

Cbject docs

Table of Contents

1. Overview	3
1.1. Features	3
1.2. Usage	3
1.3. cbject_Object model	3
2. API	6
2.1. cbject_Object	6
2.1.1. Overview	6
2.1.2. Types	6
cbject_Object	6
cbject_ObjectClass	6
struct cbject_Object	6
struct cbject_ObjectClass	7
2.1.3. Functions	8
cbject_Object_acquire()	8
cbject_Object_alloc()	8
cbject_Object_init()	8
cbject_Object_copy()	9
cbject_Object_equals()	9
cbject_Object_hashCode()	9
cbject_Object_terminate()	10
cbject_Object_release()	10
cbject_Object_dealloc()	10
cbject_Object_isOfClass()	10
cbject_ObjectClass_instance()	11
2.1.4. Macros	11
cbject_ObjectClass_setup()	11
cbject_Object_class()	11
cbject_Object_instanceSize()	12
2.1.5. Tests	12
test_cbject_ObjectClass	12
test_cbject_Object_init	12
test_cbject_Object_equals	12
test_cbject_Object_hashCode	13
test_cbject_Object_isOfClass	13
test_cbject_Object_copy	13
2.2. cbject_Singleton	14

2.2.1. Overview	14
2.2.2. Types	14
cobject_Singleton	14
cobject_SingletonClass	14
struct cobject_Singleton	15
struct cobject_SingletonClass	15
2.2.3. Functions	15
cobject_Singleton_init()	15
cobject_SingletonClass_instance()	15
2.3. cobject_utils	16
2.3.1. Overview	16
2.3.2. Macros	16
cobject_utils_acquire()	16
cobject_utils_alloc()	16
cobject_utils_stackalloc()	17
cobject_utils_hashCode()	17
cobject_utils_equals()	17
cobject_utils_copy()	18
cobject_utils_terminate()	18
cobject_utils_release()	18
cobject_utils_dealloc()	19
cobject_utils_allocPool()	19
cobject_utils_doOnce	19
cobject_utils_invokeMethod()	20
cobject_utils_invokeClassMethod()	20
cobject_utils_invokeSuperMethod()	21
cobject_utils_Array_length()	21
cobject_utils_assertStatic()	21
cobject_utils-Token_concat()	22
cobject_utils-Token_concatIndirect()	22
cobject_utils-Token_stringify()	22
cobject_utils-Token_stringifyIndirect()	22
cobject_utils_VaArgs_getFirst()	23
cobject_utils_VaArgs_getSecond()	23
cobject_utils_VaArgs_getRest()	23
cobject_utils_Pair_getFirst()	23
cobject_utils_Pair_getSecond()	24
2.4. cobject	24
2.4.1. Overview	24

1. Overview

Cbjeect makes it easier to write object oriented code in C.

1.1. Features

- Objects
- Classes
- Inheritance
- Polymorphism

1.2. Usage

Example 1. How to add it to a project

Include the following header file:

```
#include "cbjeect.h"
```

Example 2. How to create an object

```
cbjeect_Object * object = cbjeect_Object_init(cbjeect_Object_alloc(cbjeect_Object));
printf("%d\n", cbjeect_Object_hashCode(object));
cbjeect_utils_dealloc(object);
```

