Introduction to Computer Science: Programming Languages, Computation and Discrete Mathematics

James Aaron Erang [1]

[1] *Biotechnology and Analytical Laboratory, Department of Biological Sciences, College of Science, Central Luzon State University*

September 2024

Programming, Algorithms and Programs

A **program** is a sequence of instructions composed through **programming**. It includes several implemented **algorithms**, a step by step specification of procedures designed to solve a class of specific problems, and data structures.

> **Algorithm(s)** are step by step specifications of procedures or a finite sequence of mathematically rigorous instructions, designed to solve a class of specific problems or perform a computation.

1. Binary Search
2. Bubble Sort
3. Euclidean Algorithm
4. Sieves of Erastothenes

Programming languages are system of notation utilized in coding or writing a computer program. More often than not, programming languages are also used to implement a specific algorithm that solves a particular class of specific problems or perform a calculation. There are several examples programming langauges classified according its level of **abstraction**, a concept of generalizing concrete details or elements away from objects and systems.

A **high level** programming language can be defined as possessing a *strong* abstraction from the details of computer - such that it is easier to use due to its use of natural language, increasing its resemblance to human language. **Low level** programming language, on the other hand, provides little to no abstraction in the details and from instruction set architecture of computer - its resemblance is closer to *machine* language, thus, harder to use and understand:

1. High level
   (a) Python
   (b) Javascript
   (c) Java
   (d) Perl

2. Low level:
   (a) C
   (b) C++
   (c) x86 Assembly
   (d) Fortran

Code Equivalence

Programming language, as previous defined, is a system of notation that can be used to write algorithms or programs with specific function. Hence, similar functions or algorithms can be implemented in several programming languages. Consequently, due to **semantics** or **syntax** difference(s), sometimes negligible, but most of the time, notable, various code formatting or style discrepancies are observable:

```c
#include <stdio.h>
#include <string.h>

int main() { /* this is comment, this is
    ignored */
    char hello[] = "hello world!"; //
    this is variable

    if (strcmp(hello, "hello world!") ==
    0) {
        printf(
            "The tradition in programming
     is to print\n"
        );
        printf(
            "Hello World! Like this in C,
     %s\n", hello
        );

        for (int i = 1; i < 4; i++) { //
    loop over range of 1 to
            printf(                    //
    4 and print hello world!
                "%d hello world!\n", i
            );
        }
    }

    return 0;
}
```

Listing 1: `Hello world!` example in C Language.

```cpp
#include <cstring>
#include <iostream>

int main() {
    /* this is long comment */
    // this is ignored
    // this is variable
    char hello[] = "hello world!";

    // compare this variable to a string
    if ( strcmp(hello, "hello world!") ==
     0 ) {
        std::cout << "The tradition in
    programming is to"
            << std::endl;
        std::cout << "print Hello World!
    Like this in C++"
            << hello
            << std::endl;

        for (int i = 0; i < 3; i++) {
            std::cout <<
                i << " " << hello
                << std::endl;
        }
    }

    // return value
    return 0;
}
```

Listing 2: `Hello world!` example in C++ Language.

The programs exhibited above are written to (1.) set the value `"hello world!"` to variable `hello`; (2.) check if the value of variable `hello` is equal to `"hello world!"`; (3.) if the value is equal, print `"The tradition in programming is to print"` and print `"Hello World! Like this in %s"`; (4.) then *loop* or *repeat* i times while printing `"%i hello world!"`; (5.) finally, `return` a value of 0.

```
FUNCTION main:
    SET VALUE "hello world!" to hello
    CHECK IF hello IS EQUAL TO "hello world!"
    IF EQUAL
        PRINT "The tradition in programming is to print"
```

```
        PRINT "Hello World! Like this in %s"

        LOOP i TIMES
            PRINT "%i hello world!"
    RETURN 0
```

Listing 3: Pseudo-code equivalence of programs exhibited above.

As exhibited in Listing 1 and Listing 2, similar functionalities are programmed using C and C++, respectively. However, semantic differences introduces discrepancy between the formatting or styling of the code.

```rust
fn main() {
    /* this is a comment */
    // this is ignored
    // declare a variable
    let hello = String::from("hello world
    !");

    // compare this variable to a string
    if hello == "hello world!" {
        println!(
            "The tradition in programming
     is to print"
        );
        println!(
            "Hello World! Like this in
    Python, {}", hello
        );

        // loop over range of 1 to 4 and
    print hello world!
        for i in 1..4 {
            println!(
                "{} hello world!", i
            );
        }
    }
}
```

Listing 4: `Hello world!` example in Rust Language.

```python
class Main[Example]:
    """ Main object """
    def __init__(self) -> None:
        pass

    def main(self) -> int:
        """ this is a docstring """
        # this is ignored
        # this is a variable
        hello: str = "hello world!"

        # compare this variable to a
    string
        if hello == "hello world!":
            print("The tradition in
    programming is to print")
            print(f"Hello World! Like
    this in Python, {hello}")

            for i in range(1, 4):
                print(f"{i} Hello world!"
    )

        return 0

if __name__ == "__main__":
    Main().main()
```

Listing 5: `Hello world!` example in Python Language.

