LANGAGE C++

CHAÎNES DE CARACTÈRES ET STRUCTURES

Le type string

Pour utiliser des chaînes de caractères, il faut importer leur définition : `#include <string>`

La déclaration d'une variable de type chaîne de caractères se fait alors avec : `string identificateur;`

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string un_nom;  // déclaration (chaîne vide)
    string message("Bonjour ");
    // déclaration avec initialisation
    un_nom = "Aicha";
    cout << message << un_nom;
    return 0;
}
```

```
Bonjour Aicha
Process returned 0 (0x0)
Press any key to continue.
```

Le type char

Les caractères (constituants d'une chaîne) peuvent aussi se représenter en tant que tels

Leur valeurs s'écrivent avec des guillemets simples : `'a'`

```
#include <iostream>
using namespace std;
int main()
{
    char c1('O');
    char c2;
    c2 = 'k';
    cout << c1 << c2;
    return 0;
}
```

```
Ok
Process returned 0 (0x0)
Press any key to continue.
```

- Une variable de type string déclarée mais non initialisée est automatiquement initialisée à la chaîne vide ("").
- Comme pour n'importe quel autre type, une variable de type string peut être modifiée par une affectation.

```
string chaine1 ;              // -> chaine1 vaut ""
string chaine2(" de C++") ;   // -> chaine2 vaut " de C++"
chaine1 = "cours" ;           // -> chaine vaut "cours"
cout << chaine1 << chaine2;
```

```
cours de C++
Process returned 0 (0x0)
Press any key to continue.
```

Concaténation

```
string nom;
string prenom;
string identifiant;
cout << "Saisir votre nom : ";
cin >> nom;
cout << "Saisir votre prenom : ";
cin >> prenom;
identifiant = "vous vous nommez - " + nom + " " + prenom;
cout << identifiant;
```

```
Saisir votre nom : Aicha
Saisir votre prenom : Coulibaly
vous vous nommez - Aicha Coulibaly
Process returned 0 (0x0)   execution
Press any key to continue.
```

```
string reponse("solution");
...
if (n > 1) {
    reponse = reponse + 's';
}
```

Comparaison de chaînes

Comme pour les autres types, on utilise **==** pour tester l'égalité et **!=** pour la différence.

```
string reponse;
do {
    cout << "Saisir oui ou non : ";
    cin >> reponse;
} while (reponse != "oui" && reponse != "non");
```

```
Saisir oui ou non : noi
Saisir oui ou non : oun
Saisir oui ou non : oui

Process returned 0 (0x0)
Press any key to continue.
```

Une chaîne de caractères est un tableau de caractères

Indexation : Si chaine est une string , alors chaine[i] est le (i+1) ème caractère de chaine (de type char).

```
string demo("ABCD");
char premier;
char dernier;
premier = demo[0];
dernier = demo[3];
cout << premier << " " << dernier;
```

```
A D
Process returned 0 (0x0)
Press any key to continue.
```

Exercice :

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string essai("essai");
    string test;
    for(int i(1); i <= 3; ++i) {
        test = test + essai[6-2*i];
        test = essai[i] + test;
    }
    cout << test << endl;
    return 0;
}
```

Une chaîne de caractères est un tableau de caractères

Indexation : Si chaine est une string , alors chaine[i] est le (i+1) ème caractère de chaine (de type char).

```
string demo("ABCD");
char premier;
char dernier;
premier = demo[0];
dernier = demo[3];
cout << premier << " " << dernier;
```

```
A D
Process returned 0 (0x0)
Press any key to continue.
```

Exercice :