1.3. cbjeect_Object model

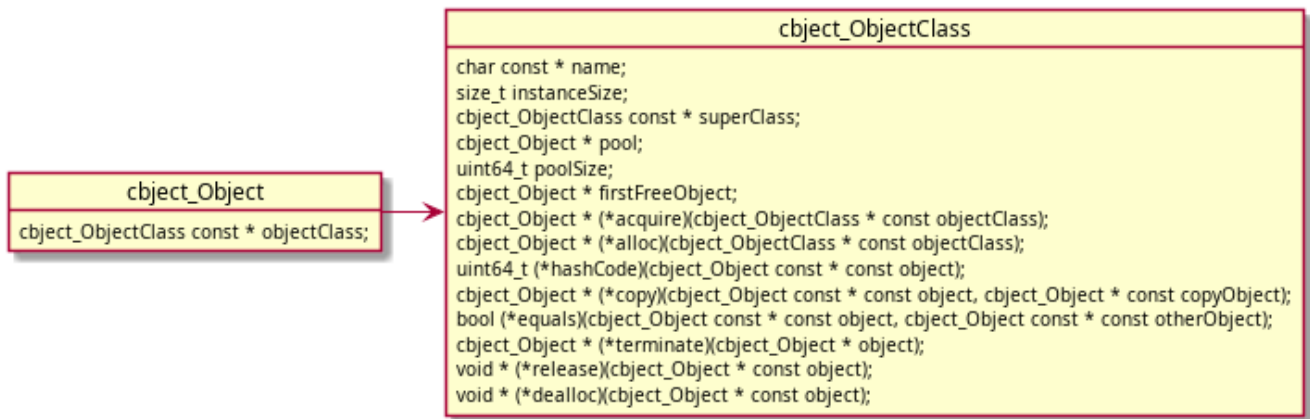


Figure 1. Building blocks

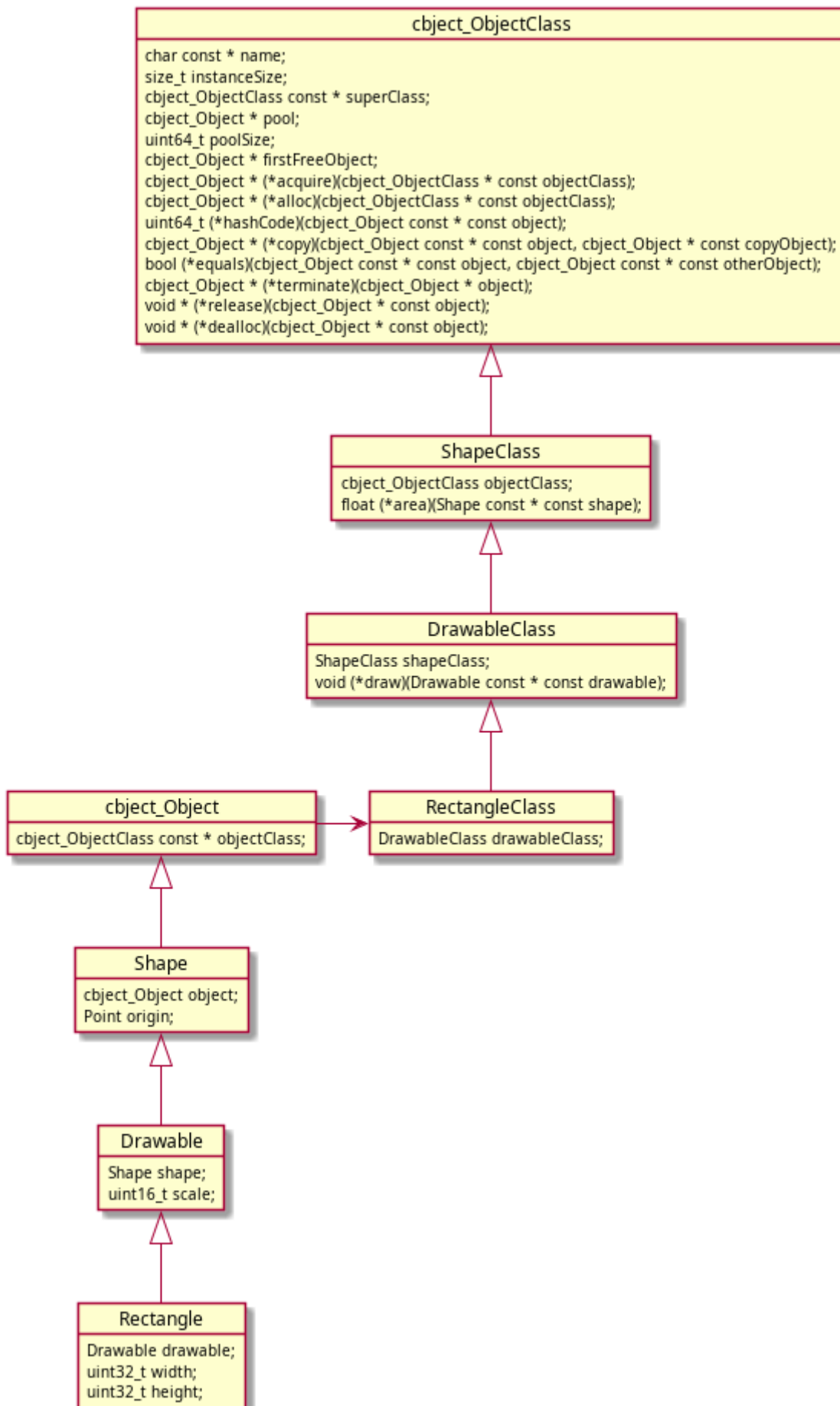


Figure 2. Rectangle class example

2. API

2.1. cbject_Object

2.1.1. Overview

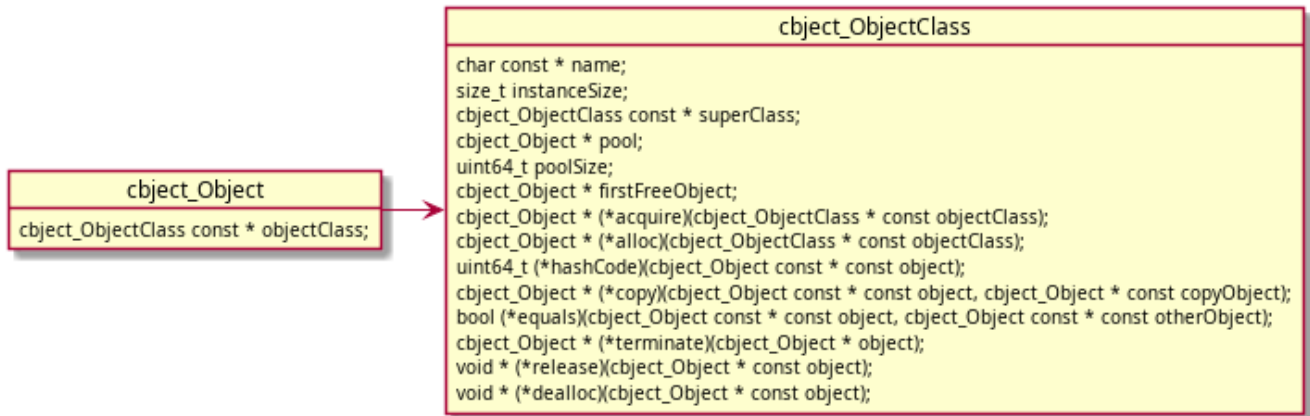


Figure 3. Context diagram

The building block. All objects defined in Cbject need to extend `cbject_Object`.

2.1.2. Types

`cbject_Object`

```
typedef struct cbject_Object cbject_Object;
```

Typedef for struct `cbject_Object`

`cbject_ObjectClass`

```
typedef struct cbject_ObjectClass cbject_ObjectClass;
```

Typedef for struct `cbject_ObjectClass`

`struct cbject_Object`

```
struct cbject_Object {  
    cbject_ObjectClass * objectClass;  
    cbject_Object_UsageStatus usageStatus;  
};
```

Definition of struct `object_Object`

Members

- `objectClass` - `object_ObjectClass` reference
- `usageStatus` - Usage status of object (free/inUse)

struct `object_ObjectClass`

```
struct object_ObjectClass {
    char const * name;
    size_t instanceSize;
    object_ObjectClass const * superClass;
    object_Object * pool;
    uint64_t poolSize;
    object_Object * firstFreeObject;
    object_Object * (*acquire)(object_ObjectClass * const objectClass);
    object_Object * (*alloc)(object_ObjectClass * const objectClass);
    uint64_t (*hashCode)(object_Object const * const object);
    object_Object * (*copy)(object_Object const * const object, object_Object *
const copyObject);
    bool (*equals)(object_Object const * const object, object_Object const * const
otherObject);
    object_Object * (*terminate)(object_Object * object);
    void * (*release)(object_Object * const object);
    void * (*dealloc)(object_Object * const object);
};
```

