

Pruebas

Este documento muestra los cálculos para 6 arreglos diferentes con los que este algoritmo (en su forma recursiva) funciona, y el propósito del documento es respaldar los resultados arrojados por el código compilado en RISC-V, mostrando que el proceso es el mismo descrito por el código de alto nivel.

Contenido :

Comparación de resultados de una versión comentada del código original (este código modificado está hasta abajo), con cálculos en hojas y un diagrama por arreglo.

El proceso para comprobar que funciona correctamente es el siguiente :

Ingresar valores de los ejemplos, en el código compilado y ejecutar el código analizando que los cálculos son los mismos por cada arreglo.

1 2 3 4 5 6 7 8 9 10
a = {-31, 0, 1, 2, 2, 4, 65, 83, 99, 782} - original

```

int k = i + ((j-i)/2)
(a, x, i, j) → (a, 5, 0, 9)
j < i? 9 < 0? X K = 0 + ((9-0)/2) = 4
a[4] = 5? 2 = 5? X
a[4] < 5? 2 < 5? ✓ → (a, 5, 5, 9)
(a, x, k, j)
(a, 5, 5, 9)
j < i? 9 < 5? X K = 5 + ((9-5)/2) = 7
a[7] = 5? 83 = 5? X
a[7] < 5? 83 < 5? X → (a, 5, 6, 7)
(a, x, k, j)
(a, 5, 5, 6)
j < i? 6 < 5? X K = 5 + ((6-5)/2) = 5
a[5] = 5? 4 = 5? X
a[5] < 5? 4 < 5? ✓ → (a, 5, 6, 6)
(a, x, k, j)
(a, 5, 6, 6)
j < i? 6 < 6? X K = 6 + ((6-6)/2) = 6
a[6] = 5? 65 = 5? X
a[6] < 5? 65 < 5? X → (a, 5, 6, 5)
(a, x, i, k-1)
(a, 5, 6, 5)
j < i? 5 < 6? ✓ → return -1

```

Objetivo (x) = 5, Número de elementos de arreglo = 10



RESULTADOS :

no está 5

resultado esperado : índice = -1

```

root@DESKTOP-REBRR6L:/mnt/c/Users/Santi/Desktop/PP/RISC-V/R# ./elef
{-31, 0, 1, 2, 2, 4, 65, 83, 99, 782};
(0, 9) k : 4

(k+1,j) : (5, 9) porque 2 < 5
(5, 9) k : 7

(i,k-1) : (5, 6) porque 83 !< 5
(5, 6) k : 5

(k+1,j) : (6, 6) porque 4 < 5
(6, 6) k : 6

(i,k-1) : (6, 5) porque 65 !< 5

5 < 6 entonces 5 no está en el arreglo
el valor del índice es -1

```

1 2 3 4 5 6 7 8 9 10
 $a = \{-31, 0, 1, 2, 3, 4, 5, 83, 99, 782\};$

```

int k = i + ((j - i) / 2)
(a, x, i, j) → (a, 5, 0, 9)
j < i? 9 < 0? X, K = 0 + ((9 - 0) / 2) = 4
a[4] = 5? 3 = 5? X
a[4] < 5? 3 < 5? ✓ → (a, 5, 5, 9)
(a, x, k, j)
(a, 5, 5, 9)
j < i? 9 < 5? X, K = 5 + ((9 - 5) / 2) = 7
a[7] = 5? 83 = 5? X
a[7] < 5? 83 < 5? X → (a, 5, 5, 6)
(a, x, i, k - 1)
(a, 5, 5, 6)
j < i? 6 < 5? X, K = 5 + ((6 - 5) / 2) = 5
a[5] = 5? 4 = 5? X
a[5] < 5? 4 < 5? ✓ → (a, 5, 6, 6)
(a, x, k + 1, j)
(a, 5, 6, 6)
j < i? 6 < 6? X, K = 6 + ((6 - 6) / 2) = 6
a[6] = 5? 5 = 5? ✓ → return K(6);

```

Objetivo (x) = 5, Número de elementos de arreglo = 10

-31	0	1	2	3	4	5	83	99	782
-31	0	1	2	3	4	5	83	99	782
-31	0	1	2	3	4	5	83	99	782
-31	0	1	2	3	4	5	83	99	782