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string essai("essai");
    string test;
    for(int i(1); i <= 3; ++i) {
        test = test + essai[6-2*i];
        test = essai[i] + test;
    }
    cout << test << endl;
    return 0;
}
```

Fonctions spécifiques aux chaînes

Certaines fonctions propres aux string sont définies.

Elles s'utilisent avec la syntaxe suivante : `nom_de_chaine.nom_de_fonction(arg1,arg2,...);`

Les fonctions suivantes sont définies (où chaine est une variable de type string) :

- chaine.size() - renvoie la taille (c'est-à-dire le nombre de caractères) de chaine .
- chaine.insert(position, chaine2) - insère, à partir de la position (indice)
- chaine.replace(position, n, chaine2) - remplace les n caractères par la string chaine2 .

Exercice :

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string exemple("abcdef");
    cout << exemple << ", taille = " << exemple.size() << endl;
    exemple.insert(1, "xx");
    cout << "insert = " << exemple << endl;
    exemple.replace(1, 2, "1234");
    cout << "replace = " << exemple << endl;
    exemple.replace(1, 2, "");
    cout << "supprimer = " << exemple << endl;
    return 0;
}
```

Fonctions spécifiques aux chaînes

- chaine.find(souschaine) - renvoie l'indice dans chaine du 1er caractère de l'occurrence la plus à gauche
- chaine.rfind(souschaine) - renvoie l'indice dans chaine du 1er caractère de l'occurrence la plus à droite

Exercice :

```
string exemple("baabbaab");
cout << "position gauche = " << exemple.find("ab") << endl;
cout << "position droite = " << exemple.rfind("ab") << endl;
```

Dans les cas où les fonctions find() et rfind() ne peuvent s'appliquer, elles renvoient la valeur prédéfinie string::npos

```
string exemple("baabbaab");
if(exemple.find("zab") != string::npos)
    cout << "position gauche = " << exemple.find("zab");
else cout << "chaine non trouve";
```

```
chaine non trouve
Process returned 0 (0x0)
Press any key to continue.
```

- chaine.substr(depart, longueur) - renvoie la sous-chaîne de chaine, de longueur longueur et commençant à la position depart .

```
string exemple("Salut à tous !");
cout << exemple.substr(8, 4);
```

```
tous
Process returned 0 (0x0)
Press any key to continue.
```

Alias de type

```
typedef array<double,4> Vecteur;
typedef array<Vecteur,2> Matrice;
Matrice m;
for(auto ligne : m){
    for(auto e : ligne){
        cout << "X";
    }
    cout << endl;
}
```

Pour des types composés complexes, dont l'utilisation directe est difficile, on peut utiliser la commande typedef pour donner un autre nom (alias) à ce type

```
XXXX
XXXX

Process returned 0 (0x0)
Press any key to continue.
```

Syntaxe

```
typedef type alias;
```

De telles définitions de nouveaux noms de types sont particulièrement utiles, pour :

- bien définir les types des objets que l'on manipule
  - meilleure identification des « concepts »
  Si tout est int, on ne distingue plus les distances des volumes, des couleurs
  - changements ultérieurs de types plus faciles (par exemple, les distances deviennent des double)

```
typedef int Distance;
Distance ma_longueur(0);
```

- les paramètres de fonctions
  Écriture plus claire, plus compacte et plus systématique

```
typedef vector<double> Vecteur;
Vecteur produit_vectoriel(Vecteur, Vecteur);
```

- les déclarations de tableaux

```
typedef vector<double> Vecteur;
typedef vector<Vecteur> Matrice;
```

Alias de type

```
typedef vector<double> Vecteur;
typedef vector<Vecteur> Matrice;

Matrice matrice(6, Vecteur(8, 0.0));
for(auto ligne : matrice) {
    for(auto element : ligne) {
        cout << element <<" ";
    }
    cout << endl;
}
```

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

Process returned 0 (0x0)
Press any key to continue.
```

Données structurées

Un programme peut avoir à représenter des données structurées, par exemple :

Âge
20
35
26
38
22

Nom	Taille	Âge	Sexe
Dupond	1.75	41	M
Dupont	1.75	42	M
Durand	1.85	26	F
Dugenou	1.70	38	M
Pahut	1.63	22	F

Les tableaux permettent de représenter des structures de données homogènes, c'est-à-dire des listes constituées d'éléments qui sont tous du même type.

