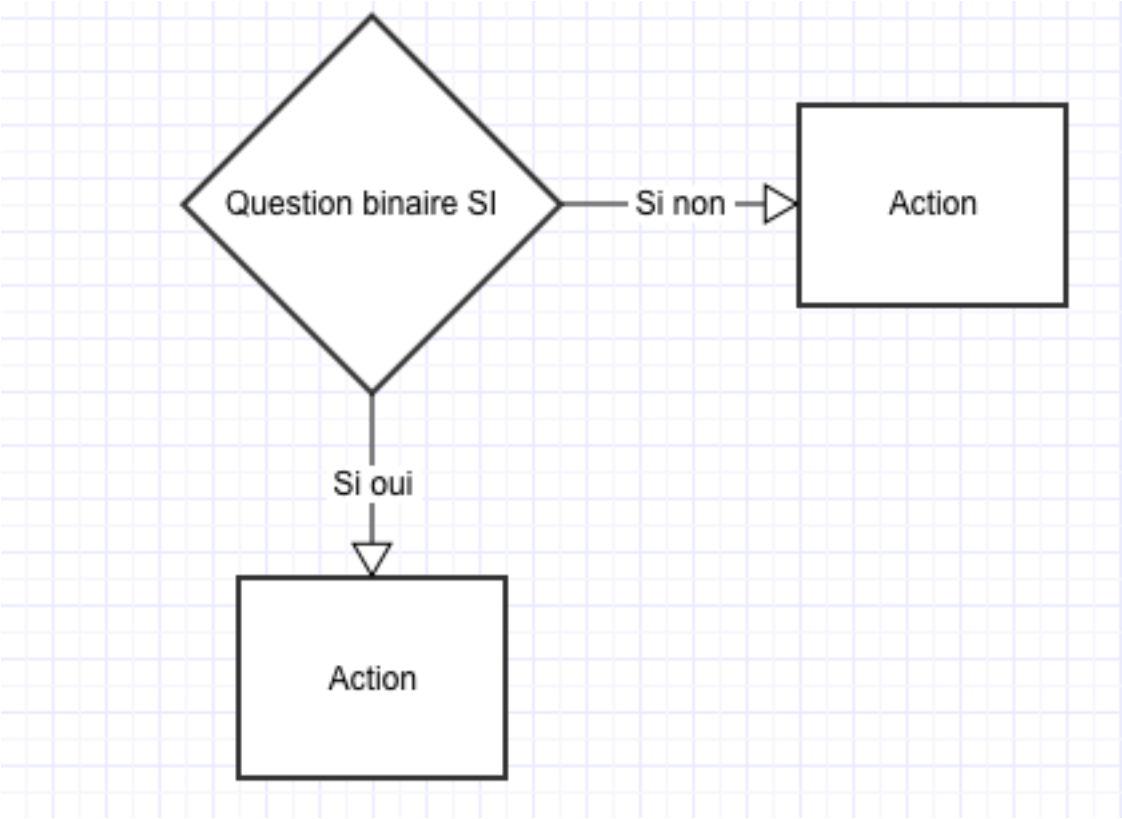


Introduction : algorithmes

- Exercice :
  - Dessiner l'algorithme permettant d'ajouter 2 nombres entre eux.

