

Chapitre 2: Introduction au C++

S1-SupNum

Les types

En C++, chaque variable doit avoir un **type** qui définit quelle sorte de donnée elle peut stocker.

```
int age = 25;           // Entier standard
long grand = 1000000;   // Entier long
unsigned int positif = 50; // Sans signe (positif uniquement)
float prix = 19.99;      // Précision simple (7 chiffres)
double distance = 384400.5; // Précision double (15 chiffres)
char lettre = 'A';       // Un seul caractère
bool estActif = true;    // Vrai ou faux
string nom = "Ali";      // Chaîne de caractères
```

Déclaration et Initialisation

```
// Déclaration simple  
int nombre;
```

```
// Déclaration avec initialisation  
int x = 10;
```

```
// Plusieurs variables  
int a, b, c;  
int d = 5, e = 10, f = 15;
```

Les Constantes

```
const double PI = 3.14159;  
const int JOURS_SEMAINE = 7;
```

Opérateurs Arithmétiques

```
int a = 10;  
int b = 3;  
  
int addition = a + b;    // 13  
int soustraction = a - b; // 7  
int multiplication = a * b; // 30  
int division = a / b;    // 3 (division entière)  
int modulo = a % b;      // 1 (reste)
```

Opérateurs d'affectation

```
int nombre = 5;  
  
nombre += 3; // nombre = nombre + 3 → 8  
nombre -= 2; // nombre = nombre - 2 → 6  
nombre *= 4; // nombre = nombre * 4 → 24  
nombre /= 6; // nombre = nombre / 6 → 4  
nombre %= 3; // nombre = nombre % 3 → 1
```

Incrémentation et Décrémentement

```
int compteur = 5;
```

```
compteur++; // Incrémente de 1 → 6
```

```
compteur--; // Décrémente de 1 → 5
```

Opérateurs de Comparaison

```
int x = 10;
```

```
int y = 20;
```

```
x == y // false (égal à)
```

```
x != y // true (différent de)
```

```
x > y // false (supérieur à)
```

```
x < y // true (inférieur à)
```

```
x >= y // false (supérieur ou égal)
```

```
x <= y // true (inférieur ou égal)
```

Opérateurs logique

```
bool a = true;
```

```
bool b = false;
```

```
a && b // false (ET logique - AND)
```

```
a || b // true (OU logique - OR)
```

```
!a // false (NON logique - NOT)
```

Structure d'un Programme

Tout programme C++ suit une structure précise et organisée.

```
#include <iostream>

int main() {

    // Votre code ici

    return 0;
}
```

Éléments obligatoires :

- Directive **#include**
- Fonction **main()**
- Instruction **return 0**

Lecture et Écriture

cout // pour afficher (sortie)

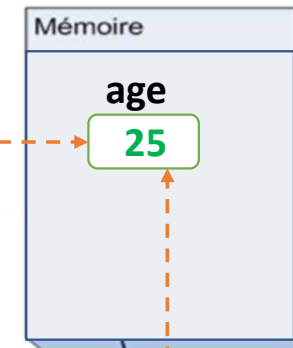
cin // pour lire (entrée)

Exemple de programme

```
#include <iostream>
using namespace std;
int main()
{
    int age; //On prépare une case mémoire pour stocker un entier.

    cout << "Quel age avez-vous ?" << endl;
    cin >> age; //On fait entrer un nombre dans cette case.
    cout << "Vous avez " << age << " ans !" << endl; //Et on l'affiche

    return 0;
}
```



Execution

```
Quel age avez-vous ?
25
Vous avez 25 ans !
```

Langage compilé/Interprété

Avant l'exécution
(en bloc)

1. LANGAGES COMPILÉS


CODE SOURCE
mon_programme.cpp




EXÉCUTION

Ex: C, C++,

Avantages

- Performance+
- Indépendance du compilateur
- Détection erreurs précoce

Inconvénients

- Temps de compilation
- L'exécutable
- Développement plus long

2. LANGAGES INTERPRÉTÉS


CODE SOURCE
mon_script.py



AFFICHE RÉSULTAT

Ex: Python, JavaScript, PHP

Avantages

- Portabilité
- Rapidité dev++
- Flexibilité
- Cycle de développement plus long

Inconvénients

- Performance—
- Détection erreurs à l'exécution
- Dépendance interpréteur

pendant l'exécution
(ligne/ligne)

