



Rimouski | Lévis

TP1 – Systèmes d’exploitation (Windows, MacOS, Linux)

Par
Frédéric Boutin

Travail présenté à Mme Lise Boudreault

Dans le cadre du cours Systèmes d’exploitation
INF34207-MS

14 février 2024

Table des matières

1. Introduction	3
1.1 Dans un tableau, présentez les caractéristiques (lignes) de l'ordinateur que vous utilisez pour ce cours :	3
1.2 Lorsque vous utilisez votre ordinateur, décrivez au moins deux applications (p.ex. un jeu vidéo, un outil de montage, un outil de modélisation 3D, etc.) :	4
2. Programmation	5
2.1 Choisissez trois langages de programmation (ex: C, C++, C#, JavaScript, Java, Python) :	5
2.2 Présentez le code requis pour effectuer trois requêtes vers le système d'exploitation :	5
Saisie d'une information (entrée de données au clavier)	5
Affichage de l'information à l'écran	6
Impression de la même information qu'à l'étape précédente, mais à partir d'une imprimante, donc, effectuer un appel à l'imprimante (par défaut)	8
2.3 Au niveau de la complexité de la programmation, que remarquez-vous entre ces trois langages ?	11
3. Bases de données	11
3.1 Quel est le rôle du système d'exploitation (séquence d'étapes) lorsqu'un langage de programmation transmet une requête vers un serveur de base de données (ex : MySQL)? ...	11
Médiagraphie	13
Annexe	14

1. Introduction

1.1 Dans un tableau, présentez les caractéristiques (lignes) de l'ordinateur que vous utilisez pour ce cours :

Avant-propos : Il est à noter que pour ce travail, j'ai utilisé mon ordinateur portable. Lors de la première séance Zoom, j'avais inscrit dans le chat Zoom les caractéristiques de mon pc de type tour de bureau. Ainsi, il est normal qu'il ne s'agisse pas des mêmes caractéristiques que celles de la première séance Zoom du 10 janvier.

Caractéristique de l'ordinateur	Valeur de la caractéristique
Nom du processeur	11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 2803 Mhz, 4 Core(s), 8 Logical Processor(s)
Manufacturier du processeur	Intel Corporation
Vitesse de base du processeur	2.80GHz
Nombre de cœur du processeur	4
Nombre de threads du processeur	8 processeurs logiques
Taille de la cache de niveau 1	320 KB
Taille de la cache de niveau 2	5.0 MB
Taille de la cache de niveau 3	12 MB
Capacité de la mémoire vive	16.0 GB
Vitesse de la mémoire vive	3200 MHz
Type de mémoire vive	DDR4
Nom du disque dur principal (modèle)	OS (C :)
Manufacturier du disque dur principal	(Standard disk drives)
Capacité du disque dur principal	475 GB
Système de fichiers utilisé du disque dur principal	NTFS (New Technology File System)
Type de connexion du disque dur principal (SATA, USB, NVMe, etc.)	NVMe INTEL SSDPEKNW512G8
Nom de la carte graphique (si présente)	Mon ordinateur n'a pas de carte graphique, il possède cependant une solution graphique intégrée, soit Intel Iris Xe Graphics . Cela signifie que l'unité de traitement graphique (GPU) est intégrée directement dans la puce du processeur, partageant des ressources telles que la mémoire avec le CPU.
Manufacturier de la carte graphique	Intel Corporation (manufacturier de la solution graphique intégrée ≠ carte graphique)
VRAM de la carte graphique	Intel Iris Xe Graphics dispose de VRAM, mais il ne s'agit pas d'une mémoire physique séparée comme pour les cartes graphiques. Au lieu de cela, elle utilise

	une partie de la mémoire principale (RAM) du système comme mémoire vidéo partagée.
Vitesse de l'horloge de la carte graphique	Après avoir essayé sans succès de trouver ce renseignement utilisant le Task Manager et Intel Graphics Command Center, je n'ai pas trouvé. Il faudrait utiliser un logiciel tiers, mais je ne préfère pas m'y risquer.