Definition of struct `object_ObjectClass`

Members

- `name` - Name of the class
- `instanceSize` - Memory size for an instance of the class
- `superClass` - Super class reference
- `pool` - Reference to the object static pool
- `poolSize` - Size of pool (number of objects in pool)
- `firstFreeObject` - Reference to the first free object in the pool
- `acquire` - Acquire method reference
- `alloc` - Alloc method reference
- `hashCode` - Hash code method reference
- `copy` - Copy method reference
- `equals` - Equals method reference
- `terminate` - Terminate method reference

- `release` - Release method reference
- `dealloc` - Dealloc method reference

2.1.3. Functions

`cbject_Object_acquire()`

```
cbject_Object * cbject_Object_acquire(cbject_ObjectClass * const objectClass);
```

Acquires an object from the static pool

Params

- `objectClass` - `cbject_ObjectClass` reference

Return

Reference of the acquired object

`cbject_Object_alloc()`

```
cbject_Object * cbject_Object_alloc(cbject_ObjectClass * const objectClass);
```

Allocates an object in heap memory

Params

- `objectClass` - `cbject_ObjectClass` reference

Return

Reference of the allocated object

`cbject_Object_init()`

```
cbject_Object * cbject_Object_init(cbject_Object * const object);
```

Initializes an object

Params

- `object` - `cbject_Object` reference

Return

Initialized object

cbject_Object_copy()

```
cbject_Object * cbject_Object_copy(cbject_Object const * const object,  
cbject_Object * const copyObject);
```

Copies the object to the provided instance.

Params

- object - cbject_Object reference
- copyObject - Reference of a new object in which to copy the original one

Return

Reference of copyObject

cbject_Object_equals()

```
bool cbject_Object_equals(cbject_Object const * const object, cbject_Object const  
* const otherObject);
```

Compares two objects

Params

- object - cbject_Object reference
- otherObject - Reference for the compared object

Return

- true - If the objects are equal
- false - If the objects are different

cbject_Object_hashCode()

```
uint64_t cbject_Object_hashCode(cbject_Object const * const object);
```

Gets the hash code of the object

Params

- object - cbject_Object reference

Return

The hash code of the object

cbject_Object_terminate()

```
cbject_Object * cbject_Object_terminate(cbject_Object * const object);
```

Terminates an object.

Params

- object - cbject_Object reference

Return

NULL

cbject_Object_release()

```
void * cbject_Object_release(cbject_Object * const object);
```

Releases the object in the static pool

Params

- object - cbject_Object reference

Return

NULL

cbject_Object_dealloc()

```
void * cbject_Object_dealloc(cbject_Object * const object);
```

Deallocates memory for an object

Params

- object - cbject_Object reference

Return

NULL

cbject_Object_isOfClass()

```
bool cbject_Object_isOfClass(cbject_Object const * const object,
```

```
cbject_ObjectClass const * const objectClass);
```

Checks if an object is of a given class

Params

- object - cbject_Object reference
- objectClass - Class reference

Return

- true - If the object is of the provided class
- false - If the object is of a different class

cbject_ObjectClass_instance()

```
cbject_ObjectClass * cbject_ObjectClass_instance(void);
```

