

Ogród zoologiczny

Generated by Doxygen 1.8.14

Contents

1	Ogród zoologiczny	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	Animal Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Field Documentation	8
4.1.2.1	age	8
4.1.2.2	comment	8
4.1.2.3	id	8
4.1.2.4	name	8
4.1.2.5	species	9
4.2	AnimalLinkedList Struct Reference	9
4.2.1	Detailed Description	10
4.2.2	Field Documentation	10
4.2.2.1	firstItem	10
4.2.2.2	lastItem	10
4.2.2.3	size	10
4.3	AnimalLinkedListItem Struct Reference	11
4.3.1	Detailed Description	11

4.3.2	Field Documentation	11
4.3.2.1	next	11
4.3.2.2	prev	12
4.3.2.3	value	12
4.4	HttpResponse Struct Reference	12
4.4.1	Detailed Description	13
4.4.2	Field Documentation	13
4.4.2.1	code	13
4.4.2.2	data	13
4.4.2.3	size	13
4.5	SortCallbackData Struct Reference	14
4.5.1	Detailed Description	14
4.5.2	Field Documentation	14
4.5.2.1	col	15
4.5.2.2	list	15
4.5.2.3	table	15
5	File Documentation	17
5.1	CMakeFiles/3.13.2/CompilerIdC/CMakeCCompilerId.c File Reference	17
5.1.1	Macro Definition Documentation	17
5.1.1.1	ARCHITECTURE_ID	17
5.1.1.2	C_DIALECT	18
5.1.1.3	COMPILER_ID	18
5.1.1.4	DEC	18
5.1.1.5	HEX	18
5.1.1.6	PLATFORM_ID	19
5.1.1.7	STRINGIFY	19
5.1.1.8	STRINGIFY_HELPER	19
5.1.2	Function Documentation	19
5.1.2.1	main()	19
5.1.3	Variable Documentation	20

5.1.3.1	info_arch	20
5.1.3.2	info_compiler	20
5.1.3.3	info_language_dialect_default	20
5.1.3.4	info_platform	20
5.2	CMakeFiles/feature_tests.c File Reference	21
5.2.1	Function Documentation	21
5.2.1.1	main()	21
5.2.2	Variable Documentation	21
5.2.2.1	features	21
5.3	include/data/animal.h File Reference	22
5.3.1	Typedef Documentation	22
5.3.1.1	Animal	22
5.3.1.2	AnimalLinkedList	23
5.3.1.3	AnimalLinkedListItem	23
5.3.2	Function Documentation	23
5.3.2.1	animal_linked_list_add_item()	23
5.3.2.2	animal_linked_list_get_item()	24
5.3.2.3	animal_linked_list_new()	24
5.3.2.4	animal_linked_list_sort()	25
5.3.2.5	animal_new()	26
5.4	include/main_window.h File Reference	26
5.4.1	Typedef Documentation	27
5.4.1.1	COLUMN	27
5.4.1.2	SortCallbackData	28
5.4.2	Enumeration Type Documentation	28
5.4.2.1	COLUMN	28
5.4.3	Function Documentation	28
5.4.3.1	add_control_buttons()	28
5.4.3.2	add_table_headers()	29
5.4.3.3	callback_sort_click()	31

5.4.3.4	fill_table()	32
5.4.3.5	main_window_new()	32
5.5	include/remove_item_window.h File Reference	34
5.5.1	Function Documentation	35
5.5.1.1	remove_item_window_new()	35
5.6	include/services/http.h File Reference	35
5.6.1	Typedef Documentation	36
5.6.1.1	HttpResponse	36
5.6.2	Function Documentation	36
5.6.2.1	http_get()	36
5.6.2.2	write_function()	37
5.7	README.md File Reference	38
5.8	src/data/animal.c File Reference	38
5.8.1	Function Documentation	38
5.8.1.1	animal_linked_list_add_item()	39
5.8.1.2	animal_linked_list_item_new()	40
5.8.1.3	animal_linked_list_new()	40
5.8.1.4	animal_linked_list_sort()	41
5.8.1.5	animal_new()	42
5.9	src/main.c File Reference	42
5.9.1	Function Documentation	43
5.9.1.1	activate()	43
5.9.1.2	main()	44
5.10	src/main_window.c File Reference	44
5.10.1	Function Documentation	45
5.10.1.1	add_animal_to_table()	45
5.10.1.2	add_control_buttons()	46
5.10.1.3	add_table_headers()	47
5.10.1.4	callback_remove_animal()	49
5.10.1.5	callback_sort_click()	49

5.10.1.6	cmp()	50
5.10.1.7	fill_table()	51
5.10.1.8	main_window_new()	52
5.10.1.9	sort_callback_data_new()	53
5.10.2	Variable Documentation	54
5.10.2.1	sort_asc	54
5.10.2.2	sort_by	54
5.11	src/remove_item_window.c File Reference	55
5.11.1	Function Documentation	55
5.11.1.1	remove_item_window_new()	55
5.12	src/services/http.c File Reference	56
5.12.1	Function Documentation	56
5.12.1.1	http_get()	56
5.12.1.2	http_response_new()	57
5.12.1.3	write_function()	58
Index		59

Chapter 1

Ogród zoologiczny

Disclaimer: THIS PROJECT DOESN'T MAKE ANY SENSE AT ALL

A simple GTK app for an university project

Warning: lots of duplicated code in order to meet the 1000-line minimum

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Animal	7
AnimalLinkedList	9
AnimalLinkedListItem	11
HttpResponse	12
SortCallbackData	14

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

CMakeFiles/ feature_tests.c	21
CMakeFiles/3.13.2/CompilerIdC/ CMakeCCompilerId.c	17
include/ main_window.h	26
include/ remove_item_window.h	34
include/data/ animal.h	22
include/services/ http.h	35
src/ main.c	42
src/ main_window.c	44
src/ remove_item_window.c	55
src/data/ animal.c	38
src/services/ http.c	56

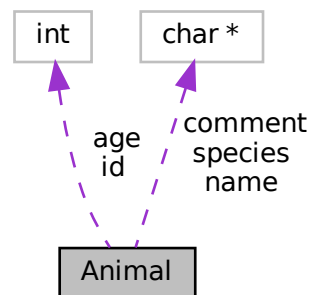
Chapter 4

Data Structure Documentation

4.1 Animal Struct Reference

```
#include <animal.h>
```

Collaboration diagram for Animal:



Data Fields

- int [id](#)
- char * [name](#)
- char * [species](#)
- int [age](#)
- char * [comment](#)

4.1.1 Detailed Description

Definition at line 20 of file animal.h.

4.1.2 Field Documentation

4.1.2.1 age

```
int Animal::age
```

Definition at line 24 of file animal.h.

Referenced by `add_animal_to_table()`, `animal_new()`, and `cmp()`.

4.1.2.2 comment

```
char* Animal::comment
```

Definition at line 25 of file animal.h.

Referenced by `add_animal_to_table()`, `animal_new()`, and `cmp()`.

4.1.2.3 id

```
int Animal::id
```

Definition at line 21 of file animal.h.

Referenced by `add_animal_to_table()`, `animal_new()`, and `cmp()`.

4.1.2.4 name

```
char* Animal::name
```

Definition at line 22 of file animal.h.

Referenced by `add_animal_to_table()`, `animal_new()`, and `cmp()`.

4.1.2.5 species

```
char* Animal::species
```

Definition at line 23 of file animal.h.

Referenced by `add_animal_to_table()`, `animal_new()`, and `cmp()`.

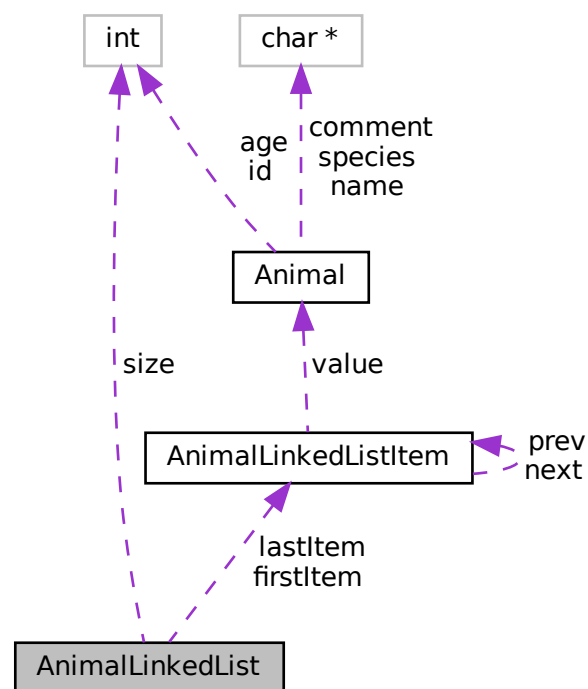
The documentation for this struct was generated from the following file:

- `include/data/animal.h`

4.2 AnimalLinkedList Struct Reference

```
#include <animal.h>
```

Collaboration diagram for AnimalLinkedList:



Data Fields

- `int` `size`
- `struct AnimalLinkedListItem *` `firstItem`
- `struct AnimalLinkedListItem *` `lastItem`

4.2.1 Detailed Description

Definition at line 31 of file animal.h.

4.2.2 Field Documentation

4.2.2.1 firstItem

```
struct AnimalLinkedListItem* AnimalLinkedList::firstItem
```

Definition at line 33 of file animal.h.

Referenced by `animal_linked_list_add_item()`, `animal_linked_list_sort()`, and `fill_table()`.

4.2.2.2 lastItem

```
struct AnimalLinkedListItem* AnimalLinkedList::lastItem
```

Definition at line 34 of file animal.h.

Referenced by `animal_linked_list_add_item()`, and `animal_linked_list_sort()`.

4.2.2.3 size

```
int AnimalLinkedList::size
```

Definition at line 32 of file animal.h.

Referenced by `animal_linked_list_add_item()`, `animal_linked_list_new()`, `animal_linked_list_sort()`, and `fill_table()`.

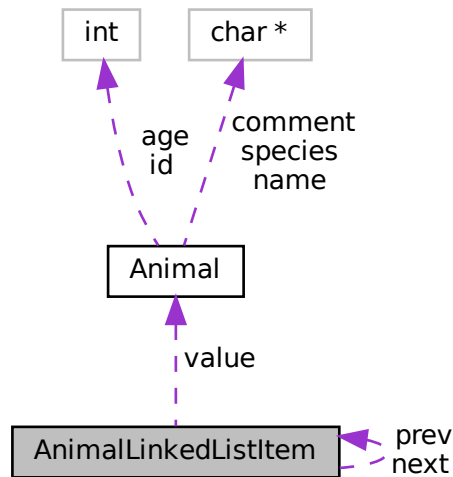
The documentation for this struct was generated from the following file:

- `include/data/animal.h`

4.3 AnimalLinkedListItem Struct Reference

```
#include <animal.h>
```

Collaboration diagram for AnimalLinkedListItem:



Data Fields

- [Animal](#) * [value](#)
- struct [AnimalLinkedListItem](#) * [prev](#)
- struct [AnimalLinkedListItem](#) * [next](#)

4.3.1 Detailed Description

Definition at line 40 of file `animal.h`.