Les outils que j'ai utilisés pour trouver les différentes informations sont les suivants :

- System Information
- Task Manager
- Device Manager
- Windows Power Shell en mode admin (commande Get-CimInstance -ClassName Win32_PhysicalMemory | Format-Table SMBIOSMemoryType pour obtenir le type de mémoire vive)

1.2 Lorsque vous utilisez votre ordinateur, décrivez au moins deux applications (p.ex. un jeu vidéo, un outil de montage, un outil de modélisation 3D, etc.) :

La première application est Google Chrome (version : 120.0.6099.225) qui est mon navigateur web. Cette application est plus efficiente en termes de temps de réponse, puisque les navigateurs modernes sont conçus pour optimiser leur temps de réponse en fonction des ressources matérielles disponibles, telles que la mémoire RAM et les processeurs multicœurs. Parmi les exemples de moyens d'optimisation, il y a l'architecture multiprocessus qui permet une meilleure isolation entre les différents processus de l'application et une meilleure répartition des tâches sur plusieurs processus (temps de réponse plus rapide), l'utilisation d'une programmation asynchrone pour la gestion des tâches, une gestion intelligente des ressources qui permet l'attribution de priorité aux onglets actifs et l'utilisation du chargement différé pour les ressources non prioritaires. Il faut noter également l'utilisation d'un moteur de rendu optimisé (patron de conception Composite) afin de permettre la gestion des pages par arborescence d'objets, ce qui améliore le temps d'opération sur les éléments des pages. Durant un test d'utilisation pour la lecture d'une vidéo YouTube (musique), l'application utilisait 2-5% du CPU, 650-750 MB de RAM, 0-0.1% du disque et la lecture était fluide.

La seconde application est mon logiciel gratuit d'antivirus AVG Internet Security System (version 23.12.8700.0). Cette application, lorsqu'elle effectue une analyse du système, doit accéder à plusieurs fichiers de celui-ci et effectuer des opérations de comparaison pour détecter des menaces. Ainsi, tout ce processus occasionne une grande utilisation des ressources matérielles de l'ordinateur, autant au niveau des calculs du processeur qu'au niveau du disque pour l'accès aux fichiers. On sait que l'accès au disque est la forme d'opération la plus lente sur une machine et l'antivirus doit y accéder souvent pour son analyse, ainsi il est possible de comprendre pourquoi

cette application possède un temps d'exécution plus lent. Une analyse intelligente a occasionné des utilisations de 20-70% du CPU, 150-300 MB de RAM et 1.5-21% du disque.

En résumé, comme il a été possible de le voir à travers ces deux applications, la différence entre l'efficacité d'exécution d'une application va beaucoup dépendre des moyens et des principes de conception qu'elle possède pour utiliser au mieux les ressources matérielles de la machine, du type de ressource qu'elle doit utiliser sur la machine, ainsi que la possibilité ou non d'optimiser ses processus. Ainsi, un navigateur web qui effectue une bonne séparation de tâches peut les exécuter plus rapidement et en parallèle. Son utilisation d'une cache et du chargement différé permettront un accès plus rapide aux informations des pages. À l'opposé, une application antivirus doit nécessairement faire son analyse de fichiers en passant régulièrement par l'accès au disque et monopolise ainsi beaucoup le processeur, ce qui augmente le temps de traitement.

2. Programmation

2.1 Choisissez trois langages de programmation (ex: C, C++, C#, JavaScript, Java, Python) :

J'ai choisi les langages Python, C# et C++. Les images ci-dessous (et en annexe) présentent les trois codes de ces langages. Les trois programmes suivent la même logique, c'est-à-dire qu'ils font des appels de fonctions que j'ai créés pour représenter la saisie d'information, l'affichage à l'écran et l'impression à l'imprimante. Pour chacun des cas, la fonction de saisie se nomme `EnteringInformation`, la fonction d'affichage à l'écran se nomme `DisplayInformation`, et la fonction d'impression se nomme `Printing`.

2.2 Présentez le code requis pour effectuer trois requêtes vers le système d'exploitation :

Avant-propos : Voici les codes des fonctions permettant d'effectuer les requêtes demandées. Il est à noter que les codes des appels à ces fonctions (Main) ainsi que les images des résultats en Terminal / Console ont été placés en annexe.