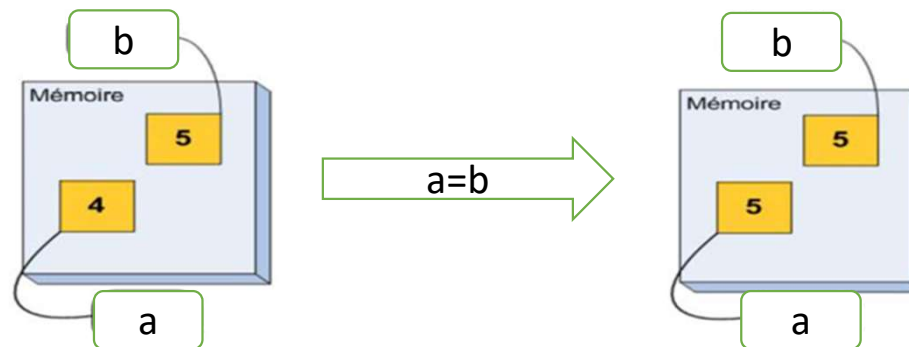
Affectation

```
#include <iostream>
using namespace std;
int main()
{
    int a=4, b=5; //Déclaration et initialisation.

    cout << "a vaut : " << a << " et b vaut : " << b << endl;
    cout << "Affectation !" << endl;
    a = b; //Affectation de la valeur de 'b' à 'a'.
    cout << "a vaut : " << a << " et b vaut : " << b << endl;

    return 0;
}
```

Exécution:



Incrémentation / Décrémentation

++ Incrémentation

-- décrémentation

Exemple 1 :

A++ ; A=A+1 ; ou A+=1 sont équivalentes.

A-- ; A=A-1 ; ou A-=1 sont équivalentes.

Exemple 2 :

Opérateur	Affectation	Description	Exemple a=5
++	X=++a	a=a+1, X=a	a=6, X=6
	X=a++	X=a, a=a+1	X=5, a=6
--	X=--a	a=a-1, X=a	a=4, X=4
	X=a--	X=a, a=a-1,	X=5, a=4,

Bibliothèque ctype

Fonctions	Description	Exemple	
isalpha (int argument)	Teste si l'argument est compris entre 'A' et 'Z' ou 'a' et 'z'	isalpha('d') vaut VRAI	 Booléennes
isdigit (int argument)	Teste si l'argument est compris entre '0' et '9'	isdigit('2') vaut VRAI	
isalnum (int argument)	Teste si l'argument est alphanumérique (lettre ou chiffre)	isalnum('X') vaut VRAI	
islower (int argument)	Teste si l'argument est compris entre 'a' et 'z'	islower('a') vaut VRAI	
isupper (int argument)	Teste si l'argument est compris entre 'A' et 'Z'	isupper('T') vaut VRAI	
tolower (int argument)	Convertit un caractère en minuscule	tolower('B') vaut 'b'	 Fonctions de transformation
toupper (int argument)	Convertit un caractère en majuscule	toupper('a') vaut 'A'	

Exemple:

```
char c = 'A';

if(isalnum(c))
{
    cout << "Alphanumérique" << endl;
}
```

```
char c = 'A';

if(isalpha(c))
{
    cout << " c'est une lettre" << endl;
}
```

```
char c = '8';

if(isdigit(c))
{
    cout << " c'est un chiffre" << endl;
}
```

```
char c = 'a';

if(islower(c))
{
    cout << "c'est une minuscule" << endl;
}
```

```
char c = 'A';

char min=tolower(c); // a
```

```
char c = 'a';

char maj=toupper(c); // A
```

Exemple d'utilisation:

Compléter le programme ci-après
en remplaçant les numéros 1-6
par les fonctions appropriées.

