Московский авиационный институт (национальный исследовательский университет)

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

Лабораторная работа по предмету "Дискретный анализ" №3

Студент: Шипилов К. Ю.

Преподаватель: Макаров Н. К.

Группа: М8О-203Б-22

Дата: 18.05.2024

Оценка:

Подпись:

Москва 2024 г.

Оглавление

Цель работы

Приобретение практических навыков в исследовании качества программ.

Постановка задачи

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.

Результатом лабораторной работы является отчёт, состоящий из:

- Дневника выполнения работы, в котором отражено что и когда делалось, какие средства использовались и какие результаты были достигнуты на каждом шаге выполнения лабораторной работы.
- Выводов о найденных недочётах.
- Сравнение работы исправленной программы с предыдущей версией.
- Общих выводов о выполнении лабораторной работы, полученном опыте.

Минимальный набор используемых средств должен содержать утилиту ***gprof*** и библиотеку ***dmalloc***, однако их можно заменять на любые другие аналогичные или более развитые утилиты (например, Valgrind или Shark) или добавлять к ним новые (например, gcov).

Дневник выполнения работы

Для поиска и устранения ошибок работы с памятью в работе использовалась утилита **Valgrind**. Для проверки программы необходимо ввести команду **valgrind --leak-check=full --show-reachable=yes --track-origins=yes ./main,** где

- **main** - скомпилированный файл, который находится в текущей директории;
- **--leak-check=full —** флаг, указывающий **Valgrind**, что нужно выполнить полную проверку утечек и вывести информацию об их местонахождении в коде программы;

- **--show-reachable=yes —** флаг, указывающий Valgrind, что нужно вывести подробную информацию об участках памяти, которые доступны при завершении программы, но не освобождены;
- **--track-origins=yes** - флаг, указывающий Valgrind, что нужно определить происхождение ошибок в работе с памятью.

При выполнении этой команды **Valgrind** выдаст подробную информацию об ошибках и утечках памяти в программе.

