

The background is a complex orange-toned abstract design. It features a grid of small circles in the upper left, a cluster of larger, rounded, bubble-like shapes in the middle left, and a dense, tangled web of thin lines in the lower half. On the right side, there is a large, faint gear with a bee inside it. The title text is centered in the middle of the slide.

STRUCTURES DE PROGRAMMATION

Instruction de décision

```
if (expression_booléenne)
```

```
    instruction;
```

```
int a, b;
```

```
cout << " Entrer deux entiers: ";
```

```
cin >> a >> b;
```

```
if (a == b)
```

```
    cout << a << " est égal à " << b;
```

Instruction de décision

```
if (expression_booléenne)
```

```
    instruction_if;
```

```
else
```

```
    instruction_else;
```

```
int a, b;
```

```
cout << " Entrer deux entiers: ";
```

```
cin >> a >> b;
```

```
if (a == b)
```

```
    cout << a << " est égal à " << b;
```

```
else
```

```
    if (a < b)
```

```
        cout << a << " est plus petit que " << b;
```

```
    else
```

```
        cout << a << " est plus grand que " << b;
```

Instruction de décision

```
switch (expression) {  
    case constante_1 : instruction_1;  
                        break;  
  
    case constante_2 :  
    case constante_3 : instruction_2_3;  
                        break;  
  
    ...  
  
    case constante_x : instruction_x;  
                        break;  
  
    ...  
  
    default           : instruction;  
}
```

Instruction de décision

```
char note;  
int points = 0;  
cout << "Note = "; cin >> note;  
switch (note) {  
    case 'A' : points +=4;  
    case 'B' : points +=3;  
    case 'C' : points +=2;  
                break;  
  
    case 'D' : points = 1;  
                break;  
  
    default  : points = 0;  
}  
cout << " Cette note vaut " << points;
```

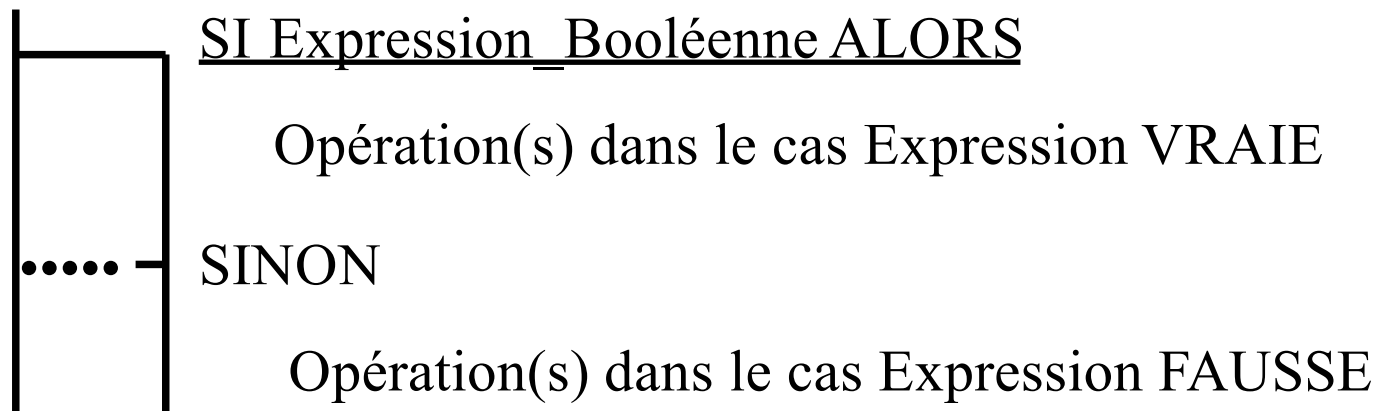
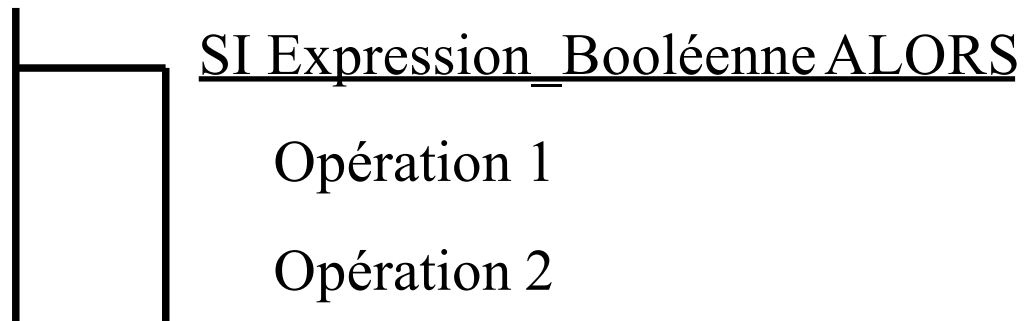
```

#include <iostream> // Pour l'utilisation de cin et cout
#include <string>    // Pour l'utilisation de chaînes de caractères
using namespace std;
int main (void)
{
    const string BISSEXTILE = " est une année bissextile. ";
    const string PASBISSEXTILE = " n'est pas une année bissextile. " ;
    int annee;
    string statut;
    cout << "Inscrire l'année dont vous désirez" << endl;
    cout << "connaître la nature (bissextile ou non) => ";
    cin >> annee;
    if (annee % 4 !=0)        // Année non un multiple de 4
        statut = PASBISSEXTILE;
    else
        if (annee % 100 != 0)        // Année multiple de 4 mais
            statut = BISSEXTILE;    // pas un multiple de 100
        else
            if (annee % 400 != 0)    // Année multiple de 4 et de 100
                statut = PASBISSEXTILE;    // mais pas de 400
            else
                statut = BISSEXTILE; // Multiple de 4, 100 et 400
    cout << endl << annee << statut;
    return 0;
}

