Addition de vecteurs

1 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.

1.1. Pré-requis

Un PC sous Windows 10 ou 11 (64 bits).
Une connexion internet pour télécharger Visual Studio
Droits administrateur sur la machine pour installer les logiciels.

1.2. Installation de Visual Studio (avec C++)

- Ouvrez votre navigateur web et allez sur le site officiel de Visual Studio (version Community).
- Téléchargez le programme d'installation de Visual Studio Community.
- Exécutez le fichier téléchargé (par exemple : vs_Community*.exe).
- Dans la fenêtre des charges de travail (Workloads), cochez :

"Développement Desktop avec C++" (Desktop development with C++).

- Vérifiez que les composants C++ de base sont sélectionnés (la sélection par défaut est suffisante pour commencer).
- Cliquez sur "Installer" en bas à droite.
- Laissez l'installation se terminer, puis redémarrez Visual Studio si cela est demandé.
- Vérifier les extensions :

  - **C/C++ (Microsoft)**

    ID : `ms-vscode.cpptools` Visual Studio Marketplace+1

  - **C/C++ Extension Pack**

    ID : `ms-vscode.cpptools-extension-pack` Visual Studio Marketplace+1

  - **CMake Tools**

ID : `ms-vscode.cmake-tools` Visual Studio Marketplace+2Visual Studio Code+2

- **Bonus**
  - **Code Runner** : lancer rapidement un fichier C++ simple sans config complexe (bien pour des petits tests).
  - **Clang-Format** ou **EditorConfig** : formatage automatique du code.
  - **GitLens** : travailler avec Git.

Ouvrir Visual Studio.

Choisir *Créer un projet*.

Choisir un *Projet vide*.

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

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*.

Sélectionner *Fichier C++*, que vous nommerez main.cpp. Créez le fichier.

Ouvrez l'onglet *Projet → Propriétés*. Vérifiez bien que vous soyez dans le cas *Configuration : Toutes les configurations* et *Plateforme : Toutes les plateformes*.

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[ ] -> 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>
```