Saisie d'une information (entrée de données au clavier)

```
def entering_information():  
    age = input("Entrez votre âge : ")  
    return age
```

Figure 1 - Code Python pour une saisie

```

using System;

public class Utils
{
    public static string EnteringInformation()
    {
        Console.Write("Entrez votre âge : ");
        string age = Console.ReadLine();
        Console.WriteLine();
        return age;
    }
}

```

Figure 2 -Code C# pour une saisie

```

#include "Utils.h"
#include <windows.h>

std::string Utils::EnteringInformation()
{
    std::string age;

    std::cout << "Entrez votre age : ";
    std::cin >> age;

    return age;
}

```

Figure 3 -Code C++ pour la saisie

Affichage de l'information à l'écran

```

def display_information(age):
    print("Âge de l'utilisateur : ", age)

```

Figure 4 -Code Python pour l'affichage

```

using System;

public class Utils
{
    public static void DisplayInformation(string age)
    {
        Console.WriteLine("Âge de l'utilisateur : " + age + "");
    }
}

```

Figure 5 -Code C# pour l'affichage

```

#include "Utils.h"
#include <windows.h>

void Utils::DisplayInformation(std::string age)
{
    std::cout << "Age de l'utilisateur : " << age << std::endl;
}

```

Figure 6 -Code C++ pour l'affichage

Impression de la même information qu'à l'étape précédente, mais à partir d'une imprimante, donc, effectuer un appel à l'imprimante (par défaut)

```
from win32 import win32print

def print_information(age):

    # -1- Trouver l'imprimante par défaut.
    printer_name = win32print.GetDefaultPrinter()

    # -2- Obtenir le handle d'imprimante pour effectuer les opérations
    # d'imprimerie.
    printer_handle = win32print.OpenPrinter(printer_name)

    doc_info = ("Age_Utilisateur", None, "TEXT")

    # -3- Notifie le print spooler de notre document d'impression (gestionnaire
    # de l'ordre des impressions). Crée une job d'impression.
    win32print.StartDocPrinter(printer_handle, 1, doc_info)

    # -4- Notifie le print spooler de notre page d'impression.
    win32print.StartPagePrinter(printer_handle)

    age_raw = age.encode("utf-8")

    # -5- Notifie le print spooler de nos données d'impression.
    win32print.WritePrinter(printer_handle, age_raw)

    # -6- Notifie le print spooler que l'application est à la fin de la page
    # d'une impression.

    win32print.EndPagePrinter(printer_handle)

    # -7- Met fin à la job d'impression.
    win32print.EndDocPrinter(printer_handle)

    # -8- Ferme l'objet handle d'imprimante.
    win32print.ClosePrinter(printer_handle)
```

Figure 7 -Code Python pour l'impression


```

using System;
using System.Drawing;
using System.Drawing.Printing;

public class Utils
{
    public static void Printing(string age)
    {
        PrintDocument printDocument = new PrintDocument();
        printDocument.DocumentName = "Age_Utilisateur";
        printDocument.PrintPage += (sender, e) =>
        {
            Font printFont = new Font("Arial", 12);
            SolidBrush brush = new SolidBrush(Color.Black);

            PointF point = new PointF(100, 100);

            e.Graphics.DrawString(age, printFont, brush, point);
        };
        printDocument.Print();
    }
}

