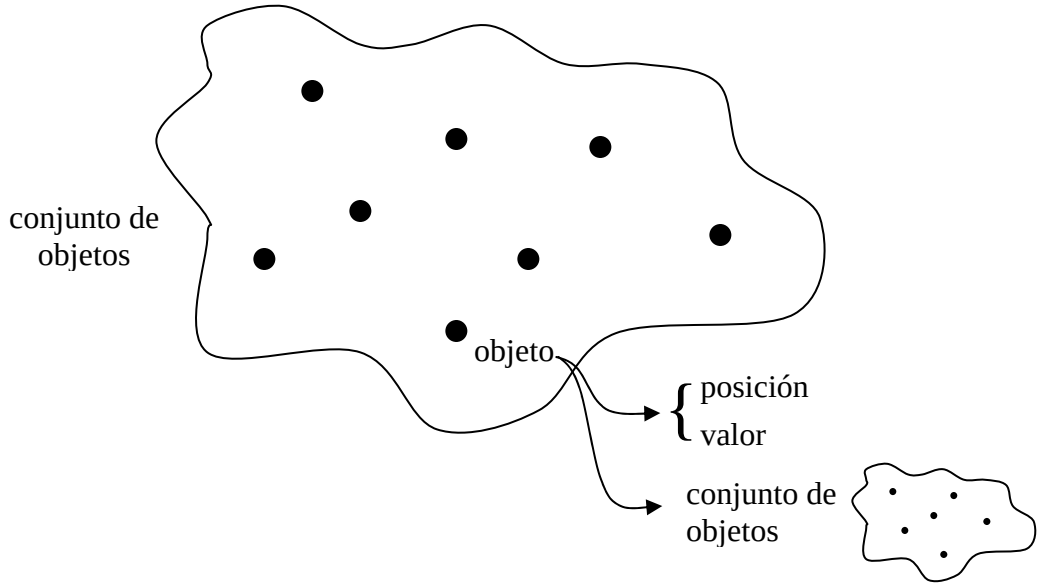
Algoritmos básicos



1. Búsqueda o localización de un objeto (o de un conjunto de objetos)
    a) Mínimo
    b) Máximo
    c) Valor
    d) Valor en un conjunto ordenado

2. Inserción de un objeto (o de un conjunto de objetos)

3. Extracción de un objeto (o de un conjunto de objetos)

4. Ordenación de objetos de un conjunto
    a) por inserción
    b) por selección

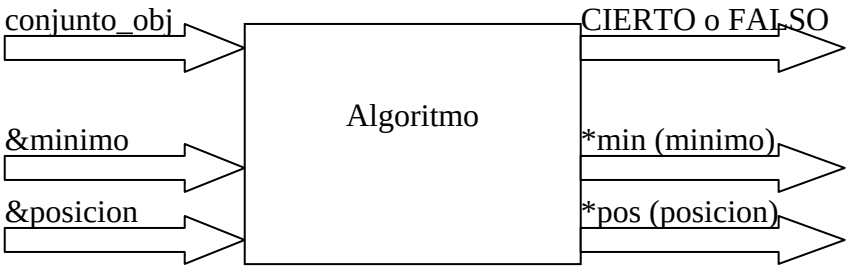
1.a) *Búsqueda de un objeto de valor mínimo*

Forma general de la llamada

```
ok = busqueda_minimo( conjunto_obj, &minimo, &posicion );
```

Cabecera de la función

```
int busqueda_minimo( ... conjunto, ... *min, ... *pos )
```



Ejemplo

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;

