}

Pruebas

Variables

int a (arreglo de elementos numéricos),
int n (tamaño de arreglo),
int x (target),
int i (target index).

arreglo original : a = {-31, 0, 1, 2, 2, 4, 65, 83, 99, 782}

main :

```
      a = {-31, 0, 1, 2, 2, 4, 65, 83, 99, 782}
      x = 5
      i = 0   → como argumento i = 0
      n = 10
      i = bsearch_r(a,x,0,n-1);
      para int i → int i = bsearch_r(a,x,0,n-1);  bsearch_r (int *a, int x, int i, int j) {
            0 bsearch_r :
            1.-  if (j < i) {              if (9 < 0) {

            return -1;
  }
  int k = i + ((j - i) / 2);           int k = 0 + ((9-0)/2) = 4
  if (a[k] == x) {                     if (a[4] == 5)  → a[4] = 2 !
     return k;
  }
  else if (a[k] < x) {                 else if (2 < 5) {
     return bsearch_r(a, x, k + 1, j);             return bsearch_r(a, 5, k+1(=5), 9);
  }
```

```
1_bsearch_r :   bsearch_r(a, 5, 5, 9);   →   bsearch_r (int *a, int x, int i, int j){
            1.-  if (j < i) {              if ( 9 < 5) {

            return -1;
  }
  int k = i + ((j - i) / 2);           int k = 5 + ((9-5)/2) = 7     → k = 7
  if (a[k] == x) {                     if (a[7] == 5)  → a[7] = 83 !
     return k;
  }
  else if (a[k] < x) {                 else if (83 < 5) {
     return bsearch_r(a, x, k + 1, j);             return bsearch_r(a, 5, k+1(=8), 9);
  }

  else {                                           else {
      return bsearch_r(a, x, i, k - 1);            return bsearch_r(a, 5, 5, k - 1(=6));
  }
```

```
2_bsearch_r :    bsearch_r(a, 5, 5, 4));     →      bsearch_r (int *a, int x, int i, int j){
                 1.-  if (j < i) {              if ( 4 < 5) {

                 return -1;        return -1;
   }
```

main :

```
i = -1;
if (i >= 0)                     if (i >= 0)  (❌)
     printf("%d is at index %d.\n", x, i);
else                            ← ✅
     printf("%d is not found.\n", x); ← ✅
return 0;
```

arreglo a :   a = {-31, 0, 1, 2, 2, 4, 5, 5, 99, 782}

main :

```
         a = {-31, 0, 1, 2, 2, 4, 5, 5, 99, 782}
         x = 5
         i = 0   → como argumento i = 0
         n = 10
         i = bsearch_r(a,x,0,n-1);
         para int i → int i = bsearch_r(a,x,0,n-1);  bsearch_r (int *a, int x, int i, int j) {
                  0 bsearch_r :
                  1.-  if (j < i) {              if (9 < 0) {

                  return -1;
   }
   int k = i + ((j - i) / 2);          int k = 0 + ((9-0)/2) = 4
   if (a[k] == x) {                    if (a[4] == 5)  → a[4] = 2 !
      return k;
   }
   else if (a[k] < x) {                else if (2 < 5) {
      return bsearch_r(a, x, k + 1, j);            return bsearch_r(a, 5, k+1(=5), 9);
   }
```

```
1_bsearch_r :    bsearch_r(a, 5, 5, 9);     →      bsearch_r (int *a, int x, int i, int j){
                 1.-  if (j < i) {              if ( 9 < 5) {

                 return -1;
   }
   int k = i + ((j - i) / 2);          int k = 5 + ((9-5)/2) = 7     → k = 7
   if (a[k] == x) {                    if (a[7] == 5)  → a[7] = 5
       return k;                           return 7;

   }
```

main :

i = -1;

```
if (i >= 0)                  if (i >= 0)  (❌)
    printf("%d is at index %d.\n", x, i);
else                         ← ✅
    printf("%d is not found.\n", x); ← ✅
return 0;
```

- prueba con arreglo de diferente objetivo
- prueba con arreglo de mayor tamaño
- prueba con arreglo de diferentes elementos