Umeå University
Institution för Datavetenskap

Datavetenskapens byggstenar 7.5 p
DV160HT15

OU1 Testing

Submitted 2015-11-17
Author: Lorenz Gerber (`dv15lgr@cs.umu.se`)
Instructor: Lena Kallin Westin / Johan Eliasson

Contents

1 Introduction

The aim of this laboration was to write unit test code for an implementation of the data type ‘queue’. The unit tests shall return feedback to the user, indicating ‘success’ or ‘failure’. The interface of the datatype may not be changed. All operations given in [1, p. 155] shall be tested.

2 Material and Methods

The datatype ‘queue’ was said to be implemented according to the specifications given in [1, pp.155 – 172]. The implementation was in ‘C’ according to standard ‘C99’. I chose to implement the unit tests according to the axiomatic specification given in [1, pp.156 + 157]. The toolchain was clang, CMake 3.3.2, GDB 7.8, IDE was CLion 1.2.1. Valgrind was used to find memory leaks. Development platform was OSX 10.10.

3 Results

3.1 General

The source code was stored in a file `queuetest.c` and submitted according instructions on the cambro web interface. The unit tests were all implemented as void functions taking no arguments. All function output goes to std out. Each functions outputs a short description of the tested operation. Depending on success, ‘pass’ or ‘fail’ is writen out. In case of ‘fail’, the function exits and returns the value `1`. All test functions are called from `main.c`.

3.2 Tests

1. Axiome 1, use `Empty` to create an empty list and check with `Isempty` if it is really emtpy.
2. Axiome 2, apply `Enqueue` to an empty list and check that `Isempty` returns `FALSE`.
3. Axiome 3, if a queue q is empty, it follows that consequtive `Enqueue`, `Dequeue` will result in the same queue q.
4. Axiome 4, if a queue q is not empty, it follows that `Dequeue` and `Enqueue` follow commutative properties, hence the resulting queue q will look the same independent in which sequence `Dequeue` and `Enqueue` are applied.
5. Axiome 5, if a queue q is empty, sequential application of `Enqueue(v, q)` and `Front(q)` will return `v`.
6. Axiome 6, if a queue q is not empty, `Enqueue` will not affect the next `Front` operation.

4 Discussion

4.1 Dynamic memory handling in C

The given implementation of queues with 2-cell uses dynamic memory allocation in C. Memory handling functions are implemented. Therefore, when comparing the outcome of two independent queues, two sets of data have to be prepared. Operations such as `Dequeue` will deallocate dynamic memory hence the data will not be available for the other queue operation. This was the case for Axiome 4 and 6.

4.2 Application of unit tests to provided datatype

The implemented unit tests where applied to the provided datatype queue (implemented with a 2-cell). The unit tests were all passed.

References

[1] L.E. Janlert and T. Wiberg. *Datatyper och algoritmer*. Studentlitteratur, 2000.