TDiagnostico busqueda_minimo( int vect[], int *min, int *pos )
{
  int i;
  *pos = 0;
  *min = vect[0];
  for( i = 1; i < DIM; i++ )
    if( *min > vect[i] )
    {
      *min = vect[i];
      *pos = i;
    }
  return CIERTO;
}
```

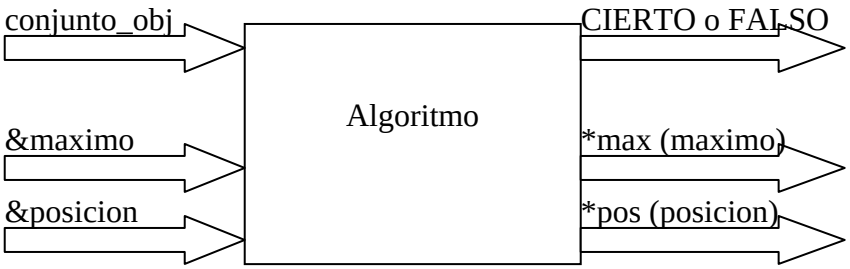
1.b) *Búsqueda de un objeto de valor máximo*

Forma general de la llamada

```
ok = busqueda_maximo( conjunto_obj, &maximo, &posicion );
```

Cabecera de la función

```
int busqueda_maximo( ... conjunto, ... *max, ... *pos )
```



Ejemplo

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;
typedef struct
{
  int num_el;
  float vect[DIM];
} Tvector;

TDiagnostico busqueda_maximo( Tvector v, float *max, int *pos )
{
  int i;
  if( v.num_el == 0 )
    return FALSO;
  *pos = 0;
  *max = v.vect[0];
  for( i = 1; i < v.num_el; i++ )
    if( *max < v.vect[i] )
    {
      *max = v.vect[i];
      *pos = i;
    }
  return CIERTO;
}
```

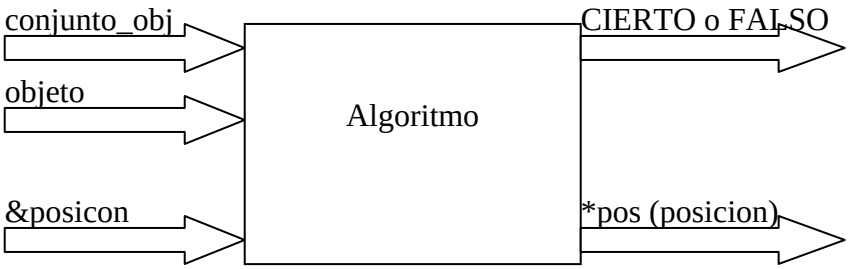
1.c) *Búsqueda de un objeto dado un valor*

Forma general de la llamada

```
ok = busqueda_valor( conjunto_obj, objeto, &posicion );
```

Cabecera de la función

```
int busqueda_valor( ... conjunto, ... obj, ... *pos )
```



Ejemplo
a) Búsqueda del primer objeto con un determinado valor

```
typedef enum {FALSO, CIERTO} TDiagnostico;

TDiagnostico busqueda_valor( char cad[], char letra, int *pos )
{
  *pos = 0;
  while( letra != cad[*pos] && (*pos) < strlen( cad ) )
    (*pos)++;
  if( *pos == strlen( cad ) )
    return FALSO;
  return CIERTO;
}
```

b) Búsqueda de todos los objetos con un determinado valor

```
typedef enum {FALSO, CIERTO} TDiagnostico;

TDiagnostico busqueda_valor( char cad[], char letra, int *pos )
{
  while( letra != cad[*pos] && (*pos) < strlen( cad ) )
    (*pos)++;
  if( *pos == strlen( cad ) )
    return FALSO;
  return CIERTO;
}

  ...
  posicion = 0;
  while( posicion < strlen( cadena ) )
  {
    if( busqueda_valor( cadena, letra, &posicion ) == CIERTO )
      ENCONTRADO!
    posicion++;
  }
  ...
```

c) Búsqueda del primer objeto con un determinado valor entre dos posiciones dadas

```
typedef enum {FALSO, CIERTO} TDiagnostico;

TDiagnostico busqueda_valor( char cad[], char letra, int *pos1, int pos2 )
{
  while( letra != cad[*pos1] && (*pos1) < pos2 )
    (*pos1)++;
  if( *pos1 == pos2 )
    return FALSO;
  return CIERTO;
}
```

