Deliverable : 2

Cross-Language Application Development

Group 1

<https://github.com/baralsamrat/MSCS632_Project_Group>

Samrat Baral

Prasanna Adhikari

Shashwat Baral

Sahithi Bontha

Spring 2025 Second Bi-term

Advanced Programming Languages -MSCS-632-B01

Dr. Dax Bradley

University of the Cumberlands

APRIL,2025

Deliverable : 2

Introduction

This repository contains an Expense Tracker Application implemented in Python and C++, demonstrating how different programming languages handle data structures, memory management, and error handling. The application allows users to:

● Add expenses with fields: Date, Amount, Category, and Description.

● Filter expenses by date range or category.

● View total expenses by category and overall.

● Compare implementations between Python and C++.

Both files are analyzed for key metrics that reflect the implementation of the core features (e.g., functions for filtering, summarizing, etc.).

Language-Specific Features

The C++ implementation shows memory management and use of STL containers, while the Python implementation (not shown here in detail) would use dictionaries, dynamic typing, and libraries like datetime. The metrics (and perhaps further analysis using tools like radon for Python or similar tools for C++) can be extended to reveal more about complexity and structure.

Side-by-Side Comparison:

By generating and visualizing these metrics, you can clearly see how each language's implementation compares in terms of code size and structure, which can help illustrate that both meet the core requirements while highlighting language-specific coding styles.

1. Memory Management:

C++:

C++ provides explicit memory management through pointers and manual allocation/deallocation using new and delete.

This offers fine-grained control but introduces the risk of memory leaks and dangling pointers if not handled carefully.

Impact:

Design: Requires careful consideration of object lifetimes and resource management.

Performance: Potential for optimized memory usage but demands meticulous coding.

Readability: Can lead to verbose code with explicit memory management constructs.

Code Snippet:

cpp

#include <iostream>

int main() {

int ptr = new int; // Allocate memory

ptr = 10;

std::cout << \*ptr << std::endl;

delete ptr; // Deallocate memory

return 0;

}

Python:

Python employs automatic memory management through garbage collection.

Developers are relieved from manual allocation/deallocation, reducing the risk of memory-related errors.

Impact:

Design: Simplifies object lifetime management.

Performance: Garbage collection overhead can impact performance in memory-intensive applications.

Readability: Results in cleaner and more concise code.

Code Snippet:

python

x = 10 # Memory allocation is automatic

print(x)

3. Standard Library Utilization:

C++:

C++ Standard Template Library (STL) provides powerful data structures and algorithms.

Offers high performance but requires understanding of templates and iterators.

Impact:

Design: Enables efficient data management and algorithm implementation.

Performance: Highly optimized for performance-critical applications.

Readability:Can be complex for beginners due to template syntax.

Code Snippet:

cpp

#include <iostream>

#include <vector>

#include <algorithm>

int main() {

std::vector<int> numbers = {3, 1, 4, 1, 5};

std::sort(numbers.begin(), numbers.end());

for (int num : numbers) {

std::cout << num << " ";

}

std::cout << std::endl;

return 0;

}

Python:

Python's standard library is extensive and provides a wide range of modules for various tasks.

Offers a high level of abstraction and ease of use.

Impact:

Design: Simplifies development with readily available modules.

Performance: Generally slower than C++ STL for performance-critical operations.

Readability: Contributes to concise and readable code.

Code Snippet:

python

numbers = [3, 1, 4, 1, 5]

numbers.sort()

print(numbers)

Conclusion:

C++ and Python offer distinct advantages and disadvantages in terms of design, performance, and readability. C++ excels in performance-critical applications with its fine-grained control and static typing. However, it requires careful memory management and can lead to complex code. Python prioritizes ease of use and rapid development with automatic memory management and a rich standard library. While it may have performance limitations, its concise syntax and readability make it suitable for a wide range of applications.

GitHub Repository:\*

The GitHub repository, [https://github.com/baralsamrat/MSCS632\_Project\_Group], contains the source code for the C++ and Python expense tracker applications, along with the metrics analysis script and this report. The repository's commit history reflects the individual contributions of each team member.