Gets cbject_ObjectClass instance

Return

Reference of the class instance

2.1.4. Macros

cbject_ObjectClass_setup()

```
cbject_ObjectClass_setup(klass)
```

Populates the class instance

Remark

cbject_Class must be defined before using this macro

Params

- klass - Class reference

cbject_Object_class()

```
cbject_Object_class(object)
```

Gets the class of an object

Params

- object - cbject_Object reference

Return

Class reference

cbject_Object_instanceSize()

```
cbject_Object_instanceSize(object)
```

Gets the size in memory of an object

Params

- object - cbject_Object reference

Return

The size in memory of the object

2.1.5. Tests

test_cbject_ObjectClass

Test setup of ObjectClass

Steps

1. Get ObjectClass instance
2. Check if object size stored in class is equal to the actual object size
3. Check that the function pointers in the class are initialized

test_cbject_Object_init

Test initialization of cbject_Object

Steps

1. Allocate object on stack an initialize it
2. Check if object class points to cbject_ObjectClass instance

test_cbject_Object_equals

Test equals method

Steps

1. Allocate object on stack and initialize it
2. Check if equals method returns true when comparing object to self
3. Allocate another object on stack and initialize it
4. Check if equals method returns false when comparing the two objects

test_cbject_Object_hashCode

Test hashCode method

Steps

1. Allocate object on stack and initialize it
2. Check if hashCode method returns the address in memory of the object

test_cbject_Object_isOfClass

Test isOfClass method

Preconditions

1. Define a dummy TestClass which extends cbject_ObjectClass

Steps

1. Allocate object on stack and initialize it
2. Check if isOfClass method returns true when checked against cbject_Object
3. Check if isOfClass method returns false when checked against Test

test_cbject_Object_copy

Test copy method

Steps

1. Allocate object on stack and initialize it
2. Allocate another object on stack and copy the first object into it
3. Check if the memory sections occupied by the two objects are equal
4. Allocate another object on heap and copy the first object into it
5. Check if the memory sections occupied by the two objects are equal
6. Deallocate the object from the heap memory