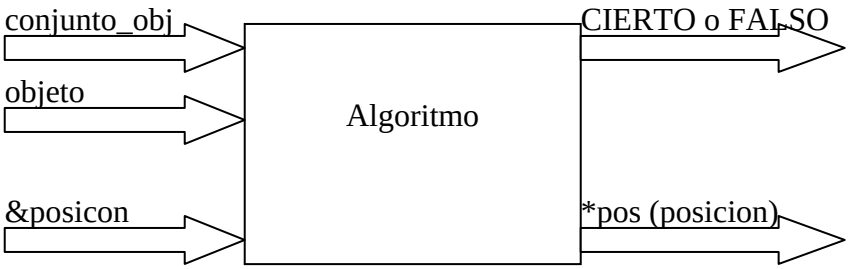
1.d) *Búsqueda de un objeto dado un valor en un conjunto ordenado*

Forma general de la llamada

```
ok = busqueda_ordenado( conjunto_obj, objeto, &posicion );
```

Cabecera de la función

```
int busqueda_ordenado( ... conjunto, ... obj, ... *pos )
```



Ejemplo

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;
typedef struct
{
  int valor;
  char nombre[DIM];
} Tobjeto;

typedef struct
{
  int num_el;
  Tobjeto obj[DIM];
} Tlista;

TDiagnostico busqueda_ordenado( Tlista lista, Tobjeto objeto, int *pos )
{
  *pos = 0;
  while( objeto.valor < lista.obj[*pos].valor && (*pos) < lista.num_el )
    (*pos)++;
  if( *pos == lista.num_el )
    return FALSO;
  if( objeto.valor == lista.obj[*pos].valor )
    return CIERTO;
  return FALSO;
}
```

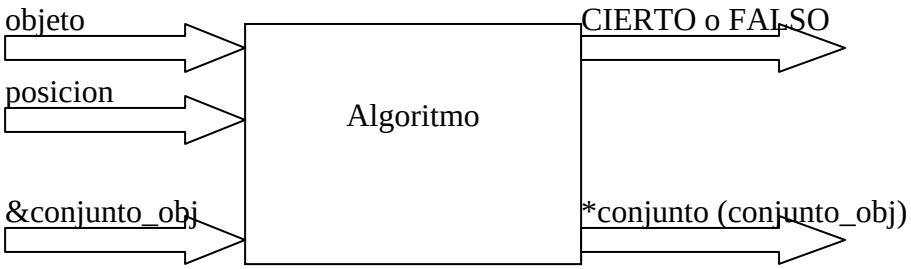
2) *Inserción de un objeto en un conjunto*

Forma general de la llamada

```
ok = insertar( &conjunto_obj, objeto, posicion );
```

Cabecera de la función

```
int insertar( ... *conjunto, ... obj, ... pos )
```



Ejemplo
a) Inserción de un objeto en una posición dada

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;
typedef struct
{
  int num_dep;
  float deposito[DIM];
} Tbanco;

TDiagnostico insertar( Tbanco *cuenta, float nuevo_dep, int pos )
{
  int i;
  if( pos < 0 || pos > cuenta->num_dep )
    return FALSO;
  if( cuenta->num_dep == DIM – 1 )
    return FALSO;
  for( i = cuenta->num_dep; i > pos; i-- )
    cuenta->deposito[i] = cuenta->deposito[i – 1];
  cuenta->deposito[pos] = nuevo_dep;
  cuenta->num_dep++;
  return CIERTO;
}
```

b) Inserción de un objeto en un conjunto ordenado

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;
typedef struct
{
  int num_dep;
  float deposito[DIM];
} Tbanco;
```

```
TDiagnostico busqueda_ordenado( Tbanco cuenta, float dep, int *pos )
{
  *pos = 0;
  while( dep < cuenta.deposito[*pos] && (*pos) < cuenta.num_dep )
    (*pos)++;
  if( *pos == cuenta.num_dep )
    return FALSO;
  if( dep == cuenta.deposito[*pos] )
    return CIERTO;
  return FALSO;
}