4.3.2 Field Documentation

4.3.2.1 next

```
struct AnimalLinkedListItem* AnimalLinkedListItem::next
```

Definition at line 43 of file `animal.h`.

Referenced by `animal_linked_list_add_item()`, `animal_linked_list_sort()`, and `fill_table()`.

4.3.2.2 prev

```
struct AnimalLinkedListItem* AnimalLinkedListItem::prev
```

Definition at line 42 of file animal.h.

Referenced by `animal_linked_list_add_item()`, and `animal_linked_list_sort()`.

4.3.2.3 value

```
Animal* AnimalLinkedListItem::value
```

Definition at line 41 of file animal.h.

Referenced by `animal_linked_list_item_new()`, `animal_linked_list_sort()`, and `fill_table()`.

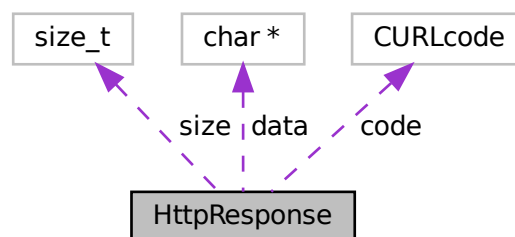
The documentation for this struct was generated from the following file:

- `include/data/animal.h`

4.4 HttpResponse Struct Reference

```
#include <http.h>
```

Collaboration diagram for HttpResponse:



Data Fields

- `char *` [data](#)
- `size_t` [size](#)
- `CURLcode` [code](#)

4.4.1 Detailed Description

Definition at line 8 of file http.h.

4.4.2 Field Documentation

4.4.2.1 code

```
CURLcode HttpResponse::code
```

Definition at line 11 of file http.h.

Referenced by `http_get()`.

4.4.2.2 data

```
char* HttpResponse::data
```

Definition at line 9 of file http.h.

Referenced by `http_get()`, `http_response_new()`, and `write_function()`.

4.4.2.3 size

```
size_t HttpResponse::size
```

Definition at line 10 of file http.h.

Referenced by `http_response_new()`, and `write_function()`.

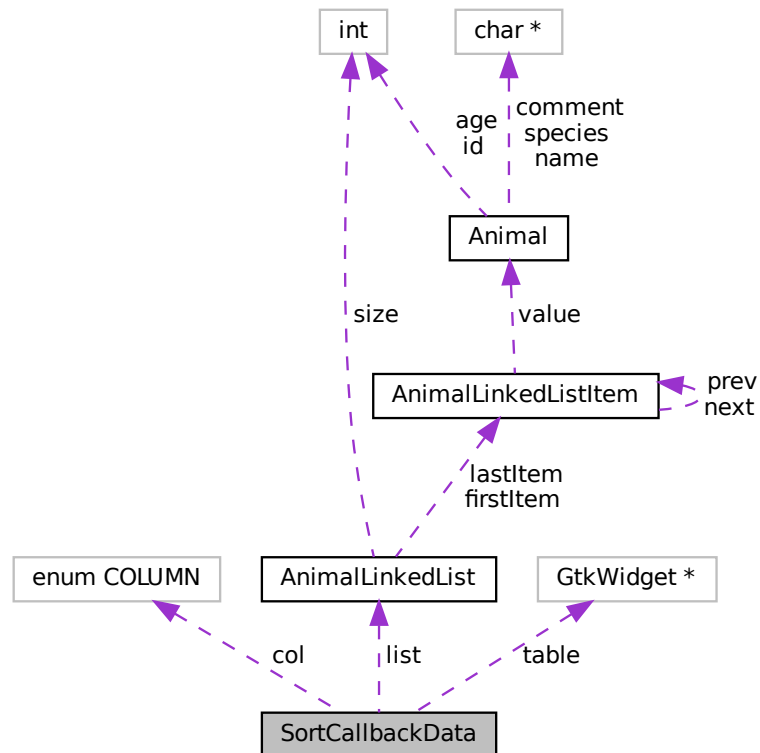
The documentation for this struct was generated from the following file:

- `include/services/http.h`

4.5 SortCallbackData Struct Reference

```
#include <main_window.h>
```

Collaboration diagram for SortCallbackData:



Data Fields

- enum `COLUMN` `col`
- `AnimalLinkedList *` `list`
- `GtkWidget *` `table`

4.5.1 Detailed Description

Definition at line 31 of file `main_window.h`.

4.5.2 Field Documentation

4.5.2.1 col

```
enum COLUMN SortCallbackData::col
```

Definition at line 32 of file `main_window.h`.

Referenced by `callback_sort_click()`, and `sort_callback_data_new()`.

4.5.2.2 list

```
AnimalLinkedList* SortCallbackData::list
```

Definition at line 33 of file `main_window.h`.

Referenced by `callback_sort_click()`, and `sort_callback_data_new()`.

4.5.2.3 table

```
GtkWidget* SortCallbackData::table
```

Definition at line 34 of file `main_window.h`.

Referenced by `callback_sort_click()`, and `sort_callback_data_new()`.

The documentation for this struct was generated from the following file:

- `include/main_window.h`

Chapter 5

File Documentation

5.1 CMakeFiles/3.13.2/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define C_DIALECT`

Functions

- `int main (int argc, char *argv[])`

Variables

- `char const * info_compiler = "INFO" ":" "compiler[" " " "]"`
- `char const * info_platform = "INFO" ":" "platform[" " "]"`
- `char const * info_arch = "INFO" ":" "arch[" " "]"`
- `const char * info_language_dialect_default`

5.1.1 Macro Definition Documentation

5.1.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 492 of file CMakeCCompilerId.c.

5.1.1.2 C_DIALECT

```
#define C_DIALECT
```

Definition at line 577 of file CMakeCCompilerId.c.

5.1.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 312 of file CMakeCCompilerId.c.

5.1.1.4 DEC

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \
('0' + ((n) / 1000000) % 10), \
('0' + ((n) / 100000) % 10), \
('0' + ((n) / 10000) % 10), \
('0' + ((n) / 1000) % 10), \
('0' + ((n) / 100) % 10), \
('0' + ((n) / 10) % 10), \
('0' + ((n) % 10))
```

Definition at line 496 of file CMakeCCompilerId.c.

5.1.1.5 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n) >> 28 & 0xF)), \
('0' + ((n) >> 24 & 0xF)), \
('0' + ((n) >> 20 & 0xF)), \
('0' + ((n) >> 16 & 0xF)), \
('0' + ((n) >> 12 & 0xF)), \
('0' + ((n) >> 8 & 0xF)), \
('0' + ((n) >> 4 & 0xF)), \
('0' + ((n) & 0xF))
```

Definition at line 507 of file CMakeCCompilerId.c.

5.1.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 429 of file CMakeCCompilerId.c.

5.1.1.7 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

Definition at line 333 of file CMakeCCompilerId.c.

5.1.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

Definition at line 332 of file CMakeCCompilerId.c.

5.1.2 Function Documentation

5.1.2.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

Definition at line 597 of file CMakeCCompilerId.c.

References `info_arch`, `info_compiler`, `info_language_dialect_default`, and `info_platform`.

```
599 {  
600     int require = 0;  
601     require += info_compiler[argc];  
602     require += info_platform[argc];  
603     require += info_arch[argc];  
604     #ifdef COMPILER_VERSION_MAJOR  
605     require += info_version[argc];  
606     #endif  
607     #ifdef COMPILER_VERSION_INTERNAL  
608     require += info_version_internal[argc];  
609     #endif  
610     #ifdef SIMULATE_ID  
611     require += info_simulate[argc];  
612     #endif  
613     #ifdef SIMULATE_VERSION_MAJOR  
614     require += info_simulate_version[argc];  
615     #endif  
616     #if defined(__CRAYXE) || defined(__CRAYXC)  
617     require += info_cray[argc];  
618     #endif  
619     require += info_language_dialect_default[argc];  
620     (void)argv;  
621     return require;  
622 }
```

5.1.3 Variable Documentation

5.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" " "]"
```

Definition at line 567 of file CMakeCCompilerId.c.

Referenced by main().

5.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" " " "]"
```

Definition at line 319 of file CMakeCCompilerId.c.

Referenced by main().

5.1.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
=  
"INFO" ":" "dialect_default[" " "]"
```

Definition at line 586 of file CMakeCCompilerId.c.

Referenced by main().

5.1.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" " "]"
```

Definition at line 566 of file CMakeCCompilerId.c.

Referenced by main().

5.2 CMakeFiles/feature_tests.c File Reference

Functions

- int `main` (int argc, char **argv)

Variables

- const char `features` []

5.2.1 Function Documentation

5.2.1.1 `main()`

```
int main (  
    int argc,  
    char ** argv )
```

Definition at line 34 of file `feature_tests.c`.

References `features`.

```
34 { (void)argv; return features[argc]; }
```

5.2.2 Variable Documentation

5.2.2.1 `features`

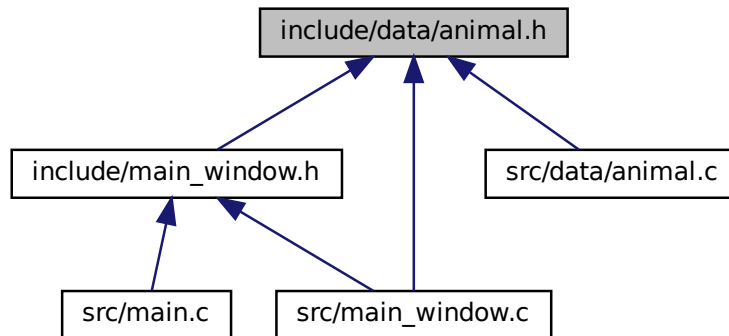
```
const char features[]
```

Definition at line 2 of file `feature_tests.c`.

Referenced by `main()`.

