C/C++: Du C au C++.

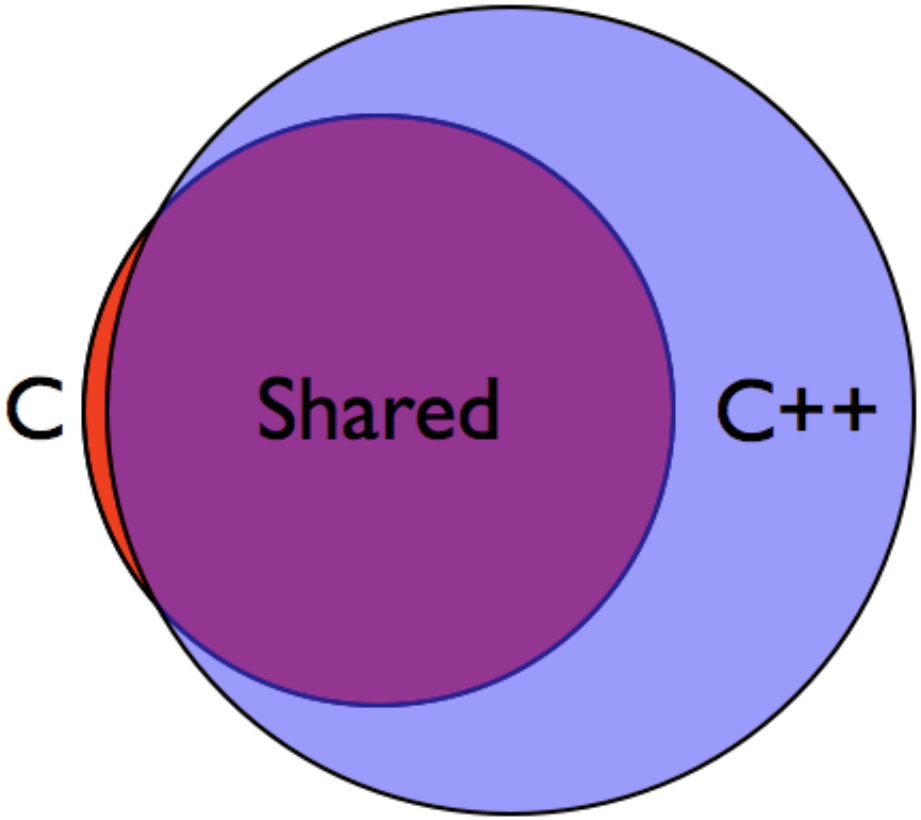
Vincent Gaudillière

Polytech Nancy

2017 / 2018

Le langage C++.

Le langage C++ est une extension du langage C.



Hello world en C++.

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello world!" << std::endl;
    return 0;
}
```

Hello world en C++ (avec directive `using`).

```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Les flux d'entrée/**sortie** en C++.

```
printf("Bonjour!\n"); // en C
cout << "Bonjour!" << endl; // en C++
```

Les flux d'entrée/**sortie** en C++.

```
printf("Bonjour!\n"); // en C
cout << "Bonjour!" << endl; // en C++
```

```
int a=12;
printf("%d\n",a); // en C
cout << a << endl; // en C++
```

Les flux d'entrée/**sortie** en C++.

```
printf("Bonjour!\n"); // en C
cout << "Bonjour!" << endl; // en C++
```

```
int a=12;
printf("%d\n",a); // en C
cout << a << endl; // en C++
```

```
int a=12;
printf("Valeur de a: %d\n",a); // en C
cout << "Valeur de a: " << a << endl; // en C++
```

Les flux d'**entrée**/sortie en C++.

```
int a;

/********** en C: **********/
printf("Entrez un nombre:\n");
scanf("%d",&a);

/********** en C++: **********/
cout << "Entrez un nombre:" << endl;
cin >> a; // pas besoin d'indiquer l'adresse de a
```

Les variables en C++.

```
/* Deux syntaxes pour l'initialisation: */
double x=1.5; // syntaxe héritée du C
double x(1.5); // syntaxe C++
```

Les variables en C++.

```
/* Deux syntaxes pour l'initialisation: */
double x=1.5; // syntaxe héritée du C
double x(1.5); // syntaxe C++
```

```
/* Type booléen: */
bool b1(true); // b1 == true
bool b2(1==2); // b2 == false
```

Les variables en C++.

```
/* Deux syntaxes pour l'initialisation: */
double x=1.5; // syntaxe héritée du C
double x(1.5); // syntaxe C++
```

```
/* Type booléen: */
bool b1(true); // b1 == true
bool b2(1==2); // b2 == false
```

```
/* Compteur d'une boucle 'for': */
for(int i(0);i<10;i++){/*...*/} // i supprimée à la fin
```

Surcharge d'une fonction en C++.

