Dr. Matthias Grabert
Johannes Mayer
Abteilung Angewandte Informationsverarbeitung

24. Oktober 2002
Blatt 2

Objektorientierte Softwareentwicklung mit C++ (WS 2002/2003)

Abgabetermin: 31. Oktober 2002

1 Dynamische Arrays (10 Punkte)

Heute haben Sie die Wahl. Na, sagen Sie schon, was gefällt Ihnen bei C nicht so? Was stört Sie denn so an C? Stimmt, Sie haben Recht, was Arrays angeht da ist C wirklich arm dran im Vergleich zu Oberon. Es wäre doch wirklich schön, denn man Arrays hätte, die Ihre Größe zur Laufzeit ändern können und die man zudem auch gleich als Stack und Queue verwenden kann wie in Perl. Da haben Sie vollkommen Recht. Und was fällt Ihnen sonst noch so an C auf? Klar, da hätt ich auch selber draufkommen können. Die Fehlerbehandlung in C ist auch nicht gerade erste Sahne. Ja wo Sie das so erwähnen, da fällt mir doch eine schöne Aufgabe für Sie ein: Schreiben Sie doch solche Arrays, deren Größe dynamisch ist. Natürlich machen Sie das mit einer vernünftigen Fehlerbehandlung durch Ausnahmen (engl. exceptions).

Also nochmal genau: Sie bekommen die Datei *main.cc* und sollen passend dazu Dateien `vector.cc`, `vector.cc` und `Makefile` schreiben. Das Makefile erzeugt eine ausführbare Datei `main`. In der Header-Datei dürfen nur die Deklarationen stehen. Die Datei `vector.cc` enthält schließlich die Implementierungen. Durch die Verwendung in der Datei *main.cc* ist schon vorgegeben, welche Funktionen zu implementieren sind und wie die Parameter und der Rückgabewert jeweils aussieht.

Beschreibung der einzelnen Funktionen, die im Namespace `vec` zu finden sein sollen:

create Neues leeres Array anlegen (noch ohne Element)

destroy Array wieder freigeben

size Anzahl der Elemente des Arrays

enlarge Vergrößert das Array auf die angegebene Größe und füllt dabei mit dem angegebenen Wert auf

get Element beim angegebenen Index auslesen

set Element beim angegebenen Index speichern (Gibt es diesen Index noch gar nicht, so wird das Array entsprechend vergrößert – dabei mit dem optionalen Füllwert aufgefüllt – und das Element geschrieben)

first Erstes Element (bei Index 0) liefern

last Letztes Element liefern

remove Element an dem angegebenen Index löschen

insert Element bei dem angegebenen Index einfügen (Element bei dem Index und die folgenden wanderen eine Position weiter)

append Element ans Ende anhängen

print Ausgabe des Vectors (via `cout`!)

Ein vector-Pointer, der NULL-wertig ist, und ein Index außerhalb des erlaubten Bereiches sollen jeweils bei allen Funktionen mit einer Exception quittiert werden.

Ihre Implementierung darf nicht auf der STL (oder sonstigen derartigen Bibliotheken basieren)!

Tipp: Die C-Standard-Funktionen `malloc`, `realloc` und `memmove` könnten sich als nützlich erweisen.