```
vector<int> ages;        array<int, 5> ages;
```

On regroupe les données hétérogènes (c'est-à-dire non homogènes) dans un type composé : les structures

- On utilise les structures lorsque l'on souhaite regrouper des types (pas nécessairement identiques) dans une même entité.
- Ce sont une version primitive de la notion « d'objet », fondement de la « programmation orientée objet ».

Les structures

Intérets

- Représenter des entités qui doivent être décrites avec plusieurs données (par exemple, une personne peut être définie avec son prénom, son nom, sa date de naissance,…)
- Elles facilitent la manipulation de telles entités en regroupant les données.
- Retourner plusieurs valeurs à une fonction
- Simplifier la conception et l'écriture des programmes (regroupements conceptuels)

Déclaration d'une structure

Pour déclarer un nouveau **type** « structure » (créer un nouveau type de variable), on utilise la syntaxe suivante :

```
struct Nom_du_type {
    type_1 identificateur_1 ;
    type_2 identificateur_2 ;
    ...
} ;
```

- Nom_du_type - nom que vous souhaitez donner à votre type structuré,
- type_i identificateur_i - déclarations des types et identificateurs des champs de la structure.

Quelques exemples de structures

```
struct Personne {
    string nom;
    double taille;
    int age;
    char sexe; // 'M' ou 'F'
};
```

```
struct Date {
    int jour;
    int mois;
    int annee;
};
```

```
struct Particule {
    array<double, 3> position;
    array<double, 3> vitesse;
    double masse;
    double charge;
};
```

Les structures

Déclaration d'une variable et initialisation

```
struct Personne {
    string nom;
    double taille;
    int age;
    char sexe; // 'M' ou 'F'
};

Personne untel =  { "Couibaly", 1.75, 41, 'F' };
```

On peut accéder aux champs d'une structure en utilisant la syntaxe suivante : `structure.champ`

Exemple

```
int main()
{
    Personne pers1 = {"Coulibaly",1.75,42,'F'};
    Personne pers2 = {"Diarra",1.84,32,'H'};
    affiche(pers1);
    affiche(pers2);
    return 0;
}
void affiche(Personne p) {
    cout << p.nom << ", ";
    if (p.sexe == 'M') cout << "homme";
    else if (p.sexe == 'F') cout << "femme";
    else cout << "alien";
    cout << ", "  << p.taille << " m, "
         << p.age << " an";
    if (p.age > 1) {
        cout << 's';
    }
    cout << endl;
}
```

```
Coulibaly, femme, 1.75 m, 42 ans
Diarra, alien, 1.84 m, 32 ans

Process returned 0 (0x0)   execution
Press any key to continue.
```

Exemple

```
#include <iostream>
using namespace std;
struct Personne {
    string nom;
    double taille;
    int age;
    char sexe; // 'M' ou 'F'
};
void naissance(Personne &p);
void affiche(Personne p);
int main()
{
    Personne pers1;
    Personne pers2;
    naissance(pers1);
    naissance(pers2);
    affiche(pers1);
    affiche(pers2);
    return 0;
}
```

```
void naissance(Personne &p) {
    cout << "Saisie d'une nouvelle personne" << endl;
    cout << " Entrez son nom : ";
    cin >> p.nom;
    cout << " Entrez sa taille (m) : ";
    cin >> p.taille;
    cout << " Entrez son age : ";
    cin >> p.age;
    do {
        cout << " Homme [M] ou Femme [F] : ";
        cin >> p.sexe;
    } while ((p.sexe != 'F') and (p.sexe != 'M'));
}
void affiche(Personne p) {
    cout << p.nom << ", ";
    if (p.sexe == 'M') cout << "homme";
    else if (p.sexe == 'F') cout << "femme";
    else cout << "alien";
    cout << ", " << p.taille << " m, "
         << p.age << " an";
    if (p.age > 1) {
        cout << 's';
    }
    cout << endl;
}
```

