Découverte du C++

2024-2025

C++

Introduction à C++ ( )

Concepts étudiés

[Num] Typage des données
[Num] Fonctions et tableaux

Mots clefs

Typage; Fonction; Tableaux

Sessions

0 Cours(s) - 1h30
2 TD(s) - 1h30
0 TD(s) Machine - 2h00
0 TP(s) - 4h30

Travail

Seul ou par binôme

Institut d'Optique
Graduate School, *France*
https://www.institutoptique.fr

GitHub - Digital Methods

https://github.com/IOGS-Digital-Methods

Maitriser les concepts de base du langage C++

Malgré une forte croissance de l'utilisation du langage **Python** dans le monde scientifique (analyse de données, machine learning, automatisation, prototypage rapide...), le langage **C++** reste encore parmi les plus performants en terme de rapidité d'exécution.

C'est notamment grâce à son étape de **compilation langage de programmation compilé** et sa compatibilité avec le langage C qui en font un des langages de programmation les plus utilisés dans les applications où la performance est critique (logiciels systèmes - exploitation et drivers, traitement de données, systèmes embarqués).

L'objectif de ces deux séances est de prendre en main un environnement de développement simple (CodeBlocks) et de découvrir les bases du langage C++, à travers des exemples à tester et à modifier.

Alors, à vos claviers...

Acquis d'Apprentissage Visés

En résolvant ces problèmes, les étudiant·e·s seront capables de :

1. **Compiler un code en langage C++**
2. **Définir des variables**
3. **Définir des fonctions** pour faciliter la réutilisabilité d'un code
4. **Définir et utiliser des tableaux**
5. **Utiliser une bibliothèque de fonctions**

Ressources

Cette séquence est basée sur le langage C++.

Nous utiliserons l'environnement **CodeBlocks** (gratuit).

Des tutoriels C++ sont en cours de développement à l'adresse suivante : https://iogs-lense-training.github.io/cpp-basics-oop/.

Travail à réaliser

Il est suggéré de créer un nouveau projet à chaque exercice afin de conserver une trace des programmes.

Séance 1

Exercice 1 / Création d'un projet sous CodeBlocks (HelloWorld)

On se propose dans un premier temps de créer un nouveau projet C++ avec l'interface de développement CodeBlocks.

1. Lancer l'application CodeBlocks.
2. Puis sélectionner File/New/Project... (Fichier/Nouveau/Projet... en français).
3. Sélectionner ensuite le modèle (*template*) Console Application.
4. Sélectionner le langage C++.
5. Donner un titre à votre projet dans le champ Project Title puis sélectionner le répertoire dans lequel vous souhaitez sauvegarder votre projet dans le champ Folder to create project in.
6. Dans la fenêtre suivante, sélectionner le compilateur (*Compiler*) : GNU GCC Compiler (installé avec CodeBlocks en sélectionnant la version incluant MingW).
7. Après validation, votre projet est créé dans la fenêtre principale de l'application. Ouvrir le dossier Sources dans le manager de projet et ouvrir le fichier intitilé main.cpp

Il faut ensuite compiler le projet (*Build*). Pour cela, il existe le raccourci clavier *CTRL + F9*.

Une fois la compilation réussie avec succès, il reste à exécuter le programme (*Run*). Pour cela, il existe le raccourci clavier *CTRL + F10*.

Exercice 2 / Structure de base d'un code C++ et interaction

On se propose d'étudier le code ***exo1_2.cpp*** fourni.

1. Créez un nouveau projet avec CodeBlocks.
2. Copiez-collez le code dans le fichier main.cpp du projet.
3. Exécutez le programme. Expliquez le code.
4. Modifiez le code pour demander à l'utilisateur de saisir le valeur de la variable *resistance* avant de l'afficher.

Exercice 3 / Mauvais calculs ?

On se propose d'étudier le code ***exo1_3.cpp*** fourni.

1. Créez un nouveau projet avec CodeBlocks.
2. Copiez-collez le code dans le fichier main.cpp du projet.
3. Exécutez le programme. Est-ce le bon résultat ?
4. Expliquez l'erreur et modifiez le programme pour qu'il affiche le bon résultat.
5. Réécrivez les lignes contenant des *printf* à l'aide de l'opérateur de sortie du C++ (*cout*).
6. Ajoutez la possibilité à l'utilisateur de saisir les valeurs des variables **a** et *b*.