2.2. cbject_Singleton

2.2.1. Overview

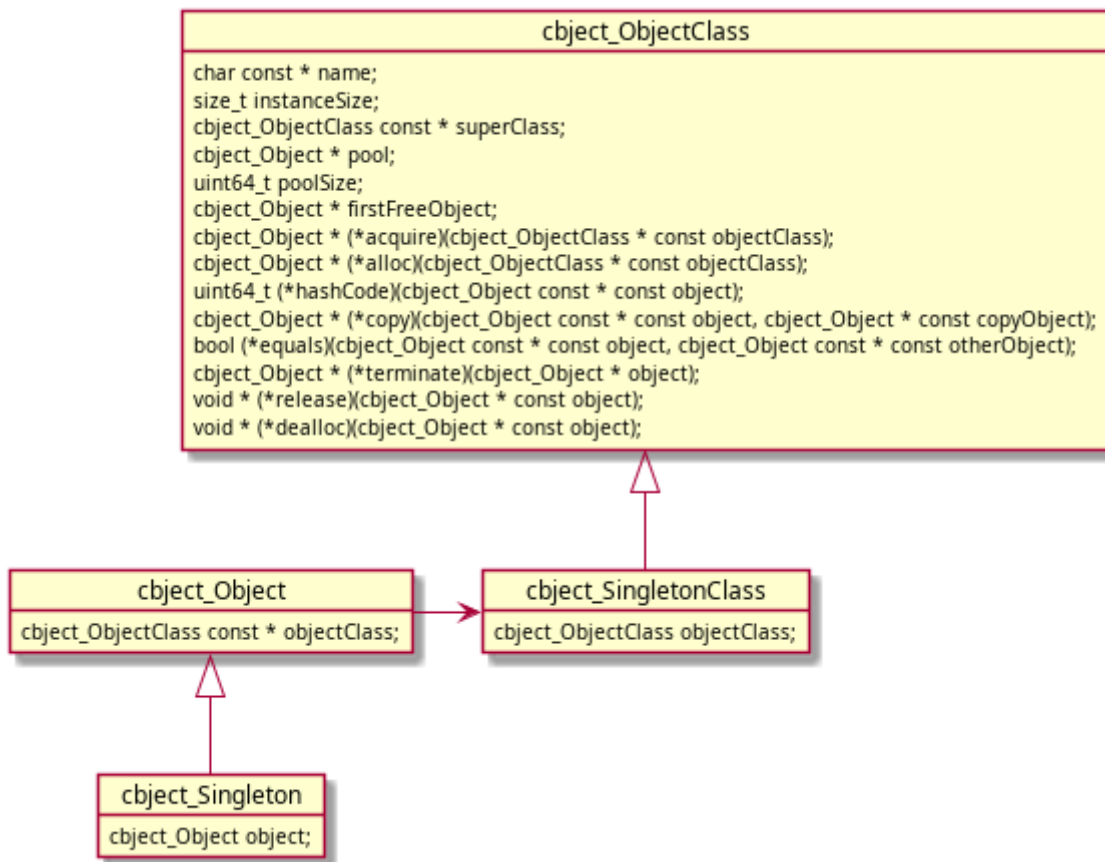


Figure 4. Context diagram

2.2.2. Types

cbject_Singleton

```
typedef struct cbject_Singleton cbject_Singleton;
```

Typedef for struct `cbject_Singleton`

cbject_SingletonClass

```
typedef struct cbject_SingletonClass cbject_SingletonClass;
```

Typedef for struct `cbject_SingletonClass`

struct cbject_Singleton

```
struct cbject_Singleton {  
    cbject_Object object;  
  
};
```

Definition of struct cbject_Singleton

Members

- object - Parent

struct cbject_SingletonClass

```
struct cbject_SingletonClass {  
    cbject_ObjectClass objectClass;  
  
};
```

Definition of struct cbject_SingletonClass

Members

- cbject_ObjectClass - class of parent

2.2.3. Functions

cbject_Singleton_init()

```
cbject_Singleton * cbject_Singleton_init(cbject_Singleton * const singleton);
```

Initializes a singleton

Params

- singleton - cbject_Singleton reference

Return

Initialized singleton

cbject_SingletonClass_instance()

```
object_SingletonClass const * object_SingletonClass_instance(void);
```

Gets object_SingletonClass instance

Return

Reference of the class instance

2.3. cbject_utils

2.3.1. Overview

TODO

2.3.2. Macros

cbject_utils_acquire()

```
cbject_utils_acquire(klass)
```

Acquires an object from the static pool

Remarks

Calls cbject_Object_acquire() and does the necessary casting

Params

- klass - Name of class

Return

Reference of the acquired object

cbject_utils_alloc()

```
cbject_utils_alloc(klass)
```

Allocates an object in heap memory

Remarks

Calls cbject_Object_alloc() and does the necessary casting

Params

- klass - Name of class

Return

Reference of the allocated object

cbject_utils_stackalloc()

```
cbject_utils_stackalloc(klass)
```

Allocates an object on the stack

Params

- klass - Name of class

Return

Reference of the allocated memory

cbject_utils_hashCode()

```
cbject_utils_hashCode(object)
```

Gets the hash code of the object

Remarks

Calls cbject_Object_hashCode() and does the necessary casting

Params

- object - cbject_Object reference

Return

The hash code of the object

cbject_utils_equals()

```
cbject_utils_equals(object, otherObject)
```

Compares two objects

Remarks

Calls cbject_Object_equals() and does the necessary casting

Params

- object - cbject_Object reference

- otherObject - Reference for the compared object

Return

- true - If the objects are equal
- false - If the objects are different

object_utils_copy()

```
object_utils_copy(object, copyObject)
```

Copies the object to the provided instance.

Remarks

Calls `object_Object_copy()` and does the necessary casting

Params

- object - `object_Object` reference
- copyObject - Reference of a new object in which to copy the original one

Return

Reference of copyObject

object_utils_terminate()

```
object_utils_terminate(object)
```

Terminates an object.

