Лабораторная работа № 2 по курсу дискретного анализа: словарь

Выполнил студент группы 08-207 МАИ *Дегтярев Денис Андреевич.*

Условие

Для реализации словаря из предыдущей лабораторной работы, необходимо провести исследование скорости выполнения и потребления оперативной памяти. В случае выявления ошибок или явных недочётов, требуется их исправить.

Дневник выполнения работы

Для проверки скорости работы функций использовал gprof, утечки памяти находил с помощью valgrind. Две эти утилиты запускал на 2 тестах разных размеров: 5(только Insert) и 10^4(все операции). gprof на двух тестах сразу же показал высокую скорость(время выполнения любой функции было принебрежимо мало), поэтому не оптимизировал код на скорость после отправки кода. Для качественной проверки на valgrind отключил оптимизаторы в начале main. При отправке в первый раз valgrind ругается на неизвестный key внутри класса TNode. В деструкторе я совершаю delete[] key, а key неицилизирован, пишет valgrind... Понял, что его нужно удалять, тк это new char[], но удалять не дают, тк не знают, что это. Решил данную проблему инициализировав данный TNode* как nullptr и при каждом добавлении nullptr приравнивался к new TNode, те мы не создавали безполезные TNode.

P.S. код оптимизировал только после "OK"отправки, тк о 3 лабе узнал замного позже отправки...

Первый запуск программы с помощью valgrind

```
==29454==
==29454== HEAP SUMMARY:
==29454==     in use at exit: 435,456 bytes in 9,072 blocks
==29454==   total heap usage: 18,148 allocs, 9,076 frees, 740,128 bytes allocated
==29454==
==29454== 435,456 bytes in 9,072 blocks are definitely lost in loss record 1 of 1
==29454==    at 0x4E05833: operator new(unsigned long) (vg_replace_malloc.c:434)
==29454==    by 0x10B78C: main (main.cpp:396)
==29454==
==29454== LEAK SUMMARY:
==29454==    definitely lost: 435,456 bytes in 9,072 blocks
==29454==    indirectly lost: 0 bytes in 0 blocks
==29454==      possibly lost: 0 bytes in 0 blocks
==29454==    still reachable: 0 bytes in 0 blocks
==29454==         suppressed: 0 bytes in 0 blocks
==29454==
==29454== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
denis@den850-CJK:~/code/c++/diskran/l3/lab$
```