5.3 include/data/animal.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct [Animal](#)
- struct [AnimalLinkedList](#)
- struct [AnimalLinkedListItem](#)

Typedefs

- typedef struct [Animal](#) [Animal](#)
- typedef struct [AnimalLinkedList](#) [AnimalLinkedList](#)
- typedef struct [AnimalLinkedListItem](#) [AnimalLinkedListItem](#)

Functions

- [Animal](#) * [animal_new](#) (int id, char *name, char *species, int age, char *comment)
- void [animal_linked_list_sort](#) ([AnimalLinkedList](#) *list, int(*cmp)([Animal](#) *a, [Animal](#) *b))
- [AnimalLinkedList](#) * [animal_linked_list_new](#) ()
- void [animal_linked_list_add_item](#) ([AnimalLinkedList](#) *list, [Animal](#) *value)
- [Animal](#) [animal_linked_list_get_item](#) (int index)

5.3.1 Typedef Documentation

5.3.1.1 Animal

```
typedef struct Animal Animal
```

5.3.1.2 AnimalLinkedList

```
typedef struct AnimalLinkedList AnimalLinkedList
```

5.3.1.3 AnimalLinkedListItem

```
typedef struct AnimalLinkedListItem AnimalLinkedListItem
```

5.3.2 Function Documentation

5.3.2.1 animal_linked_list_add_item()

```
void animal_linked_list_add_item (
    AnimalLinkedList * list,
    Animal * value )
```

Definition at line 93 of file animal.c.

References `animal_linked_list_item_new()`, `AnimalLinkedList::firstItem`, `AnimalLinkedList::lastItem`, `AnimalLinkedListItem::next`, `AnimalLinkedListItem::prev`, and `AnimalLinkedList::size`.

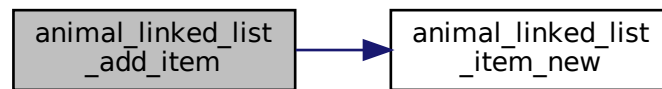
Referenced by `main_window_new()`.

```

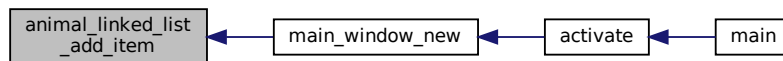
93                                     {
94
95     AnimalLinkedListItem* cur = list->firstItem;
96     AnimalLinkedListItem* toAdd =
animal_linked_list_item_new(value);
97
98     if(list->size == 0){
99         list->firstItem = toAdd;
100         list->lastItem = toAdd;
101         list->size ++;
102         return;
103     }
104     int i;
105     for(i=0; i<list->size-1; ++i){
106         cur = cur->next;
107     }
108
109     cur->next = toAdd;
110     list->lastItem = toAdd;
111     toAdd->prev = cur;
112
113     list->size ++;
114 };

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.2 animal_linked_list_get_item()

```

Animal animal_linked_list_get_item (
    int index )
  
```

5.3.2.3 animal_linked_list_new()

```

AnimalLinkedList* animal_linked_list_new ( )
  
```

Definition at line 48 of file `animal.c`.

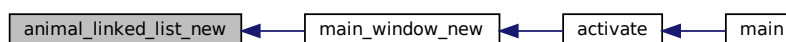
References `AnimalLinkedList::size`.

Referenced by `main_window_new()`.

```

48     {
49         AnimalLinkedList* list = malloc(sizeof(AnimalLinkedList));
50         list->size = 0;
51         return list;
52     }
  
```

Here is the caller graph for this function:



5.3.2.4 animal_linked_list_sort()

```
void animal_linked_list_sort (
    AnimalLinkedList * list,
    int (*) (Animal *a, Animal *b) cmp )
```

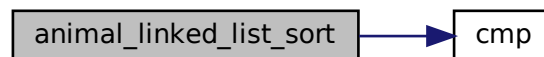
Definition at line 55 of file animal.c.

References `cmp()`, `AnimalLinkedList::firstItem`, `AnimalLinkedList::lastItem`, `AnimalLinkedListItem::next`, `AnimalLinkedListItem::prev`, `AnimalLinkedList::size`, and `AnimalLinkedListItem::value`.

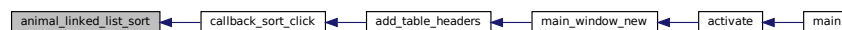
Referenced by `callback_sort_click()`.

```
57 {
58     int changed = 1;
59     while(changed){
60         changed = 0;
61         AnimalLinkedListItem* cur = list->firstItem;
62         for(int i=0; i<list->size-1; ++i){
63             if((*cmp)(cur->value, cur->next->value)){
64                 changed = 1;
65                 int has_prev = list->firstItem != cur;
66                 int has_next = list->lastItem != cur->next;
67                 AnimalLinkedListItem* node_0 = has_prev ? cur->
prev : NULL;
68                 AnimalLinkedListItem* node_1 = cur;
69                 AnimalLinkedListItem* node_2 = cur->next;
70                 AnimalLinkedListItem* node_3 = has_next ? cur->
next->next : NULL;
71                 node_1->next = node_2->next;
72                 node_1->prev = node_2;
73                 node_2->next = node_1;
74                 node_2->prev = node_0;
75                 if(has_prev){
76                     node_0->next = node_2;
77                 } else {
78                     list->firstItem = node_2;
79                 }
80                 if(has_next){
81                     node_3->prev = node_1;
82                 } else {
83                     list->lastItem = node_1;
84                 }
85                 cur = node_2;
86             }
87             cur = cur->next;
88         }
89     }
90 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.5 animal_new()

```
Animal* animal_new (
    int id,
    char * name,
    char * species,
    int age,
    char * comment )
```

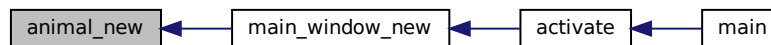
Definition at line 33 of file animal.c.

References `Animal::age`, `Animal::comment`, `Animal::id`, `Animal::name`, and `Animal::species`.

Referenced by `main_window_new()`.

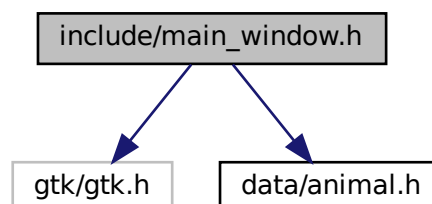
```
33                                     {
34     Animal* animal = malloc(sizeof(Animal));
35     animal->id = id;
36     animal->name = malloc(strlen(name)+1);
37     strcpy(animal->name, name);
38     animal->species = malloc(strlen(species)+1);
39     strcpy(animal->species, species);
40
41     animal->age = age;
42     animal->comment = malloc(strlen(comment)+1);
43     strcpy(animal->comment, comment);
44     return animal;
45 }
```

Here is the caller graph for this function:

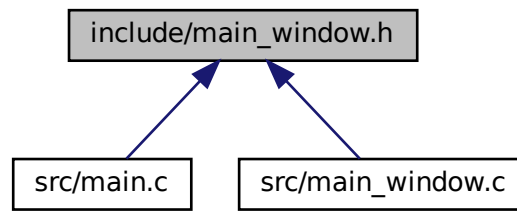


5.4 include/main_window.h File Reference

```
#include <gtk/gtk.h>
#include "data/animal.h"
Include dependency graph for main_window.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [SortCallbackData](#)

Typedefs

- typedef enum [COLUMN](#) [COLUMN](#)
- typedef struct [SortCallbackData](#) [SortCallbackData](#)

Enumerations

- enum [COLUMN](#) {
[ID](#), [NAME](#), [SPECIES](#), [AGE](#),
[COMMENT](#) }

Functions

- GtkWidget * [main_window_new](#) (GtkApplication *app)
Initialize the main window.
- void [callback_sort_click](#) (GtkWidget *widget, gpointer callback_data)
- void [add_table_headers](#) (GtkWidget *mainContainer, GtkWidget *table, [AnimalLinkedList](#) *animals)
Add table header buttons.
- void [add_control_buttons](#) (GtkApplication *app, GtkWidget *mainContainer)
- void [fill_table](#) (GtkWidget *table, [AnimalLinkedList](#) *animals)

5.4.1 Typedef Documentation

5.4.1.1 COLUMN

```
typedef enum COLUMN COLUMN
```

5.4.1.2 SortCallbackData

```
typedef struct SortCallbackData SortCallbackData
```

5.4.2 Enumeration Type Documentation

5.4.2.1 COLUMN

```
enum COLUMN
```

Enumerator

ID	
NAME	
SPECIES	
AGE	
COMMENT	

Definition at line 24 of file main_window.h.

```
24             {ID,  
25             NAME,  
26             SPECIES,  
27             AGE,  
28             COMMENT}
```

5.4.3 Function Documentation

5.4.3.1 add_control_buttons()

```
void add_control_buttons (  
    GtkApplication * app,  
    GtkWidget * mainContainer )
```

Add edit and delete buttons below the table

Parameters

<i>mainContainer</i>	
----------------------	--

Definition at line 97 of file main_window.c.

References `callback_remove_animal()`.

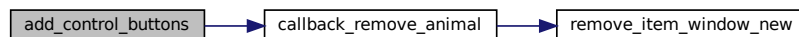
Referenced by `main_window_new()`.

```

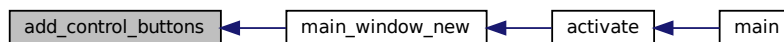
99     {
100         GtkWidget* buttonRemoveAnimal = gtk_button_new_with_label("Usuń element");
101         gtk_widget_set_hexpand(buttonRemoveAnimal, 1);
102         gtk_grid_attach(GTK_GRID(mainContainer), buttonRemoveAnimal, 0, 2, 2, 1);
103         g_signal_connect(G_OBJECT(buttonRemoveAnimal), "clicked", G_CALLBACK(
callback_remove_animal), (gpointer) app);
104
105
106         GtkWidget* buttonAddAnimal = gtk_button_new_with_label("Dodaj element");
107         gtk_widget_set_hexpand(buttonAddAnimal, 1);
108         gtk_widget_set_margin_start(buttonAddAnimal, 5);
109         gtk_grid_attach(GTK_GRID(mainContainer), buttonAddAnimal, 2, 2, 3, 1);
110     }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.2 add_table_headers()

```

void add_table_headers (
    GtkWidget * mainContainer,
    GtkWidget * table,
    AnimalLinkedList * animals )

```

Add table header buttons.

Parameters

<i>mainContainer</i>	the main GtkGrid of the window
<i>table</i>	table
<i>animals</i>	list of animals

Definition at line 147 of file `main_window.c`.

References `AGE`, `callback_sort_click()`, `COMMENT`, `ID`, `NAME`, `sort_callback_data_new()`, and `SPECIES`.