Exercice 4 / Conditions et itérations

Le code suivant permet de **faire un test** sur la variable a et d'afficher si sa valeur est supérieure à 8 ou non.

```
int a = 10;

if(a > 8){
        printf("a > 8\n");
}
else{
        printf("a <= 8\n");
}
```

Le code suivant permet d'**itérer un nombre de fois prédéfini** des actions (ici l'affichage des 5 premiers nombres entiers impairs).

```
int i;

for(i = 0; i < 5; i++){
        printf("[%d] -> %d \n", i, i*2+1);
}
```

On propose d'étudier à présent les deux codes suivants :

```
int i = 0;

do{
        printf("[%d] -> %d \n", i, i*2+1);
        i++;
}while(i < 5);
```

```
int i = 0;

while(i < 5){
        printf("[%d] -> %d \n", i, i*2+1);
        i++;
};
```

1. A quoi servent ces deux codes ? Quelle est leur différence ?
2. Écrire un programme qui demande à l'utilisateur de taper un entier N entre 0 et 20 bornes incluses et continue de demander tant que la valeur n'est pas correcte. La valeur finale sera affichée.

Exercice 5 / Fonctions

On se propose d'étudier le code ***exo1_5.cpp*** fourni, qui contient une fonction *somme* et un programme principal qui appelle cette fonction avec des paramètres particuliers.

```
/* Definition de la fonction somme */
int somme(int a, int b){
        int s = a + b;
        return s;
}
```

1. Écrire une fonction qui calcule u(N) défini par :
   u(n+1)=3*u(n)+4
2. Ecrire un programme de test qui demande à l'utilisateur de saisir un nombre entre 0 et 20 et qui exécute la fonction écrite précédemment.

Travail à réaliser (suite)

Séance 2

Exercice 1 / Bibliothèque de fonctions - tab1D

Soit les fichiers ***tab1D.h*** et ***tab1D.cpp*** contenant des fonctions associées à la gestion de tableaux en C++ et le fichier ***exo2_1.cpp*** contenant un test des fonctions contenues dans les fichiers précédemment cités.

1. Créez un nouveau projet avec CodeBlocks.
2. Copiez-collez le code du fichier ***exo2_1.cpp*** fourni dans le fichier main.cpp du projet.
3. Ajoutez les fichiers ***tab1D.h*** et ***tab1D.cpp*** dans le répertoire du projet.
4. Exécutez le programme.
5. Expliquez le code fourni et sa structure. Que contiennent notamment les fichiers ***.h*** et ***.cpp*** ?

Exercice 2 / Maximum, somme et moyenne d'un tableau 1D

Ajoutez les fonctions *maxTabInt(...)*, *sumTabInt(...)* et *moyTabInt(...)* à la bibliothèque de fonctions sur les tableaux 1D, permettant respectivement de retourner le maximum, la somme et la moyenne d'un tableau d'entiers passé en argument.

Exercice 3 / Masques binaires

Soit les variables **int a = 478;** et **int k;**.

1. Quel est le résultat des opérations **k = a << 3;** et **k = a >> 2;** ?
2. Quel est le résultat de l'opération **k = a & 0b11111111;** ?

   Soit les variables **int k1 = (a >> 8) & 0b11111111;** et **int k2 = (a) & 0b11111111;**.
3. Quel est le résultat de l'opération **k = (k1 << 8) + k2;** ? Expliquez ces trois instructions.

Exercice 4 / Chaine de caractères

On se propose d'étudier le code ***exo1_5.cpp*** fourni.

1. Testez ce code et expliquez comment est gérée une chaine de caractères en C/C++.
2. Commentez les différents affichages de la variable **chaine[2]**.
3. Qu'arrive-t-il à la variable **chaine[3]** stockée dans la variable **c**.
4. A quoi sert l'instruction **sprintf (...)** ? Combien d'octets utilise la variable **chaine2** ?

Exercice 5 / Transmission d'informations

On souhaite transmettre 5 données entières qui sont chacune comprise entre 0 et 179.

1. Proposez une solution pour les transmettre de manière asynchrone octet par octet de manière fiable, en utilisant le moins d'octets possible.
2. Cette solution est-elle toujours la moins gourmande en octets dans le cas de 5 nombres entiers compris entre 0 et 3599 ?

Exercice B1 / Echange de variables

Écrire une fonction qui prend deux entiers A et B en paramètre, qui échange le contenu des variables A et B.

Exercice B2 / Remplissage d'un tableau 1D

Écrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. Le programme doit ensuite afficher l'indice du plus grand élément.
