
TASK 1 - Understand Object-Oriented Programming in C

---

animal.h

```
/* animal.h */
#ifndef ANI_INC
#define ANI_INC

struct animal;    /*Creating a struct animal class*/
typedef void speakfunc_t(const struct animal *a);

typedef struct animal
{
        speakfunc_t *vspeak;    /*The class consists of a virtual method*/
        float weight;
}
animal_t;

struct animal *animal_new(float weight);    /*Creating a method new*/
struct animal *animal_create(float weight);    /*Creating method create*/

void animal_del(struct animal *a);
void animal_cleanup(struct animal **a_ref);

void animal_speak(const struct animal *a);

float animal_get_weight(const struct animal *a);

void animal_set_weight(struct animal *a, float weight);

#endif
```

animal.c

```
/* animal.c */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "animal.h"

static void _animal_speak(const struct animal *const a)
{
	(void)a;
	puts("grrrr.");
}

struct animal *animal_new(const float weight)
{
	struct animal *a = calloc(1, sizeof *a);
	if( a != NULL )
		*a = animal_create(weight);
	return a;
}

struct animal animal_create(const float weight)
{
	struct animal *a;
	a->weight = weight;
	a->vspeak = &_animal_speak;
	return *a;
}

void animal_del(struct animal *const a)
{
	if( a==NULL )
		return;
	else {
		memset(a, 0, sizeof *a);
		free(a);
	}
```

```
}

void animal_cleanup(struct animal **a_ref)
{
	if( a_ref==NULL || *a_ref==NULL )
		return;
	else {
		animal_del(*a_ref), *a_ref = NULL;
	}
}

void animal_speak(const struct animal *const a)
{
	if( a==NULL )
		return;
	else (*a->vspeak)(a);
}

float animal_get_weight(const struct animal *const a)
{
	return a != NULL ? a->weight : 0.f;
}

void animal_set_weight(struct animal *const a, const float weight)
{
	if( a==NULL )
		return;
	else {
		a->weight = weight;
	}
}
```