Remarks

Calls `object_Object_terminate()` and does the necessary casting

Params

- object - `object_Object` reference

Return

NULL

object_utils_release()

```
object_utils_release(object)
```

Releases the object in the static pool

Remarks

Calls `cbject_Object_release()` and does the necessary casting

Params

- `object` - `cbject_Object` reference

Return

NULL

cbject_utils_dealloc()

```
cbject_utils_dealloc(object)
```

Deallocates memory for an object

Remarks

Calls `cbject_Object_dealloc()` and does the necessary casting

Params

- `object` - `cbject_Object` reference

Return

NULL

cbject_utils_allocPool()

```
cbject_utils_allocPool(poolSize)
```

Allocates a static pool

Remarks

`cbject_Class` must be defined before using this macro

Params

- `poolSize` - Size of pool (number of objects in pool)

cbject_utils_doOnce

```
cbject_utils_doOnce
```

Runs a block of code only once

Usage

```
cbject_utils_doOnce {  
    functionCall();  
    anotherFunctionCall();  
}
```

Remark

Not thread safe

cbject_utils_invokeMethod()

```
cbject_utils_invokeMethod(method, ...)
```

Polymorphic call of an object method

Remarks

cbject_Class must be defined before using this macro

Params

- method - Name of the method
- ...
 - object - cbject_Object reference
 - ... - Method params

Return

Depends on the called method

cbject_utils_invokeClassMethod()

```
cbject_utils_invokeClassMethod(method, ...)
```

Polymorphic call of a class method

Remarks

cbject_Class must be defined before using this macro

Params

- method - Name of the method
- ... - Method params

Return

Depends on the called method

cbject_utils_invokeSuperMethod()

```
cbject_utils_invokeSuperMethod(type, method, ...)
```

Polymorphic call of a super method (object or class)

Remarks

cbject_Class must be defined before using this macro

Params

- klass - Name of the class
- method - Name of the method
- ...
 - object - cbject_Object reference (optional - in case of object method)
 - ... - Method params

Return

Depends on the called method

cbject_utils_Array_length()

```
cbject_utils_Array_length(array)
```

Gets length of an array

Params

- array - Array for which to get the length

cbject_utils_assertStatic()

```
cbject_utils_assertStatic(expression, identifier)
```

Compile time assert

Params

- expression - Expression to assert

- identifier - An identifier to describe the assertion

cbject_utils-Token_concat()

```
cbject_utils-Token_concat(token, otherToken)
```

Concatenates otherToken after the provided token

Params

- token - Token
- otherToken - Token to add after the provided token

cbject_utils-Token_concatIndirect()

```
cbject_utils-Token_concatIndirect(token, otherToken)
```

Concatenates otherToken after the provided token indirectly

Params

- token - Token
- otherToken - Token to add after the provided token

cbject_utils-Token_stringify()

```
cbject_utils-Token_stringify(token)
```

Stringifies the provided token

Params

- token - Token

cbject_utils-Token_stringifyIndirect()

```
cbject_utils-Token_stringifyIndirect(token)
```

Stringifies the provided token indirectly

Params

- token - Token

cbject_utils_VaArgs_getFirst()

```
cbject_utils_VaArgs_getFirst(...)
```

Gets first argument from *VA_ARGS*

Params

- ... - *VA_ARGS*

cbject_utils_VaArgs_getSecond()

```
cbject_utils_VaArgs_getSecond(...)
```

Gets second argument from *VA_ARGS*

Params

- ... - *VA_ARGS*

cbject_utils_VaArgs_getRest()

```
cbject_utils_VaArgs_getRest(...)
```

Gets list of arguments from *VA_ARGS* except the first

Remark

- Comma is added before the list
- Supports max 99 arguments

Params

- ... - *VA_ARGS*

cbject_utils_Pair_getFirst()

```
cbject_utils_Pair_getFirst(pair)
```

Gets first element from pair

Params

- pair - (first, second)

cbject_utils_Pair_getSecond()

```
cbject_utils_Pair_getSecond(pair)
```

Gets second element from pair

Params

- pair - (first, second)

2.4. cbject

2.4.1. Overview

todo