Unit Testing & cuTAGI

Miquel Florensa

Polythecnique Montréal

February 24, 2023

Table of Contents

What are Unit Tests?

- Unit tests are automated tests that are designed to verify the behavior of individual units or components of software applications.
- A unit can be a method, function, or class in the codebase, and the test ensures that the unit functions as intended and produces the expected output for a given set of inputs.
- Unit tests usually wants to compute extreme and not frequents values as well as big datasets to ensure the program will always work.

Why are Unit Tests Important?

- Unit tests help catch bugs early in the development process, before they become harder and more expensive to fix.
- They also help ensure that changes to the code don't break existing functionality.
- Unit tests can serve as documentation for the code, making it easier for other developers to understand how it works.
- They can also help improve the design of the code, by forcing developers to write code that is testable and modular.

Example: Testing a Function

Function to Test

```
int square(int x);
```

Returns the square of the input value.

Test Cases

- square(2) $\rightarrow$ 4
- square(0) $\rightarrow$ 0
- square(-3) $\rightarrow$ 9

Integration Tests

- Integration tests check how different parts of your system work together.
- They are used to ensure that components that work well in isolation also work well together.
- Integration tests are typically written by developers or testers who are familiar with the system as a whole, and are run less frequently than unit tests.

Functional Tests

- Functional tests are used to check that your system meets its functional requirements.
- They simulate user interactions with your system, and check that it behaves correctly in response to those interactions.
- Functional tests are typically written by testers or product owners, and are run less frequently than unit tests or integration tests.

Other Tests

- **Performance Tests:** Used to test the software's performance under specific conditions.
- **Security Tests:** Used to test the software's security and identify vulnerabilities or weaknesses in the system.
- **Acceptance Tests:** Used to test whether the software meets the acceptance criteria.
- **Regression Tests:** Used to ensure that changes to the software do not introduce new bugs or break existing functionality.

What is C++?

- C++ is a **general-purpose** programming language that was developed by Bjarne Stroustrup at Bell Labs in 1983. It is an extension of the C language and supports object-oriented programming (OOP) principles.
- C++ is a **statically typed language**, which means that variables must be declared with a specific data type and their types cannot be changed during runtime.

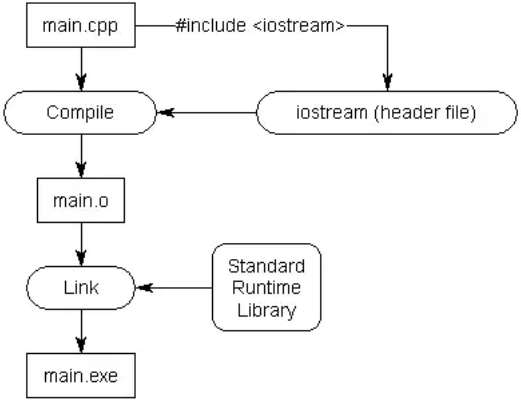
Key Features

- **Efficiency:** C++ is a high-performance language that is designed to be fast and efficient. It is **compiled** into machine code, which allows it to run quickly and use system resources efficiently.
- **Object-oriented programming:** supports OOP principles such as encapsulation, inheritance, and polymorphism, which allow for modular, reusable code.
- **Standard library:** has a rich standard library that includes data structures, algorithms, and I/O functions, which can be used to simplify programming tasks.
- **Low-level memory access:** allows for direct access to system memory, which can be useful for tasks that require low-level control over system resources.

Why is C++ faster than Python?

Python is an **interpreted** language, which means that the code is executed by an interpreter, rather than **compiled** into machine code. This allows for rapid development and prototyping, but can make Python slower than compiled languages like C++.

How C++ Compilation works:



Compilation Example:

main.cpp

```cpp
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Compilation Example:

bash

```
g++ -c main.cpp
```

This command will give us an object: `main.o`

Compilation Example:

bash

```
g++ -o main.exe main.o
```

This command will give us an executable: `main.exe`

Compilation Example:

We can now execute the code by doing:

bash input

```
./main.exe
```

bash output

```
Hello, World!
```

Cuda

CUDA stands for **"Compute Unified Device Architecture"** and is a parallel computing platform and programming model developed by NVIDIA. It allows developers to write programs that can run on NVIDIA GPUs to accelerate computationally intensive tasks.

Cuda example:

matrix_multiplication.cu

```
__global__ void matrix_multiply(int *A, int *B, int *C, int N) {
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;

    int sum = 0;
    for (int i = 0; i < N; i++) {
        sum += A[row * N + i] * B[i * N + col];
    }

    C[row * N + col] = sum;
}
```

cuTAGI Demo

cuTAGI

pyTAGI Demo

pyTAGI