Cours 4:

Structure de décision: construction ***if*/*else*** et ***switch***

Prof. Mohamed Ibrahim, PhD, Jr. Ing.
University of Ottawa

La structure de décision *if/else*

- La structure de `if/else` nous permet de spécifier un ensemble d'instructions à exécuter si l'expression logique est **vraie** et un autre ensemble d'instructions si l'expression logique est fausse.
- La syntaxe pour isoler les commandes seules de C:

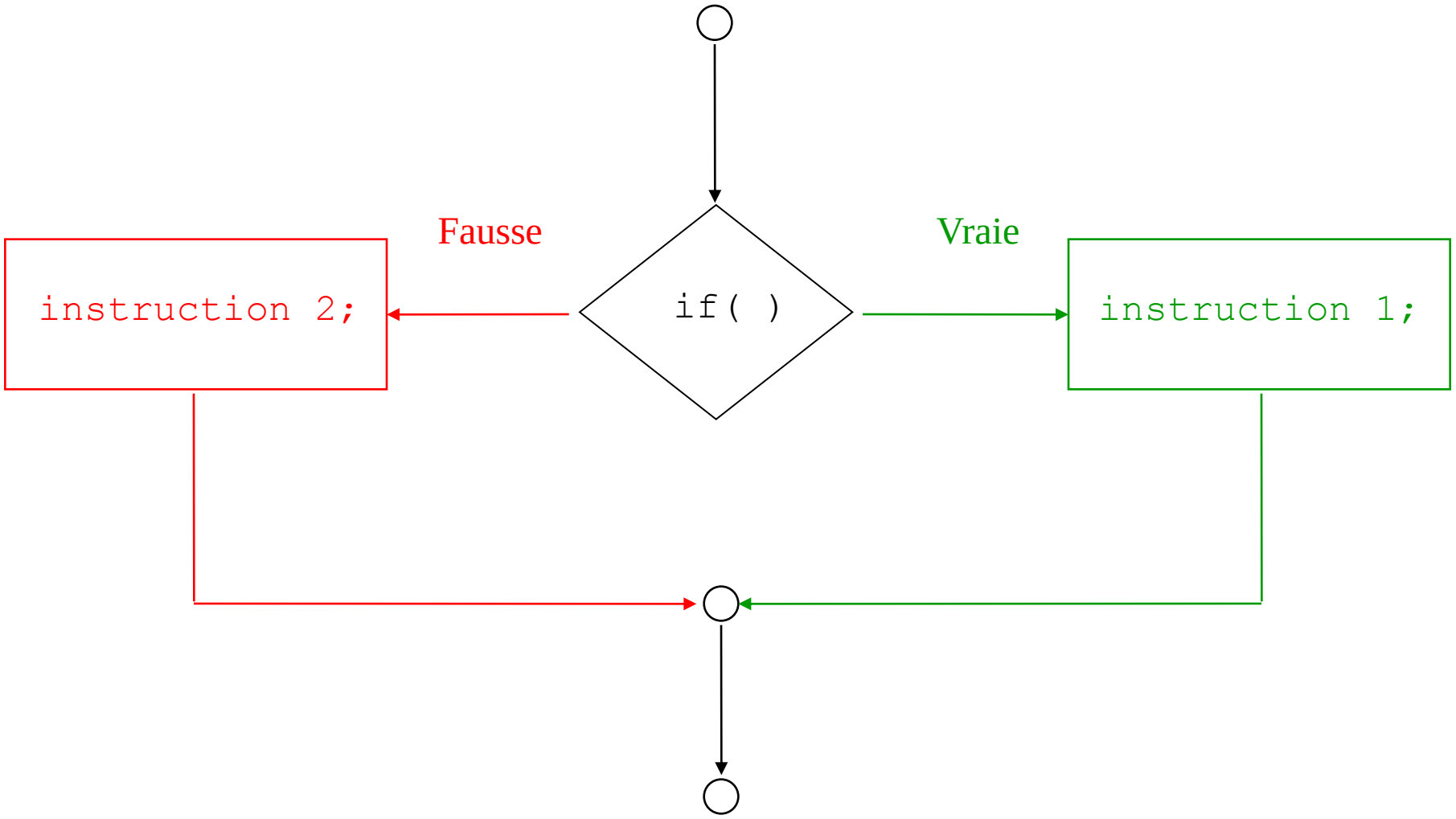
```
if (expression logique)
  instruction 1;
else
  instruction 2;
```

- La syntaxe pour isoler les commandes multiples de C:

```
if (xpression logique)
{
    instruction 1;
    instruction n;
}
else
{
    instruction 1;
    instruction n;
}
```

Attention, il n'y a pas un ; ici.

