# Code Conventions

Thomas Versteeg  Job Talle

April 6, 2014

## 1 Conventions

### 1.1 Example Code

*Header:*

```
1   #pragma once
2
3   #include <stdlib.h>
4   #include "dingen.h"
5
6   /* Does random shit */
7   int function(char *destination, const char *source, int value);
```

*Body:*

```
1   #include "header.h"
2
3   typedef struct{
4       int variable;
5       char *string;
6   } Struct;
7
8   typedef struct _StructStruct{
9       struct _StructStruct *next;
10  } StructStruct;
11
12  typedef enum{
13      A = 1,
14      BETA = 2
15  } Enum;
16
17  int function(char *destination, const char *source, int value)
18  {
19      int i, j, output;
20      /* Random loop */
21      for(i = 0; i < strlen(source); i++){
22          for(j = 0; j < value; j++){
23              destination[i] += j;
24          }
25      }
26      return output;
27  }
```

## 1.2 Naming

CamelCase is applied for naming all variables and structures. All variables and functions should start with a lowercase letter. Some exceptions exist.

*Structures* and *enumerations* are the only entities starting with an uppercase letter.

Locally used structures and variables can start with an underscore to avoid naming conflicts, or to clarify their local nature.

## 1.3 spaces

Spaces are used around operators, as seen in the example code. They are not used before the {.

## 1.4 OOP

Structs are initialized with a function named newStruct (where Struct is the structure name), and freed using freeStruct. Function pointers are assigned in the newStruct function.