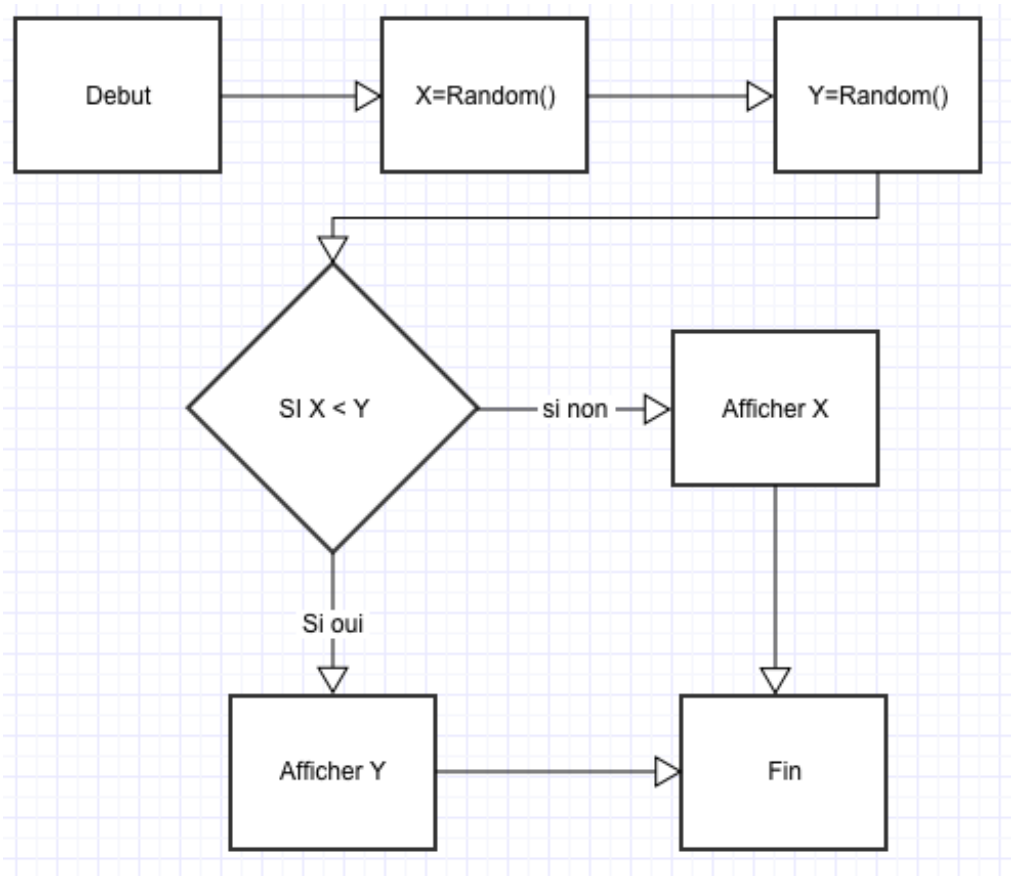
Introduction : algorithmes

- Les autres briques : le choix conditionnel



Introduction : algorithmes

- Exemple : afficher le max de deux nombres aléatoires

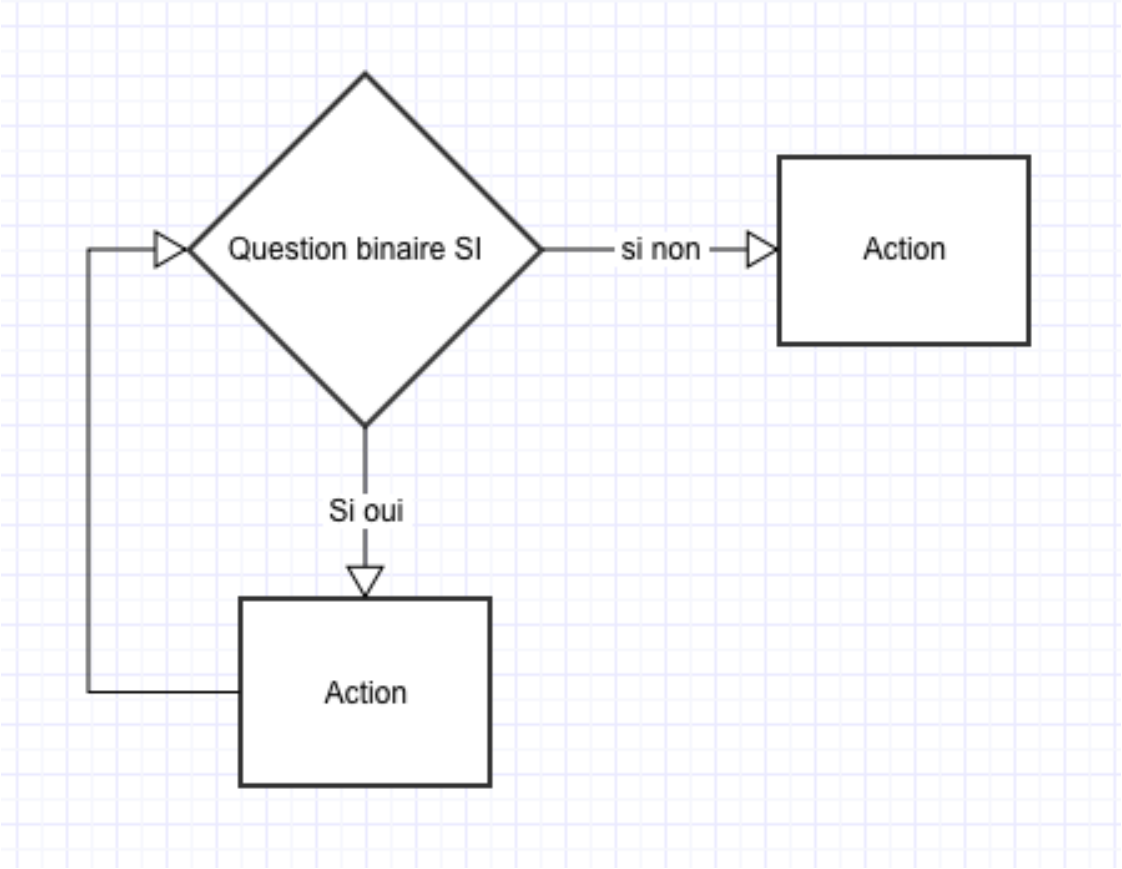


Introduction : algorithmes

- Exercices :
  - Afficher le max de trois nombres aléatoires.
  - Afficher la différence de trois nombres aléatoires.
  - Afficher le produit de deux nombres aléatoires .
  - Afficher si un nombre aléatoire est pair.
  - Afficher la moyenne de trois nombres aléatoires.