TDiagnostico insertar( Tbanco *cuenta, float nuevo_dep, int pos )
{
  int i;
  if( pos < 0 || pos > cuenta->num_dep )
    return FALSO;
  if( cuenta->num_dep == DIM – 1 )
    return FALSO;
  for( i = cuenta->num_dep; i > pos; i-- )
    cuenta->deposito[i] = cuenta->deposito[i – 1];
  cuenta->deposito[pos] = nuevo_dep;
  cuenta->num_dep++;
  return CIERTO;
}

  ...
  busqueda_ordenado( lista, dinero, &posicion );
  ok = insertar( &lista, dinero, posicion );
  ...
```

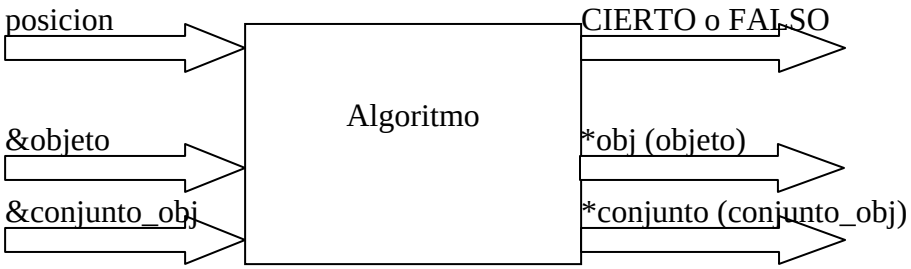
3) *Extracción de un objeto en un conjunto*

Forma general de la llamada

```
ok = extraer( &conjunto_obj, &objeto, posicion );
```

Cabecera de la función

```
int extraer( ... *conjunto, ... *obj, ... pos )
```



Ejemplo
a) Extracción de un objeto de una posición dada

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;
typedef struct
{
  int num_dep;
  float deposito[DIM];
} Tbanco;

TDiagnostico extraer( Tbanco *cuenta, float *dep, int pos )
{
  int i;
  if( pos < 0 || pos >= cuenta->num_dep )
    return FALSO;
  if( cuenta->num_dep == 0 )
    return FALSO;
  *dep = cuenta->deposito[pos];
  for( i = pos; i < cuenta->num_dep - 1; i++ )
    cuenta->deposito[i] = cuenta->deposito[i + 1];
  cuenta->num_dep--;
  return CIERTO;
}
```

b) Extracción de un objeto dado

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;
typedef struct
{
  int num_dep;
  float deposito[DIM];
} Tbanco;
```

```
TDiagnostico busqueda_valor( Tbanco cuenta, float dep, int *pos )
{
  *pos = 0;
  while( dep != cuenta.deposito[*pos] && (*pos) < cuenta.num_dep )
    (*pos)++;
  if( *pos == cuenta.num_dep )
    return FALSO;
  return CIERTO;
}

TDiagnostico extraer( Tbanco *cuenta, int pos )
{
  int i;
  if( pos < 0 || pos >= cuenta->num_dep )
    return FALSO;
  if( cuenta->num_dep == 0 )
    return FALSO;
  for( i = pos; i < cuenta->num_dep - 1; i++ )
    cuenta->deposito[i] = cuenta->deposito[i + 1];
  cuenta->num_dep--;
  return CIERTO;
}

  ...
  ok = busqueda( lista, dinero, &posicion );
  if( ok )
  {
    ok = extraer( &lista, posicion );
    ...
```

4.a) *Ordenación por inserción*

Forma general de la llamada

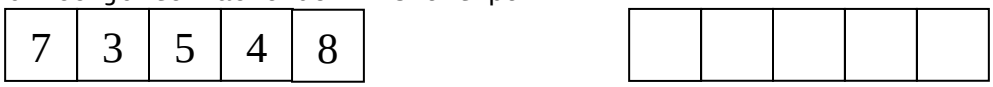
```
ok = ordena_ins( &conjunto_obj );
```

Cabecera de la función

