<C++>

Variables et Constantes

Dorian.H Mekni

09 | APR | 2020

C++

var

- elle s’écrit et se declare en camelCase
- Elle s’écrit de deux manières différentes ->

1. `int dataMan = 12;`
2. `int dataMan{12}; // Cette méthode est préférée`

C++

Var -> écriture + affichage

```cpp
#include <iostream>

/*

 */

int main()

{
    int dataMan = 12;

    std::cout << dataMan << std::endl;

    return 0;

}
```

```
12
Program ended with exit code: 0
```

C++

var -> héritée du C

```
4  #include <iostream>
5
6  /*
7
8   */
9
10 int main()
11
12 {
13     int dataMan = 12; // Ancienne declaration heritée du C -> valide
14
15     std::cout << dataMan << std::endl;
16
17     return 0;
18
19 }
20
21
```

```
12
Program ended with exit code: 0
```

C++

Var -> C++ Moderne

```cpp
#include <iostream>

/*

 */

int main()

{
    int dataMan {12}; // Déclaration moderne du C++
    std::cout << dataMan << std::endl;

    return 0;

}
```

```
12
Program ended with exit code: 0
```

C++

const -> Déclaration héritée du C

```cpp
#include <iostream>

/*

 */

int main()

{
   const int DATAMAN = 1; //Déclaration héritée du C pour l'écriture en capitale d'imprimerie

    std::cout << DATAMAN << std::endl;

    return 0;

}
```

```
1
Program ended with exit code: 0
```

C++

const -> Déclaration propre au C++

```cpp
#include <iostream>

/*

 */

int main()

{
    constexpr int DATAMAN = 1; //Déclaration d'une constante propre au C++
    // Elle peut s'écrire également DATA_MAN

     std::cout << DATAMAN << std::endl;

     return 0;

}
```

```
1
Program ended with exit code: 0
```

C++

auto ->

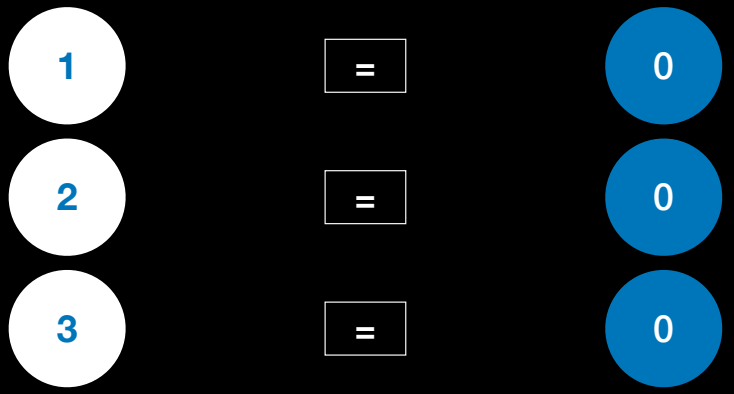
```cpp
#include <iostream>

/*

 */

int main()

{
   auto whichData = 12; //Déclaration type, propre au C++, qui permet la déduction type de la valeur


    std::cout << whichData << std::endl;

    return 0;

}
```

```
12
Program ended with exit code: 0
```

C++

Initialisation.0 x 3

- `int dataType{};`
- `int dataType{0};`
- `int dataType = 0;`



C++

decltype()

- decltype() enable a new variable to adopt the same datatype used by the one between bracket in decltype ->

```
int main()
{

int dataOne{12};

std::cout << dataOne << std::endl;

decltype(dataOne) dataTwo{};

return 0;

}
```

// dataTwo will be initialised with the same datatype of dataOne

C++

suffixe de précision

```cpp
#include <iostream>

/*

    u, U  (unsigned)
    l, L (long int or long double)
    ll, LL (long long int)
    f, F (float)

*/

int main()

{
    float dataSerie = 14.0f;
    //Le datatype est forcé afin d'obtenir un résultat précis
    std::cout << dataSerie << std::endl;


    return 0;

}

```

```
14
Program ended with exit code: 0
```

C++

suffixe de précision | Mémo

```
/*

    u, U  (unsigned)
    l, L (long int or long double)
    ll, LL (long long int)
    f, F (float)

*/
```

C++

static<>

```cpp
#include <iostream>

int main()

{
    const auto number = static_cast<int>(12);
    std::cout << number << std::endl;
    // Conversion statique active au moment de la compilation
    return 0;

}

```

C++

Bibliothèque

```
#include <limits>
```

C++

min() | max()

```cpp
#include <iostream>
#include <limits>
int main()

{

    std::cout << std:: numeric_limits<int>::max() << std::endl;
    // Cette méthode nous donne accés au nombre maximum utilisable par notre variable
    std::cout << std:: numeric_limits<int>::min() << std::endl;
    // Cette méthode nous donne accés au nombre minimum utilisable par notre variable

    return 0;

```

```
-2147483648
2147483647
Program ended with exit code: 0
```