

# Addition de vecteurs

## 1 Installation IDE

L'utilisation de Visual Studio va grandement nous faciliter la tâche, l'IDE offrant tous les outils nécessaires à la mise en place du projet. Nous allons commencer par mettre en place la solution Visual Studio dans laquelle nous écrirons notre premier code.

1. Ouvrir Visual Studio (**2017 ou supérieur**).
2. Choisir *Créer un projet*.
3. Choisir un *Projet vide*.
4. Choisissez le nom de votre projet et son emplacement. Cochez de préférence la case afin de mettre la solution et le projet dans le même répertoire.

Nous allons maintenant préparer l'environnement de travail

1. A droite de votre écran se trouve une sous-fenêtre *Explorateur de solutions*. Sur le dossier nommé *Fichiers sources*, faites un clic-droit et choisissez *Ajouter... → Nouvel élément*.
2. Sélectionner *Fichier C++*, que vous nommerez *main.cpp*. Créez le fichier.
3. Ouvrez l'onglet *Projet → Propriétés*. Vérifiez bien que vous soyez dans le cas *Configuration : Toutes les configurations* et *Plateforme : Toutes les plateformes*.
4. Dans le premier onglet (*Propriétés de configuration → Général*), assurez-vous de sélectionner *Norme du langage C++ : Norme ISO C++ 17*.

## 2 Premier code sur CPU

Nous allons implémenter en C++ l'addition de 2 vecteurs.

$A[] + B[] \rightarrow C[]$

### 1. Vector\_add

Un premier cas sans utiliser le multithread en séquentiel. Nous choisirons la taille des vecteurs puis les valeurs. On rajoutera un chronomètre pour mesurer le temps d'exécution des boucles de calculs.

Les includes du programme :

```
#include <iostream>
#include <vector>
#include <chrono>

using namespace std ;
using namespace std :: chrono ;
```

## 2. Vector\_add\_Thread

Le deuxième cas sera d'utiliser les threads (thread C++11 : concurrency support library) pour paralléliser sur CPU le programme. Chaque thread additionne une portion des vecteurs.

Le chrono mesure uniquement la phase d'addition parallèle.

`thread::hardware_concurrency()` → détecte le nombre de threads disponibles.

```
#include <iostream>
#include <vector>
#include <thread>
#include <chrono>
using namespace std ;
using namespace std :: chrono ;
```

### 3. Vector\_add\_CPU

Le troisième cas sera la comparaison de temps d'exécution entre le cas 1 et le cas 2 dans le même programme toujours sur CPU. Nous verrons que la taille des vecteurs est importantes pour pouvoir comparer, il faudra donc générer automatiquement des vecteurs de grandes tailles (on pourra choisir la taille des vecteurs).

```
#include <iostream>
#include <vector>
#include <thread>
#include <chrono>
#include <random>
using namespace std ;
using namespace std :: chrono ;
```

#### **4. Vector\_add\_CPU\_Threads**

Le quatrième cas sera le cas 3 en faisant varier le nombre de threads utilisés pour pouvoir construire des courbes de comparaison de temps d'exécution en fonction du nombre de threads utilisés (comme dans le cours).

## 5. Vector\_add\_GPU

Le cinquième cas sera le portage du programme sur GPU.

```
#include <vulkan/vulkan.h>
#include <iostream>
#include <vector>
#include <stdexcept>
#include <cstring>
#include <cstdint>
#include <cstdlib>
#include <fstream>
#include <chrono>
#include <optional>
```