Итоговый запуск программы с помощью valgrind

```
denis@den850-CJK:~/code/c++/diskran/l3/lab$ g++ -Wall -ggdb3 main.cpp -o lab
denis@den850-CJK:~/code/c++/diskran/l3/lab$ valgrind -s --leak-check=full --show-leak-kinds=all --track-origins=yes ./lab < test1.txt
==24633== Memcheck, a memory error detector
==24633== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==24633== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==24633== Command: ./lab
==24633==
OK
OK
OK
OK
OK
==24633==
==24633== HEAP SUMMARY:
==24633==     in use at exit: 0 bytes in 0 blocks
==24633==   total heap usage: 14 allocs, 14 frees, 78,122 bytes allocated
==24633==
==24633== All heap blocks were freed -- no leaks are possible
==24633==
==24633== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

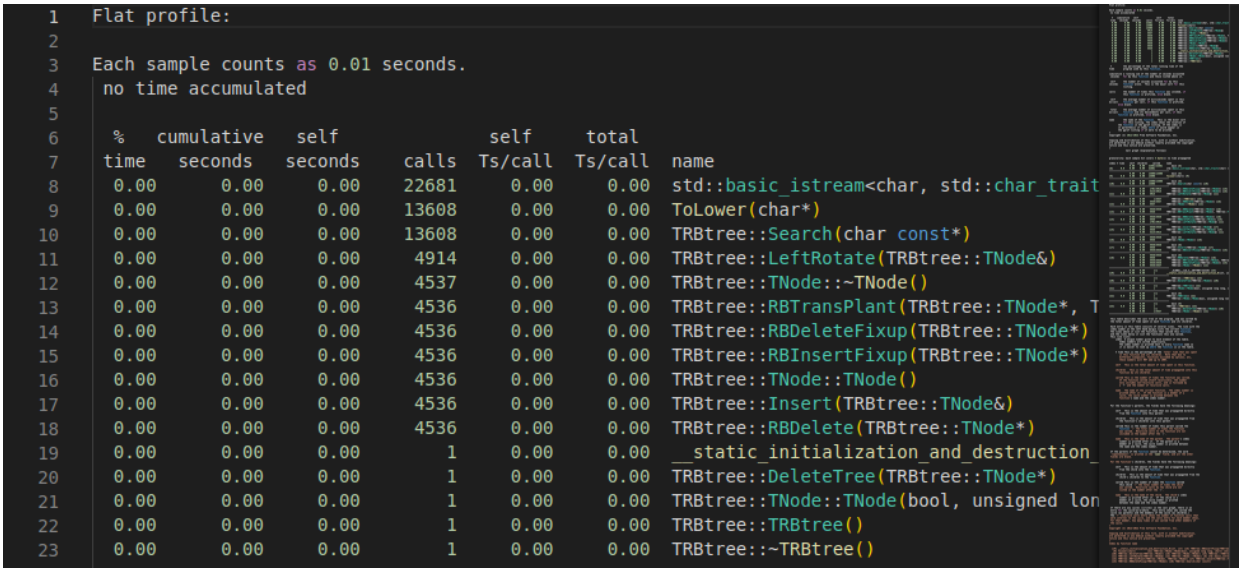
```
==24894== HEAP SUMMARY:
==24894==     in use at exit: 0 bytes in 0 blocks
==24894==   total heap usage: 9,076 allocs, 9,076 frees, 304,672 bytes allocated
==24894==
==24894== All heap blocks were freed -- no leaks are possible
==24894==
==24894== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Итоговый запуск программы с помощью gprof

```
Flat profile:

Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
  0.00      0.00     0.00       11     0.00     0.00  std::basic_istream<char, std::char_trait
  0.00      0.00     0.00        6     0.00     0.00  TRBtree::TNode::~TNode()
  0.00      0.00     0.00        5     0.00     0.00  ToLower(char*)
  0.00      0.00     0.00        5     0.00     0.00  TRBtree::RBInsertFixup(TRBtree::TNode*)
  0.00      0.00     0.00        5     0.00     0.00  TRBtree::TNode::TNode()
  0.00      0.00     0.00        5     0.00     0.00  TRBtree::Insert(TRBtree::TNode&)
  0.00      0.00     0.00        5     0.00     0.00  TRBtree::Search(char const*)
  0.00      0.00     0.00        2     0.00     0.00  TRBtree::LeftRotate(TRBtree::TNode&)
  0.00      0.00     0.00        1     0.00     0.00  __static_initialization_and_destruction_
  0.00      0.00     0.00        1     0.00     0.00  TRBtree::DeleteTree(TRBtree::TNode*)
  0.00      0.00     0.00        1     0.00     0.00  TRBtree::TNode::TNode(bool, unsigned lon
  0.00      0.00     0.00        1     0.00     0.00  TRBtree::TRBtree()
  0.00      0.00     0.00        1     0.00     0.00  TRBtree::~TRBtree()
```



```
Flat profile:

Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
  0.00      0.00     0.00    22681     0.00     0.00  std::basic_istream<char, std::char_trait
  0.00      0.00     0.00    13608     0.00     0.00  ToLower(char*)
  0.00      0.00     0.00    13608     0.00     0.00  TRBtree::Search(char const*)
  0.00      0.00     0.00     4914     0.00     0.00  TRBtree::LeftRotate(TRBtree::TNode&)
  0.00      0.00     0.00     4537     0.00     0.00  TRBtree::TNode::~TNode()
  0.00      0.00     0.00     4536     0.00     0.00  TRBtree::RBTransPlant(TRBtree::TNode*, T
  0.00      0.00     0.00     4536     0.00     0.00  TRBtree::RBDeleteFixup(TRBtree::TNode*)
  0.00      0.00     0.00     4536     0.00     0.00  TRBtree::RBInsertFixup(TRBtree::TNode*)
  0.00      0.00     0.00     4536     0.00     0.00  TRBtree::TNode::TNode()
  0.00      0.00     0.00     4536     0.00     0.00  TRBtree::Insert(TRBtree::TNode&)
  0.00      0.00     0.00     4536     0.00     0.00  TRBtree::RBDelete(TRBtree::TNode*)
  0.00      0.00     0.00        1     0.00     0.00  __static_initialization_and_destruction_
  0.00      0.00     0.00        1     0.00     0.00  TRBtree::DeleteTree(TRBtree::TNode*)
  0.00      0.00     0.00        1     0.00     0.00  TRBtree::TNode::TNode(bool, unsigned lon
  0.00      0.00     0.00        1     0.00     0.00  TRBtree::TRBtree()
  0.00      0.00     0.00        1     0.00     0.00  TRBtree::~TRBtree()
```

Выводы

Существуют множество утилит, которые помогают нам оптимизировать код программы и найти различные утечки в ходе выполнения. Для проверки моего кода на надежность и скорость помогли gprof и valgrind. Эти мощные утилиты не только сказали мне, что у меня где-то есть проблемы, но и указали где(удобно, когда тебе пишут в какой строчке у тебя ошибка и почему или когда ты знаешь в какой функции беда...). Данные инструменты очень полезны и ими нельзя принебреать при работе с указателями.