```
int ordena_ins( ... *conjunto )
```

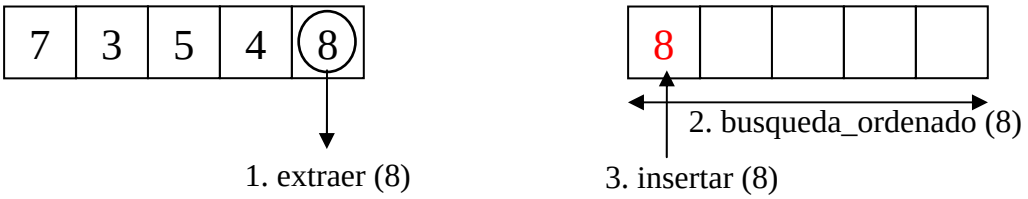
Ejemplo de principio

1. Se crea un conjunto vacío del mismo tipo



2. Bucle de 0 a número de elementos – 1

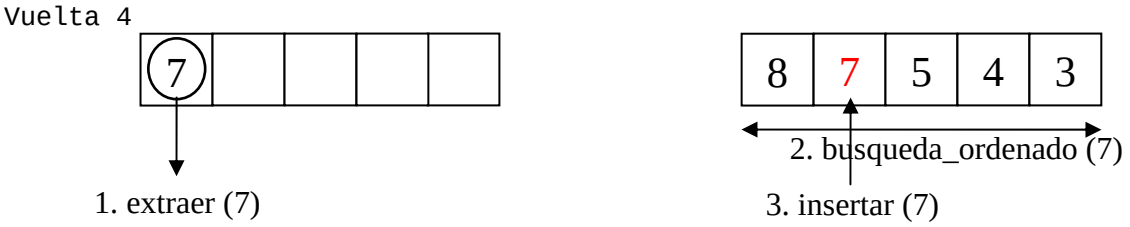
Vuelta 0



Vuelta 1

Vuelta 2

Vuelta 3



Ejemplo

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;
typedef struct
{
  int num_dep;
  float deposito[DIM];
} Tbanco;

TDiagnostico extraer( Tbanco *cuenta, float *dep, int pos )
{
  int i;
  if( pos < 0 || pos >= cuenta->num_dep )
    return FALSO;
  if( cuenta->num_dep == 0 )
    return FALSO;
  *dep = cuenta->deposito[pos];
  for( i = pos; i < cuenta->num_dep - 1; i++ )
    cuenta->deposito[i] = cuenta->deposito[i + 1];
  cuenta->num_dep--;
  return CIERTO;
}

TDiagnostico busqueda_ordenado( Tbanco cuenta, float dep, int *pos )
{
  *pos = 0;
  while( dep < cuenta.deposito[*pos] && (*pos) < cuenta.num_dep )
    (*pos)++;
  if( *pos == cuenta.num_dep )
    return FALSO;
  if( dep == cuenta.deposito[*pos] )
    return CIERTO;
  return FALSO;
}

TDiagnostico insertar( Tbanco *cuenta, float nuevo_dep, int pos )
{
  int i;
  if( pos < 0 || pos > cuenta->num_dep )
    return FALSO;
  if( cuenta->num_dep == DIM – 1 )
    return FALSO;
  for( i = cuenta->num_dep; i > pos; i-- )
    cuenta->deposito[i] = cuenta->deposito[i – 1];
  cuenta->deposito[pos] = nuevo_dep;
  cuenta->num_dep++;
  return CIERTO;
}
```

```
TDiagnostico ordena_ins( Tbanco *cuenta1 )
{
  int i, n, posicion;
  Tbanco cuenta2;
  float dep;
  if( cuenta1->num_dep == 0 )
    return FALSO;
  n = cuenta1->num_dep;
  cuenta2.num_dep = 0;
  for( i = 0; i < n; i++ )
  {
    extraer( cuenta1, &dep, cuenta1->num_dep – 1 );
    busqueda_ordenado( cuenta2, dep, &posicion );
    insertar( &cuenta2, dep, posicion );
  }
  *cuenta1 = cuenta2;
  return CIERTO;
}
```