1:

2:

3:

4:

5:

6:

```
#include <iostream>
#include <cctype>

using namespace std;

int main() {
    char caractere;

    cout << "Entrez un caractere : ";
    cin >> caractere;

    if (1) {
        cout << "-> C'est une LETTRE" << endl;

        if (2) {
            cout << "-> C'est une MAJUSCULE" << endl;
            cout << "-> Sa minuscule est : '" << 3 << "'" << endl;
        }
        else if (4) {
            cout << "-> C'est une MINUSCULE" << endl;
            cout << "-> Sa majuscule est : '" << 5 << "'" << endl;
        }
    }
    else if (6) {
        cout << "-> C'est un CHIFFRE" << endl;
    }
    else {
        cout << "-> C'est un AUTRE CARACTERE" << endl;
    }

    return 0;
}
```

Tableau récapitulatif – Opérations sur les chaînes en C++

Opération	Opérateur	Description	Exemple
Affectation	=	Remplace complètement le contenu de la chaîne cible par la chaîne source.	chaîne = "Bonjour";
Ajout	+=	Ajoute du texte à la fin de la chaîne existante <i>(modifie directement la variable).</i>	chaîne += 'le monde';
Concaténation	+	Construit une nouvelle chaîne en combinant deux chaînes.	total = chaîne1 + chaîne2;

Note importante Pour utiliser le type `<string>` et ces opérateurs, il faut inclure la bibliothèque suivante tout en haut du fichier :

```
#include <string>
```

Fonctions sur les chaînes de caractères (suite)

Fonction / Opérateur	Description	Exemple	Résultat / Remarque
Comparaison ==, !=, <, >, <=, >=	Compare le contenu des chaînes alphabétiquement (ordre lexicographique).	if (s1 == "mot") if (s1 < s2)	Renvoie un booléen (true ou false). ⚠ Sensible à la casse.
Taille s.size()	Renvoie le nombre de caractères dans la chaîne.	string s = "Salut"; int len = s.size();	Renvoie un entier (ex : 5).
Recherche s.find(texte)	Cherche la première position (index) où apparaît un texte.	string s = "Bonjour"; int pos = s.find("jour");	Renvoie l'index (ex : 3). Si non trouvé :

Exercice :

Objectif : Écrire un programme en C++ qui simule un assistant virtuel basique. Le programme devra manipuler des chaînes de caractères pour gérer l'identité de l'utilisateur, vérifier un mot de passe et analyser une phrase saisie.

Travail à faire :

1. Déclarez une variable chaîne nommée *monNom* contenant le nom "Toto".
2. Déclarez une variable chaîne *codeSecret* contenant le mot de passe "123".
3. Affichez un message d'accueil Bienvenue suivi de *monNom*.
4. Demandez à l'utilisateur de saisir son *prénom* (variable *monPrenom*).
5. Si le prénom contient moins de 3 lettres, alors son prénom sera par défaut "Nono".
6. Demandez à l'utilisateur de saisir un mot de passe.
7. Comparez le mot de passe saisi avec le *codeSecret*.
8. Si le mot de passe est correct, alors affichez un message " Bravo " suivi du nom suivi du prénom. Sinon, affichez simplement "mot de passe incorrect".

Solution :

```
#include <string>
```

```
using namespace std;
```

```
int main() {
```

```
    // 1. Déclarez une variable chaîne nommée monNom contenant le nom "Toto".
```

```
    string monNom = "Toto";
```

```
    // 2. Déclarez une variable chaîne codeSecret contenant le mot de passe "123".
```

```
    string codeSecret = "123";
```

```
    // 3. Affichez un message d'accueil Bienvenue suivi de monNom.
```

```
    cout << "Bienvenue " + monNom << endl;
```

```
    // 4. Demandez à l'utilisateur de saisir son prénom (variable monPrenom).
```

```
    string monPrenom;
```

```
    cout << "Veuillez saisir votre prénom : ";
```

```
    cin >> monPrenom;
```

```
    // 5. Si le prénom contient moins de 3 lettres, alors son prénom sera par défaut "Nono".
```

```
    if (monPrenom.size() < 3) {
```

```
        monPrenom = "Nono";
```

```
        cout << "Prénom trop court, remplace par 'Nono'." << endl;
```

```
    }
```

Suite de la solution

```
// 6. Demandez à l'utilisateur de saisir un mot de passe.
string motDePasse;
cout << "Veuillez saisir le mot de passe : ";
cin >> motDePasse;

// 7. Comparez le mot de passe saisi avec le codeSecret.
if (motDePasse == codeSecret) {
    cout << "Bravo " << monNom << " " << monPrenom << endl;
} else {
    cout << "mot de passe incorrect" << endl;
}

return 0;
}
```