Similarly, Listing 4 and Listing 5 depicts notable difference between semantics and syntax of Rust and Python. Moreover, due to difference between the paradigm of two languages, significantly larger discrepancy was observable despite of its similar functionality.

An Example of Algorithm

One problems in computation is searching or finding an item in an array of elements. For example, in array of all biochemical compounds A, there exists a curative compound C, how would its *index* or *position* can be determined?

Given an array A of length n:

i_0	i_1	i_2	i_3	i_4	i_5	$\ldots$	i_{n-1}

How can the index i_n of the curative compound C be determined? One notable solution in this kind of problem is an algorithm called **binary search**.

```python
class BinarySearch:
    """ Binary search solution in Python """

    def __init__(self, arr: list[int]) -> None:
        self.arr: list[int] = arr

    def solution(self, T: int) -> int:
        """ Solution """

        while (L := 0) <= (H := (len(arr) - 1)):
            M: int = (H + L) // 2

            if self.arr[M] < T:
                L = M + 1
            elif self.arr[M] > T:
                H = M - 1
            else: # if self.arr[M] == T
                return M

        return 1
```

Listing 6: Implementation of binary search in Python 3.12

The algorithm involves several specified steps:

1. Set $L = 0$ and $H = n - 1$
2. Find the middle index M:

$$M = \left\lfloor \frac{(L + H)}{2} \right\rfloor$$

3. If $A_M < T$; set $L = M + 1$
4. However, if $A_M > T$; set $H = M - 1$
5. Return M if $A_M = T$
6. If $L > n$ return 1

Location of Antimicrobial Peptide MAGAININ1

Given a 1×9 array A retrieved from database of EMBL-EBI (European Molecular Biology Laboratory-European Bioinformatics Institute) ChEMBL using the query `magainin`:

$$A = \begin{bmatrix} \text{CHEMBL437357} \\ \text{CHEMBL409372} \\ \text{CHEMBL414933} \\ \text{CHEMBL4088094} \\ \text{CHEMBL1673385} \\ \text{CHEMBL1673394} \\ \text{CHEMBL1673389} \\ \text{CHEMBL1673395} \\ \text{CHEMBL412693} \end{bmatrix} \quad B = \begin{bmatrix} \text{Protein} \\ \text{Protein} \\ \text{Protein} \\ \text{Small Molecule} \\ \text{Protein} \\ \text{Protein} \\ \text{Protein} \\ \text{Small Molecule} \\ \text{Protein} \end{bmatrix}$$

What is the associated ChEMBLID of MAGAININ 1 and its molecule type from 1×9 array B if its index n is 0?

Mathematical Equations in Programming

Programming languages can also be used to implement mathematical functions in order to solve complex computational tasks or perform a repetitive mathematical calculations on a system:

Saturation Function in C

The saturation function r is defined as the quotient from the portion of bound ligand to the total amount of the macromolecule $r = [\mathrm{L}]_{\mathrm{bound}}/[\mathrm{M}]_0$:

$$r = \frac{\sum_{i=1}^{n} i \binom{n}{i} \left(\frac{[\mathrm{L}]}{K_D}\right)^i}{1 + \sum_{i=1}^{n} \binom{n}{i} \left(\frac{[\mathrm{L}]}{K_D}\right)^i}$$

$$\binom{n}{i} = \frac{n!}{(n-i)!i!}$$

Equation Set 1: Saturation function r of bound ligand to the macromolecule .

```c
#include <math.h>

unsigned int factorial(int N) {
    /*
     * c does not have any factorial
     * function, so you have to implement
     * your own factorial function if
     * you need to do factorial operations.
     */
    if (N <= 1) { return 1; }
    else {
        int prod = 1;
        for (int k = 1; k <= N; k++) {
            prod *= k;
        }
        return prod;
    }
}

int r_sat_func() {
    float r, r_num, r_den;
    // example arbitrary value, does not
    represent any
    float L = 50.05; float Kd = 1.65;
    int n = 5; int i = 4;

    unsigned int n_i_matrix =
        factorial(n)/factorial(n - i) * factorial
    (i);

    for (int j = 1; j == n; j++) {
        r_num += i * n_i_matrix * pow((L/Kd), i);
        r_den += 1 + n_i_matrix * pow((L/Kd), i);
    }
    return r_num + r_den;
}
```

Listing 7: Implementation of saturation function (Equation Set 1) in C.