Multi-Paradigm Programming

Repeat Assessment

Short report Procedural approach x object-oriented approach

Languages utilized: C, Python and Java

PROCEDURAL: C and Python

OBJECT-ORIENTED: Java

This report describes and compares different approaches for the GPA calculator

What is ***GPA***?

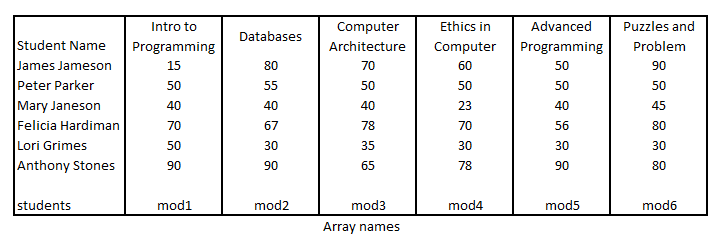
Based on the [Institute of Art Design + Technology](https://iadt.ie/for-students/exams-assessments/grade-point-average-gpa/#:~:text=Grade%20Point%20Average%20(GPA),-Your%20GPA%20is&text=GPA%20is%20the%20average%20grade,credits%2Fpostgraduate%20programmes%20vary).) website, “*GPA is the average grade points you have earned over a period of time (for instance a stage or your entire programme)”.*

With that in mind, I calculated the GPA based on the grade average of each student (sum all modules grades and divided it by the number of modules)

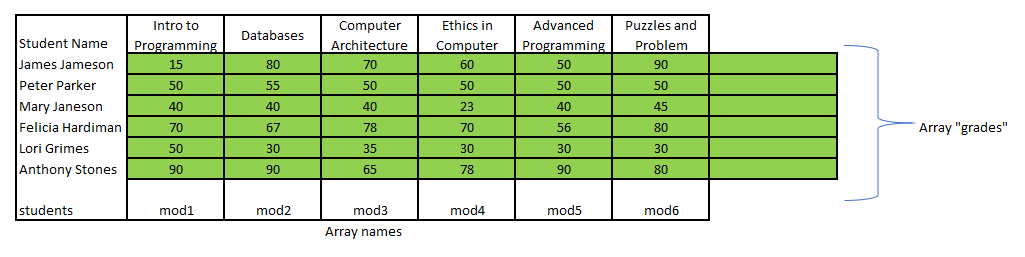
The CSV calculator reads a CSV file, that contains student names and their grades for different modules, and then output the results to the console.

The file can be divided into 7 different columns. The first column is the STUDENT NAME, and the other columns are the grades

I opted to convert each of the columns of the CSV file into arrays and work with them along all the program



Then, I created another array which would hold the grades for each student, based on the first arrays I created for the grade modules:



By following the procedure above, I was able to assign one array with all grades for each one of the elements in the array “students”

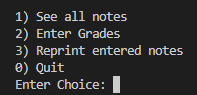
I opted to hardcode the grade names into an array called “module\_names”, as it was not said in the project description that these values would have to be changed

Finally, with all arrays ready for use, I created a series of looping to perform mathematic operations and then output the results

It was also much simpler to use these arrays as input values to all the functions that were needed to calculate the GPA and assign the correspondent “letter” for each grade

In the live session, I created an option for user to reprint previous entered values, in case of he wishes to reprint it

A “menu” was also created to facilitate the use of the program, making its use very simple, with the following options:



APPROACH COMPARISON

When I was creating the calculator, I could notice that both Java and C languages are way more verbose than the Python language. As an example, while Python would require only 1 single row to ask user to “enter an information and then read it”, in C it would take at least 2 rows, depending on the type of information that is being read:

printf("\nModule %d:\n", i + 1);

        scanf("%s", nM[i].new\_module\_name);

User input in C

System.out.println(String.format("\nModule %d:", i + 1));

            new\_mod\_name[i] = sc.next();

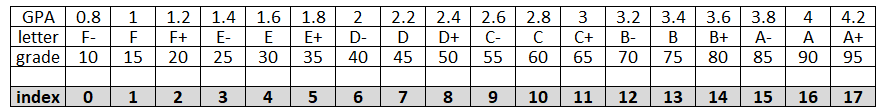
User input in Java

modName = str(input('Module %d:\n' %(i + 1)))

User input in Python

The 3 code statements above will perform the same operation: ask for a module name. In C, the information entered by user is being stored in an Struct array position; in Java, the information will be stored directly into an array; and in Python, the information is passed to a variable, which will at the end of the loop be appended into an array

In order to convert the percentage grade into a letter and then calculate the GPA, I created the following table schema, based on the project description:



The following aspects will be compared across the 3 implementations:

-Create arrays.