Organigramme de la structure de décision `if/else`:



- Exemple:

```
if (note >= 90)
    printf("Tu as A+.\n");
else
{
    printf("moins que 90.\n");
    printf("moins que A+.\n");
}
```

→ Exécutée si note est supérieure ou égale à 90.

→ Exécutée si note est inférieure à 90.

- Nous pouvons aussi avoir d'autres structures `if/else` à l'intérieur d'une structure `if/else`:

```
if (x > y)
    if (y < z)
        k = k + 1;
    else
        m = m + 1;
else
    j = j + 1;
```

→ Exécutée si x>y et y<z

→ Exécutée si x>y et y≥z

→ Exécutée si x≤y

- **REMARQUE:** *sinon* est associée à <u>la plus récente</u> *si* indépendamment de l'indentation. La seule exception est, comme toujours, les parenthèses qui dictent la priorité. L'indentation consiste à ajouter des tabulations au début de certaines lignes d'un programme afin d'en améliorer la lisibilité.

- L'utilisation des {} est fortement recommandée dans une structure logique complexe pour clarifier la portée prévue de `if` et `else` afin d'éviter des erreurs de logique de programmation.
  - Considérons la structure suivante avec l'indentation trompeuse:

```
if (x > y)
    if (y < z)
        k = k + 1;
else
    j = j + 1;
```

Executée si x>y et y<z.

Executée si x>y et y≥z;
sinon x≤y comme le programmeur a écrit.
Remarque : *sinon* est toujours associé au plus récente *si*.

  - Le programmeur voulait:

```
if (x > y)
{
    if (y < z)
    k = k + 1;
}
else
    j = j + 1;
```

Executée si x>y et y<z.

Executée si x≤y.

Exercice 1

Écrire un programme qui demande à l'utilisateur d'entrer la température en degrés centigrades. Le programme doit ensuite imprimer "Température de congélation" si la température est inférieure à zéro, "Température Normale" si la température est entre 0 et 32, " Température trop chaud" si la température est supérieure à 32.

Solution de l'exercice 1

```
#include <stdio.h>
int main()
{
    float temperature;
    printf("Entrez la temperature en degres centigrades: ");
    scanf("%f",&temperature);
    if(temperature < 0)
        printf("Temperature de congelation\n");
    else if((temperature >= 0) && (temperature <= 32))
        printf("Temperature normale\n");
    else
        printf("Temperature trop chaude\n");

    system("PAUSE");
    return 0;
}
```

La structure de décision suivante est généralement rencontrée et est créée à partir de plusieurs niveaux de `if/elses` imbriqués:

```
if (choice == 1)
  printf("Premier choix sélectionné.\n");
else if (choice == 2)
  printf("Deuxième choix sélectionné.\n");
else if (choice == 3)
  printf("Troisième choix sélectionné.\n");
else
  printf("Mauvaise sélection!\n");
```

- La variable `choice` est comparée avec les valeurs entières de 1 à 3.
- Si une expression logique est vraie (par exemple la première) alors le message qui suit est imprimé à l'écran (`Premier choix sélectionné`.) et l'exécution continuera avec la commande après la structure; c'est-à-dire, `printf("Mauvaise sélection!\n");`
- La dernière `else (Mauvaise sélection!)` est exécutée seulement si aucune de `if`s n'est vraie.
- La susdite est une structure de décision si populaire que les programmeurs ajoutent rarement l'indentation et les {}. Exceptionnellement, ceci n'est pas considéré comme un mauvais style de programmation.

Exercice 2

Écrire un programme qui demande un utilisateur à entrer la note d'un étudiant et imprime la marque de lettre équivalente. Utilisez le système de notation suivante:

Note entre 90 et 100 => A
Note entre 80 et 90 => B
Note entre 70 et 80 => C
Note entre 60 et 70 => D
Note inférieure 60 => F
Sinon => imprime: “Cette note est invalide!”