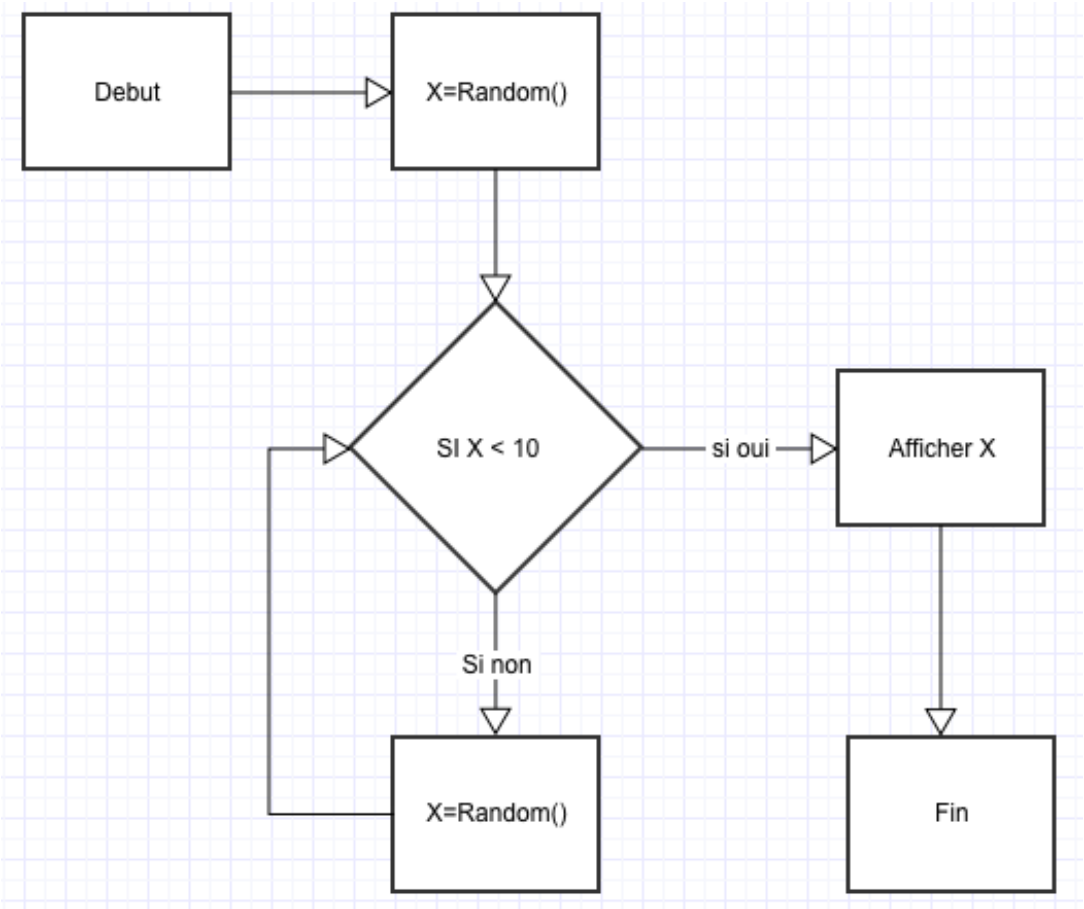
Introduction : algorithmes

- Les autres briques : la boucle



Introduction : algorithmes

- Exemple : afficher un nombre aléatoire plus petit que 10

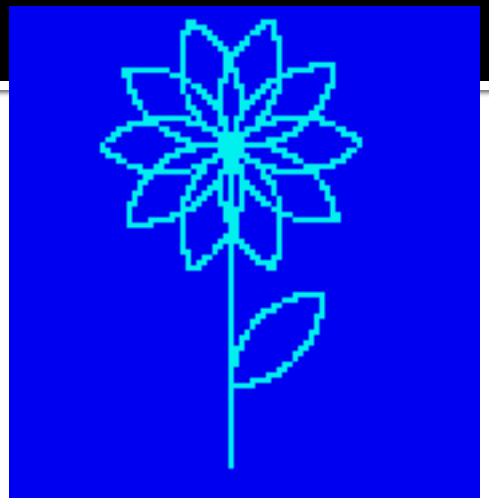


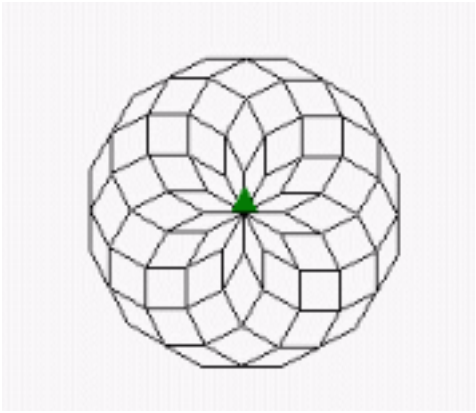
Introduction : algorithmes

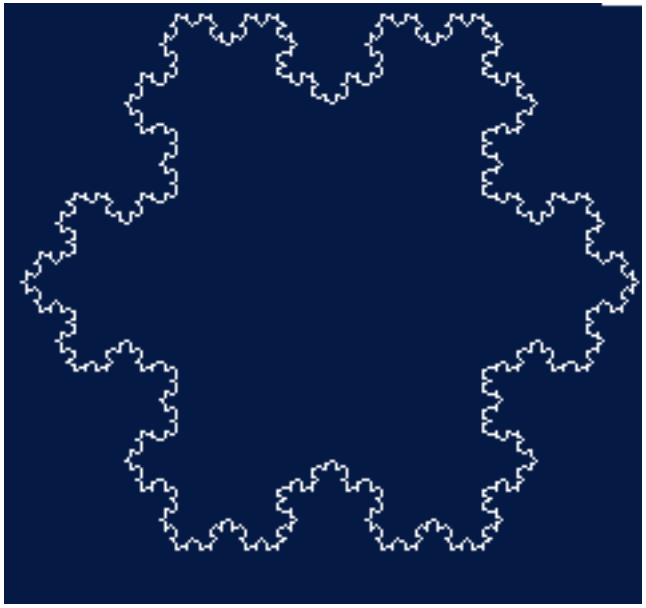
- Exercices :
  - Additionner des nombres aléatoires jusqu'à 100.
  - Afficher 10 nombres aléatoires.
  - Faire un compteur de 1 à 10.
  - Additionner 10 nombres aléatoires.
  - Additionner jusqu'à 100 des nombres aléatoires seulement si ils sont inferieur a 10.

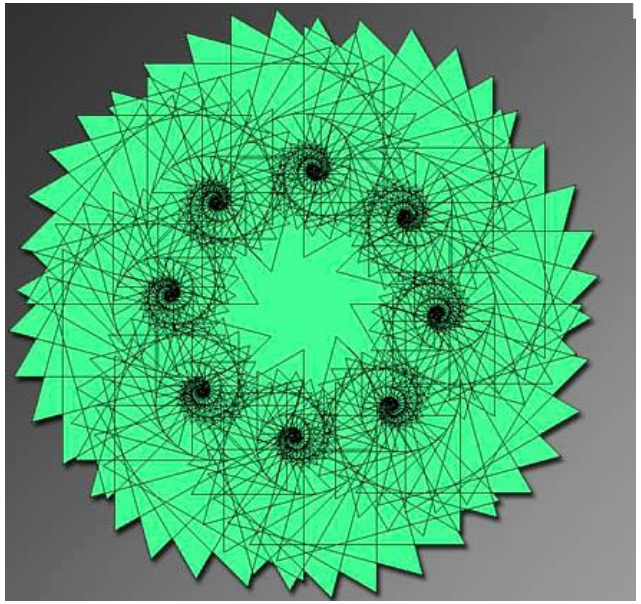
Introduction : programmation

- Un peu de pratique : le LOGO
  - Un langage graphique



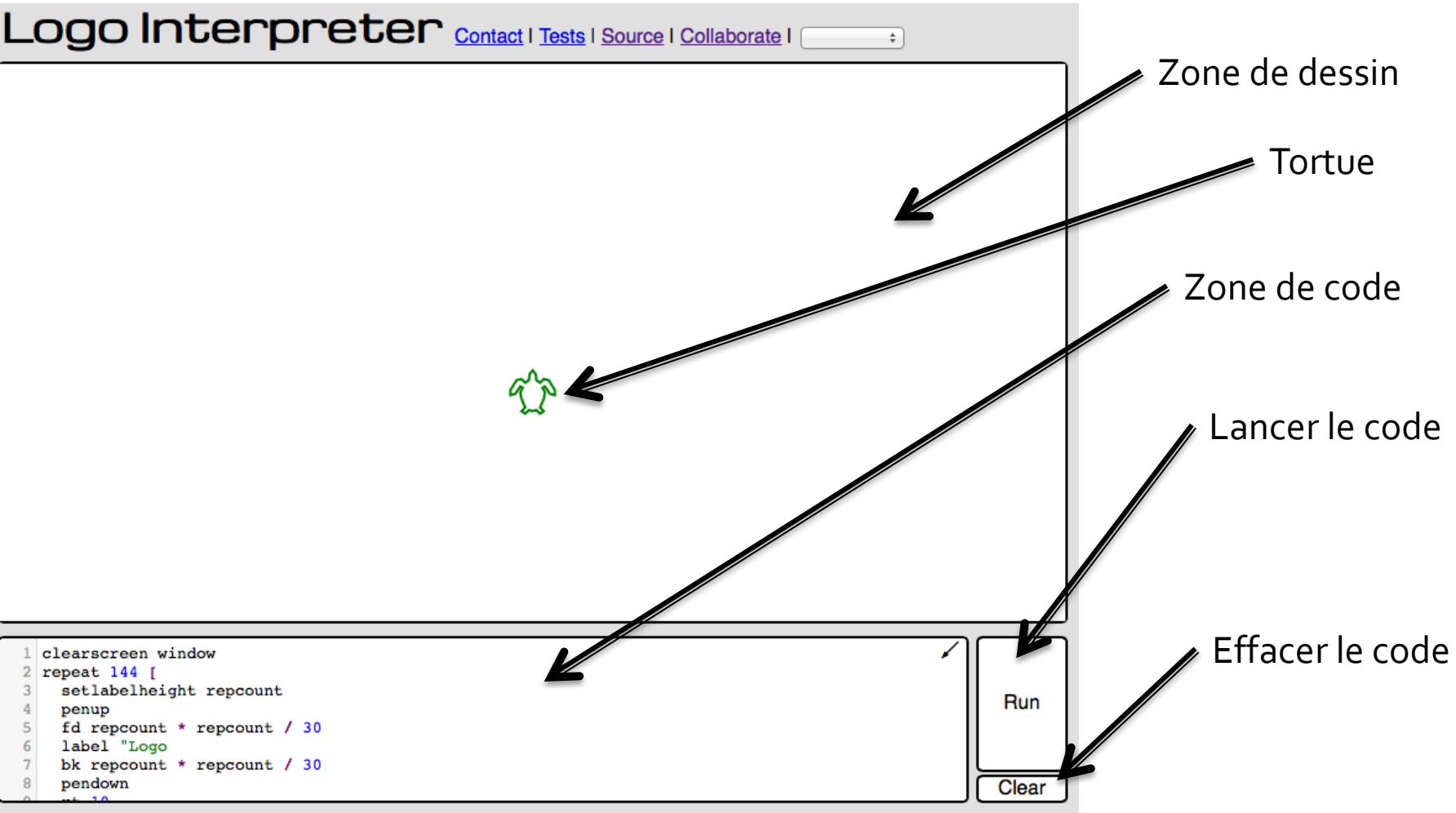






Introduction : programmation

- www.dafflon.fr/logo



Introduction : programmation

- Un peu de pratique: le LOGO
  - Quelles sont les « boites disponibles » ?

