```
Input: function f(x), float a, float b
Output: solution vector results

begin bisection:
    array results
    f_a <- function(a)
    f_b <- function(b)
    if (f_a * f_b >= 0) then
        return 0
    else:
                array aux
        mp <- (a + b)/2
        f_mp <- (function(mp)
                aux <- [a,mp,b]
        results add(aux)
        cont <- 1
        while (cont <= 2) do
            if (f_a * f_mp < 0) then
                b <- mp
            else:
                a <- mp
            p_0 <- mp
            mp <- (a + b)/2
            f_mp <- function(mp)
            cont <- cont + 1
            aux <- [a,mp,b]
                        results add(aux)
        return results

end Bisection

Input: function f(x), float x_0, x_1, float tolerance, int iterations
Output: solution vector results

begin aitkent:
    array results
    bisectionResult <- bisection(function,x_0,x_1)
    infinite <- MAXIMUM FLOAT VALUE
    if (bisectionResult != 0) then
        count <- 1
        error <- infinite
        xAitken0 <- 0
        while (count <= iterations
        and error > tolerance and error != 0 and bisectionResult != 0) do
            x1 <- bisectionResult[0][1]
            x2 <- bisectionResult[1][1]
```

```
            x3 <- bisectionResult[2][1]
            xAitken <- (x1 * x3 - (x2 ** 2)) / (x3 - 2 * x2 + x1)
            f_xAitken <- function(xAitken)
            error <- |xAitken0 - xAitken|
            if (error == 0) then
                error <- infinite
            xAitken0 <- xAitken
                        array aux = [count,xAitken,f_xAitken,error]
            results[count] <- aux
            x_0 <- bisectionResult[1][0]
            x_1 <- bisectionResult[1][2]
            bisectionResult <- bisection(function,x_0,x_1)
            count <- count + 1
    for key in results do:
        if (results[key][3] == infinite) then
            results[key][3] <- 0
    return results

end aitken
```