Solution de l'exercice 2

```
#include <stdio.h>
int main()
{
    float noteEtudiant;
    /*Demander d'entrer la note a l'utilisateur*/
    printf("Entrez la note de l'etudiant: ");
    scanf("%f", &noteEtudiant); /*Lire la note */
    if((noteEtudiant >= 90) & (noteEtudiant <= 100))
        printf("La note est: A\n");
     else if((noteEtudiant >= 80) & (noteEtudiant <= 90))
        printf("La note est: B\n");
      else if((noteEtudiant >= 70) & (noteEtudiant <= 80))
        printf("La note est: C\n");
    else if((noteEtudiant >= 60) & (noteEtudiant <= 70))
        printf("La note est: D\n");
    else if((noteEtudiant >= 0) & (noteEtudiant < 60))
        printf("La note est: F\n");
    else
        printf("Cette note est invalide!\n");

    system("PAUSE");
    return 0;
}
```

Attention aux structures logiques `if/else` imbriquées

- suposons que nous avons la structure de décision suivante:

```
if (note >= 90)
  printf(“Vous avez un A+\n”);
else if (note >= 85)
  printf(“Vous avez un A\n”);
```

- Qu’arrive-t-il, si nous avons?

```
if (note >= 85)
  printf(“Vous avez un A\n”);
else if (grade >= 90)
  printf(“Vous avez un A+\n”);
```

- Vous voyez le problème? Qu’arrive-t-il arrive si la note est égal à 90?
  - `Vous avez un A`

La structure de décision *switch*

- La structure de décision `switch` est généralement utilisée quand une variable doit être comparée à de nombreuses valeurs différentes et une action appropriée est prise lorsqu'un cas est vrai.
- Exemple de la syntaxe (la variable `choice` est déclarée `int`):

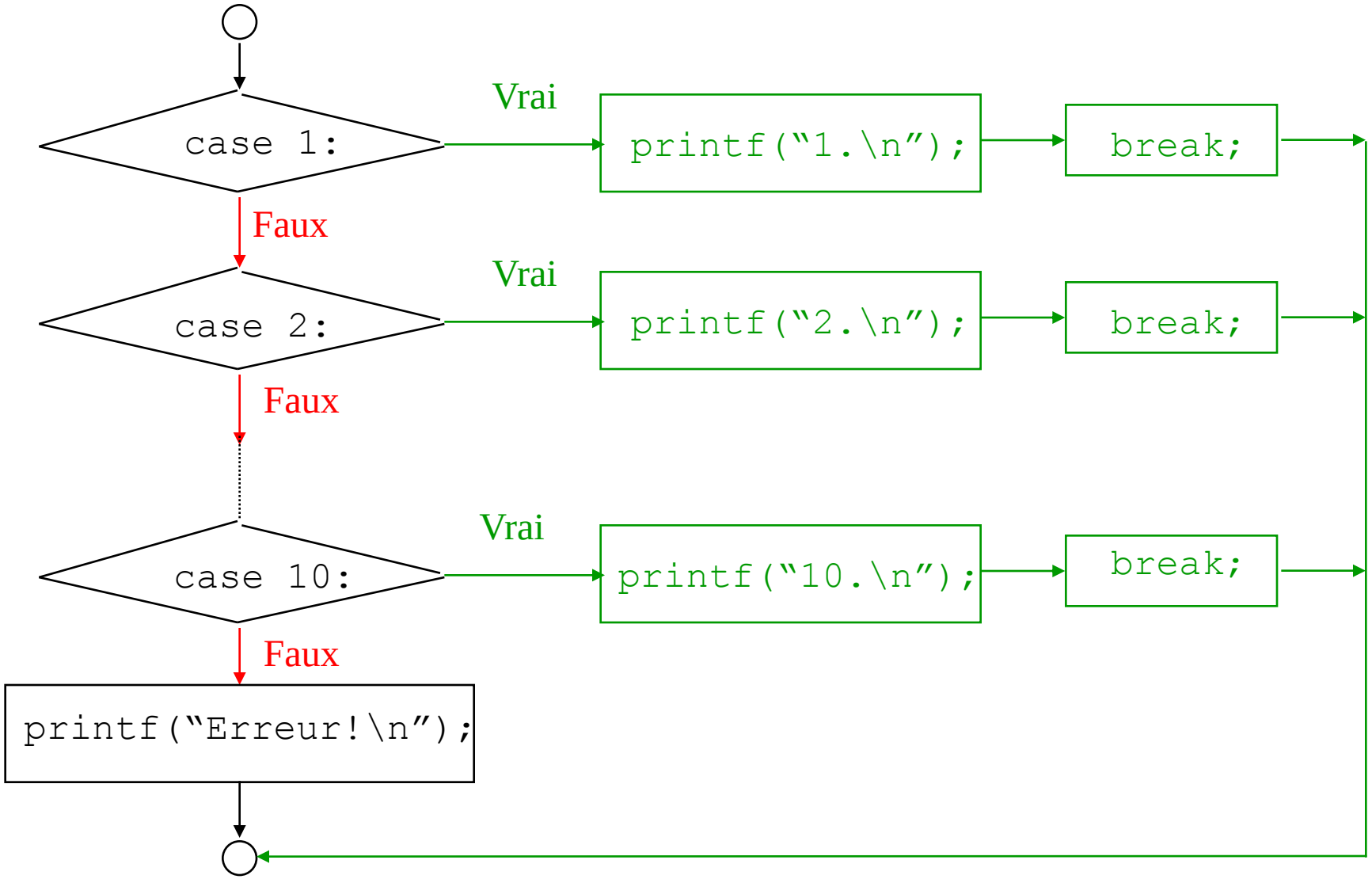
```
switch (choice)
{
  case 1:
    printf("1.\n");
    break;

  case 2:
    printf("2.\n");
    break;

  default:
    printf("Erreur!\n");
    break;
}
```