  - **Clearscreen**
    efface l' écran
  - **forward** x
    - avance de *x*
  - **back** x
    - Recule de x
  - **left** x
    - Tourne de x degrés sur la gauche
  - **right** x
    - Tourne de x degrés sur la droite

  - **Random** x
    - Pioche un nombre random entre 0 et x
  - **penup**
    - Lève le crayon
  - **Pendown**
    - Baisse le crayon

Introduction : programmation

- Exercices :
  - Dessiner une ligne verticale de 100 pixels.
  - Dessiner une ligne horizontale de 50 pixels.
  - Dessiner une ligne verticale d'un nombre aléatoire (entre 0 et 400) de pixels.
  - Dessiner un carré de 100 pixels de largeur.
  - Dessiner un rectangle de 100 pixels de largeur et 50 de longueur.
  - Créer un polygone de « rayon » 100 et à N côté (n entier variable)

Plan

- Plan du cours :
  - Introduction
    - Introduction aux algorithmes
    - Découverte de la programmation avec Logo
  - Programmation
    - Le langage C
    - Structure d'un programme
    - Variables
    - Premier programme
    - Entrées / sorties
    - Structures de contrôle
      - Conditions
      - Boucles
    - Architecture
      - Fonctions
      - Bibliothèques
    - Structures de données (tableaux)

Programmation : le langage C

- Le C : un langage de programmation impératif
  - De quoi est composé le C ?
    - De Variables ;
    - De Fonctions ;
    - De bibliothèques ;
    - De structures de contrôle.
  - Pour transformer du code C en programme, nous avons aussi besoin d'un **compilateur** (ex: visual studio).

Programmation : Structure d'un programme

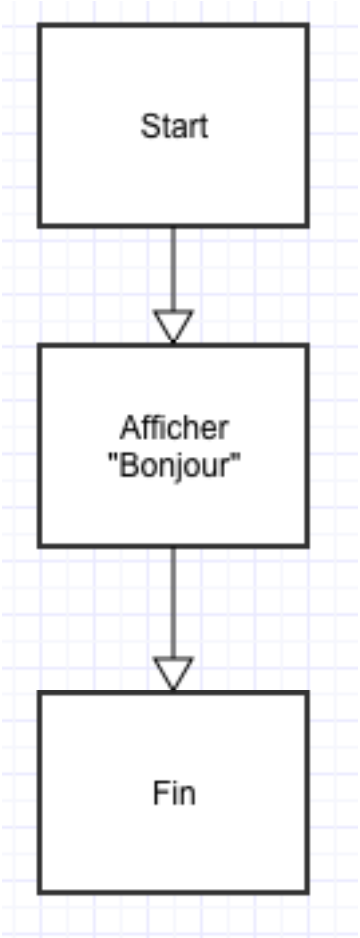
- Structure d'un programme en C :
  - **Inclusion de bibliothèques** : permet l'utilisation de fonctions déjà écrites, définies dans ces bibliothèques
  - **Déclarations de variables** : nommées intelligemment, et typées
  - **Déclarations de fonctions** : internes (définies dans ce programme source)
  - **Corps du programme principal** : liste des instructions et appels de fonctions

Programmation : Structure d'un programme

- Exemple

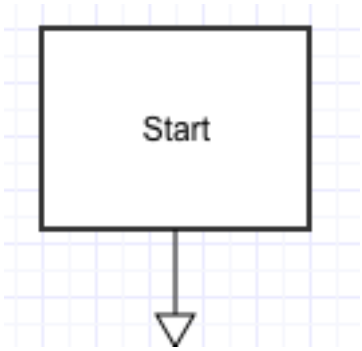
```
#include <stdio.h>
int main()
{
   /* Ce programme affiche ”Bonjour" */
   printf(”Bonjour");

   return 0;
}
```



Programmation : Structure d'un programme

- Exemple




```
#include <stdio.h>
int main()
{
    /* Ce programme affiche ”Bonjour" */
    printf(”Bonjour");

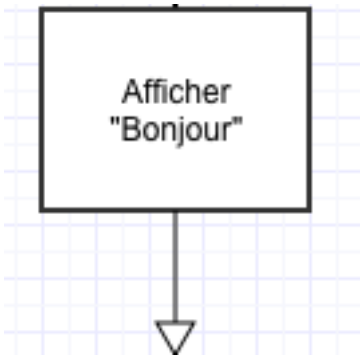
    return 0;
}
```

Programmation : Structure d'un programme

- Exemple

```
#include <stdio.h>
int main()
{
   /* Ce programme affiche ”Bonjour" */
   printf(”Bonjour");

   return 0;
}
```

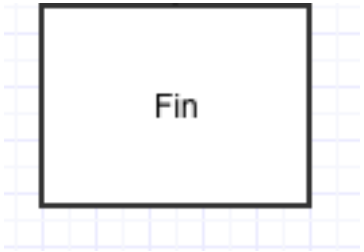


Programmation : Structure d'un programme

- Exemple

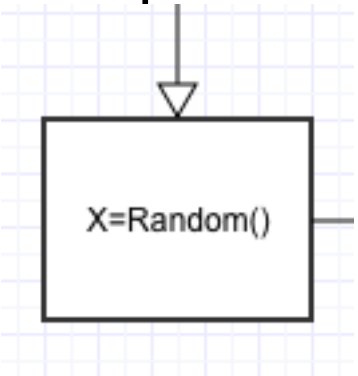
```
#include <stdio.h>
int main()
{
   /* Ce programme affiche ”Bonjour" */
   printf(”Bonjour");

   return 0;
}
```



Programmation : variables

- Les variables
  - Une variable liée à un emplacement précis en mémoire.
  - Le nom de la variable désigne la valeur qui se trouve à cet endroit
  - La valeur est en principe modifiable, mais une variable ne peut contenir qu'une seule valeur à un instant donné.



Programmation : variables

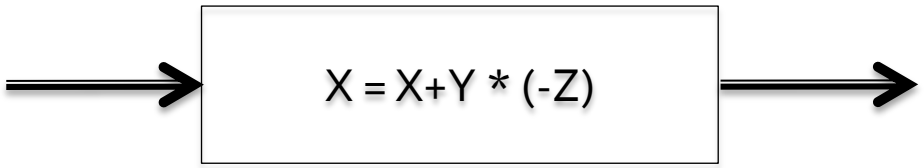
- Les variables
  - En C, les variables sont typées :
    - *Int* : nombre entier
    - *Float* : nombre flottant (nombre a virgule)
    - *Char* : caractère (ex: lettre)
  - En C, il faut « déclarer » les variables avant utilisation :
    - Exemple : type nom_variable = valeur;
    - Exemples en C :
      - `int nombre_d_eleves;`
      - `Int age_du_capitaine = 35;`

nombre_d_eleves=0

age_du_capitaine = 35

Programmation : variables

- Operations sur les variables :
  - + : pour l'addition
  - - : pour la soustraction (ou l'opposé)
  - * : pour la multiplication
  - / : pour la division entière : exemple : 7/2 vaudra 3
  - % : pour le modulo, reste de la division entière :
- Priorités : opposé, puis *, /, %, puis + -
  - Exemple :
    - X = X + Y * -Z



Programmation : premier programme

- Les bons reflexes pour programmer efficacement :
  1. Nommer son programme.
  2. L'enregistrer dans votre dossier personnel.
  3. Ajouter votre premier commentaire.
     - Un commentaire est une ligne de code ignorée par le compilateur.
     - Exemple :

```
/* Calcul des moyennes du semestre – Auteur : B. Dafflon – Septembre 2015 */
```

Programmation : premier programme

- Exercice :
  - Ecrire un programme qui additionne deux nombres entiers (a = 25 et b=77).
    1. Créer la structure vide de votre programme.
    2. Réaliser l'algorithme sur papier.
    3. Implémenter l'algorithme.
    4. Afficher le résultat.
       - Utiliser la fonction *printf*.
       - Exemple : `printf("%d",c);`