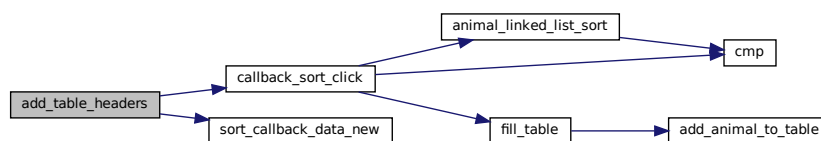
Referenced by `main_window_new()`.

```

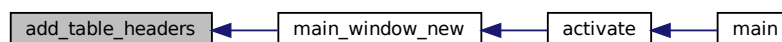
150 {
151     GtkWidget *buttonId = gtk_button_new();
152     gtk_button_set_label(GTK_BUTTON(buttonId), "Id");
153     gtk_widget_set_hexpand(buttonId, TRUE);
154
155     g_signal_connect(G_OBJECT(buttonId), "clicked", G_CALLBACK(
callback_sort_click), (gpointer) sort_callback_data_new(
ID, animals, table));
156     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonId), 0, 0, 1, 1);
157
158     GtkWidget *buttonName = gtk_button_new();
159     gtk_button_set_label(GTK_BUTTON(buttonName), "Imię");
160     gtk_widget_set_hexpand(buttonName, TRUE);
161     g_signal_connect(G_OBJECT(buttonName), "clicked", G_CALLBACK(
callback_sort_click), (gpointer) sort_callback_data_new(
NAME, animals, table));
162     gtk_widget_set_margin_start(buttonName, 5);
163     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonName), 1, 0, 1, 1);
164
165     GtkWidget *buttonSpecies = gtk_button_new();
166     gtk_button_set_label(GTK_BUTTON(buttonSpecies), "Gatunek");
167     gtk_widget_set_hexpand(buttonSpecies, TRUE);
168     g_signal_connect(G_OBJECT(buttonSpecies), "clicked", G_CALLBACK(
callback_sort_click),
169         (gpointer) sort_callback_data_new(
SPECIES, animals, table));
170     gtk_widget_set_margin_start(buttonSpecies, 5);
171     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonSpecies), 2, 0, 1, 1);
172
173     GtkWidget* buttonAge = gtk_button_new();
174     gtk_button_set_label(GTK_BUTTON(buttonAge), "Wiek");
175     gtk_widget_set_hexpand(buttonAge, TRUE);
176     g_signal_connect(G_OBJECT(buttonAge), "clicked", G_CALLBACK(
callback_sort_click),
177         (gpointer) sort_callback_data_new(AGE, animals, table));
178     gtk_widget_set_margin_start(buttonAge, 5);
179     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonAge), 3, 0, 1, 1);
180
181     GtkWidget* buttonComment = gtk_button_new();
182     gtk_button_set_label(GTK_BUTTON(buttonComment), "Komentarz");
183     gtk_widget_set_hexpand(buttonComment, TRUE);
184     g_signal_connect(G_OBJECT(buttonComment), "clicked", G_CALLBACK(
callback_sort_click),
185         (gpointer) sort_callback_data_new(
COMMENT, animals, table));
186     gtk_widget_set_margin_start(buttonComment, 5);
187     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonComment), 4, 0, 1, 1);
188 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.3 callback_sort_click()

```
void callback_sort_click (
    GtkWidget * widget,
    gpointer callback_data )
```

On click on any of the header buttons

Parameters

<i>widget</i>	
<i>callback_data</i>	an instance of SortCallbackData

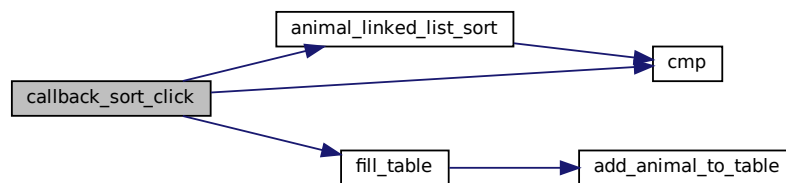
Definition at line 197 of file main_window.c.

References [animal_linked_list_sort\(\)](#), [cmp\(\)](#), [SortCallbackData::col](#), [fill_table\(\)](#), [SortCallbackData::list](#), [sort_asc](#), [sort_by](#), and [SortCallbackData::table](#).

Referenced by [add_table_headers\(\)](#).

```
200 {
201     SortCallbackData* cd = (SortCallbackData*) callback_data;
202     if(sort_by == cd->col) sort_asc = !sort_asc;
203     sort_by = cd->col;
204     animal_linked_list_sort(cd->list, cmp);
205     fill_table(cd->table, cd->list);
206 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.4 fill_table()

```
void fill_table (
    GtkWidget * table,
    AnimalLinkedList * animals )
```

Definition at line 264 of file main_window.c.

References `add_animal_to_table()`, `AnimalLinkedList::firstItem`, `AnimalLinkedListItem::next`, `AnimalLinkedList::size`, and `AnimalLinkedListItem::value`.

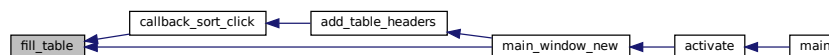
Referenced by `callback_sort_click()`, and `main_window_new()`.

```
266 {
267     GList *children, *iter;
268
269     children = gtk_container_get_children(GTK_CONTAINER(table));
270     for(iter = children; iter != NULL; iter = g_list_next(iter))
271         gtk_widget_destroy(GTK_WIDGET(iter->data));
272     g_list_free(children);
273
274     AnimalLinkedListItem* cur = animals->firstItem;
275     for(int i=0; i<animals->size; ++i){
276         add_animal_to_table(table, cur->value, i+1);
277         cur = cur->next;
278     }
279 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.5 main_window_new()

```
GtkWidget* main_window_new (
    GtkApplication * app )
```

Initialize the main window.

Parameters

<i>app</i>	Application
------------	-------------

Returns

main window, not shown yet

Definition at line 49 of file main_window.c.

References `add_control_buttons()`, `add_table_headers()`, `animal_linked_list_add_item()`, `animal_linked_list_new()`, `animal_new()`, and `fill_table()`.

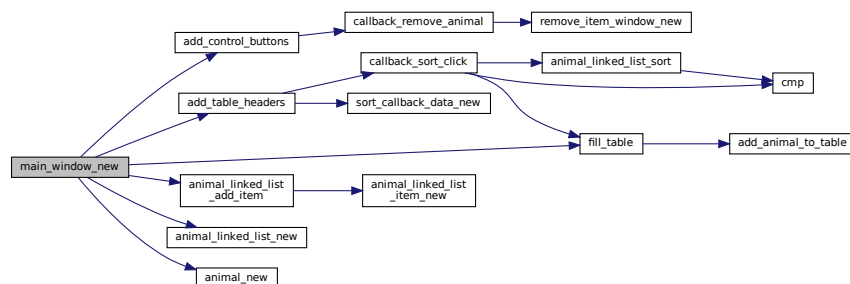
Referenced by `activate()`.

```

50 {
51     GtkWidget* window;
52     GtkWidget* mainContainer;
53
54     window = gtk_application_window_new (app);
55     gtk_window_set_title (GTK_WINDOW (window), "Ogród zoologiczny");
56     gtk_window_set_default_size (GTK_WINDOW (window), 500, 500);
57
58     mainContainer = gtk_grid_new();
59     gtk_container_add(GTK_CONTAINER (window), GTK_WIDGET (mainContainer));
60
61     GtkWidget* containerTable;
62     containerTable = gtk_scrolled_window_new(NULL, NULL);
63     gtk_widget_set_hexpand(containerTable, 1);
64     gtk_widget_set_vexpand(containerTable, 1);
65     GtkWidget* table;
66     table = gtk_grid_new();
67     AnimalLinkedList* animals = animal_linked_list_new();
68
69     for(int i=0; i<200; ++i){
70         Animal* a = animal_new(i, "Blazej", "Wielblad", i/4+3, "Je orzeszki");
71         animal_linked_list_add_item(animals, a);
72     }
73     gtk_container_add(GTK_CONTAINER(containerTable), GTK_WIDGET(table));
74     gtk_grid_set_column_homogeneous(GTK_GRID(table), gtk_true());
75     gtk_grid_set_column_homogeneous(GTK_GRID(mainContainer), gtk_true());
76     add_table_headers(mainContainer, table, animals);
77     fill_table(table, animals);
78     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (containerTable), 0, 1, 5, 1);
79     add_control_buttons(app, mainContainer);
80
81     return window;
82 }
83

```

Here is the call graph for this function:



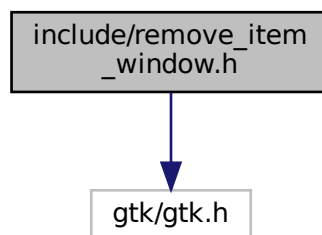
Here is the caller graph for this function:



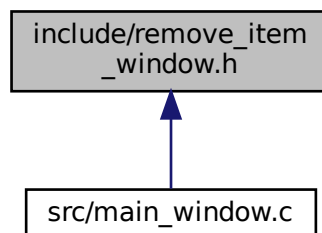
5.5 include/remove_item_window.h File Reference

```
#include <gtk/gtk.h>
```

Include dependency graph for remove_item_window.h:



This graph shows which files directly or indirectly include this file:



Functions

- GtkWidget * [remove_item_window_new](#) (GtkApplication *app)

5.5.1 Function Documentation

5.5.1.1 remove_item_window_new()

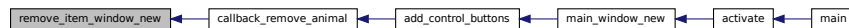
```
GtkWidget* remove_item_window_new (
    GtkApplication * app )
```

Definition at line 22 of file remove_item_window.c.

Referenced by callback_remove_animal().

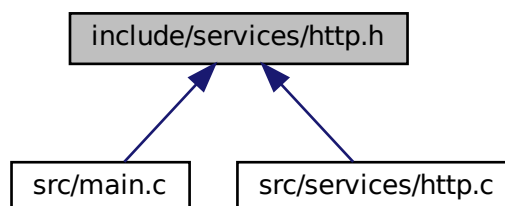
```
22                                     {
23     GtkWidget* window;
24     GtkWidget* mainContainer;
25     mainContainer = gtk_grid_new();
26     window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
27     gtk_container_add(GTK_CONTAINER(window), mainContainer);
28     GtkEntry* entryId = gtk_entry_new();
29     gtk_entry_set_placeholder_text(entryId, "Id elementu");
30     gtk_grid_attach(GTK_GRID(mainContainer), entryId, 0, 0, 1, 1);
31     gtk_widget_set_hexpand(entryId, 1);
32     gtk_widget_set_vexpand(entryId, 1);
33     gtk_window_set_title (GTK_WINDOW (window), "Uzuwanie elementu");
34
35     return window;
36 }
```

Here is the caller graph for this function:



5.6 include/services/http.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct [HttpResponse](#)

Typedefs

- typedef struct [HttpResponse](#) [HttpResponse](#)

Functions

- size_t [write_function](#) (void *ptr, size_t size, size_t nmemb, [HttpResponse](#) *r)
- [HttpResponse](#) * [http_get](#) (char *url)

Gets HTTP response for url.

5.6.1 Typedef Documentation

5.6.1.1 HttpResponse

```
typedef struct HttpResponse HttpResponse
```

5.6.2 Function Documentation

5.6.2.1 http_get()

```
HttpResponse* http_get (  
    char * url )
```

Gets HTTP response for url.

Parameters

<i>url</i>	Site address
------------	--------------

Returns

Filled [HttpResponse](#)

Definition at line 46 of file http.c.

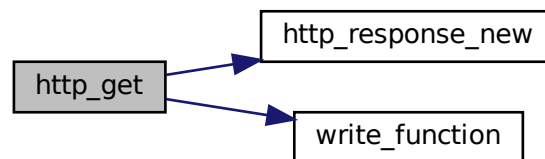
References [HttpResponse::code](#), [HttpResponse::data](#), [http_response_new\(\)](#), and [write_function\(\)](#).