En C, une fonction est caractérisée par son nom.
En C++, une fonction est caractérisée par son nom **et ses paramètres**. (Le type de retour ne caractérise pas une fonction).

```
void triple(int * a); // Signature: triple(int*)
{
    (*a) *= 3;
}

void triple(double * a); // Signature: triple(double*)
{
    (*a) *= 3;
}
```

Valeurs par défaut pour les paramètres en C++.

```
int somme(int a, int b=0, int c=0); // à droite!

int main(){
    int a(1),b;
    b = somme(a); // 1+0+0=1
    b = somme(a,8); // 1+8+0=9
    b = somme(a,8,3); // 1+8+3=12
    return 0;
}

int somme(int a, int b, int c) // pas ici!
{
    return a+b+c;
}
```

Les chaînes de caractères en C++.

```cpp
#include <iostream>

using namespace std;

int main()
{
    string nom("Lucien");
    cout << nom << endl; // Lucien
    nom[0] = 'J';
    nom[2] = 'l';
    cout << nom << endl; // Julien
    cout << nom.size() << endl; // 6

    return 0;
}
```

Références en C++.

Dans une déclaration, le symbole & signifie *référence*. Partout ailleurs, le symbole & signifie *adresse*.

```
int &reference; // déclaration d'une référence sur un 'int'
```

- Règle 1 : une référence doit être initialisée dès sa déclaration.
- Règle 2 : une fois initialisée, une référence ne peut plus être modifiée.

```
int maVariable = 6;
int &referenceSurMaVariable = maVariable;
```

Références en C++.

```
int main(){
    int maVariable = 21;
    int &referenceSurMaVariable = maVariable;
    /* Deux noms différents pour une seule variable! */

    cout << referenceSurMaVariable << endl; // 21
    cout << maVariable << endl; // 21

    referenceSurMaVariable = 40;

    cout << referenceSurMaVariable << endl; // 40
    cout << maVariable << endl; // 40

    return 0;
}
```

Passage par valeur en C et C++.

```
void remiseAZero(int n) // paramètre: int
{
    n = 0;
}

int main()
{
    int a = 9;
    // ici, a est égal à 9
    remiseAZero(a);
    // ici, a est toujours égal à 9

    return 0;
}
```

Passage par adresse en C.

```c
void remiseAZero(int * n) // paramètre: int*
{
    *n = 0;
}

int main()
{
    int a = 9;
    printf("%d\n",a); // 9
    remiseAZero(&a);
    printf("%d\n",a); // 0

    return 0;
}
```

Passage par référence en C++.

```
void remiseAZero(int & n) // paramètre: int&
{
    n = 0;
}

int main()
{
    int a = 9;
    cout << a << endl; // 9
    remiseAZero(a);
    cout << a << endl; // 0

    return 0;
}
```

Les structures en C++.

```cpp
#include <iostream>

using namespace std;

struct Point{ // typedef automatique, donc...
    double x;
    double y;
};

int main()
{
    Point P1; // ... pas besoin d'écrire 'struct' ici!
    return 0;
}
```

Référence sur structure.

```
int main()
{
    Point P1 = {0,1};
    Point &ref_P1 = P1;

    cout << P1.x << endl; // 0
    cout << ref_P1.x << endl; // 0
    cout << P1.y << endl; // 1
    cout << ref_P1.y << endl; // 1

    return 0;
}
```

Ecrire dans un fichier en C++.

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

int main(){
    ofstream file("fichier.txt"); // o: output
    if(file){ // <==> if(file==true)
        file << "Une phrase à écrire." << endl;
    }
    else{
        cout << "Impossible d'ouvrir le fichier." << endl;
    }

    return 0;
}
```

Lire dans un fichier en C++.

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

int main(){
    ifstream file("fichier.txt"); // i: input
    if(file){ // <==> if(file==true)
        int a; // entier à lire
        file >> a;
    }
    else{
        cout << "Impossible d'ouvrir le fichier." << endl;
    }

    return 0;
}
```

Allocation dynamique en C++.

```
#include <iostream>
int main()
{
    int *tableau = NULL;
    int taille = 20;

    tableau = new int[taille]; // Allocation (malloc en C)

    delete[] tableau; // Libération (free en C)

    return 0;
}
```