Programmation : premier programme

- Exercice :
  - Comment afficher le résultat ?
    - La **fonction** *printf* permet d'afficher du texte a l'écran :
      - `printf("hello world");`
      - `printf("La valeur de la variable A est %d",A);`
    - Pour pouvoir utiliser *printf* il faut déclarer la **bibliothèque** « stdio » :
      - `#include <stdio.h>`

Programmation : entrées/sorties

- *printf : explications*

format	explications
%d, %X	d comme décimal, X pour obtenir le nombre en hexadecimal
%3d	affiché sur 3 caractères(dont le signe -), ou plus si nécessaire
%f	f comme float, affiché par défaut avec 6 décimales
%8f	affiché sur 8 caractères (ou plus si nécessaire) (8 caractères dont le point décimal et le signe -) et 6 décimales
%8.2f	affiché sur 8 caractères (ou plus si nécessaire) (dont le point décimal et le signe -) et 2 décimales
%.u	u comme unsigned
%lf	lf comme long float, pour un réel de type double, affiché avec 6 décimales
%20.15lf	affiché sur 20 caractères dont 15 décimales
%c	c comme caractère unique
%e, %E	e comme exponentielle, affiché par défaut avec 6 décimales
%8e	affiché sur 8 caractères, dont le point décimal, le E et les signes et 6 décimales
%8.2e	affiché sur 8 caractères, dont le point décimal, le E et les signes et 2 décimales
%hd	short decimal (entier court)

Programmation : entrées/sorties

- *printf :* interprétation
  - Exercice :
    - Qu'affiche le programme suivant ?

```
int a=3,b=16,c;
float z,x,y;
a=b/a;
x=b%a;
x=x+a;
y=b+x/4;
z=x-y;
printf("x=%4.2f\n",x);
printf("y=%4.2f\n",y);
printf("z=%4.2f\n",z);
```

Programmation : entrées/sorties

- Acquisition avec *scanf* :
  - Cette instruction nécessite, comme *printf*, l'inclusion de la bibliothèque « stdio » :
    - `#include <stdio.h>`
  - Elle permet de transférer dans une variable, tout ou partie de l'entrée clavier.
  - Syntaxe:
    - `scanf ("type de la variable",&variable) ;`
  - Exemple :
    - ```
      int qte= 0;
      printf("\nTapez la quantité: ");
      scanf ("%d",&qte) ;
      ```
      ```

Programmation : entrées/sorties

- Acquisition avec *scanf* :
  - Attention :
    - On ne peut lire qu'une valeur à la fois ;
    - Le type de la variable doit correspondre à la valeur attendue ;
    - Il faut ajouter le caractère « & » devant le nom de la variable ;
    - En principe la lecture cesse dès qu'on appuie sur la touche Entrée.

Programmation : entrées/sorties

- Exercices :
  1. Ecrire un programme qui :
     1. Demande à l'utilisateur deux nombres.
     2. Mémorise ces nombres dans deux variables.
     3. Echange les deux valeurs entre les variables.
     4. Affiche le résultat et le nom des variables.
  2. Ecrire un second programme qui :
     1. Demande à l'utilisateur de saisir un nombre réel.
     2. Affiche sa valeur absolue.

Programmation : structures de contrôle – conditions

- Les expressions booléennes :
  - Elles sont construites à l'aide de :
    - variables,
    - valeurs,
    - expressions mathématiques,
    - opérateurs de comparaison (==, >, <, >=, <=, !=),
    - opérateurs booléens (ET, OU, NOT).
  - Le résultat est toujours Vrai (TRUE) ou Faux (FALSE)
  - Exemples :
    - 3 == 5 -> faux
    - 3<=5 -> vrai
    - 5 != 5 -> faux
    - 4 = 4 -> vrai

Programmation : structures de contrôle – conditions

- Les Conditions (Si … alors … sinon …)
  - Syntaxe :
    - Si (valeur booléenne)
    - Alors
      - instruction
    - Sinon
      - instruction

```
if (expression booleénne)
{
    bloc1
}
else
{
    bloc2
}
```

Programmation : structures de contrôle – conditions

- Les Conditions (Si … alors … sinon …)
  - Exemple :
    - Comparer et afficher la plus grande valeur entre 2 variables :

```
int x;
int y;
scanf("%d",&x);
scanf("%d",&y);
if(x<y)
  printf("%d", &y);
else
  printf("%d", &x);
```

Programmation : structures de contrôle – conditions

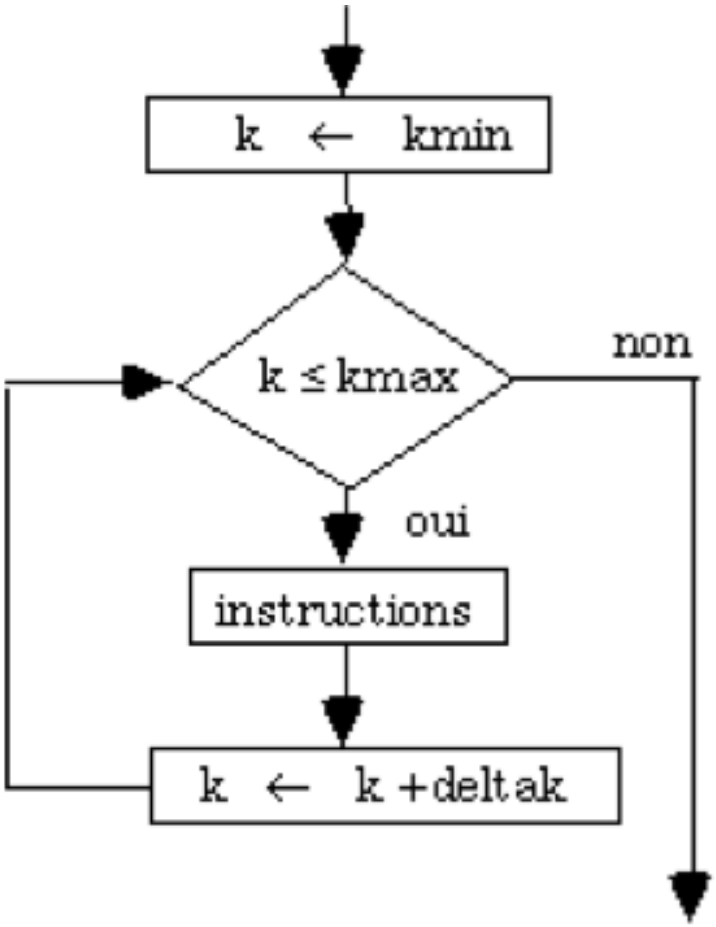
- Exercices :
  1. Demander à l'utilisateur de saisir trois nombres entiers.
     Afficher plus grand et le plus petit des trois.
  2. Demander à l'utilisateur de saisir sa note du bac sur 20 et afficher sa mention.
  3. Demander à l'utilisateur de saisir deux nombres entiers. Afficher si le premier est divisible par le deuxième ou non.

Programmation : structures de contrôle – conditions

- Exercices :
  4. Demander à l'utilisateur de saisir un numéro de ligne de bus. Suivant le numéro saisi, afficher la fréquence de circulation : 10 minutes pour la ligne 15, 8 minutes pour la ligne 22, 12 minutes pour la ligne 35, et le message « Ne passe pas par ici » pour tout autre numéro saisi.
  5. Demander à l'utilisateur un nombre et afficher la lettre de l'alphabet qui correspond (a=1, b=2, etc.).
  6. Demander à l'utilisateur 2 nombres et un operateur (1=+, 2=-, 3=/, 4=* et 5=%) et afficher le résultat.