```

46         {
47     CURL *curl;
48     HttpResponse * res = http_response_new();
49
50     curl = curl_easy_init();
51     if(curl) {
52         curl_easy_setopt(curl, CURLOPT_URL, url);
53         /* example.com is redirected, so we tell libcurl to follow redirection */
54         curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
55         curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_function);
56         curl_easy_setopt(curl, CURLOPT_WRITEDATA, res);
57
58         /* Perform the request, res->code will get the return code */
59         res->code = curl_easy_perform(curl);
60         /* Check for errors */
61         if(res->code != CURLE_OK)
62             fprintf(stderr, "curl_easy_perform() failed: %s\n",
63                     curl_easy_strerror(res->code));
64         else {
65             /* Print data for debug */
66             printf("Data: %s\n", res->data);
67         }
68
69         /* Cleanup */
70         curl_easy_cleanup(curl);
71     }
72     return res;
73 }

```

Here is the call graph for this function:



5.6.2.2 write_function()

```

size_t write_function (
    void * ptr,
    size_t size,
    size_t nmemb,
    HttpResponse * r )

```

Definition at line 25 of file `http.c`.

References `HttpResponse::data`, and `HttpResponse::size`.

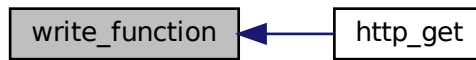
Referenced by `http_get()`.

```

25     {
26         size_t new_len = r->size + size*nmemb;
27         r->data= realloc(r->data, new_len+1);
28         memcpy(r->data+r->size, ptr, size*nmemb);
29         r->data[new_len] = '\0';
30         return size*nmemb;
31     }

```

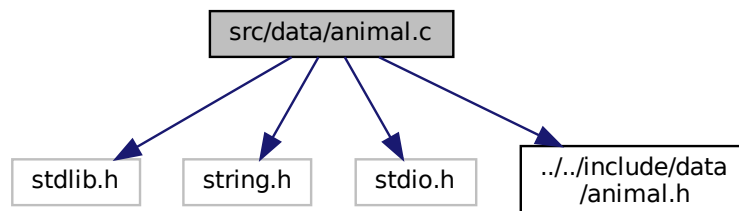
Here is the caller graph for this function:



5.7 README.md File Reference

5.8 src/data/animal.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "../include/data/animal.h"
Include dependency graph for animal.c:
```



Functions

- [AnimalLinkedListItem](#) * [animal_linked_list_item_new](#) ([Animal](#) *value)
- [Animal](#) * [animal_new](#) (int id, char *name, char *species, int age, char *comment)
- [AnimalLinkedList](#) * [animal_linked_list_new](#) ()
- void [animal_linked_list_sort](#) ([AnimalLinkedList](#) *list, int(*cmp)([Animal](#) *a, [Animal](#) *b))
- void [animal_linked_list_add_item](#) ([AnimalLinkedList](#) *list, [Animal](#) *value)

5.8.1 Function Documentation

5.8.1.1 animal_linked_list_add_item()

```
void animal_linked_list_add_item (
    AnimalLinkedList * list,
    Animal * value )
```

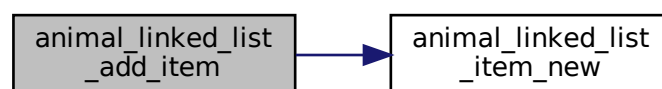
Definition at line 93 of file animal.c.

References `animal_linked_list_item_new()`, `AnimalLinkedList::firstItem`, `AnimalLinkedList::lastItem`, `AnimalLinkedListItem::next`, `AnimalLinkedListItem::prev`, and `AnimalLinkedList::size`.

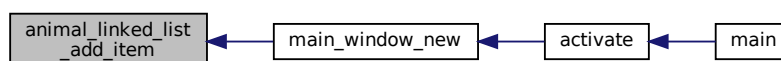
Referenced by `main_window_new()`.

```
93                                     {
94
95     AnimalLinkedListItem* cur = list->firstItem;
96     AnimalLinkedListItem* toAdd =
    animal_linked_list_item_new(value);
97
98     if(list->size == 0){
99         list->firstItem = toAdd;
100        list->lastItem = toAdd;
101        list->size ++;
102        return;
103    }
104    int i;
105    for(i=0; i<list->size-1; ++i){
106        cur = cur->next;
107    }
108
109    cur->next = toAdd;
110    list->lastItem = toAdd;
111    toAdd->prev = cur;
112
113    list->size ++;
114 };
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.1.2 animal_linked_list_item_new()

```
AnimalLinkedListItem* animal_linked_list_item_new (
    Animal * value )
```

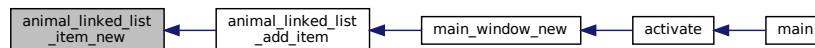
Definition at line 25 of file animal.c.

References `AnimalLinkedListItem::value`.

Referenced by `animal_linked_list_add_item()`.

```
25     {
26         AnimalLinkedListItem* item = malloc(sizeof(
AnimalLinkedListItem));
27         item->value = value;
28         return item;
29     }
```

Here is the caller graph for this function:



5.8.1.3 animal_linked_list_new()

```
AnimalLinkedList* animal_linked_list_new ( )
```

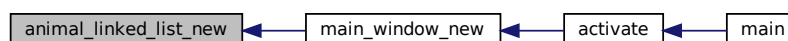
Definition at line 48 of file animal.c.

References `AnimalLinkedList::size`.

Referenced by `main_window_new()`.

```
48     {
49         AnimalLinkedList* list = malloc(sizeof(AnimalLinkedList));
50         list->size = 0;
51         return list;
52     }
```

Here is the caller graph for this function:



5.8.1.4 animal_linked_list_sort()

```
void animal_linked_list_sort (
    AnimalLinkedList * list,
    int (*) (Animal *a, Animal *b) cmp )
```

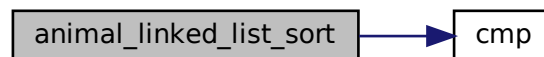
Definition at line 55 of file animal.c.

References `cmp()`, `AnimalLinkedList::firstItem`, `AnimalLinkedList::lastItem`, `AnimalLinkedListItem::next`, `AnimalLinkedListItem::prev`, `AnimalLinkedList::size`, and `AnimalLinkedListItem::value`.

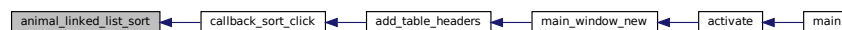
Referenced by `callback_sort_click()`.

```
57 {
58     int changed = 1;
59     while(changed){
60         changed = 0;
61         AnimalLinkedListItem* cur = list->firstItem;
62         for(int i=0; i<list->size-1; ++i){
63             if((*cmp)(cur->value, cur->next->value)){
64                 changed = 1;
65                 int has_prev = list->firstItem != cur;
66                 int has_next = list->lastItem != cur->next;
67                 AnimalLinkedListItem* node_0 = has_prev ? cur->
prev : NULL;
68                 AnimalLinkedListItem* node_1 = cur;
69                 AnimalLinkedListItem* node_2 = cur->next;
70                 AnimalLinkedListItem* node_3 = has_next ? cur->
next->next : NULL;
71                 node_1->next = node_2->next;
72                 node_1->prev = node_2;
73                 node_2->next = node_1;
74                 node_2->prev = node_0;
75                 if(has_prev){
76                     node_0->next = node_2;
77                 } else {
78                     list->firstItem = node_2;
79                 }
80                 if(has_next){
81                     node_3->prev = node_1;
82                 } else {
83                     list->lastItem = node_1;
84                 }
85                 cur = node_2;
86             }
87             cur = cur->next;
88         }
89     }
90 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.1.5 animal_new()

```
Animal* animal_new (
    int id,
    char * name,
    char * species,
    int age,
    char * comment )
```

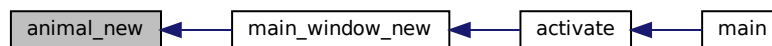
Definition at line 33 of file animal.c.

References `Animal::age`, `Animal::comment`, `Animal::id`, `Animal::name`, and `Animal::species`.

Referenced by `main_window_new()`.

```
33                                     {
34     Animal* animal = malloc(sizeof(Animal));
35     animal->id = id;
36     animal->name = malloc(strlen(name)+1);
37     strcpy(animal->name, name);
38     animal->species = malloc(strlen(species)+1);
39     strcpy(animal->species, species);
40
41     animal->age = age;
42     animal->comment = malloc(strlen(comment)+1);
43     strcpy(animal->comment, comment);
44     return animal;
45 }
```

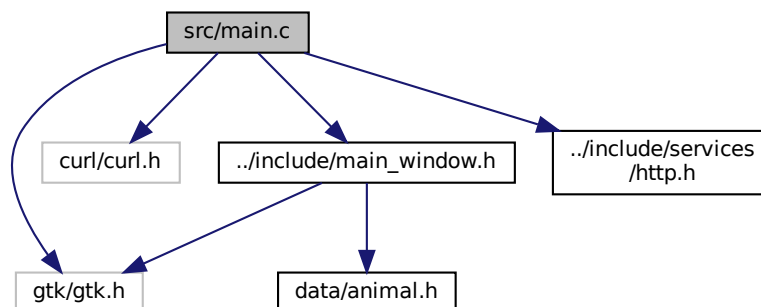
Here is the caller graph for this function:



5.9 src/main.c File Reference

```
#include <gtk/gtk.h>
#include <curl/curl.h>
#include "../include/main_window.h"
#include "../include/services/http.h"
```

Include dependency graph for main.c:



Functions

- static void [activate](#) (GtkApplication *app, gpointer user_data)
Initialize UI.
- int [main](#) (int argc, char **argv)

5.9.1 Function Documentation

5.9.1.1 activate()

```
static void activate (
    GtkApplication * app,
    gpointer user_data ) [static]
```

Initialize UI.

