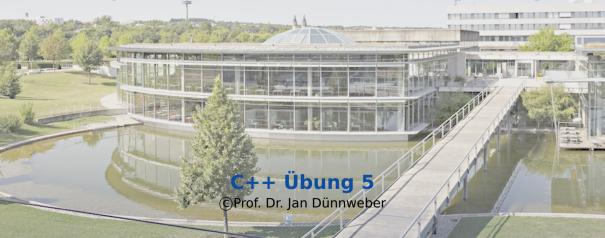


Übungen zu Programmieren 2

Ostbayerische Technische Hochschule Regensburg





 Ein Queue dient zur dient zur Zwischenspeicherung von Objekten nach dem Warteschlangenprinzip

Array-basierte Implementierung der Datenstruktur in ANSI C

```
#define MAX 100
int queue[MAX + 1], head, tail, size;
void enqueue(int value) {
  if (size == MAX) {
    printf("queue_overflow.\n");
    exit(1); }
  queue[tail] = value; tail = (tail + 1) % MAX; size++; }
/* Fortsetzung auf der naechsten Folie ... */
```



enqueue/dequeue sind FIFO-Operationen auf dem limitierten Puffer

```
/* ... weitere Operationen: */
void queueInit(void) { head = tail = size = 0; }
int dequeue(void) { if(queueEmpty()) {
    printf("attempt_to_remove_element_from_empty_queue\n");
    exit(1); } size--; int element = queue[head];
    head = (head + 1) % max; return element; }
int queueEmpty() { return size == 0; }
```

- Portieren Sie den C-Code in eine C++ Klasse
 - Ergänzen Sie Konstruktor & Destruktor

 - Implementieren Sie das unäre Minus (−) als Operator für die Entnahme von Elementen, d.h. e = -q = e = q.dequeue()



Eine Variante von BubbleSort

```
#include <vector>
int main() {
 constexpr int max = 100; queue myQueue { max };
 mvOueue + 4 + 7 + 1 + 1 + 4 + 2;
 int limit = myQueue.elements( );
 for (int i = 0; i < limit; ++i)
   int a = -mvOueue;
   for (int i = 0: i < limit - 1: ++i) {
     int b = -mvOueue:
     if (a < b) mv0ueue + b:
     else mvOueue + a, a = b: }
   myQueue + a; }
 print(myQueue); return 0; }:
```

Ergänzen Sie die member-Fkt. elements() und die friend-Fkt. print