```

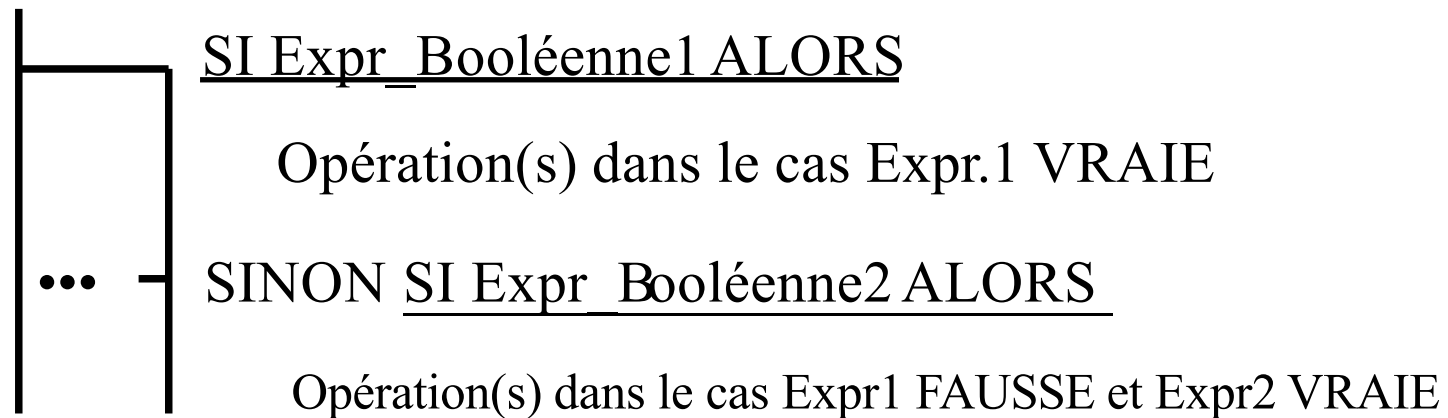
Algorithme

Structures décisionnelles en pseudo-code schématique



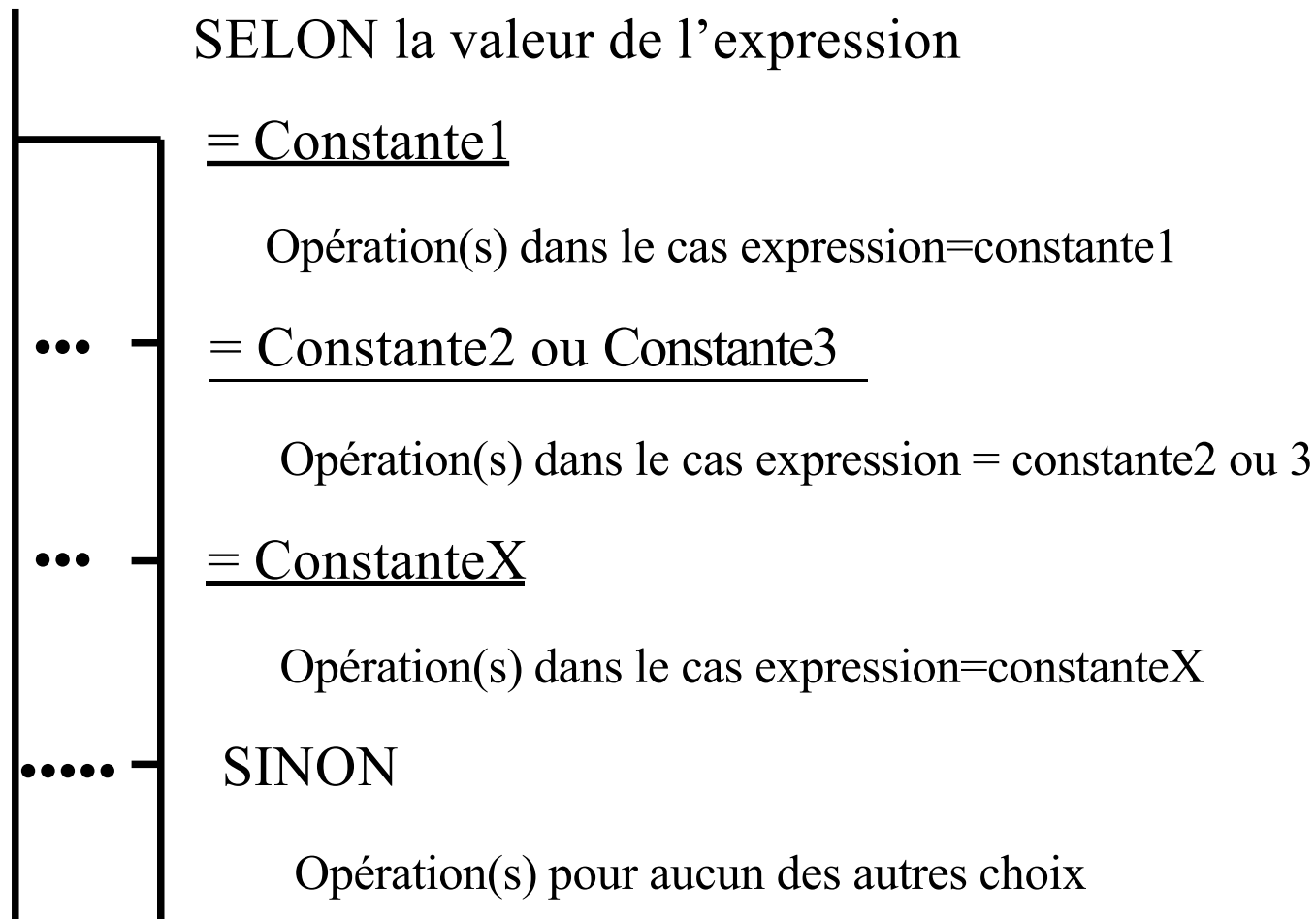
Algorithme

Structures décisionnelles en pseudo-code schématique



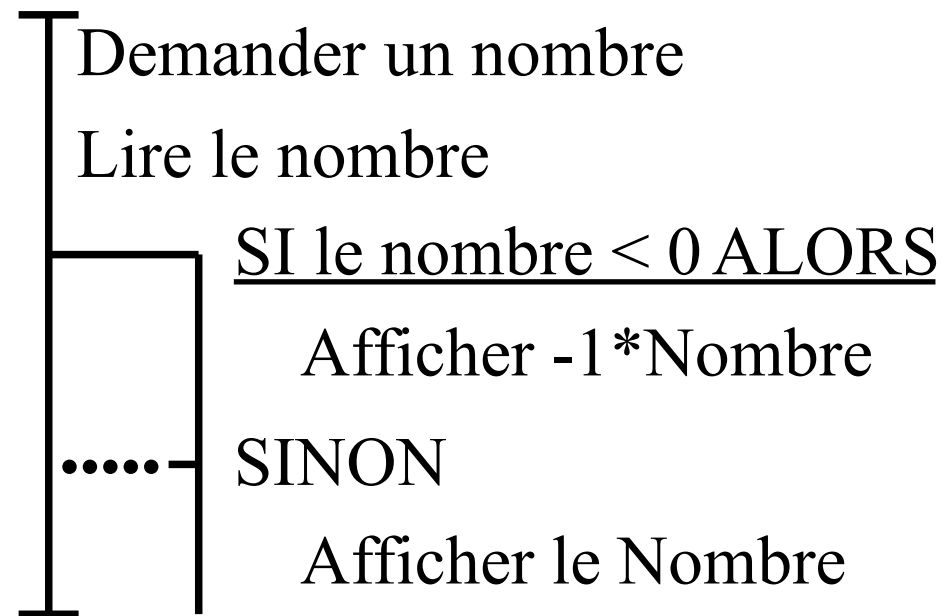
Algorithme

Structures décisionnelles en pseudo-code schématique



Algorithme

- Lire un nombre et afficher la valeur absolue de ce nombre



Instruction de répétition

```
while (expression_booléenne)  
    instruction;
```

Points clés:

- une initialisation
- critère d'arrêt
- instruction affectant l'expression booléenne

Instruction de répétition

```
do  
  instruction;  
while (expression_booléenne);
```

Points clés :

- critère d'arrêt
- instruction affectant l'expression booléenne
- le corps de la boucle est exécuté au moins une fois

Instruction de répétition

