btest.cpp - Ein automatisiertes Testprogramm
Anders als in simpletest laufen die Tests in btest automatisiert ab, gesteuert durch compiler macros und command line options. Wie in simpletest gilt: Wenn btest keine Fehler findet, kann die Implementierung korrekt sein. Wenn btest Fehler findet, dann ist die Implementierung sicher nicht korrekt.

compiler macros (Auszug)

PH1 nur die Funktionalität für die erste Projektphase wird getestet
PH2 zusätzliche Tests für die zweite Projektphase werden aktiviert
SIZE=$value Wert für den zweiten Templateparameter (N), andernfalls wird der Vorgabewert verwendet
BST Schaltet die geordneten Stresstests ab, nur für den Binären Suchbaum gedacht

command line options

```
$ ./btest -h
usage: ./btest opts
where opt in opts is one of the following:

  -n $value ... number of values for first test, default: 10
  -m $value ... stepsize for n, default: 10
  -o $value ... maximum value for n, default: 100
  -v $value ... maximum value for first test, default: 10
  -w $value ... stepsize for v, default: 10
  -x $value ... maxmimum value for v, default: 100
  -s $value ... first seed, default: 666
  -t $value ... number of seeds (will be drawn from rng w/ previous seed), default: 1
               will do the full test suite t times!
  -b        ... only do benchmark
  -B        ... don't do benchmark
  -h        ... this message
```

btest is a program that will try to find a simple way to mess an ADS_set up. it should theoretically touch upon the full functionality, but is of course not capable of finding all possible error cases.

Der Test erzeugt viel (nützlichen) output. Es empfiehlt sich daher, diesen in einer Datei zu speichern.

Beispiel (1. Projektphase, Standardwert für N, Output wird in log.txt geschrieben)

```
$ \g++ -Wall -Wextra -Og -g -std=c++17 -pedantic-errors -pthread -DPH1 btest.cpp -o btest
$ ./btest 2> log.txt
```

OK

Beispiel (2. Projektphase, N=1, Output wird in log.txt geschrieben)

```
$ \g++ -Wall -Wextra -Og -g -std=c++17 -pedantic-errors -pthread -DPH2 -DSIZE=1 btest.cpp -o btest
$ ./btest 2> log.txt
```

OK

Kompilieren Sie jedenfalls OHNE -lmcheck.

Bei Abstürzen oder unerklärlichem Verhalten zahlt es sich oft aus, btest in Valgrind auszuführen. Valgrind (besser gesagt Memcheck) kontrolliert alle Speicherzugriffe und kann in 99% Prozent der Fälle eine gute Diagnostik zu der bestehenden Problematik stellen.

```
$ \g++ -Wall -Wextra -Og -g -std=c++17 -pedantic-errors -pthread btest.cpp -o btest
$ valgrind ./btest
... output omitted
```

Beachten Sie dass aufgrund der Instrumentation die Valgrind vornimmt Ihr Programm innerhalb von Valgrind viel langsamer läuft und möglicherweise den Stresstest in btest nicht mehr besteht, was dann zu einem Abort führt. Da der Stresstest der aller letzte Test ist, ist das nicht weiter dramatisch. Der Stresstest kann mit der Option -B deaktiviert werden (siehe oben).

btest kompiliert nicht auf Windows, funktioniert aber eventuell unter Cygwin oder Mingw sofern diese isatty und getopt zur Verfügung stellen. Es sollte auch auf Windows' Linux Subsystem funktionieren.