- La structure de `switch` compare une expression à une valeur et exécute la commande après le premier `case` qui révèle être vrai.
- La commande `break;` force l'exécution du programme de reprendre après la structure `switch`
  - Si `break;` n'est pas inclus, alors toutes les commandes qui se produisent après le premier `case` vrai seront exécutées.
- Si aucun `case` est vrai, alors les commandes après `default` seront exécutées.
- Les `{}` ne sont pas exigé dans `case` puisque toutes les commandes seront exécutées jusqu'à `break;` ou la fin de la structure `switch`.
- REMARQUE: l'expression dans `switch` peut être seulement de type integer, char, short et long. Les types **Float and double ne sont pas permis**

Organigramme de la structure de décision `switch`:



Exercice 3

Écrire un programme qui demande à l'utilisateur d'entrer une valeur de type entier comprise entre 1 et 5. Le programme doit imprimer le mot correspondant du nombre entré par l'utilisateur. Par exemple, si l'utilisateur entre le nombre 4, le programme doit afficher : 'Quatre'.

Solution de l'exercice 3

```
#include <stdio.h>
int main()
{
    int nombre;
    printf("Entrez un nombre entier: ");
    scanf("%d", &nombre);
    switch (nombre)
    {
        case 1:
        printf("Un\n");
        break;
        case 2:
        printf("Deux\n");
        break;
        case 3:
        printf("Trois\n");
        break;
        case 4:
        printf("Quatre\n");
        break;
        case 5:
        printf("Cinq\n");
        break;
        default:
        printf("Erreur!\n");
        break;
    }
    system("PAUSE");
    return 0;
}
```

Les opérateurs d'affectation = et de comparaison d'égalité ==

Attention lors de l'utilisation des opérateurs d'affectation = et de comparaison d'égalité ==. Une interchangeabilité ne provoque pas une erreur de compilation mais une erreur d'exécution qui peut être difficile à détecter.

- Exemple:

code souhaité
```
if (var1 == 2)
printf("…\n");
```
Vraie seulement lorsque la variable var1 est égale à 2.

code écrit par erreur
```
if (var1 = 2)
printf("…\n");
```
L'affectation est toujours vrai.

- Une autre erreur très courante:

code souhaité
```
var1 = 2;
printf("…\n");
```
On assigne la valeur entière 2 à la variable var1.

code écrit par erreur
```
var1 == 2;
```
L'expression logique est évaluée lors de l'exécution mais aucune affectation n'est réalisée.