

# ObjAlloc.c

— Version 1.1 —

*This describes the code for manipulating dynamically  
growing arrays*

Eidon (Eidon@tutanota.com)

# Contents

<b>1</b>	<b>Error codes .....</b>	<b>3</b>
<b>2</b>	<b>Defined Macros .....</b>	<b>4</b>
<b>3</b>	<b>Type definitions .....</b>	<b>5</b>
	3.1 The ObjAlloc_t type .....	5
<b>4</b>	<b>The Functions .....</b>	<b>6</b>
	<b>Class Graph .....</b>	<b>11</b>

**Error codes****Names**

```
#define    E_OA_NULL_PTRS
           A null pointer was given

#define    E_OA_COULD_NOT_REALLOC
           Could not reallocate some memory

#define    E_OA_INDEX_INCONSISTENCY
           Indexes where inconsistent

#define    E_OA_INCONSISTENCY
           There is an inconsistency problem

#define    E_OA_MALLOC
           A Malloc failed
```

## Defined Macros

### Names

#define	<b>OA_array</b> (oa)	<i>Return the included data array of the object</i>
#define	<b>OA_cardinality</b> (oa)	<i>Return the number of elements currently in the data structure</i>
#define	<b>OA_size</b> (oa)	<i>Return the size of the elements making up the data structure</i>

3

Type definitions

## Names

3.1	The <code>ObjAlloc_t</code> type .....	5
-----	--	---

3.1

The `ObjAlloc_t` type

This is the allocation object type.

```
typedef struct
{
    size_t  nelem;
    size_t  elem_sz;
    size_t  inc_asize;
    size_t  allocated;
    void    *array;
    Semaphore_t *sem;
} ObjAlloc_t;
```

**nelem** The amount of elements in the data list.

**elem\_sz** The size of the elements.

**inc\_asize** The incrementor size.

**allocated** The currently allocated space.

**\*array** The data list.

**\*sem** The semaphore.

## 4 The Functions

### Names

4.1	ObjAlloc_t*	<b>OA_open</b> (size_t elem_sz, size_t init_size, size_t inc_size) .....	7
4.2	int	<b>OA_close</b> (ObjAlloc_t *oa) .....	7
4.3	void*	<b>OA_insert</b> (ObjAlloc_t *oa, size_t index, void *obj) .....	7
4.4	int	<b>OA_remove</b> (ObjAlloc_t *oa, size_t index) .....	8
	void	<b>OA_wait</b> (ObjAlloc_t *oa)	
	int	<b>OA_testwait</b> (ObjAlloc_t *oa)	
	void	<b>OA_signal</b> (ObjAlloc_t *oa)	
4.5	int	<b>OA_bsearch</b> (ObjAlloc_t *oa, void *q, int (*compare)(void *, void *), int *index) .....	8
4.6	int	<b>OA_lsearch</b> (ObjAlloc_t *oa, void *q, int (*compare)(void *, void *), int *index) .....	9
4.7	char*	<b>OA_error_description</b> () .....	9
4.8	ObjAlloc_t*	<b>OA_import</b> (size_t elem_size, size_t init_size, size_t inc_size, void *array, size_t elem_count) .....	9

This section describes the function used for manipulating the dynamically growing arrays.

## 4.1

```
ObjAlloc_t* OA_open (size_t elem_sz, size_t init_asize,
                      size_t inc_asize)
```

. Create and initialize a new allocation object. This function complies with an object constructor in the OOP vocabular.

**Return Value:** This function returns a pointer to the newly initialized allocation object and sets the global variable `OA_err` to 0 upon success. Upon failure it returns `NULL`.

**Parameters:** `elem_sz` — The size of the elements contained in the allocation object.  
`init_asize` — The initial amount of elements contained in the object.  
`inc_asize` — The incrementor size for the object.

**See Also:** `OA_close`, `OA_array`, `OA_cardinality`, `OA_insert`, `OA_remove`, `OA_error_description`

## 4.2

```
int OA_close (ObjAlloc_t *oa)
```

. Close an existing allocation object. This function complies with an object destructor in the OOP vocabular.

**Return Value:** The function and sets the global variable `OA_err` to 0 and returns this upon success. Upon failure it returns `NULL_PTRS`.

**Parameters:** `*oa` — The object to be closed.

**See Also:** `errorcodes`, `OA_open`, `OA_array`, `OA_cardinality`, `OA_insert`, `OA_remove`, `OA_error_description`

## 4.3

```
void* OA_insert (ObjAlloc_t *oa, size_t index, void *obj)
```

. Add a new entry to the allocation object data field.

**Return Value:** This function returns a pointer to the datastructure included in the object, upon success. Upon failure it returns NULL and puts the OA\_err global variable to an appropriate error value.

**Parameters:** **\*oa** — The allocation object to be changed.  
**index** — The index where an entry should be inserted.  
**\*obj** — The data to be inserted.

**See Also:** OA\_open, OA\_close, OA\_array, OA\_cardinality, OA\_remove, OA\_error\_description

## 4.4

```
int OA_remove (ObjAlloc_t *oa, size_t index)
```

. Remove an entry from the data field of an object.

**Return Value:** This function returns 0 upon success. Upon failure the function returns NULL\_PTRS.

**Parameters:** **\*oa** — The allocation object to be changed.  
**index** — The index where an entry should be removed.

**See Also:** OA\_open, OA\_close, OA\_array, OA\_cardinality, OA\_insert, OA\_error\_description

## 4.5

```
int OA_bsearch (ObjAlloc_t *oa, void *q, int (*compare)(void *, void *), int *index)
```

. This functions allows for searching in the Objects data field.

**Return Value:** The function returns 1 if the entry was found and 0 if it was not found. In case of an error a standard error message is returned.

**Parameters:** **\*oa** — The Object to be searched in.  
**\*q** — The entry to be searched for.  
**\*compare()** — The function to use for comparing the searched and the listed items.  
**\*index** — The index variable pointing to the entry where \*q was found.



**See Also:** errorcodes

#### 4.6

```
int OA_lsearch (ObjAlloc_t *oa, void *q, int (*com-
                pare)(void *, void *), int *index)
```

. This functions allows for searching linearly in the Objects data field.

**Return Value:** The function returns 1 if the entry was found and 0 if it was not found. In case of an error a standard error message is returned.

**Parameters:** **\*oa** — The Object to be searched in.  
**\*q** — The entry to be searched for.  
**\*compare()** — The function to use for comparing the searched and the listed items.  
**\*index** — The index variable pointing to the entry where \*q was found.

**See Also:** errorcodes

#### 4.7

```
char* OA_error_description ()
```

. A description function to get more intelligible information out of error codes. This function makes use of the global variable OA\_err.

**Return Value:** The function returns a string, explaining the currently registered error.

**See Also:** OA\_open, OA\_close, OA\_array, OA\_cardinality, OA\_insert, OA\_remove

#### 4.8

```
ObjAlloc_t* OA_import (size_t elem_size, size_t init_size,
                        size_t inc_size, void *array, size_t
                        elem_count)
```

. This function creates a new object from the given data.

**Return Value:** The function returns the the newly created object or NULL in case of a failure. For fault code return the global value OA\_err is used.

**Parameters:** **elem\_size** — The size of the elements included in the array.  
**init\_size** — The initial number of elements the object's data structure should have.  
**inc\_size** — The incrementor value.  
**\*array** — The array of data to be imported into the object.  
**elem\_count** — The number of elements listed in the array.

**See Also:** error\_codes, OA\_open

# Class Graph