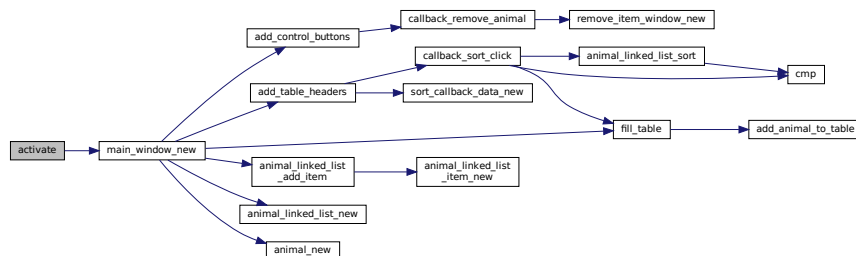
Definition at line 27 of file main.c.

References [main_window_new\(\)](#).

Referenced by [main\(\)](#).

```
28 {
29     gtk_widget_show_all (main\_window\_new(app));
30 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.1.2 main()

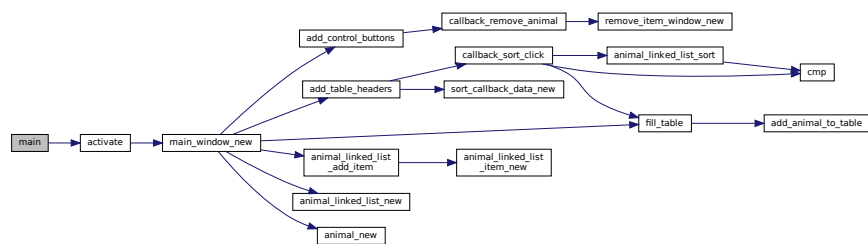
```
int main (
    int argc,
    char ** argv )
```

Definition at line 33 of file main.c.

References activate().

```
33      {
34      GtkApplication *app;
35
36      int status;
37
38      app = gtk_application_new ("pl.mrokita.pri-proj-3", G_APPLICATION_FLAGS_NONE);
39      g_signal_connect (app, "activate", G_CALLBACK (activate), NULL);
40      status = g_application_run (G_APPLICATION (app), argc, argv);
41      g_object_unref (app);
42
43      return status;
44 }
```

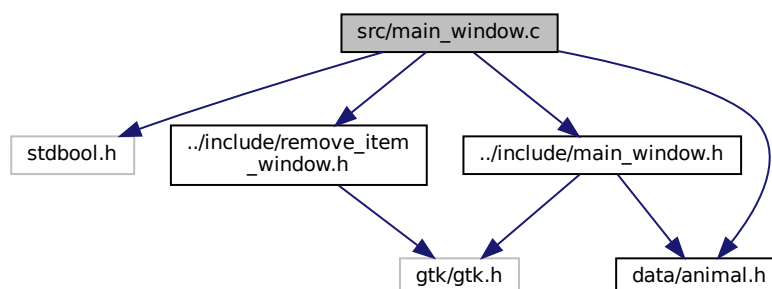
Here is the call graph for this function:



5.10 src/main_window.c File Reference

```
#include <stdbool.h>
#include "../include/main_window.h"
#include "../include/data/animal.h"
#include "../include/remove_item_window.h"
```

Include dependency graph for main_window.c:



Functions

- [SortCallbackData](#) * [sort_callback_data_new](#) ([COLUMN](#) col, [AnimalLinkedList](#) *list, [GtkWidget](#) *table)
- [GtkWidget](#) * [main_window_new](#) ([GtkApplication](#) *app)

Initialize the main window.
- void [callback_remove_animal](#) ([GtkWidget](#) *widget, gpointer callback_data)
- void [add_control_buttons](#) ([GtkApplication](#) *app, [GtkWidget](#) *mainContainer)
- int [cmp](#) ([Animal](#) *a, [Animal](#) *b)

Animal comparator Used to sort the table.
- void [add_table_headers](#) ([GtkWidget](#) *mainContainer, [GtkWidget](#) *table, [AnimalLinkedList](#) *animals)

Add table header buttons.
- void [callback_sort_click](#) ([GtkWidget](#) *widget, gpointer callback_data)
- void [add_animal_to_table](#) ([GtkWidget](#) *table, [Animal](#) *animal, int row)
- void [fill_table](#) ([GtkWidget](#) *table, [AnimalLinkedList](#) *animals)

Variables

- int [sort_by](#) = -1
- int [sort_asc](#) = 1

5.10.1 Function Documentation

5.10.1.1 add_animal_to_table()

```
void add_animal_to_table (
    GtkWidget * table,
    Animal * animal,
    int row )
```

Add animal to the table

Parameters

<i>table</i>	
<i>animal</i>	
<i>row</i>	row number, should be empty.

Definition at line 215 of file main_window.c.

References [Animal::age](#), [Animal::comment](#), [Animal::id](#), [Animal::name](#), and [Animal::species](#).

Referenced by [fill_table\(\)](#).

```
219 {
220     char* idString = malloc(12);
221     sprintf(idString, "%d", animal->id);
222     GtkWidget* labelId = gtk_label_new(idString);
223     gtk_label_set_xalign(GTK_LABEL (labelId), 0.0);
224     gtk_widget_set_margin_start(labelId, 5);
```

```

225     gtk_widget_set_margin_top(labelId, 5);
226     gtk_widget_set_hexpand(labelId, TRUE);
227     gtk_grid_attach(GTK_GRID (table), labelId, 0, row, 1, 1);
228
229     GtkWidget* labelName = gtk_label_new(animal->name);
230     gtk_label_set_xalign(GTK_LABEL (labelName), 0.0);
231     gtk_widget_set_margin_start(labelName, 5);
232     gtk_widget_set_margin_top(labelName, 5);
233     gtk_widget_set_hexpand(labelName, TRUE);
234     gtk_grid_attach(GTK_GRID (table), labelName, 1, row, 1, 1);
235
236     GtkWidget* labelSpecies = gtk_label_new(animal->species);
237     gtk_label_set_xalign(GTK_LABEL (labelSpecies), 0.0);
238     gtk_widget_set_margin_start(labelSpecies, 5);
239     gtk_widget_set_margin_top(labelSpecies, 5);
240     gtk_widget_set_hexpand(labelSpecies, TRUE);
241     gtk_grid_attach(GTK_GRID (table), labelSpecies, 2, row, 1, 1);
242
243     char* ageString = malloc(12);
244     sprintf(ageString, "%d", animal->age);
245
246     GtkWidget* labelAge = gtk_label_new(ageString);
247     gtk_label_set_xalign(GTK_LABEL (labelAge), 0.0);
248     gtk_widget_set_margin_start(labelAge, 5);
249     gtk_widget_set_margin_top(labelAge, 5);
250     gtk_widget_set_hexpand(labelAge, TRUE);
251     gtk_grid_attach(GTK_GRID (table), labelAge, 3, row, 1, 1);
252
253     GtkWidget* labelComment = gtk_label_new(animal->comment);
254     gtk_label_set_xalign(GTK_LABEL (labelComment), 0.0);
255     gtk_widget_set_margin_start(labelComment, 5);
256     gtk_widget_set_margin_top(labelComment, 5);
257     gtk_widget_set_hexpand(labelComment, TRUE);
258     gtk_grid_attach(GTK_GRID (table), labelComment, 4, row, 1, 1);
259     gtk_widget_show_all(table);
260 }

```

Here is the caller graph for this function:



5.10.1.2 add_control_buttons()

```

void add_control_buttons (
    GtkApplication * app,
    GtkWidget * mainContainer )

```

Add edit and delete buttons below the table

Parameters

<i>mainContainer</i>	
----------------------	--

Definition at line 97 of file main_window.c.

References `callback_remove_animal()`.

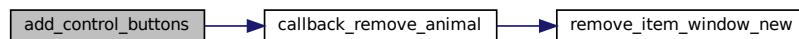
Referenced by `main_window_new()`.

```

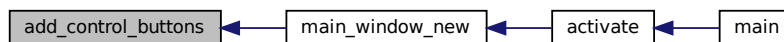
99      {
100      GtkWidget* buttonRemoveAnimal = gtk_button_new_with_label("Usuń element");
101      gtk_widget_set_hexpand(buttonRemoveAnimal, 1);
102      gtk_grid_attach(GTK_GRID(mainContainer), buttonRemoveAnimal, 0, 2, 2, 1);
103      g_signal_connect(G_OBJECT(buttonRemoveAnimal), "clicked", G_CALLBACK(
callback_remove_animal), (gpointer) app);
104
105
106      GtkWidget* buttonAddAnimal = gtk_button_new_with_label("Dodaj element");
107      gtk_widget_set_hexpand(buttonAddAnimal, 1);
108      gtk_widget_set_margin_start(buttonAddAnimal, 5);
109      gtk_grid_attach(GTK_GRID(mainContainer), buttonAddAnimal, 2, 2, 3, 1);
110  }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.3 add_table_headers()

```

void add_table_headers (
    GtkWidget * mainContainer,
    GtkWidget * table,
    AnimalLinkedList * animals )

```

Add table header buttons.

Parameters

<i>mainContainer</i>	the main GtkGrid of the window
<i>table</i>	table
<i>animals</i>	list of animals

Definition at line 147 of file `main_window.c`.

References `AGE`, `callback_sort_click()`, `COMMENT`, `ID`, `NAME`, `sort_callback_data_new()`, and `SPECIES`.