Programmation : structures de contrôle – boucles

- Les boucles



Programmation : structures de contrôle – boucles

- Les boucles
  - Il existe deux sortes de boucles :
    - Les inconditionnelles, dont on connaît le nombre d'itérations à l'avance :
      - POUR … FAIRE …
    - Les conditionnelles dont la répétition dépend d'une expression booléenne :
      - REPETER… TANT QUE …
      - TANT QUE … REPETER …

Programmation : structures de contrôle – boucles

- Les boucles
  - Quelques règles :
    - Une boucle doit toujours être exécutée un nombre entier de fois : 0, 1 ou plusieurs fois.
    - Il est interdit d'interrompre prématurément l'exécution d'une boucle (il existe forcément une solution alternative).
    - On ne mettra dans une boucle que des instructions qui devront toutes être exécutées le même nombre de fois.

Programmation : structures de contrôle – boucles

- La boucle « pour »
  - On l'utilisera chaque fois qu'on sera certain de toujours pouvoir déterminer avant d'y entrer, le nombre exact de fois qu'elle sera exécutée
  - Exemple :

```
for(init;condition;evolution){
    instruction;
}

for(int i=0;i<10;i=i+1){
    printf("%d",i);
}
```

Programmation : structures de contrôle – boucles

- La boucle « Tant que »
  - On l'utilisera chaque fois :
    - Que l'on ne pourra pas utiliser correctement la boucle Pour
    - Que l'on sera certain qu'il faudra toujours l'exécuter au moins une fois
  - Exemple :

```
do {
    bloc d'instructions
}
while (expression booleénne) ;
```

Programmation : structures de contrôle – boucles

- La boucle « Tant que … Répéter»
  - On l'utilisera chaque fois :
    - Que l'on ne pourra pas utiliser correctement la boucle Pour
    - Que l'on ne sera pas certain qu'il faudra toujours l'exécuter au moins une fois
  - Exemple :

```
while (expression booleénne)
{
    bloc d'instructions
}
```

Programmation : structures de contrôle – boucles

- Exercices :
  1. Demander à l'utilisateur combien font 2 fois 2 et répéter cette question aussi longtemps que la réponse est fausse. Ajouter le message "faux, recommencez" à chaque mauvaise réponse, et "bravo !" pour la bonne réponse.
  2. Demander à l'utilisateur de saisir un nombre réel (y) non-négatif. Calculer approximativement (à 0.001 près) sa racine carrée (x) et l'afficher. Pour ce faire, initialiser x à zéro et l'incrémenter de 0.001 aussi longtemps que $x^2 < y$.
  3. Demander à l'utilisateur un nombre entier (i) entre 1 et 9. Afficher sa table de multiplication. Exemple : pour i=3 afficher "1 fois 3 font 3, 2 fois 3 font 6, 3 fois 3 font 9… »
  4. Demander à l'utilisateur de saisir des notes (entre 0 et 20) et lui expliquer qu'une valeur hors de cet intervalle arrêtera la saisie. Compter les notes saisies. Une fois la saisie terminée, afficher le nombre de notes saisies.

Programmation : structures de contrôle – boucles

- Exercices :
  4. *Emploi du temps.* On suppose que l'EdT est le suivant : 8h maths, 9h anglais, 10h TDM, 11h élec, 14h info, 15h autom. Proposer à l'utilisateur de saisir l'heure (nombre entier) et afficher le nom du cours ou « pas de cours » en fonction de l'heure saisie.
  5. *Prix TTC.* Demander le prix unitaire HT et le nombre d'exemplaires. Calculer et afficher le prix total HT, la TVA et le prix total (TTC) à payer.
  6. *Catalogue et facture pour un produit.* Afficher une liste numérotée de produits avec, pour chacun, son prix unitaire HT. Demander à l'utilisateur de choisir le numéro de produit (un seul) ou zéro s'il ne veut rien. Si le choix est différent de zéro, demander le nombre d'exemplaires souhaités, afficher le nom du produit choisi, calculer et afficher les prix, comme dans l'exercice précédent.

Programmation : structures de contrôle – boucles

- Exercices :
  7. *Catalogue et facture pour plusieurs produits.* Même chose que précédemment, mais on ajoute de nouveaux produits à la facture tant que le choix n'a pas été zéro.
  8. *Dépenses.* Demander à l'utilisateur, combien d'argent il avait initialement et combien de dépenses il a réalisé. Pour chaque dépense, demander le montant et calculer au fur et à mesure leur somme. A la fin, afficher ce qui lui reste.
  9. *Moyenne d'une classe.* Demander à l'utilisateur le nombre d'élèves dans la classe. Pour chaque élève, demander la note. Calculer et afficher la moyenne.

Plan

- Plan du cours :
  - Introduction
    - Introduction aux algorithmes
    - Découverte de la programmation avec Logo
  - Programmation
    - Le langage C
    - Structure d'un programme
    - Variables
    - Premier programme
    - Entrées / sorties
    - Structures de contrôle
      - Conditions
      - Boucles
    - Architecture
      - Fonctions
      - Bibliothèques
    - Structures de données (tableaux)

Programmation : architecture

- Un programme écrit en C, comme un logiciel, est un ensemble d'instructions qui peuvent être regroupées en :
  - Fichiers
  - Fonctions (aussi appelées routines)
  - Bibliothèques

Programmation : architecture – fonctions

- Pourquoi utiliser des fonctions ?
  - Rendre le code plus lisible ;
  - Factoriser les instructions répétitives ;
  - Réutiliser des séquences d'instructions avec des paramètres différents ;
  - Travailler à plusieurs ;
  - Créer des bibliothèques ;
  - Utiliser des bibliothèques existantes.

