========== 🚀 C++ KIT DE SURVIE –

BASES DU LANGAGE

Hello World

```cpp
#include <iostream>
int main() {
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Commentaires

```cpp
// Ligne simple
/* Bloc de commentaires */
```

Variables et Types

```cpp
int age = 30;        // entier
float poids = 72.5;  // nombre à virgule
char initiale = 'T'; // caractère
bool majeur = true;  // booléen
std::string nom = "Thomas"; // chaîne
```

Entrées / Sorties

```cpp
std::cout << "Ton âge ? ";
std::cin >> age;
```

Constantes

```cpp
const double PI = 3.14159;
```

Opérateurs courants

```cpp
+ - * / %   // arithmétiques
== != < > <= >= // comparaisons
&& || !     // logiques
```

🚃STRUCTURES DE CONTRÔLE

Condition

```
if (age >= 18) std::cout << "Majeur";
else std::cout << "Mineur";
```

Switch

```
switch (jour) {
  case 1: std::cout << "Lundi"; break;
  default: std::cout << "Autre";
}
```

Boucles

```
for (int i=0; i<5; ++i) std::cout << i;
while (x < 10) x++;
do { x--; } while (x > 0);
```

FONCTIONS ET PORTÉE

Définition & Appel

```
int somme(int a, int b) { return a + b; }
int main() { std::cout << somme(2,3); }
```

Passage par référence / valeur

```
void ajoute1(int& x) { x++; }
```

Surcharge simple

```
double somme(double a, double b) { return a + b; }
```

🛖TABLEAUX ET CHAÎNES

Tableau statique

```
int t[3] = {1,2,3};
for (int i : t) std::cout << i;
```

Vecteur dynamique

```
#include <vector>
std::vector<int> v = {1,2,3};
v.push_back(4);
```

Chaîne

```
std::string nom = "P2P";
std::cout << nom.size();
```

🛖CLASSES & OBJETS

Définition simple

```
class Point {
  public:
    int x, y;
    Point(int a, int b): x(a), y(b) {}
    void afficher() { std::cout << x << "," << y; }
};
```

Utilisation

```
Point p(2,3);
p.afficher();
```

Héritage

```
class Point3D : public Point {
  int z;
  public: Point3D(int a,int b,int c): Point(a,b), z(c) {}
};
```

POINTEURS ET RÉFÉRENCES

Base

```cpp
int n = 5;
int* p = &n;
std::cout << *p; // affiche 5
```

Référence

```cpp
int a=1, b=2;
int& ref = a;
ref = b; // a devient 2
```

Allocation dynamique

```cpp
int* ptr = new int(10);
delete ptr;
```

STANDARDS & UTILITAIRES

auto & boucles modernes

```cpp
for (auto& e : v) std::cout << e;
```

Namespace

```cpp
namespace math { int x=5; }
std::cout << math::x;
```

Enums

```cpp
enum Couleur { ROUGE, VERT, BLEU };
Couleur c = ROUGE;
```

FICHIERS & FLUX

```cpp
#include <fstream>
std::ofstream f("test.txt");
f << "Bonjour";
f.close();
```

Lecture

```cpp
std::ifstream f("test.txt");
std::string ligne;
while (getline(f, ligne)) std::cout << ligne;
```

🚨 ERREURS & EXCEPTIONS

```cpp
try {
    throw std::runtime_error("Erreur!");
} catch (std::exception& e) {
    std::cout << e.what();
}
```

BONNES PRATIQUES

- Toujours initialiser les variables.
- Utiliser `std::vector` plutôt que `new[]`.
- Préférer `const &` en paramètre.
- Compiler avec `-Wall -Wextra -Wpedantic`.
- Indenter et commenter clairement.

🧭 COMMANDE DE COMPILATION

```
g++ -std=c++20 -Wall -Wextra -O0 -g main.cpp -o main
./main
```