Identification of the problem and requirements analysis of the integrative task

Oscar Stiven Munoz Ramirez

Programing and Algorithm

Nicolas Salazar

Systems Engineers

Cali, 1 March of 2023

Input and output variables

Double numberA: It is the initial number that the user will enter as at the beginning of the interval

Double number: It is the initial number that the user will enter as the end of the interval

Int option: This option is used to choose a condition from the program menus.

Double epsilon: It is the tolerance value that compares the validation of the bisection method

Output variables:

NumberC: It is the root result between the two arrays, if the interval entered exists

Identification of the problem and requirements analysis

Case of study: bisection method

Customer	Teacher Nicolas
Users	Users and Teacher Nicolas
Functional Requirements	RQ0-functionABS RQ1-functionPow RQ2-factorial RQ3-functionCosine RQ4-operationOne RQ5-operationTwo RQ6-operationsThree RQ7-operationsfunction RQ8-validateDouble RQ9-validateInt RQ10-bisectionMethod
Context of problem	This integrator focuses on finding the root of a function using the bisection method. The bisection method is a numerical search algorithm used to find a solution to an equation. It is particularly useful for nonlinear equations that do not have an analytical solution or are difficult to solve using other methods. The bisection method is based on the intermediate value theorem and is applied to a continuous function f(x) over an interval [a, b] where f(a) and f(b) have opposite signs. The method successively divides the interval in half, finding the midpoint c = (a + b) / 2 and evaluating f(c). Depending on the sign of f(c), a new interval [a, c] or [c, b] is established, which is again divided in half. This process is repeated until a desired precision or a number close to the error tolerance is reached.
Functional not Requirements	-Can not use Java Math API library -Be recursive with loops, conditionals, functionals, etc.

Identifier and name	RQ0-functionABS		
Summary	This control method checks the absolute value of a number using a conditional statement. Its parameter is a double data type called "number", which passes through a condition to return the absolute value and store it in the variable "result".		
Input	**Input name**	**Typedate**	**Conditions of valid values**
	number	double	all real number
Result or Postcondition	Check if the variable "number" is negative to return a positive and if it is, do not make any changes		
Exit	**Exit name**	**Typedate**	**Format**
	result	double	Double number

Identifier and name	RQ1-functionPow		
Summary	This control method calculates the power of a number using a loop. Its parameters are two double data types. The first parameter is called "base", which is the number that will be multiplied by itself the number of times specified by the "index" variable. The result will then be assigned to the variable "result", which stores the value of the power.		
Input	**Input name**	**Typedate**	**Conditions of valid values**
	base	double	All real number
	indice	int	All integer
Result or Postcondition	Once the loop ends, the variable "result" will be returned		
Exit	**Exit name**	**Typedate**	**Format**
	result	double	All real number

Identifier and name	RQ2-factorial		
Summary	This control method calculates the factorial of a number using a loop. The method takes a single parameter called "number", which is multiplied from 1 up to its value, and the result is returned in a variable called "result".		
Input	**Input name**	**Typedate**	**Conditions of valid values**

	number	int	All integer
Result or Postcondition	Once the loop ends, the variable "result" will be returned		
Exit	**Exit name**	**Typedate**	**Format**
	result	double	All real number

Identifier and name	RQ3-functionCosine		
Summary	This control method calculates the cosine of a number using a loop. The method takes a single parameter called "number", which is the value to be entered as an argument. The method works by using the formula: Summation(i=0 to infinite) (((-1)^i)* (x^2i))/(2i)!, where other methods such as "functionPow(number, 2i)" and "factorial(2.i)" are called to assist in the calculation.		
Input	**Input name**	**Typedate**	**Conditions of valid values**
	number	double	All real number
Result or Postcondition	Once the loop ends, the variable "result" will be returned		
Exit	**Exit name**	**Typedate**	**Format**
	result	double	Real number