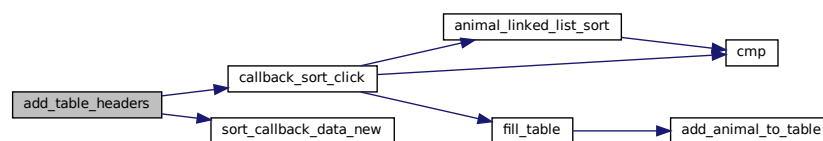
Referenced by `main_window_new()`.

```

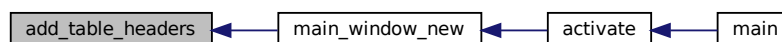
150 {
151     GtkWidget *buttonId = gtk_button_new();
152     gtk_button_set_label(GTK_BUTTON(buttonId), "Id");
153     gtk_widget_set_hexpand(buttonId, TRUE);
154
155     g_signal_connect(G_OBJECT(buttonId), "clicked", G_CALLBACK(
callback_sort_click), (gpointer) sort_callback_data_new(
ID, animals, table));
156     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonId), 0, 0, 1, 1);
157
158     GtkWidget *buttonName = gtk_button_new();
159     gtk_button_set_label(GTK_BUTTON(buttonName), "Imię");
160     gtk_widget_set_hexpand(buttonName, TRUE);
161     g_signal_connect(G_OBJECT(buttonName), "clicked", G_CALLBACK(
callback_sort_click), (gpointer) sort_callback_data_new(
NAME, animals, table));
162     gtk_widget_set_margin_start(buttonName, 5);
163     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonName), 1, 0, 1, 1);
164
165     GtkWidget *buttonSpecies = gtk_button_new();
166     gtk_button_set_label(GTK_BUTTON(buttonSpecies), "Gatunek");
167     gtk_widget_set_hexpand(buttonSpecies, TRUE);
168     g_signal_connect(G_OBJECT(buttonSpecies), "clicked", G_CALLBACK(
callback_sort_click),
169         (gpointer) sort_callback_data_new(
SPECIES, animals, table));
170     gtk_widget_set_margin_start(buttonSpecies, 5);
171     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonSpecies), 2, 0, 1, 1);
172
173     GtkWidget* buttonAge = gtk_button_new();
174     gtk_button_set_label(GTK_BUTTON(buttonAge), "Wiek");
175     gtk_widget_set_hexpand(buttonAge, TRUE);
176     g_signal_connect(G_OBJECT(buttonAge), "clicked", G_CALLBACK(
callback_sort_click),
177         (gpointer) sort_callback_data_new(AGE, animals, table));
178     gtk_widget_set_margin_start(buttonAge, 5);
179     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonAge), 3, 0, 1, 1);
180
181     GtkWidget* buttonComment = gtk_button_new();
182     gtk_button_set_label(GTK_BUTTON(buttonComment), "Komentarz");
183     gtk_widget_set_hexpand(buttonComment, TRUE);
184     g_signal_connect(G_OBJECT(buttonComment), "clicked", G_CALLBACK(
callback_sort_click),
185         (gpointer) sort_callback_data_new(
COMMENT, animals, table));
186     gtk_widget_set_margin_start(buttonComment, 5);
187     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonComment), 4, 0, 1, 1);
188 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.4 `callback_remove_animal()`

```
void callback_remove_animal (
    GtkWidget * widget,
    gpointer callback_data )
```

Definition at line 85 of file `main_window.c`.

References `remove_item_window_new()`.

Referenced by `add_control_buttons()`.

```
88     {
89         gtk_widget_show_all(remove_item_window_new((GtkApplication*)callback_data));
90     }
```

Here is the call graph for this function:



Here is the caller graph for this function:

5.10.1.5 `callback_sort_click()`

```
void callback_sort_click (
    GtkWidget * widget,
    gpointer callback_data )
```

On click on any of the header buttons

Parameters

<i>widget</i>	
<i>callback_data</i>	an instance of SortCallbackData

Definition at line 197 of file `main_window.c`.

References `animal_linked_list_sort()`, `cmp()`, `SortCallbackData::col`, `fill_table()`, `SortCallbackData::list`, `sort_asc`, `sort_by`, and `SortCallbackData::table`.

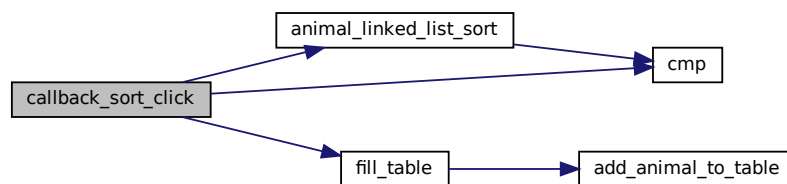
Referenced by `add_table_headers()`.

```

200 {
201     SortCallbackData* cd = (SortCallbackData*) callback_data;
202     if(sort_by == cd->col) sort_asc = !sort_asc;
203     sort_by = cd->col;
204     animal_linked_list_sort(cd->list, cmp);
205     fill_table(cd->table, cd->list);
206 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.6 cmp()

```

int cmp (
    Animal * a,
    Animal * b )

```

Animal comparator Used to sort the table.

Parameters

<i>a</i>	
<i>b</i>	

Returns

boolean, true if *a* and *b* should be swapped

Definition at line 124 of file main_window.c.

References `Animal::age`, `AGE`, `Animal::comment`, `COMMENT`, `Animal::id`, `ID`, `Animal::name`, `NAME`, `sort_asc`, `sort_by`, `Animal::species`, and `SPECIES`.

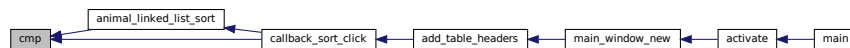
Referenced by `animal_linked_list_sort()`, and `callback_sort_click()`.

```

124         {
125             int res = 0;
126             if(!sort_asc) {
127                 Animal* t = a;
128                 a = b;
129                 b = t;
130             }
131             if(sort_by == ID) res = a->id > b->id;
132             if(sort_by == AGE) res = a->age > b->age;
133             if(sort_by == NAME) res = strcmp(a->name, b->name) > 0;
134             if(sort_by == SPECIES) res = strcmp(a->species, b->
species) > 0;
135             if(sort_by == COMMENT) res = strcmp(a->comment, b->
comment) > 0;
136             return res;
137     }

```

Here is the caller graph for this function:



5.10.1.7 fill_table()

```

void fill_table (
    GtkWidget * table,
    AnimalLinkedList * animals )

```

Definition at line 264 of file main_window.c.

References `add_animal_to_table()`, `AnimalLinkedList::firstItem`, `AnimalLinkedListItem::next`, `AnimalLinkedList::size`, and `AnimalLinkedListItem::value`.

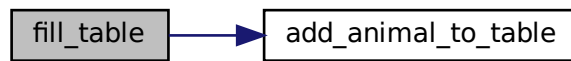
Referenced by `callback_sort_click()`, and `main_window_new()`.

```

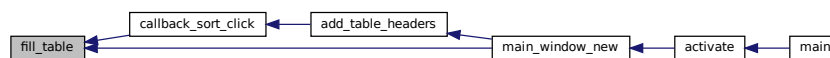
266 {
267     GList *children, *iter;
268
269     children = gtk_container_get_children(GTK_CONTAINER(table));
270     for(iter = children; iter != NULL; iter = g_list_next(iter))
271         gtk_widget_destroy(GTK_WIDGET(iter->data));
272     g_list_free(children);
273
274     AnimalLinkedListItem* cur = animals->firstItem;
275     for(int i=0; i<animals->size; ++i){
276         add_animal_to_table(table, cur->value, i+1);
277         cur = cur->next;
278     }
279 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.8 main_window_new()

```
GtkWidget* main_window_new (
    GtkApplication * app )
```

Initialize the main window.

Parameters

<i>app</i>	Application
------------	-------------

Returns

main window, not shown yet

Definition at line 49 of file `main_window.c`.

References `add_control_buttons()`, `add_table_headers()`, `animal_linked_list_add_item()`, `animal_linked_list_new()`, `animal_new()`, and `fill_table()`.

Referenced by `activate()`.

```

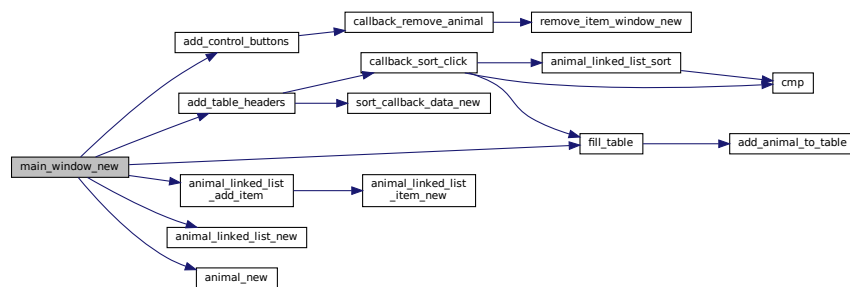
50 {
51
52     GtkWidget* window;
53     GtkWidget* mainContainer;
54
55     window = gtk_application_window_new (app);
56     gtk_window_set_title (GTK_WINDOW (window), "Ogród zoologiczny");
57     gtk_window_set_default_size (GTK_WINDOW (window), 500, 500);
58
59     mainContainer = gtk_grid_new();
```

```

60     gtk_container_add(GTK_CONTAINER (window), GTK_WIDGET (mainContainer));
61
62     GtkWidget* containerTable;
63     containerTable = gtk_scrolled_window_new(NULL, NULL);
64     gtk_widget_set_hexpand(containerTable, 1);
65     gtk_widget_set_vexpand(containerTable, 1);
66     GtkWidget* table;
67     table = gtk_grid_new();
68     AnimalLinkedList* animals = animal_linked_list_new();
69
70     for(int i=0; i<200; ++i){
71         Animal* a = animal_new(i, "Blazej", "Wielblad", i/4+3, "Je orzeszki");
72         animal_linked_list_add_item(animals, a);
73     }
74     gtk_container_add(GTK_CONTAINER(containerTable), GTK_WIDGET(table));
75     gtk_grid_set_column_homogeneous(GTK_GRID(table), gtk_true());
76     gtk_grid_set_column_homogeneous(GTK_GRID(mainContainer), gtk_true());
77     add_table_headers(mainContainer, table, animals);
78     fill_table(table, animals);
79     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (containerTable), 0, 1, 5, 1);
80     add_control_buttons(app, mainContainer);
81
82     return window;
83 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.9 sort_callback_data_new()

```

SortCallbackData* sort_callback_data_new (
    COLUMN col,
    AnimalLinkedList * list,
    GtkWidget * table )

```

A container for data required to sort the table

Parameters

<i>col</i>	Column
<i>list</i>	List of animals
<i>table</i>	GtkGrid of the table

Returns

All arguments neatly packed in an [SortCallbackData](#) instance.

Definition at line 32 of file main_window.c.

References [SortCallbackData::col](#), [SortCallbackData::list](#), and [SortCallbackData::table](#).

Referenced by [add_table_headers\(\)](#).

```

34     {
35         SortCallbackData* sortCallbackData = malloc(sizeof(
SortCallbackData));
36         sortCallbackData->col = col;
37         sortCallbackData->list = list;
38         sortCallbackData->table = table;
39         return sortCallbackData;
40     }

```

Here is the caller graph for this function:



5.10.2 Variable Documentation

5.10.2.1 sort_asc

```
int sort_asc = 1
```

Definition at line 113 of file main_window.c.

Referenced by [callback_sort_click\(\)](#), and [cmp\(\)](#).

5.10.2.2 sort_by

```
int sort_by = -1
```

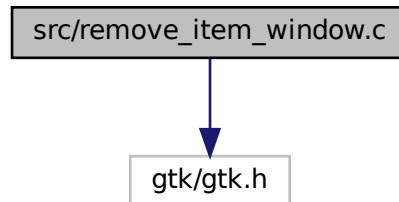
Definition at line 112 of file main_window.c.

