1 Programmieren in C

1.10 Dynamischer Speicher

```c
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
   int *buffer = NULL;
   int count = 7;
   buffer = malloc (count * sizeof (int));
   for (int i = 0; i < count; i++)
      buffer[i] = i;
   for (int i = 0; i < count; i++)
      printf ("%d ", buffer[i]);
   printf ("\n");
   free (buffer);
   return 0;
}
```

1 Programmieren in C

1.10 Dynamischer Speicher

```
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
   int *buffer = NULL;
   int count = 7;
   buffer = malloc (count * sizeof (int));
   for (int i = 0; i < count; i++)
      buffer[i] = i;
   for (int i = 0; i < count; i++)
      printf ("%d ", buffer[i]);
   printf ("\n");
   free (buffer);
   return 0;
}
```

Man sieht buffer nicht an,
wie groß das Array ist.

1 Programmieren in C

1.10 Dynamischer Speicher

```
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  int *buffer = NULL;
  int count = 7;
  buffer = malloc (count * sizeof (int));
  for (int i = 0; i < count; i++)
    buffer[i] = i;
  for (int i = 0; i < count; i++)
    printf ("%d ", buffer[i]);
  printf ("\n");
  free (buffer);
  return 0;
}
```

Man sieht buffer nicht an,
wie groß das Array ist.

$\longrightarrow$ buffer und count gemeinsam
in einem **struct** speichern

1 Programmieren in C

1.10 Dynamischer Speicher

```
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  int *buffer = NULL;
  int count = 7;
  buffer = malloc (count * sizeof (int));
  for (int i = 0; i < count; i++)
    buffer[i] = i;
  for (int i = 0; i < count; i++)
    printf ("%d ", buffer[i]);
  printf ("\n");
  free (buffer);
  return 0;
}
```

Man sieht buffer nicht an,
wie groß das Array ist.

$\longrightarrow$ buffer und count gemeinsam
in einem **struct** speichern

$\longrightarrow$ „Container-Klasse“

1 Programmieren in C

1.10 Dynamischer Speicher

```
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  int *buffer = NULL;
  int count = 7;
  buffer = malloc (count * sizeof (int));
  for (int i = 0; i < count; i++)
    buffer[i] = i;
  for (int i = 0; i < count; i++)
    printf ("%d ", buffer[i]);
  printf ("\n");
  free (buffer);
  return 0;
}
```

Man sieht buffer nicht an,
wie groß das Array ist.

⟶ buffer und count gemeinsam
in einem **struct** speichern
⟶ „Container-Klasse“

Aufgabe

Schreiben Sie eine Bibliothek
zur Nutzung eines dynamischen Arrays
für selbstgewählte Daten
(z. B. Integer, String)

1.11 Rekursive Datenstrukturen

```
#include <stdio.h>
#include <stdlib.h>

typedef struct string_list
{
   char *content;
   struct string_list *next;
}
string_list;

int main (void)
{
   string_list *first = malloc (sizeof (string_list));
   first->content = "Dies ist ein Test.";
   first->next = malloc (sizeof (string_list));
   first->next->content = "Was ist das?";
   first->next->next = malloc (sizeof (string_list));
   first->next->next->content = "Ein Test.";
   first->next->next->next = NULL;
   for (string_list *p = first; p; p = p->next)
      printf ("%s\n", p->content);
}
```

1.11 Rekursive Datenstrukturen

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct string_list
{
   char *content;
   struct string_list *next;
}
string_list;
```

```c
int main (void)
{
   string_list *first = malloc (sizeof (string_list));
   first->content = "Dies ist ein Test.";
   first->next = malloc (sizeof (string_list));
   first->next->content = "Was ist das?";
   first->next->next = malloc (sizeof (string_list));
   first->next->next->content = "Ein Test.";
   first->next->next->next = NULL;
   for (string_list *p = first; p; p = p->next)
      printf ("%s\n", p->content);
}
```

Verkettete Liste

1.11 Rekursive Datenstrukturen

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct string_list
{
   char *content;
   struct string_list *next;
}
string_list;
```

```c
int main (void)
{
   string_list *first = malloc (sizeof (string_list));
   first->content = "Dies ist ein Test.";
   first->next = malloc (sizeof (string_list));
   first->next->content = "Was ist das?";
   first->next->next = malloc (sizeof (string_list));
   first->next->next->content = "Ein Test.";
   first->next->next->next = NULL;
   for (string_list *p = first; p; p = p->next)
      printf ("%s\n", p->content);
}
```

Aufgabe

Verkettete Liste

Schreiben Sie eine Bibliothek
zur Nutzung einer verketteten Liste
für selbstgewählte Daten
(z. B. Integer, String)