Identifier and name	RQ4-operationOne		
Summary	This control method takes a single parameter called "number", which receives a number to perform the formula 1.F(x) = 2 * Cos(x^2). The method uses the "functionPow" and "functionCosine" methods to assist in the calculation, and the result is stored in the variable called "result".		
Input	**Input name**	**Typedate**	**Conditions of valid values**
	number	double	All real number
Result or Postcondition	Once the operation is finished, the variable "result" will be returned		
Exit	**Exit name**	**Typedate**	**Format**
	result	double	Real number

Identifier and name	RQ5-operationTwo

Summary	This control method takes a single parameter called "number", which receives a number to perform the formula 2. F(x) = 3x^3 + 7x^2 + 5. The method uses the "functionPow" method to assist in the calculation, and the result is stored in the variable called "result".		
Input	**Input name**	**Typedate**	**Conditions of valid values**
	number	double	All real number
Result or Postcondition	Once the operation is finished, the variable "result" will be returned		
Exit	**Exit name**	**Typedate**	**Format**
	result	double	Real number

Identifier and name	RQ6-operationsThree		
Summary	This control method takes a single parameter called "number", which receives a number to perform the formula F(x) = x * Cos(x). The method uses the "functionCosine" method to assist in the calculation, and the result is stored in the variable called "result".		
Input	**Input name**	**Typedate**	**Conditions of valid values**
	number	double	All real number
Result or Postcondition	Once the operation is finished, the variable "result" will be returned		
Exit	**Exit name**	**Typedate**	**Format**
	result	double	Real number

Identifier and name	RQ7-operationsfunction		
Summary	This control method takes two parameters called "number" and "option". Depending on the value of "option", one of three functions will be performed: "functionOne(number)", "functionTwo(number)", or "functionThree(number)". The method uses a switch statement to determine which function to call. A convergence criterion is used to ensure that the three functions converge properly. The result is stored in a variable called "result" and returned by the method.		
Input	**Input name**	**Typedate**	**Conditions of valid values**
	number	double	All number real
	option	int	It only accepts the values: 1,2 and 3

Result or Postcondition	Once the operation is finished, the variable "result" will be returned		
Exit	**Exit name**	**Typedate**	**Format**
	result	double	Real number

Identifier and name	RQ8-validateDouble		
Summary	This conditional control method checks if the entered data is of type "double". If it is, the data is stored in the variable "option". If not, the method prompts the user to enter the data again using a loop until a valid "double" value is entered.		
Input	**Input name**	**Typedate**	**Conditions of valid values**
	Does not apply	Does not apply	Does not apply
Result or Postcondition	Returns the variable "option"		
Exit	**Exit name**	**Typedate**	**Format**
	option	double	Real number

Identifier and name	RQ9-validateInt		
	This conditional control method checks if the entered data is of type "int". If it is, the data is stored in the variable "option". If not, the method prompts the user to enter the data again using a loop until a valid "int" value is entered.		
Summary	**Input name**	**Typedate**	**Conditions of valid values**
	Does not apply	Does not apply	Does not apply
Result or Postcondition	Returns the variable "option"		
Exit	**Exit name**	**Typedate**	**Format**
	option	double	Real number

Identifier and name	RQ10-bisectionMethod

Summary	This control method has the objective of searching for the root within two intervals in a function, as parameters are the two intervals, the option to search for the chosen function within a subroutine. The method is:
	f(a) × f(b) < 0{
	`do{`
	`    c = (a + b)/2`

```
            if( f(a) * f(b) < 0)
                   b = c
            si no
                   a = c
      }While
            (f(c) > épsilon)

}
```

to return the value of the root at the end

Input	**Input name**	**Typedate**	**Conditions of valid values**
	NumberA	Double	It is the first interval and accept all real number
	NumberB	Double	It is the first number of the interval and accept all real number
	epsilon	Double	It is the tolerance value that compares the validation of the bisection method, accept all real number
	option	int	Alone is accept the numbers 1,2 or 3.
Result or Postcondition	Returns the closest possible root of the two intervals according to the function		
Exit	**Exit name**	**Typedate**	**Format**
	Root	Double	Real number