for (initialisation;^① expression^②_booléenne; actualisation^④)
instruction;^③

- ① L'initialisation est effectuée qu'une seule fois au départ.
- ② L'expression booléenne est vérifiée, si elle est vraie l'instruction ③ est exécutée, autrement la boucle est terminée.
- ③ L'instruction à répéter; une fois terminée on passe à l'actualisation.
- ④ L'actualisation doit permettre la modification de l'expression booléenne.
ensuite...on boucle 2-3-4-2-3-4-2-3-4-...

Instructions de répétition

while	do-while	for
Il faut tester la condition avant l'exécution de l'instruction.	Il faut exécuter l'instruction au moins une fois avant que la condition soit testée.	Idéalement l'instruction n'affecte pas la condition.
Le nombre de répétitions de l'instruction n'est pas connu d'avance.	Le nombre de répétitions de l'instruction n'est pas connu d'avance.	Le nombre de répétitions de l'instruction est généralement connu d'avance.

Exemple While

```
char reponse = 'o';
int nombre;
int somme = 0;

cout << "Voulez vous entrer un nombre ? (o/n) ";
cin >> reponse;
while (reponse == 'o')
{
    cout << "Rentre un nombre entier: ";
    cin >> nombre;
    somme += nombre;
    cout << "Voulez vous entrer un nombre ? (o/n) ";
    cin >> reponse;

}
cout << "La somme est:" << somme << endl;
```

Si la réponse est autre chose que o, la boucle ne sera jamais exécutée

Changement de l'expression booléenne

Exemple Do While

```
char response='o';  
int nombre;  
int somme=0;  
  
do  
{  
    cout << "Rentrer un nombre entier: ";  
    cin >> nombre;  
    somme += nombre;  
  
    cout << "Voulez vous entrer un autre nombre ? (o/n) ";  
    cin >> response;  
}  
while(response=='o');  
cout << "La somme est:" << somme << endl;
```

