Programmation C / C++

#4 C / C++ avancé

par David Albert

GEMA
GRANDES ÉCOLES DES MÉTIERS D'AVENIR

2024

Table des matières

01

Les conteneurs STL

Conteneurs

En C, l'unique type permettent de regrouper des variables de même type au sein d'une même entité est le tableau.

En C++, il existe de nombreux autres conteneurs (vector, list, map, stack, queue, ...).

En C++, on privilégiera toujours l'utilisation de conteneurs STL par rapport à une implémentation manuelle.

Liste des conteneurs STL : https://cplusplus.com/reference/stl/

Les tableaux dynamiques - vector

Définition

Un **vector** est un conteneur séquentiel qui encapsule les tableaux de taille dynamique.

Usage

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> vect(5); // un vecteur de 5 entiers

    vect[0] = 1; // accès direct à l'indice 0 pour affecter la valeur 1

    vect.push_back( 6 ); // ajoute l'entier 6 à la fin

    vect.pop_back(); // supprime le dernier élément

    cout << "Le vecteur vect contient " << vect.size() << " entiers : \n";

    cout << "Le 3 ème élément contient la valeur " << vect[2] << "\n";

    return 0;
}
```

Notion d'itérateur - iterator

Définition

Les **iterator** sont une généralisation des pointeurs : ce sont des objets qui pointent sur d'autres objets. Ils sont utilisés pour **parcourir une série d'objets**.

Usage

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> v2(4, 100); // un vecteur de 4 entiers initialisés avec la valeur 100

    cout << "Le vecteur v2 contient " << v2.size() << " entiers : ";

    // utilisation d'un itérateur pour parcourir le vecteur v2
    for (vector<int>::iterator it = v2.begin(); it != v2.end(); ++it) {
        cout << " " << *it;
    }

    return 0;
}
```

La table associative - map

Définition

Une **map** permet d'associer une clé à une donnée.

Usage

```cpp
#include <iostream>
#include <iomanip>
#include <map>
#include <string>
using namespace std;

int main()
{
    map<string, unsigned int> nbJoursMois;

    nbJoursMois["janvier"] = 31;
    nbJoursMois["février"] = 28;
    cout << "La map contient " << nbJoursMois.size() << " elements : \n";

    for (map<string,unsigned int>::iterator it=nbJoursMois.begin(); it!=nbJoursMois.end(); ++it)
    {
        cout << it->first << " -> \t" << it->second << endl;
    }

    cout << "Nombre de jours du mois de janvier : " << nbJoursMois.find("janvier")->second << '\n';

}
```

02

Les fichiers

Manipuler des fichiers

en C

Etapes à suivre pour manipuler des fichiers (en C) :

1. inclure les fichiers les fichiers en-tête

```c
#include <stdlib.h>
#include <stdio.h>
```

2. ouvrir un fichier (en lecture ou écriture)

```c
FILE* rfile, wfile, rwfile;
rfile = fopen("file-read.txt", "r"); // en lecture
wfile = fopen("file-write.txt", "w"); // en écriture
rwfile = fopen("file-read-write.txt", "r+"); // en lecture et écriture
```

3. lire ou écrire dans le fichier

```c
double note;
fscanf(rfile, "%d", &note); // lit depuis le fichier
fprintf(wfile, "Bonjour aux %d étudiants", 20); // écrit dans le fichier
```

4. Fermer le fichier

```c
fclose(rfile);
```

Manipuler des fichiers

en C

Principaux modes d'ouverture :

Syntaxe	Explication
"r"	lecture seule
"w"	écriture seule
"a"	mode d'ajout
"a+"	ajout en lecture / écriture à la fin
"r+"	lecture et écriture.
"w+"	lecture et écriture, avec suppression du contenu au préalable

Ecrire dans un fichier

en C

```c
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    FILE* wfile = NULL;
    int age = 0;

    wfile = fopen("user.txt", "w");

    if (wfile != NULL)
    {
        // On demande l'âge
        printf("Quel age avez-vous ? ");
        scanf("%d", &age);

        // On l'écrit dans le fichier
        fprintf(wfile, "Le Monsieur qui utilise le programme, il a %d ans", age);
        fclose(wfile);
    }

    return 0;
}
```

Lire dans un fichier

en C

```c
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    FILE* rfile = NULL;
    int score[3] = {0}; // Tableau des 3 meilleurs scores

    rfile = fopen("scores.txt", "r");

    if (rfile != NULL)
    {
        fscanf(rfile, "%d %d %d", &score[0], &score[1], &score[2]); // lit depuis le fichier
        printf("Les meilleurs scores sont : %d, %d et %d", score[0], score[1], score[2]);

        fclose(rfile);
    }

    return 0;
}
```

Manipuler des fichiers

en C++

Etapes à suivre pour manipuler des fichiers (en C++) :

1. inclure les fichiers les fichiers en-tête

```cpp
#include <iostream>
#include <fstream>
```

2. ouvrir un fichier (en lecture ou écriture)

```cpp
ifstream rfile("data.txt");  // en lecture
ofstream wfile("scores.txt");  // en écriture
```

3. lire ou écrire dans le fichier (à la manière de `cin` et `cout`)

```cpp
double note;
rfile >> note; // lire depuis un fichier
wfile << "Bonjour à tous" << endl; // écrire dans un fichier
```

4. Fermer le fichier

```cpp
momFluxSortie.close()
```

Ecrire dans un fichier

Exemple C++

```cpp
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main()
{
    string const nomFichier("scores.txt");
    ofstream wfile(nomFichier.c_str()); // ouverture d'un fichier en écriture

    if(wfile)
    {
        // l'ouverture s'est bien passée, on peut donc lire
        wfile << "Bonjour, je suis une phrase écrite dans un fichier." << endl;
        wfile << 42.1337 << endl;
        int age(100);
        wfile << "J'ai " << age << " ans." << endl;
    }
    else
    {
        cout << "ERREUR: Impossible d'ouvrir le fichier." << endl;
    }
    return 0;
}
```

Lire un fichier

Exemple C++

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
   ifstream rfile("data.txt");  // ouverture d'un fichier en lecture

   if(fichier)
   {
      // l'ouverture s'est bien passée, on peut donc lire
      string mot;
      rfile >> mot; // lecture d'un mot

      rfile.ignore(); // on change de mode

      string ligne;
      getline(rfile, ligne); // lecture d'une ligne complète

   }
   else
   {
      cout << "ERREUR: Impossible d'ouvrir le fichier en lecture." << endl;
   }

   return 0;
}
```