`struct List`: Doppelt verkettete Liste

Die Datenstruktur `struct List` (im Folgenden *Liste*) und die zugehörigen Funktionen implementieren eine doppelt verkettete Liste.

```c
struct Item // type to be stored in the list
{
  char name[128];
};
typedef struct Item T;


struct ListNode // type for nodes in the list
{
  struct ListNode *prev; // pointer to previous node
  struct ListNode *next; // pointer to next node
  T value;               // value (struct Item)
};

struct List // list type itself
{
  struct ListNode *entry; // cyclic entry node (before first AND after last)
};

struct List *list_new();
void list_delete(struct List **self);
void list_clear(struct List *self);
size_t list_size(const struct List *self);
void list_push_back(struct List *self, const T *value);
void list_print(const struct List *self);
```

`list_new`: Dynamisches Listen-Objekt erzeugen

```c
struct List *list_new();
```

Funktionsbeschreibung: Alloziert ein `struct List`-Objekt, initialisert es und gibt einen Zeiger darauf zurück.

Rückgabewert:

- Zeiger auf eine dynamisch allozierte Liste.
- Der Aufrufende wird alleiniger Besitzer des dynamischen Speichers (und übernimmt somit auch die Pflicht des Freigebens (`list_delete`).

`list_delete`: Dynamisches Listen-Objekt freigeben

```c
void list_delete(struct List **self);
```

Funktionsbeschreibung: Dealloziert ein per doppelter Referenz übergebenes alloziertes `struct List`-Objekt. Gibt zuvor den Speicher aller Knoten (inklusive `entry`-Knoten) frei und setzt den Zeiger auf die Liste auf `NULL`.

Parameter `self`: Zeiger auf einen Zeiger auf eine dynamisch allozierte Liste (Doppelzeiger).

`list_clear`: Alle Elemente/Knoten freigeben

```c
void list_clear(struct List *self);
```

Funktionsbeschreibung: Entfernt alle Knoten aus der übergebenen Liste und gibt den zugehörigen Speicher der jeweiligen Knoten frei (der `entry`-Knoten bleibt erhalten).

Parameter `self`: Zeiger auf die Liste

`list_size`: Anzahl an Elementen/Knoten

```
size_t list_size(const struct List *self);
```

Funktionsbeschreibung: Gibt die momentane Anzahl an Knoten in der Liste zurück (der `entry`-Knoten wird nicht gezählt).

Parameter `self`: Zeiger auf die Liste

Rückgabewert: Anzahl an Knoten

`list_push_back`: Anhängen eines Elements am Ende

```
void list_push_back(struct List *self, const T *value);
```

Funktionsbeschreibung: Fügt ein weiteres Element am Ende hinzu.

Parameter:

- `self`: Zeiger auf die Liste
- `value`: Zeiger auf den einzufügenden Wert

`list_print`: Ausgabe aller Elemente

```
void list_print(const struct List *self);
```

Funktionsbeschreibung: Gibt alle Elemente der übergebenen Liste in die Konsole aus.

Parameter `self`: Zeiger auf die Liste