Au moins une fois

Expression vérifiée à la fin

Exemple For

```
string phrase = "il pleut, il vente, il neige et il gresille";
```

```
int occurrence = 0;
```

```
char lettre = 'e';
```

```
int i ;
```

```
for ( i=0 ; i < phrase.size(); i++ )
```

```
{
```

```
    if (phrase[i] == lettre)
```

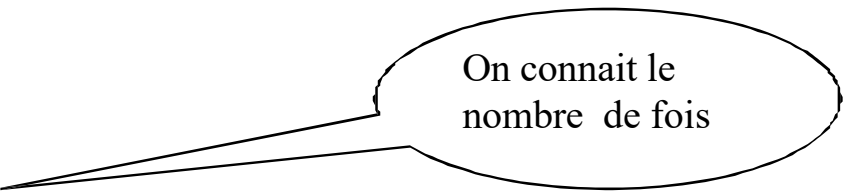
```
        occurrence++;
```

```
}
```

```
cout << "Le nombre occurrence de la lettre "
```

```
    << lettre << '\t' << occurrence
```

```
    << endl;
```



On connaît le
nombre de fois

Les tableaux

- ☐ Une structure homogène constituée d'un nombre déterminé d'éléments de même type.
- ☐ On peut repérer chaque élément à l'aide d'un indice qui sert à indiquer sa position.
- ☐ Déclaration
 - `type_des_elts tableau[dim1][dim2]...[dimN];`

Déclaration d'un tableau

```
int liste[3]= {1,2,3};
```

```
int i;
```

```
for ( i = 0; i < 3; i++)
```

```
    cout << liste[i]<< endl;
```

1

2

3

Les tableaux

```
int unTableau [10] [5] [15];
```

- On ne peut pas effectuer de lecture, d'affichage, de comparaison ni d'autres opérations sur des tableaux complets (sauf les chaînes de caractères)
- Le premier élément d'un tableau est toujours à l'indice 0
- Accès aux éléments d'un tableau
 - `unTableau[3][2][5] = 12;`

Les tableaux

- `int classe[6] [75];`

Ce tableau contient 450 éléments (6×75). Cependant pour accéder à un élément de ce tableau, l'indice de la première dimension doit être compris entre 0 et 5, l'indice de la deuxième dimension doit être compris entre 0 et 74.

***I
M
P
O
R
T
A
N
T***

Tableau

```
int matrice[6][7];
```

The diagram shows a 6x7 matrix with the following values:

0	1	0	1	0	1	0
0	1	0	1	1	1	1
1	1	1	1	0	0	0
0	0	0	0	0	0	1
1	1	1	1	1	0	0
0	0	0	1	1	1	1

Three elements are highlighted with arrows:

- `matrice[2][2]` points to the value 1 in the third row, third column.
- `matrice[1][6]` points to the value 1 in the second row, seventh column.
- `matrice[5][0]` points to the value 0 in the sixth row, first column.

Les tableaux

- Il n'est pas possible de manipuler le tableau dans son entité.
- Pour ces opérations, il faut accéder à chacun de ses éléments.
 - ▮ Affectation d'un tableau à un autre
 - ▮ Comparaison de deux tableaux
 - ▮ Lecture et écriture

Traitement des éléments d'un tableau

```
double vecteurA[10], vecteurB[10], norme;  
bool pareil;  
int indice;
```

- Initialisation du vecteur^y
for (indice = 0; indice < 10; indice++)
 vecteurA[indice] = 0;
- Affectation d'un tableau à un autre
for (indice = 0; indice < 10; indice++)
 vecteurB[indice] = vecteurA[indice];

Traitement des éléments d'un tableau

- Comparaison de deux tableaux

```
pareil = true;  
for (indice = 0; indice < 10 && pareil; indice++)  
    if (vecteurA[indice] != vecteurB[indice])  
        pareil = false;
```

- Calculer la norme d'un vecteur

```
norme = 0;  
for (indice = 0; indice < 10; indice++)  
    norme += pow(vecteurA[indice], 2);  
norme = sqrt(norme);
```