A detailed illustration of a futuristic living room. On the left, a large, rectangular screen displays a soft, white, cloud-like texture. To the right of the screen is a control panel with several circular dials and buttons. The room features a large, curved wall on the right side, a small table with a glass and a bowl, and a potted plant in the foreground. The overall color palette is warm, with shades of yellow, orange, and brown.

Du C
au C++

Origines

- Créé en 1982 par Bjarne Stroustrup
- Programmation orientée objet



Wikipedia






<https://www.stroustrup.com/>

Programme de base

```
#include <stdio.h>  
  
using namespace std;  
  
int main()  
{  
    ...  
}
```




Déclaration de variables

```
int main() {  
    int n;   
    ...  
    n = 2;  
    ...  
    int i = 2 * n + 1;   
    ...  
    for (int v = 0 ; v < 10 ; v++) {   
        ...  
    }  
}
```

Entrées/Sorties

- cin : lecture sur l'entrée standard (le clavier)

cin >> a

- cout : écriture sur la sortie standard (l'écran)

cout << "texte"

Pas de format ! 🤖

```
#include <iostream>

using namespace std;

int main()
{
    int a, b, c;
    cout << "Saisissez la valeur de a : ";
    cin >> a;
    cout << "a= " << a << "\n";
    cout << "Saisissez les valeurs de b et de c : ";
    cin >> b >> c;
    cout << "b= " << b << ", c= " << c << endl;
}
```

Les namespaces

Prévenir les conflits de nommage

Appel "normal" de cout :

```
std::cout << "texte"
```

sinon :

```
using namespace std;
```

...

```
cout << "texte"
```



```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a, b, c;
```

```
    std::cout << "Saisissez la valeur de a : ";
```

```
    std::cin >> a;
```

```
    std::cout << "a= " << a << "\n";
```

```
    std::cout << "Saisissez les valeurs de b et de c : ";
```

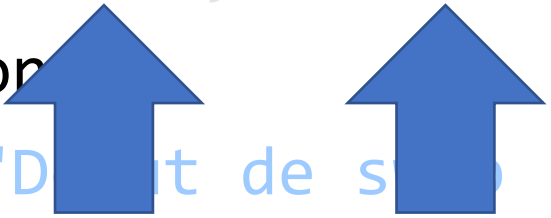
```
    std::cin >> b >> c;
```

```
    std::cout << "b= " << b << ", c= " << c << std::endl;
```

```
}
```

Passage par référence

```
void swap (int & a, int & b) {  
    int tampon;  
    cout << "Début de swap : " << a << " " << b << endl;  
    tampon = a;  
    a = b;  
    b = tampon;  
    cout << "Fin de swap : " << a << " " << b << endl;  
}
```



```
int main () {  
    int n = 24, p = 48;  
    cout << "Avant : " << n << " " << p << endl;  
    swap (n, p);  
    cout << "Après : " << n << " " << p << endl;  
}
```

donne :

Avant : 24 48

Début de swap : 24 48

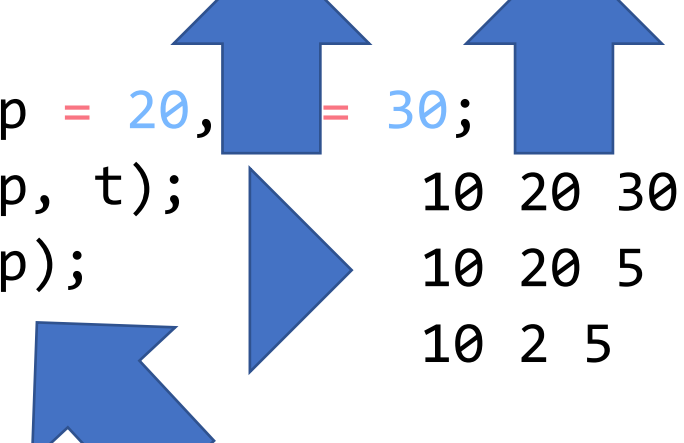
Fin de swap : 48 24

Après : 48 24

Arguments optionnels

```
void afficher (int, int = 2, int = 5);
```

```
int main() {  
    int n = 10, p = 20, t = 30;  
    afficher(n, p, t);  
    afficher(n, p);  
    afficher(n);  
}
```