Referenced by [callback_sort_click\(\)](#), and [cmp\(\)](#).

5.11 src/remove_item_window.c File Reference

```
#include <gtk/gtk.h>
```

Include dependency graph for remove_item_window.c:



Functions

- GtkWidget* [remove_item_window_new](#) (GtkApplication *app)

5.11.1 Function Documentation

5.11.1.1 remove_item_window_new()

```
GtkWidget* remove_item_window_new (
    GtkApplication * app )
```

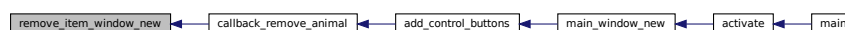
Definition at line 22 of file remove_item_window.c.

Referenced by `callback_remove_animal()`.

```

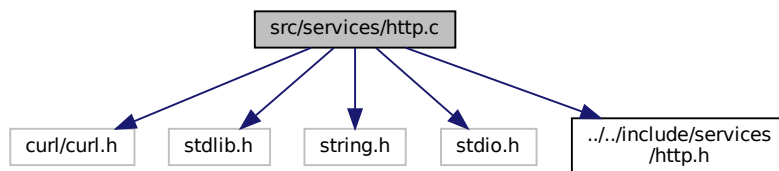
22                                     {
23     GtkWidget* window;
24     GtkWidget* mainContainer;
25     mainContainer = gtk_grid_new();
26     window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
27     gtk_container_add(GTK_CONTAINER(window), mainContainer);
28     GtkEntry* entryId = gtk_entry_new();
29     gtk_entry_set_placeholder_text(entryId, "Id elementu");
30     gtk_grid_attach(GTK_GRID(mainContainer), entryId, 0, 0, 1, 1);
31     gtk_widget_set_hexpand(entryId, 1);
32     gtk_widget_set_vexpand(entryId, 1);
33     gtk_window_set_title (GTK_WINDOW (window), "Uzuwanie elementu");
34
35     return window;
36 }
```

Here is the caller graph for this function:



5.12 src/services/http.c File Reference

```
#include <curl/curl.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "../include/services/http.h"
Include dependency graph for http.c:
```



Functions

- `size_t write_function` (`void *ptr`, `size_t size`, `size_t nmemb`, [HttpResponse](#) *`r`)
- [HttpResponse](#) * `http_response_new` ()
- [HttpResponse](#) * `http_get` (`char *url`)
Gets HTTP response for url.

5.12.1 Function Documentation

5.12.1.1 http_get()

```
HttpResponse* http_get (
    char * url )
```

Gets HTTP response for url.

Parameters

<code>url</code>	Site address
------------------	--------------

Returns

Filled [HttpResponse](#)

Definition at line 46 of file http.c.

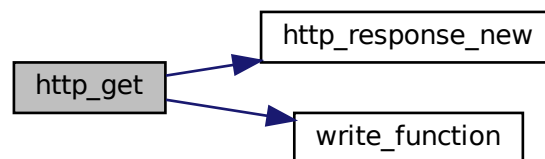
References [HttpResponse::code](#), [HttpResponse::data](#), [http_response_new\(\)](#), and [write_function\(\)](#).


```

46         {
47     CURL *curl;
48     HttpResponse * res = http_response_new();
49
50     curl = curl_easy_init();
51     if(curl) {
52         curl_easy_setopt(curl, CURLOPT_URL, url);
53         /* example.com is redirected, so we tell libcurl to follow redirection */
54         curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
55         curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_function);
56         curl_easy_setopt(curl, CURLOPT_WRITEDATA, res);
57
58         /* Perform the request, res->code will get the return code */
59         res->code = curl_easy_perform(curl);
60         /* Check for errors */
61         if(res->code != CURLE_OK)
62             fprintf(stderr, "curl_easy_perform() failed: %s\n",
63                     curl_easy_strerror(res->code));
64         else {
65             /* Print data for debug */
66             printf("Data: %s\n", res->data);
67         }
68
69         /* Cleanup */
70         curl_easy_cleanup(curl);
71     }
72     return res;
73 }

```

Here is the call graph for this function:



5.12.1.2 http_response_new()

```
HttpResponse* http_response_new ( )
```

Definition at line 33 of file `http.c`.

References `HttpResponse::data`, and `HttpResponse::size`.

Referenced by `http_get()`.

```

33     {
34     HttpResponse * res = malloc(sizeof(HttpResponse));
35     res->size = 0;
36     res->data = malloc(res->size+1);
37     res->data[res->size] = '\0';
38     return res;
39 };

```

Here is the caller graph for this function:



5.12.1.3 write_function()

```
size_t write_function (
    void * ptr,
    size_t size,
    size_t nmemb,
    HttpResponse * r )
```

Definition at line 25 of file `http.c`.

References `HttpResponse::data`, and `HttpResponse::size`.

Referenced by `http_get()`.

```
25                                     {
26     size_t new_len = r->size + size*nmemb;
27     r->data= realloc(r->data, new_len+1);
28     memcpy(r->data+r->size, ptr, size*nmemb);
29     r->data[new_len] = '\0';
30     return size*nmemb;
31 }
```

Here is the caller graph for this function:



Index

ARCHITECTURE_ID

CMakeCCompilerId.c, 17

activate

main.c, 43

add_animal_to_table

main_window.c, 45

add_control_buttons

main_window.c, 46

main_window.h, 28

add_table_headers

main_window.c, 47

main_window.h, 29

age

Animal, 8

Animal, 7

age, 8

animal.h, 22

comment, 8

id, 8

name, 8

species, 8

animal.c

animal_linked_list_add_item, 38

animal_linked_list_item_new, 39

animal_linked_list_new, 40

animal_linked_list_sort, 40

animal_new, 41

animal.h

Animal, 22

animal_linked_list_add_item, 23

animal_linked_list_get_item, 24

animal_linked_list_new, 24

animal_linked_list_sort, 24

animal_new, 25

AnimalLinkedList, 22

AnimalLinkedListItem, 23

animal_linked_list_add_item

animal.c, 38

animal.h, 23

animal_linked_list_get_item

animal.h, 24

animal_linked_list_item_new

animal.c, 39

animal_linked_list_new

animal.c, 40

animal.h, 24

animal_linked_list_sort

animal.c, 40

animal.h, 24

animal_new

animal.c, 41

animal.h, 25

AnimalLinkedList, 9

animal.h, 22

firstItem, 10

lastItem, 10

size, 10

AnimalLinkedListItem, 11

animal.h, 23

next, 11

prev, 11

value, 12

C_DIALECT

CMakeCCompilerId.c, 17

CMakeCCompilerId.c

ARCHITECTURE_ID, 17

C_DIALECT, 17

COMPILER_ID, 18

DEC, 18

HEX, 18

info_arch, 20

info_compiler, 20

info_language_dialect_default, 20

info_platform, 20

main, 19

PLATFORM_ID, 18

STRINGIFY_HELPER, 19

STRINGIFY, 19

CMakeFiles/3.13.2/CompilerIdC/CMakeCCompilerId.c,
17

CMakeFiles/feature_tests.c, 21

COLUMN

main_window.h, 27, 28

COMPILER_ID

CMakeCCompilerId.c, 18

callback_remove_animal

main_window.c, 48

callback_sort_click

main_window.c, 49

main_window.h, 30

cmp

main_window.c, 50

code

HttpResponse, 13

col

SortCallbackData, 14

comment

Animal, 8

- DEC
 - CMakeCCompilerId.c, 18
- data
 - HttpResponse, 13
- feature_tests.c
 - features, 21
 - main, 21
- features
 - feature_tests.c, 21
- fill_table
 - main_window.c, 51
 - main_window.h, 31
- firstItem
 - AnimalLinkedList, 10
- HEX
 - CMakeCCompilerId.c, 18
- http.c
 - http_get, 56
 - http_response_new, 57
 - write_function, 58
- http.h
 - http_get, 36
 - HttpResponse, 36
 - write_function, 37
- http_get
 - http.c, 56
 - http.h, 36
- http_response_new
 - http.c, 57
- HttpResponse, 12
 - code, 13
 - data, 13
 - http.h, 36
 - size, 13
- id
 - Animal, 8
- include/data/animal.h, 22
- include/main_window.h, 26
- include/remove_item_window.h, 34
- include/services/http.h, 35
- info_arch
 - CMakeCCompilerId.c, 20
- info_compiler
 - CMakeCCompilerId.c, 20
- info_language_dialect_default
 - CMakeCCompilerId.c, 20
- info_platform
 - CMakeCCompilerId.c, 20
- lastItem
 - AnimalLinkedList, 10
- list
 - SortCallbackData, 15
- main
 - CMakeCCompilerId.c, 19
 - feature_tests.c, 21
 - main.c, 43
- main.c
 - activate, 43
 - main, 43
- main_window.c
 - add_animal_to_table, 45
 - add_control_buttons, 46
 - add_table_headers, 47
 - callback_remove_animal, 48
 - callback_sort_click, 49
 - cmp, 50
 - fill_table, 51
 - main_window_new, 52
 - sort_asc, 54
 - sort_by, 54
 - sort_callback_data_new, 53
- main_window.h
 - add_control_buttons, 28
 - add_table_headers, 29
 - COLUMN, 27, 28
 - callback_sort_click, 30
 - fill_table, 31
 - main_window_new, 32
 - SortCallbackData, 27
- main_window_new
 - main_window.c, 52
 - main_window.h, 32
- name
 - Animal, 8
- next
 - AnimalLinkedListItem, 11
- PLATFORM_ID
 - CMakeCCompilerId.c, 18
- prev
 - AnimalLinkedListItem, 11
- README.md, 38
- remove_item_window.c
 - remove_item_window_new, 55
- remove_item_window.h
 - remove_item_window_new, 35
- remove_item_window_new
 - remove_item_window.c, 55
 - remove_item_window.h, 35
- STRINGIFY_HELPER
 - CMakeCCompilerId.c, 19
- STRINGIFY
 - CMakeCCompilerId.c, 19
- size
 - AnimalLinkedList, 10
 - HttpResponse, 13
- sort_asc
 - main_window.c, 54
- sort_by
 - main_window.c, 54

- sort_callback_data_new
 - main_window.c, [53](#)
- SortCallbackData, [14](#)
 - col, [14](#)
 - list, [15](#)
 - main_window.h, [27](#)
 - table, [15](#)
- species
 - Animal, [8](#)
- src/data/animal.c, [38](#)
- src/main.c, [42](#)
- src/main_window.c, [44](#)
- src/remove_item_window.c, [55](#)
- src/services/http.c, [56](#)
- table
 - SortCallbackData, [15](#)
- value
 - AnimalLinkedListItem, [12](#)
- write_function
 - http.c, [58](#)
 - http.h, [37](#)