RESULTADOS :

está 5

resultado esperado : índice = 6

```

root@DESKTOP-REBRR6L:/mnt/c/Users/Santi/Desktop/PP/RISC-V/R# ./elef
{-31, 0, 1, 2, 3, 4, 5, 83, 99, 782}
(0, 9) k : 4

(k+1,j) : (5, 9) porque 3 < 5

(5, 9) k : 7

(i,k-1) : (5, 6) porque 83 !< 5

(5, 6) k : 5

(k+1,j) : (6, 6) porque 4 < 5

(6, 6) k : 6

el valor del índice es 6

```

1 2 3 4 5 6 7 8 9 10
 $a = \{-31, 0, 1, 2, 2, 4, 5, 65, 99, 782\}$

```

int k = i + ((j - i) / 2)
(a, x, i, j) → (a, 5, 0, 9)
j < i? 9 < 0? X, K = 0 + ((9 - 0) / 2) = 4
a[4] = 5? 2 = 5? X
a[4] < 5? 2 < 5? ✓ → (a, 5, 5, 9)
(a, 5, 5, 9)
j < i? 9 < 5? X, K = 5 + ((9 - 5) / 2) = 7
a[7] = 5? 65 = 5? X
a[7] < 5? 65 < 5? X → (a, 5, 5, 6)
(a, 5, 5, 6)
j < i? 6 < 5? X, K = 5 + ((6 - 5) / 2) = 5
a[5] = 5? 4 = 5? X
a[5] < 5? 4 < 5? ✓ → (a, 5, 6, 6)
(a, 5, 6, 6)
j < i? 6 < 6? X, K = 6 + ((6 - 6) / 2) = 6
a[6] = 5? 5 = 5? ✓ → return K (6);

```

Objetivo (x) = 5, Número de elementos de arreglo = 10

-31	0	1	2	2	4	5	65	99	782
-31	0	1	2	2	4	5	65	99	782
-31	0	1	2	2	4	5	65	99	782
-31	0	1	2	2	4	5	65	99	782

RESULTADOS :

está 5, y en el mismo índice aunque con algunos
 elementos diferentes,
 pero debería dar mismo el resultado
 resultado esperado : índice = 6

```

root@DESKTOP-REBRR6L:/mnt/c/Users/Santi/Desktop/PP/RISC-V/R# ./elef
{-31, 0, 1, 2, 2, 4, 5, 65, 99, 782}
(0, 9) k : 4

(k+1,j) : (5, 9) porque 2 < 5

(5, 9) k : 7

(i,k-1) : (5, 6) porque 65 !< 5

(5, 6) k : 5

(k+1,j) : (6, 6) porque 4 < 5

(6, 6) k : 6

el valor del índice es 6

```

1 2 3 4 5 6 7 8 9 10
 $a = \{-31, -15, -13, -2, 0, 3, 4, 5, 99, 300\}$

```

int K = i + ((j-i)/2)

(a, x, i, j) → (a, 5, 0, 9)
j < i? 9 < 0? X, K = 0 + ((9-0)/2) = 4
a[4] = 5? 0 = 5? X,
a[4] < 5? 0 < 5? ✓ → (a, 5, 5, 9)
                      (a, x, K+1, j)
(a, 5, 5, 9)
j < i? 9 < 5? X, K = 5 + ((9-5)/2) = 7
a[7] = 5? 5 = 5? ✓ → return K(7);
  
```

Objetivo (x)=5, Número de elementos de arreglo=10

-31	-15	-13	-2	0	3	4	5	99	300
-31	-15	-13	-2	0	3	4	5	99	300

RESULTADOS:

está 5, en diferente índice con algunos
 elementos diferentes,
 resultado esperado : índice = 7

```

el valor del índice es 6
root@DESKTOP-REBRR6L:/mnt/c/Users/Santi/Desktop/PP/RISC-V/R# gcc binsearch_ro.c -o elef
root@DESKTOP-REBRR6L:/mnt/c/Users/Santi/Desktop/PP/RISC-V/R# ./elef
{-31, -15, -13, -2, 0, 3, 4, 5, 99, 300}
(0, 9) k : 4

(k+1, j) : (5, 9) porque 0 < 5

(5, 9) k : 7
  