4.b) *Ordenación por selección*

Forma general de la llamada

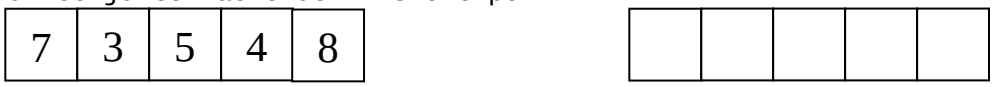
```
ok = ordena_sel( &conjunto_obj );
```

Cabecera de la función

```
int ordena_sel( ... *conjunto )
```

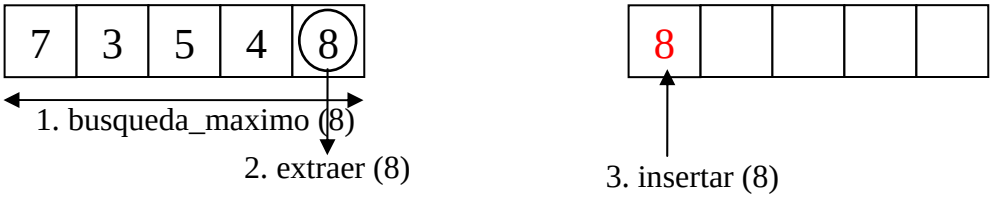
Ejemplo de principio

1. Se crea un conjunto vacío del mismo tipo



2. Bucle de 0 a número de elementos – 1

Vuelta 0



Vuelta 1

Vuelta 2

Vuelta 3

Vuelta 4

Ejemplo

```
#define DIM 10

typedef enum {FALSO, CIERTO} TDiagnostico;
typedef struct
{
  int num_dep;
  float deposito[DIM];
} Tbanco;

TDiagnostico busqueda_maximo( Tbanco cuenta, float *dep, int *pos )
{
  int i;
  if( cuenta.num_dep == 0 )
    return FALSO;
  *pos = 0;
  *max = cuenta.deposito[0];
  for( i = 1; i < cuenta.num_dep; i++ )
    if( *max < cuenta.deposito[i] )
    {
      *max = cuenta.deposito[i];
      *pos = i;
    }
  return CIERTO;
}

TDiagnostico extraer( Tbanco *cuenta, float *dep, int pos )
{
  int i;
  if( pos < 0 || pos >= cuenta->num_dep )
    return FALSO;
  if( cuenta->num_dep == 0 )
    return FALSO;
  *dep = cuenta->deposito[pos];
  for( i = pos; i < cuenta->num_dep - 1; i++ )
    cuenta->deposito[i] = cuenta->deposito[i + 1];
  cuenta->num_dep--;
  return CIERTO;
}

TDiagnostico insertar( Tbanco *cuenta, float nuevo_dep, int pos )
{
  int i;
  if( pos < 0 || pos > cuenta->num_dep )
    return FALSO;
  if( cuenta->num_dep == DIM – 1 )
    return FALSO;
  for( i = cuenta->num_dep; i > pos; i-- )
    cuenta->deposito[i] = cuenta->deposito[i – 1];
```

```
  cuenta->deposito[pos] = nuevo_dep;
  cuenta->num_dep++;
  return CIERTO;
}

TDiagnostico ordena_sel( Tbanco *cuenta1 )
{
  int i, n, posicion, ok;
  float dep;
  Tbanco cuenta2;
  if( cuenta1->num_dep == 0 )
    return FALSO;
  n = cuenta1->num_dep;
  cuenta2.num_dep = 0;
  for( i = 0; i < n; i++ )
  {
    busqueda_maximo( *cuenta1, &dep, &posicion );
    extraer( cuenta1, &dep, posicion );
    insertar( &cuenta2, dep, i );
  }
  *cuenta1 = cuenta2;
  return CIERTO;
}
```