Programmation : architecture – fonctions

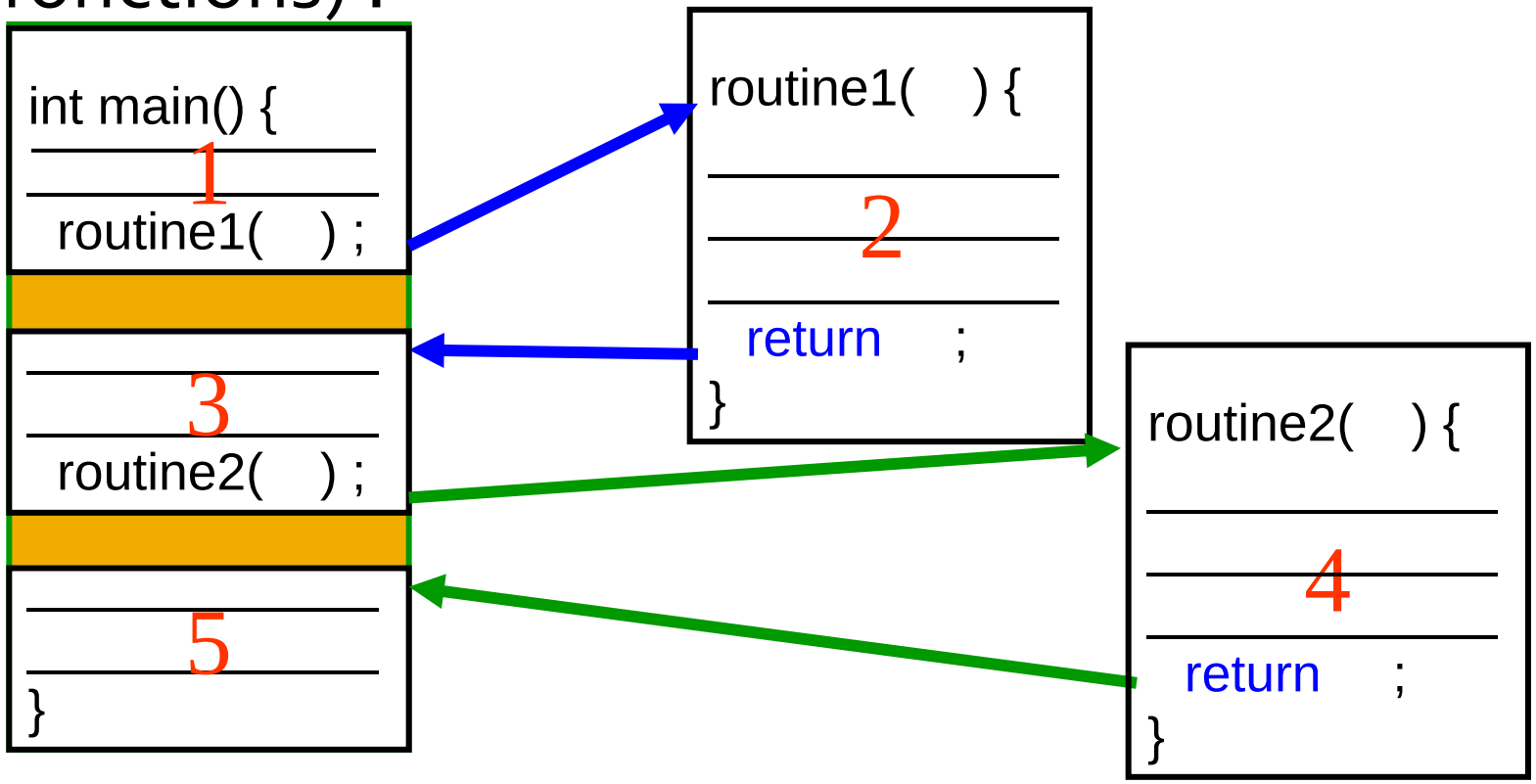
- Pourquoi utiliser des fonctions ?
  - Rendre le code plus lisible ;
  - Factoriser les instructions répétitives ;
  - Réutiliser des séquences d'instructions avec des paramètres différents ;
  - Travailler à plusieurs ;
  - Créer des bibliothèques ;
  - Utiliser des bibliothèques existantes.

Programmation : architecture – fonctions

- Pourquoi utiliser des fonctions ?
  - Rendre le code plus lisible ;
  - Factoriser les instructions répétitives ;
  - Réutiliser des séquences d'instructions avec des paramètres différents ;
  - Travailler à plusieurs ;
  - Créer des bibliothèques ;
  - Utiliser des bibliothèques existantes.

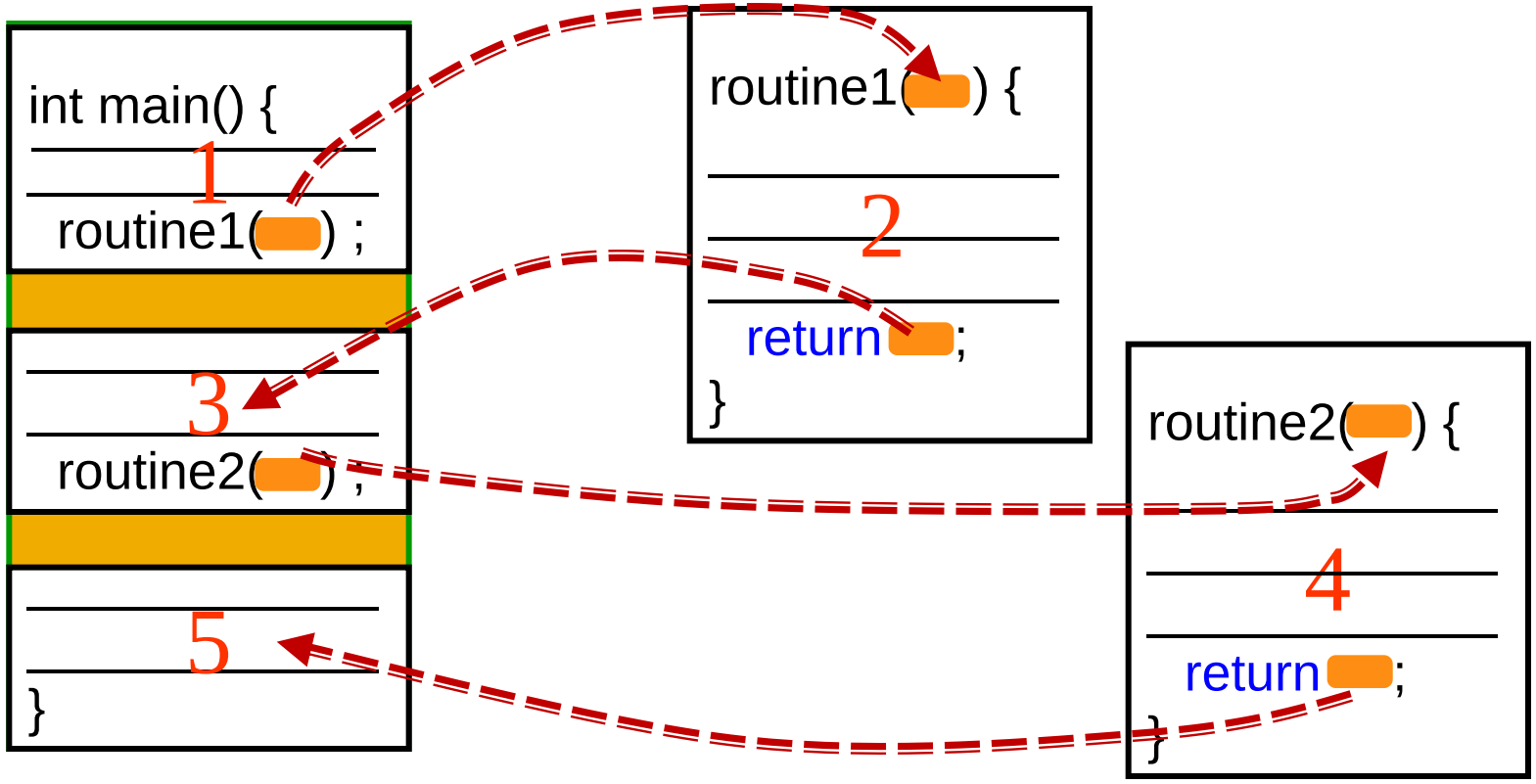
Programmation : architecture – fonctions

- Découpage du code en sous-programmes (les fonctions) :



Programmation : architecture – fonctions

- Passage de paramètre(s) :



Programmation : architecture – fonctions

- Passage de paramètre(s) :
  - pour réaliser ses calculs, le sous-programme a généralement besoin de connaître les valeurs d'un certain nombre de variables : ce sont les paramètres de la fonction.
  - au retour, le programme appelant a besoin de récupérer les valeurs calculées : ce sont les valeurs de retour de la fonction.
  - on évite autant que possible l'usage de variables globales (connues par tous les sous-programmes) !
  - la portée des variables déclarées dans le corps d'une fonction est limitée : elles sont détruites après le retour.