```

1 2 3 4 5 6 7 8 9 10
a = {3, 15, 213, 402, 500, 640, 1135, 1200, 1399, 2782}

```
int K = i + ((j-i)/2);

(a, x, i, j) → (a, 15, 0, 9)
j < i? 9 < 0? X, K = 0 + ((9-0)/2) = 4

a[4] = 15? 500 = 15? X,
a[4] < 15? 500 < 15? X, → (a, 15, 0, 3)
(a, x, i, K-1)

(a, 15, 0, 3)
j < i? 3 < 0? X, K = 0 + ((3-0)/2) = 1

a[1] = 15? 15 = 15? ✓, → return K(1);
```

Objetivo (x)=15, Número de elementos de arreglo=10

3	15	213	402	500	640	1135	1200	1399	2782
3	15	213	402	500	640	1135	1200	1399	2782
3	15	213	402	500	640	1135	1200	1399	2782

RESULTADOS:

se busca un número diferente + (15) con
 elementos diferentes,
 resultados esperados : índice = 1

```
el valor del índice es 6
root@DESKTOP-REBRR6L:/mnt/c/Users/Santi/Desktop/PP/RISC-V/R# gcc binsearch_ro.c -o elef
root@DESKTOP-REBRR6L:/mnt/c/Users/Santi/Desktop/PP/RISC-V/R# ./elef
{-31, -15, -13, -2, 0, 3, 4, 5, 99, 300}
(0, 9) k : 4

(k+1,j) : (5, 9) porque 0 < 5
(5, 9) k : 7
```


1 2 3 4 5 6 7 8 9 10
 $a = \{-782, -499, -203, -120, -42, -24, -15, -7, -2, 0\}$

```

int k = i + ((j-i)/2)
(a, x, i, j) → (a, -203, 0, 9)
j < i? 9 < 0? X, K = 0 + ((9-0)/2) = 4
a[4] = -203? -42 = -203? X, → (a, -203, 0, 3)
a[4] ≠ -203? -42 < -203? X, → (a, x, i, K-1)
(a, -203, 0, 3)
j < i? 3 < 0? X, K = 0 + ((3-0)/2) = 1
a[1] = -203? -499 = -203? X, → (a, -203, 2, 3)
a[1] < -203? -499 < -203? ✓, → (a, x, K+1, j)
(a, -203, 2, 3)
j < i? 3 < 2? X, K = 2 + ((3-2)/2) = 2
a[2] = -203? -203 = -203? ✓ → return K(2);
  
```

Objetivo (x) = -203, Número de elementos de arreglo = 10

-782	-499	-203	-120	-42	-24	-15	-7	-2	0
-782	-499	-203	-120	-42	-24	-15	-7	-2	0
-782	-499	-203	-120	-42	-24	-15	-7	-2	0

RESULTADOS:

se busca un número diferente - (-203)
 elementos diferentes,
 resultados esperados : índice = 2

```

root@DESKTOP-REBRR6L:/mnt/c/Users/Santi/Desktop/PP/RISC-V/R# ./elef
{-782, -499, -203, -120, -42, -24, -15, -7, -2, 0}
(0, 9) k : 4

(i,k-1) : (0, 3) porque -42 !< -203

(0, 3) k : 1

(k+1,j) : (2, 3) porque -499 < -203

(2, 3) k : 2

el valor del índice es 2
  
```

Código modificado

nombre del archivo : binsearch_ro.c

```
#include <stdio.h>

int bsearch_r (int *a, int x, int i, int j) {
    if (j < i) {
        printf("\n\n %d < %d entonces %d no está en el arreglo", j, i, x);
        return -1;
    }
    int k = i + ((j - i) / 2);
    printf("\n (%d, %d)   k : %d\n", i, j, k);
    if (a[k] == x) {
        return k;
    }
    else if (a[k] < x) {
        printf("\n (k+1,j) : (%d, %d) porque %d < %d\n", k+1, j,
a[k], x);
        return bsearch_r(a, x, k + 1, j);
    }
    else {
        printf("\n (i,k-1) : (%d, %d) porque %d !< %d\n", i, k-1,
a[k], x);
        return bsearch_r(a, x, i, k - 1);
    }
}

int main () {
    int a[] = {-782, -499, -203, -120, -42, -24, -15, -7, -2, 0};
    int n = sizeof a / sizeof a[0];
    int x = -203;
    printf("{-782, -499, -203, -120, -42, -24, -15, -7, -2, 0}");
    int i = bsearch_r(a, x, 0, n - 1);
    printf("\nel valor del índice es %d\n", i);
    return 0;
}
```