```

Figure 8 -Code C# pour l'impression

```

#include "Utils.h"
#include <windows.h>

void Utils::Printing(std::string age)
{
    DWORD dwSize = 0;
    GetDefaultPrinter(NULL, &dwSize);

    // Allouer de la mémoire pour le nom de l'imprimante par défaut.
    wchar_t* pDefaultPrinterName = new wchar_t[dwSize];

    if (GetDefaultPrinter(pDefaultPrinterName, &dwSize)) {
        std::wcout << L"Imprimante par défaut : " << pDefaultPrinterName << std::endl;
    }

    HANDLE handlePrinter = NULL;
    LPHANDLE lphandlePrinter = &handlePrinter;

    PRINTER_DEFAULTS printerDefaults;
    printerDefaults.pDatatype = NULL;
    printerDefaults.pDevMode = NULL;
    printerDefaults.DesiredAccess = PRINTER_ACCESS_USE;

    OpenPrinter(pDefaultPrinterName, lphandlePrinter, NULL);

    DWORD level = 1;

    const int length = age.length();
    char* data = new char[length + 1];
    strcpy_s(data, length+1, age.c_str());

    DWORD dataSize = sizeof(data);
    dataSize = dataSize - 1;

    DOC_INFO_1 docInfo;

    const wchar_t* docName = L"Age_Utilisateur";
    LPTSTR lpDocName = const_cast<LPTSTR>(docName);

    const wchar_t* dataType = L"TEXT";
    LPTSTR lpDataType = const_cast<LPTSTR>(dataType);

    docInfo.pOutputFile = NULL;
    docInfo.pDocName = lpDocName;
    docInfo.pDatatype = lpDataType;

    DWORD printJob = StartDocPrinter(handlePrinter, level, (LPBYTE)&docInfo);

    StartPagePrinter(handlePrinter);

    DWORD bytesWritten = 0;

    if (!WritePrinter(handlePrinter, (LPVOID)data, dataSize, &bytesWritten)) {
        std::cerr << "Echec de l'impression!" << std::endl;
        ClosePrinter(handlePrinter);
        return;
    }

    EndPagePrinter(handlePrinter);

    EndDocPrinter(handlePrinter);

    ClosePrinter(handlePrinter);

    delete[] pDefaultPrinterName;
}

```

Figure 9 -Code C++ pour l'impression

2.3 Au niveau de la complexité de la programmation, que remarquez-vous entre ces trois langages ?

Avant-propos : Il est à noter que le code C#, pour l'impression à l'imprimante, utilise une bibliothèque graphique faisant partie du système d'exploitation de Windows, contrairement aux codes Python et C++ qui utilisent les mêmes fonctions liées à l'impression sous Windows.

Il est possible de remarquer que plus le langage de programmation est de bas niveau, autrement dit, plus le langage est près des instructions-machine, plus la programmation est complexe. Si l'on compare les deux codes Python (code haut niveau) et C++ (code bas niveau) pour l'impression avec l'imprimante par exemple, il est possible de voir que le code Python ne nécessite que 10 lignes de code, alors que celui en C++ nécessite 39 lignes. De plus, l'implémentation des fonctions sous Python est beaucoup plus simple, du fait que les paramètres qui les composent sont plus directement accessibles. Il est donc possible d'en conclure que bien que les deux codes utilisent les mêmes fonctions d'imprimerie pour interagir avec le système d'impression de Windows, il y a un certain nombre d'opérations intermédiaires qui sont effectuées en arrière-plan par l'interpréteur du code Python pour en arriver au même résultat qu'avec le code C++. Après tout, puisque le système d'exploitation est l'interface permettant aux applications de communiquer avec le pilote de l'imprimante (par la suite, du pilote à l'imprimante), il doit donc recevoir les mêmes instructions d'entrées si l'on veut obtenir un même résultat. Ce sont précisément les opérations intermédiaires d'arrière-plan qui rend le langage Python plus facile à utiliser, cependant cela peut représenter un inconvénient lorsqu'on veut obtenir un plus grand contrôle de notre programmation, puisqu'on ne sait pas toujours tout ce qui est fait en arrière-plan.

3. Bases de données

3.1 Quel est le rôle du système d'exploitation (séquence d'étapes) lorsqu'un langage de programmation transmet une requête vers un serveur de base de données (ex : MySQL)?

Le rôle du système d'exploitation est celui d'un intermédiaire entre le langage de programmation (application du langage) et le serveur de base de données. Il va ainsi s'occuper des opérations d'allocation des ressources, de création et de destruction du réseau de communication entre les deux entités, de gestion de ce dernier ainsi que de la transmission des données. Voici un résumé de la séquence d'étapes :

1. Le code de l'application (côté client) utilise l'interface de programmation d'application ou API (via une librairie par exemple) propre à la base de données, permettant ainsi l'interaction avec son système de gestion (SGBD), pour transmettre une requête.