Programmation : architecture – fonctions

- Type de retour :
  - Une fonction retourne toujours une valeur dont le type doit être précisé dans le prototype :
    - Exemple : `int main();`
  - Il est possible de créer des procédures (c'est-a-dire, des fonctions qui ne retournent rien). Dans ce cas, le type de retour est *void.*
    - Exemple : `void direBonjour();`
  - Attention : une fonction ne peut retourner qu'une seule valeur.

Programmation : architecture – fonctions

- Exercice :
  1. Ecrire, sous la forme d'une fonction, un programme qui calcule et affiche la moyenne de 3 nombres aléatoires.
  2. Ecrire un nouveau programme qui calcule et affiche 5 fois la moyenne de 3 nombres aléatoires. Pour cela, utiliser la fonction définie.
  3. Ecrire une nouvelle fonction qui calcule le maximum de 2 flottants passés en paramètres.
  4. Dans le programme principal, afficher la meilleure des moyennes calculées.

Programmation : architecture – bibliothèques

- Une bibliothèque est :
  - Un ensembles de prototypes de fonction ;
  - Des sous-programmes (le corps des fonctions).

- Pour créer et utiliser une bibliothèque, il faut :
  - Ecrire l'entête de la bibliothèque ;
  - L'inclure dans le programme principale (instruction *include*) ;
  - Ecrire le contenu des fonctions.

Programmation : architecture – bibliothèques

- Exemple d'utilisation d'une bibliothèque :

Main.c

```
#include "Biblio.h"

 int main(){
 …
 int f=fonction1();
 …
float g=fonction2(f);
 …
 for(int i=0;i<10;++i)
    f=f+g+fonction1();
}
```

Programme principal

Biblio.h

```
#ifndef _BIBLIO_
#define _BIBLIO_


int fonction1();

float fonction2(int a);



#endif
```

Fichier d'entête (prototypes)

Biblio.c

```
#include "Biblio.h"


int fonction1(){
    return 75;
}

float fonction2(int x){
    float a = 5f;
    return x*x+2-a;
}
```

Implémentation (corps des fonctions)

Bibliothèque

Programmation : architecture – bibliothèques

- Exercice :
  1. Créer et écrire un fichier d'entête d'une bibliothèque nommée « minmaxbib » avec les prototypes des fonctions suivantes
     a. Min de 2 entiers
     b. Max de 2 entiers
     c. Min de 3 entiers (réutiliser la fonction du a.)
     d. Max de 3 entiers (réutiliser la fonction du b.)
     e. Min, Max de 2 et 3 nombres flottants
  2. Créer et écrire l'implémentation de toutes les fonctions
  3. Créer et écrire le programme principal qui permet de calculer trouver le minimum et le maximum de 3 entiers aléatoires (utiliser la fonction *rand()* de la bibliothèque *stdlib*) ainsi que le minimum et maximum de 3 nombres flottants de votre choix.

Plan

- Plan du cours :
  - Introduction
    - Introduction aux algorithmes
    - Découverte de la programmation avec Logo
  - Programmation
    - Le langage C
    - Structure d'un programme
    - Variables
    - Premier programme
    - Entrées / sorties
    - Structures de contrôle
      - Conditions
      - Boucles
    - Architecture
      - Fonctions
      - Bibliothèques
    - Structures de données (tableaux)

Programmation : structures de données – tableaux

- Un tableau est une variable « multiple » dans laquelle on peut gérer un grand nombre d'informations, **toutes de même type**, sous un seul nom de variable.

- Lors de la création d'une variable de type tableau, il faut l'initialiser (c'est-à-dire donner des valeurs pour chaque élément du tableau) car l'initialisation automatique change d'un compilateur à un autre.

Programmation : structures de données – tableaux

- Exemple de tableau :

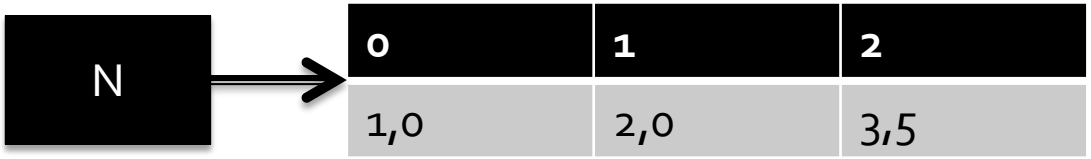
Indices	0	1	2	3	4	5	6
Valeurs	256	4455	1455	157	25	12	-1

  - En langage C, le premier élément du tableau se trouve à l'indice zéro (0) ;
  - Le dernier élément d'un tableau de taille n se trouve donc à la position n-1 !
  - Comme les tableaux sont stockés de manière contiguë en mémoire, il faut faire attention à ne pas lire une « case » qui n'existe pas.

Programmation : structures de données – tableaux

- Déclaration d'un tableau :
  - La taille du tableau est initialisée lors de la déclaration.
  - Exemple en C :
    - `float N [3] = {1.,2.,3.5};`
    - `float N[3];`

      ```
      N[0]=1.;
      N[1]=2.;
      N[2]=3.5;
      ```
  - Représentation en mémoire :



Programmation : structures de données – tableaux

- Exercice (tableaux) :
  1. Créer un programme qui initialise un tableau de 100 éléments avec des entiers aléatoires. Puis, calculer et afficher la moyenne des 100 valeurs contenues dans le tableau.
  2. Créer un programme qui remplit un tableau de 10 nombres flottants saisis par l'utilisateur. Puis, afficher :
     1. Tous les nombres du tableau dans l'ordre de la saisie;
     2. Tous les nombres du tableau dans le sens inverse ;
     3. Le plus grand élément du tableau (maximum) et son indice

Programmation : structures de données – tableaux

- Exercice (tableaux et fonctions) :
  1. Ecrire une fonction de remise à zéro des élement d'un tableau dont le prototype est :

     ```
     void raz(int[] tab, int taille);
     ```
  2. Ecrire une fonction qui initialise un tableau avec un nombre d'entier aléatoire passé en paramètre.
  3. Ecrire une fonction : `void affiche(int[] tab, int taille);`
  4. Ecrire un programme qui :
     1. Demande à l'utilisateur une taille de tableau ;
     2. Crée et affiche un tableau de nombres aléatoires de cette taille ;
     3. Remet le tableau à zéro et l'affiche.

Programmation : structures de données – tableaux

- Exercice (tableaux ,fonctions et bibliothèque) :
  1. Créer une bibliothèque pour la gestion de tableaux de 100 entiers maximum contenant les fonctions :
     a. Initialisation de la taille d'un tableau avec remise à zéro.
     b. Remplissage aléatoire.
     c. Remplissage par l'utilisateur.
     d. Fonctions minimum, maximum et moyenne des éléments.
     e. Comptage du nombre d'occurrences.
     f. Tableau des indices des occurrences.
     g. Tri dans l'ordre croissant.
  2. Ecrire un programme qui :
     a. Crée deux tableaux dont la taille est donnée par l'utilisateurs. L'un est rempli pas l'utilisateur, l'autre par des nombres aléatoires ;
     b. Calcule et affiche les min, max et moyenne des deux tableaux ;
     c. Demande une valeur à chercher à l'utilisateur, et affiche le nombre et la position des occurrences trouvées.
     d. Affiche les deux tableaux triés.