10	20	30
10	20	5
10	2	5

```
void afficher (int a, int b, int c) {  
    cout << a << " " << b << " " << c << endl;  
}
```

Les arguments optionnels doivent obligatoirement être **les derniers de la liste !**

Surdéfinition de fonctions

Prototypes :

```
void afficher(int);
```

```
void afficher(float);
```

```
void afficher(int, float);
```

Surdéfinition de fonctions

Déclarations :

```
void afficher(int a) {  
    cout << "Valeur entière = " << a << endl;  
}
```

```
void afficher(float a) {  
    cout << "Valeur réelle = " << a << endl;  
}
```

```
void afficher(int a, float b) {  
    cout << "Valeur entière = " << a << endl << "Valeur réelle = " << b << endl;  
}
```


Surdéfinition de fonctions

Utilisation :

```
int main() {  
    int n = 10;  
    float p = 8.8;  
    afficher(n);  
    afficher(p);  
    afficher(n, p);  
}
```

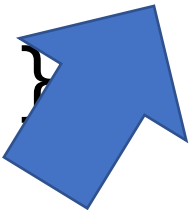


```
Valeur entière = 10  
Valeur réelle = 8.8  
Valeur entière = 10  
Valeur réelle = 8.8
```

Déclaration des structures

```
struct Point {  
    int x;  
    int y;  
};
```

```
int main() {  
    Point A;
```



Gestion dynamique de la mémoire

```
type * var = new type;
```

alloue l'emplacement pour 1 élément du type indiqué.

```
type * var = new type[n];
```

alloue l'emplacement pour n éléments du type indiqué (tableau).

→ remplace `malloc/calloc`

Gestion dynamique de la mémoire

En cas d'erreur, renvoie null ou lève une exception
bad_alloc

delete var;

delete [] var;

libèrent les emplacements.

→ remplace free

Mots-clés

<code>alignas (since C++11)</code> <code>alignof (since C++11)</code> <code>and</code> <code>and_eq</code> <code>asm</code> <code>atomic_cancel (TM TS)</code> <code>atomic_commit (TM TS)</code> <code>atomic_noexcept (TM TS)</code> <code>auto(1)</code> <code>bitand</code> <code>bitor</code> <code>bool</code> <code>break</code> <code>case</code> <code>catch</code> <code>char</code> <code>char8_t (since C++20)</code> <code>char16_t (since C++11)</code> <code>char32_t (since C++11)</code> <code>class(1)</code> <code>compl</code> <code>concept (since C++20)</code> <code>const</code> <code>constexpr (since C++11)</code> <code>constinit (since C++20)</code> <code>const_cast</code> <code>continue</code> <code>co_await (since C++20)</code> <code>co_return (since C++20)</code> <code>co_yield (since C++20)</code> <code>decltype (since C++11)</code>	<code>default(1)</code> <code>delete(1)</code> <code>do</code> <code>double</code> <code>dynamic_cast</code> <code>else</code> <code>enum</code> <code>explicit</code> <code>export(1)(3)</code> <code>extern(1)</code> <code>false</code> <code>float</code> <code>for</code> <code>friend</code> <code>goto</code> <code>if</code> <code>inline(1)</code> <code>int</code> <code>long</code> <code>mutable(1)</code> <code>namespace</code> <code>new</code> <code>noexcept (since C++11)</code> <code>not</code> <code>not_eq</code> <code>nullptr (since C++11)</code> <code>operator</code> <code>or</code> <code>or_eq</code> <code>private</code> <code>protected</code> <code>public</code> <code>reflexpr (reflection TS)</code>	<code>register(2)</code> <code>reinterpret_cast</code> <code>requires (since C++20)</code> <code>return</code> <code>short</code> <code>signed</code> <code>sizeof(1)</code> <code>static</code> <code>static_assert (since C++11)</code> <code>static_cast</code> <code>struct(1)</code> <code>switch</code> <code>synchronized (TM TS)</code> <code>template</code> <code>this</code> <code>thread_local (since C++11)</code> <code>throw</code> <code>true</code> <code>try</code> <code>typedef</code> <code>typeid</code> <code>typename</code> <code>union</code> <code>unsigned</code> <code>using(1)</code> <code>virtual</code> <code>void</code> <code>volatile</code> <code>wchar_t</code> <code>while</code> <code>xor</code> <code>xor_eq</code>
---	---	--