2. Il y a transformation de l'appel API de haut niveau en plusieurs appels de plus bas niveau, que le système d'exploitation peut comprendre. Ces appels de bas niveau comprennent des directives pour l'établissement des communications entre les applications.
3. Lorsque le système d'exploitation aura donné la priorité d'exécution à l'application, côté client, il va exécuter ces instructions de bas niveau tout en allouant les ressources nécessaires pour effectuer la communication entre l'application client et l'application côté serveur (base de données).
4. Le système d'exploitation va créer un réseau de communication entre l'application client et le serveur de base de données, il va donc ouvrir les ports de communication.
5. La requête initiale est ainsi adaptée au formatage de la base de données. Le système d'exploitation la transmet par le réseau de communication précédemment créé.
6. Le serveur de base de données va ensuite traiter la requête, suit à sa réception par son interface réseau. Une fois la requête traitée, le serveur va renvoyer la réponse sur le port du réseau de communication créé initialement par le système d'exploitation de l'application client.
7. Le système d'exploitation gère la transmission des données de la réponse de retour via le réseau de communication, jusqu'à l'application client en cours.
8. À la fin de l'échange, le système d'exploitation ferme les ports de communication et relâche les ressources y étant associées.

Médiagraphie

Chargement différé des images au niveau du navigateur pour le Web | Articles. (s. d.). web.dev.

Consulté 14 février 2024, à l'adresse <https://web.dev/articles/browser-level-image-lazy-loading?hl=fr>

Chromium Docs—Threading and Tasks in Chrome. (s. d.). Consulté 14 février 2024, à l'adresse

[https://chromium.googlesource.com/chromium/src/+main/docs/threading_and_tasks.md](https://chromium.googlesource.com/chromium/src/+/main/docs/threading_and_tasks.md)

Client Server Communication in Operating System. (2023, décembre 26). *GeeksforGeeks.*

<https://www.geeksforgeeks.org/client-server-communication-in-operating-system/>

Collins, E. (2021, février 23). *Does Antivirus Software Slow Down Your Computer?* MUO.

<https://www.makeuseof.com/does-antivirus-slow-down-computer/>

dotnet-bot. (s. d.). *Graphics Class (System.Drawing).* Consulté 14 février 2024, à l'adresse

<https://learn.microsoft.com/en-us/dotnet/api/system.drawing.graphics?view=dotnet-plat-ext-8.0>

drewbatgit. (2021, janvier 7). *Print Spooler API - Win32 apps.* [https://learn.microsoft.com/en-](https://learn.microsoft.com/en-us/windows/win32/printdocs/print-spooler-api)

[us/windows/win32/printdocs/print-spooler-api](https://learn.microsoft.com/en-us/windows/win32/printdocs/print-spooler-api)

Le, J. (2020, juin 9). *How Operating Systems Work : 10 Concepts you Should Know as a Developer.*

Medium. <https://data-notes.co/how-operating-systems-work-10-concepts-you-should-know-as-a-developer-8d63bb38331f>

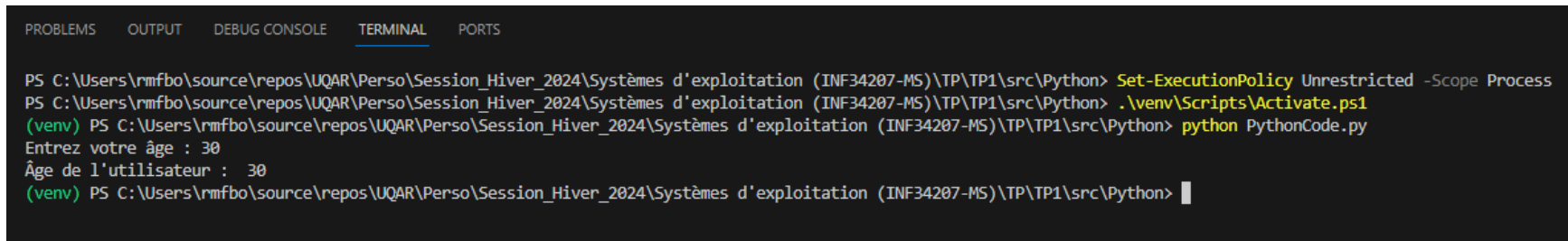
Why Does Google Chrome Have Multiple Processes Running. (2024, février 5). *Robots.Net.*

<https://robots.net/software-and-applications/browsers-and-extensions/why-does-google-chrome-have-multiple-processes-running/>

Annexe

```
# Main
user_age = entering_information()
display_information(user_age)
print_information(user_age)
```