```
Saisie d'une nouvelle personne
 Entrez son nom : Coulibaly
 Entrez sa taille (m) : 1.75
 Entrez son age : 42
 Homme [M] ou Femme [F] : F
Saisie d'une nouvelle personne
 Entrez son nom : Diarra
 Entrez sa taille (m) : 1.82
 Entrez son age : 32
 Homme [M] ou Femme [F] : M
Coulibaly, femme, 1.75 m, 42 ans
Diarra, homme, 1.82 m, 32 ans

Process returned 0 (0x0)   execution
Press any key to continue.
```

Autre exemple

```
#include <iostream>
#include <array>
using namespace std;
struct Personne {
    string nom;
    double taille;
    int age;
    char sexe; // 'M' ou 'F'
};
void naissance(Personne &p);
void affiche(Personne p);
int main()
{
    array<Personne,3> tab;
    Personne pers1, pers2, pers3;
    naissance(pers1);
    naissance(pers2);
    naissance(pers3);
    tab[0]=pers1;
    tab[1]=pers2;
    tab[2]=pers3;
    for(int i=0;i<tab.size();i++){
        affiche(tab[i]);
    }
    return 0;
}
```

```
void naissance(Personne &p) {
    cout << "Saisie d'une nouvelle personne" << endl;
    cout << " Entrez son nom : ";
    cin >> p.nom;
    cout << " Entrez sa taille (m) : ";
    cin >> p.taille;
    cout << " Entrez son age : ";
    cin >> p.age;
    do {
        cout << " Homme [M] ou Femme [F] : ";
        cin >> p.sexe;
    } while ((p.sexe != 'F') and (p.sexe != 'M'));
}
void affiche(Personne p) {
    cout << p.nom << ", ";
    if (p.sexe == 'M') cout << "homme";
    else if (p.sexe == 'F') cout << "femme";
    else cout << "alien";
    cout << ", "  << p.taille << " m, "
         << p.age << " an";
    if (p.age > 1) {
        cout << 's';
    }
    cout << endl;
}
```

Autre exemple

```
#include <iostream>
#include <vector>
using namespace std;
struct Cours {
    string intitule;
    string professeur;
    int nbHeure;
};
 struct Etudiant {
        string nom;
        string section;
        vector<Cours> cours;
        double moyenne;
};
void afficherEtudiant(Etudiant e);
void afficherCours(Cours c);
```

```
int main()
{
    Cours cours1 = {"Langage C++","Tophe",72};
    Cours cours2 = {"Base de donnees relationnelle","Diarra",84};
    Cours cours3 = {"UML","Coulibaly",88};
    Etudiant etudiant;
    cout << "Saisir les informations d'inscription d'un etudiant:";
    cout << endl << "Saisir nom: ";
    cin >> etudiant.nom;
    cout << "Saisir section : ";
    cin >> etudiant.section;
    etudiant.moyenne = 12.4;
    etudiant.cours.push_back(cours1);
    etudiant.cours.push_back(cours2);
    etudiant.cours.push_back(cours3);
    afficherEtudiant(etudiant);
    return 0;
}
void afficherEtudiant(Etudiant e){
    cout << e.nom << ", " << e.section << ", " << e.moyenne << endl;
    for(int i=0;i<e.cours.size();i++){
        afficherCours(e.cours[i]);
    }
}
 void afficherCours(Cours c){
     cout << "COURS - " << c.intitule << ", " << c.professeur
          << ", " << c.nbHeure << " heures." << endl;
}
```

```
Saisir les informations d'inscription d'un etudiant:
Saisir nom: Diallo
Saisir section : License Genie Logiciel
Diallo, License, 12.4
COURS - Langage C++, Tophe, 72 heures.
COURS - Base de donnees relationnelle, Diarra, 84 heures.
COURS - UML, Coulibaly, 88 heures.

Process returned 0 (0x0)   execution time : 39.903 s
Press any key to continue.
```