-Convert percentage grade to a letter.

-Calculate GPA.

-Calculate highest, minimum, mean and standard deviation values

1. Create arrays

In C, the CSV file was read into an *Struct Array*, and each column was split into a different array; we need to use a special function named “strcpy” to read strings in C, as it won’t accept a string to be directly assigned to a variable. I opted to parse the grades from struct into an array and coverts its value from *string* to *integer* using the command *“atof”*. For some reason, I was not able to populate the grades as integer values into the struct and had to use this workaround

In Python, the CSV file was read straight into arrays by using the function “append”, and no struct was used.

In Java, the CSV file was read into ArrayLists, by using the function “add”

Language C takes more commands to get an array populated than in Java or in Python

1. Convert percentage grade to a letter

A function called “get\_LETTER” was created in all the 3 implementations. The array “letter” was hardcoded and a “for” looping was created to compare the grades with the index

Then, if the grade value matches with the index value, the function will return the LETTER correspondent to that index value

Java and C were very similar on the way the looping was implemented:

for (int i = 10; i < 100; i += 5) {

        if (grade >= i) {

            if ((i + 5) > grade) {

                letter\_index = ((i / 5) - 2);

            }

        }

    }

    return letter[letter\_index];

Getting letter in C

for(int i = 10; i < 100; i += 5){

            if (grade >= i) {

                if ((i + 5) > grade) {

                    letter\_index = ((i / 5) - 2);

                }

            }

        }

        return letter[letter\_index];

Getting letter in Java

In Python, the function needed to be returned in the IF condition:

for i in range(10,100, 5):

        if(grade >= i):

            if ((i + 5) > grade):

                 letter\_index = ((i / 5) - 2)

                 return letter[int(letter\_index)]

To use the function “get\_LETTER” in Python and in C, it is only needed to call it as in the example below:

**In C:**

get\_LETTER(myNumbers[j])

**In Python:**

get\_LETTER(int(grades[j]))

**In Java:**

Because it is object oriented, we need first to create the constructor for it:

Get\_LETTER getletter = new Get\_LETTER();

Then, it can be applied wherever we need:

getletter.get\_Letter(grades[j])

1. Calculate GPA

The approach here was similar with the function to get a letter for each grade: a “for” looping was created to compare the grades with the index, as in the table schema above.

Then, if the grade value matches with the index value, the function will return the GPA value correspondent to that index value

To use the function “calculate\_GPA” in Python and in C, it is only needed to call it as in the example below:

**In C:**

calcule\_GPA(grades\_average)

**In Python:**

calculate\_GPA(grades\_average)

**In Java:**

Because it is object oriented, we need first to create the constructor for it:

Calculate\_GPA calcgpa = new Calculate\_GPA();

Then, it can be applied wherever we need:

calcgpa.calculate\_GPA(grades\_average)

1. Calculate highest, minimum, mean and standard deviation values

Similarities: The call for a function is similar in all implementations. The Standard Deviation function can be called on the same way on Python and C (both procedural):

In C:

StandardDeviation(myNumbers)

In Python:

(stddev(grades)

In Java it is a bit different: we need firstly to create the constructor for it:

Calc\_STD calcstd = new Calc\_STD();

Then, we can use it:

calcstd.calc\_std(grades)

Final Notes

I really enjoyed coding this calculator because it was a big challenge, specially for C and Java OOP.

Regarding the language C, it was very tricky because most of things needs to be done in a different way than in Java or Python, whereas in python everything is easier. I took only 3 days to complete the calculator in Python, while it took me 3 weeks to complete it in C

I’ve learned a lot from this challenge, and I hope I can use this knowledge in the future