Figure 10 - Code Python d'appel de fonctions (Main)



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\rmfbo\source\repos\UQAR\Perso\Session_Hiver_2024\Systèmes d'exploitation (INF34207-MS)\TP\TP1\src\Python> Set-ExecutionPolicy Unrestricted -Scope Process
PS C:\Users\rmfbo\source\repos\UQAR\Perso\Session_Hiver_2024\Systèmes d'exploitation (INF34207-MS)\TP\TP1\src\Python> .\venv\Scripts\Activate.ps1
(venv) PS C:\Users\rmfbo\source\repos\UQAR\Perso\Session_Hiver_2024\Systèmes d'exploitation (INF34207-MS)\TP\TP1\src\Python> python PythonCode.py
Entrez votre âge : 30
Âge de l'utilisateur : 30
(venv) PS C:\Users\rmfbo\source\repos\UQAR\Perso\Session_Hiver_2024\Systèmes d'exploitation (INF34207-MS)\TP\TP1\src\Python> 
```

Figure 11 - Capture d'écran du terminal du code Python

```

using System;
using Utils;

namespace CsharpCode
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            // 2.2 Présentez le code requis pour effectuer trois requêtes
            // vers le système d'exploitation :

            // saisie d'une information (entrée de données au clavier)
            string userAge = Utils.Utils EnteringInformation();

            // affichage de l'information à l'écran
            Utils.Utils.DisplayInformation(userAge);

            // impression de la même information qu'à l'étape précédente,
            // mais à partir d'une imprimante,
            // donc, effectuer un appel à l'imprimante (par défaut)
            Utils.Utils.Printing(userAge);

            Console.WriteLine("Done");
        }
    }
}

```

Figure 12 - Code C# d'appel de fonctions (Main)



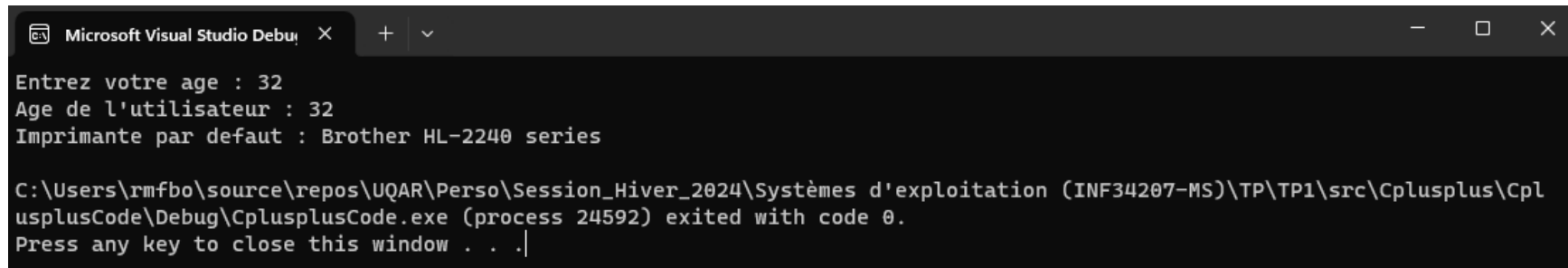
```
C:\Windows\system32\cmd.exe
Entrez votre âge : 31
Âge de l'utilisateur :31
Done
Appuyez sur une touche pour continuer... |
```

Figure 13 - Capture d'écran de la console du code C#

```
#include "Utils.h"

int main()
{
    std::string userAge = Utils::EnteringInformation();
    Utils::DisplayInformation(userAge);
    Utils::Printing(userAge);
}
```

Figure 14 - Code C++ d'appel de fonctions (Main)



```
Microsoft Visual Studio Debug Console
Entrez votre age : 32
Age de l'utilisateur : 32
Imprimante par défaut : Brother HL-2240 series

C:\Users\rmfbo\source\repos\UQAR\Perso\Session_Hiver_2024\Systèmes d'exploitation (INF34207-MS)\TP\TP1\src\Cplusplus\CplusplusCode\Debug\CplusplusCode.exe (process 24592) exited with code 0.
Press any key to close this window . . .
```

Figure 15 - Capture d'écran de la console du code C++