При проверке своей программы на сгенерированных данных я получил следующий результат:

```
❯ valgrind --leak-check=full --show-reachable=yes --track-origins=yes
./main_copy
==7943== Memcheck, a memory error detector
==7943== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==7943== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==7943== Command: ./main_copy
==7943==
OK
...
OK: 402627350
==7943== Invalid read of size 4
==7943==    at 0x10B43D: RedBlackTree::eraseFixup(Node*) (main-copy.cpp:149)
==7943==    by 0x10BCF2: RedBlackTree::erase(Node*) (main-copy.cpp:358)
==7943==    by 0x10C46C: Dictionary::erase(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&) (main-copy.cpp:454)
==7943==    by 0x10A9EC: main (main-copy.cpp:502)
==7943==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==7943==
==7943==
==7943== Process terminating with default action of signal 11 (SIGSEGV)
==7943==  Access not within mapped region at address 0x0
==7943==    at 0x10B43D: RedBlackTree::eraseFixup(Node*) (main-copy.cpp:149)
```

```
==7943==    by 0x10BCF2: RedBlackTree::erase(Node*) (main-copy.cpp:358)
==7943==    by 0x10C46C: Dictionary::erase(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&) (main-copy.cpp:454)
==7943==    by 0x10A9EC: main (main-copy.cpp:502)
==7943==  If you believe this happened as a result of a stack
==7943==  overflow in your program's main thread (unlikely but
==7943==  possible), you can try to increase the size of the
==7943==  main thread stack using the --main-stacksize= flag.
==7943==  The main thread stack size used in this run was 8388608.
==7943==
==7943== HEAP SUMMARY:
==7943==     in use at exit: 86,873 bytes in 80 blocks
==7943==   total heap usage: 375 allocs, 295 frees, 128,303 bytes allocated
==7943==
==7943== 72 bytes in 1 blocks are still reachable in loss record 1 of 8
==7943==    at 0x4846FA3: operator new(unsigned long) (in
/usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==7943==    by 0x10B774: RedBlackTree::RedBlackTree() (main-copy.cpp:232)
==7943==    by 0x10C95F: Dictionary::Dictionary() (main-copy.cpp:424)
==7943==    by 0x10A90A: main (main-copy.cpp:491)
==7943==
==7943== 481 bytes in 1 blocks are still reachable in loss record 2 of 8
==7943==    at 0x4846FA3: operator new(unsigned long) (in
/usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==7943==    by 0x49D1EFB: std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >::_M_mutate(unsigned long, unsigned
long, char const*, unsigned long) (in /usr/lib/x86_64-linux-gnu/libstdc+
+.so.6.0.33)
==7943==    by 0x49439CB: std::basic_istream<char, std::char_traits<char> >&
std::operator>><char, std::char_traits<char>, std::allocator<char>
>(std::basic_istream<char, std::char_traits<char> >&,
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&)
(in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.33)
==7943==    by 0x10AAE9: main (main-copy.cpp:493)
```

```
==7943==
==7943== 481 bytes in 1 blocks are still reachable in loss record 3 of 8
==7943==    at 0x4846FA3: operator new(unsigned long) (in
/usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==7943==    by 0x49D1EFB: std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >::_M_mutate(unsigned long, unsigned
long, char const*, unsigned long) (in /usr/lib/x86_64-linux-gnu/libstdc+
+.so.6.0.33)
==7943==    by 0x49439CB: std::basic_istream<char, std::char_traits<char> >&
std::operator>><char, std::char_traits<char>, std::allocator<char>
>(std::basic_istream<char, std::char_traits<char> >&,
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&)
(in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.33)
==7943==    by 0x10A9D6: main (main-copy.cpp:501)
==7943==
==7943== 1,024 bytes in 1 blocks are still reachable in loss record 4 of 8
==7943==    at 0x4846828: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==7943==    by 0x4B9E1A4: _IO_file_doallocate (filedoalloc.c:101)
==7943==    by 0x4BAE513: _IO_doallocbuf (genops.c:347)
==7943==    by 0x4BABF7F: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:745)
==7943==    by 0x4BACA9E: _IO_new_file_xsputn (fileops.c:1244)
==7943==    by 0x4BACA9E: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1197)
==7943==    by 0x4B9FA01: fwrite (iofwrite.c:39)
==7943==    by 0x49C0CA4: std::basic_ostream<char, std::char_traits<char> >&
std::__ostream_insert<char, std::char_traits<char> >(std::basic_ostream<char,
std::char_traits<char> >&, char const*, long) (in
/usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.33)
==7943==    by 0x49C102A: std::basic_ostream<char, std::char_traits<char> >&
std::operator<< <std::char_traits<char> >(std::basic_ostream<char,
std::char_traits<char> >&, char const*) (in /usr/lib/x86_64-linux-gnu/libstdc+
+.so.6.0.33)
```

```
==7943==    by 0x10C0A4: Dictionary::insert(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&, unsigned long const&)
(main-copy.cpp:429)
==7943==    by 0x10A989: main (main-copy.cpp:498)
==7943==
==7943== 2,736 bytes in 38 blocks are still reachable in loss record 5 of 8
==7943==    at 0x4846FA3: operator new(unsigned long) (in
/usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==7943==    by 0x10B9FE: RedBlackTree::insert(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&, unsigned long const&)
(main-copy.cpp:295)
==7943==    by 0x10C07F: Dictionary::insert(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&, unsigned long const&)
(main-copy.cpp:428)
==7943==    by 0x10A989: main (main-copy.cpp:498)
==7943==
==7943== 4,096 bytes in 1 blocks are still reachable in loss record 6 of 8
==7943==    at 0x4846828: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==7943==    by 0x4B9E1A4: _IO_file_doallocate (filedoalloc.c:101)
==7943==    by 0x4BAE513: _IO_doallocbuf (genops.c:347)
==7943==    by 0x4BAB873: _IO_file_underflow@@GLIBC_2.2.5 (fileops.c:486)
==7943==    by 0x4BAE5C1: _IO_default_uflow (genops.c:362)
==7943==    by 0x49937E0: __gnu_cxx::stdio_sync_filebuf<char,
std::char_traits<char> >::underflow() (in /usr/lib/x86_64-linux-gnu/libstdc+
+.so.6.0.33)
==7943==    by 0x49A1A1E: std::istream::sentry::sentry(std::istream&, bool)
(in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.33)
==7943==    by 0x4943764: std::basic_istream<char, std::char_traits<char> >&
std::operator>><char, std::char_traits<char>, std::allocator<char>
>(std::basic_istream<char, std::char_traits<char> >&,
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&)
(in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.33)
==7943==    by 0x10AAE9: main (main-copy.cpp:493)
```

```
==7943==
==7943== 4,255 bytes in 36 blocks are still reachable in loss record 7 of 8
==7943==    at 0x4846FA3: operator new(unsigned long) (in
/usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==7943==    by 0x49D4EF9: void std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >::_M_construct<char*>(char*, char*,
std::forward_iterator_tag) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.33)
==7943==    by 0x10ADBC: Node::Node(colorType, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&, unsigned long const&,
Node*, Node*, Node*) (main-copy.cpp:27)
==7943==    by 0x10BA39: RedBlackTree::insert(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&, unsigned long const&)
(main-copy.cpp:295)
==7943==    by 0x10C07F: Dictionary::insert(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&, unsigned long const&)
(main-copy.cpp:428)
==7943==    by 0x10A989: main (main-copy.cpp:498)
==7943==
==7943== 73,728 bytes in 1 blocks are still reachable in loss record 8 of 8
==7943==    at 0x4846828: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==7943==    by 0x4926401: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.33)
==7943==    by 0x400571E: call_init.part.0 (dl-init.c:74)
==7943==    by 0x4005823: call_init (dl-init.c:120)
==7943==    by 0x4005823: _dl_init (dl-init.c:121)
==7943==    by 0x401F59F: ??? (in /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2)
==7943==
==7943== LEAK SUMMARY:
==7943==    definitely lost: 0 bytes in 0 blocks
==7943==    indirectly lost: 0 bytes in 0 blocks
==7943==      possibly lost: 0 bytes in 0 blocks
==7943==    still reachable: 86,873 bytes in 80 blocks
==7943==         suppressed: 0 bytes in 0 blocks
==7943==
```

```
==7943== For lists of detected and suppressed errors, rerun with: -s
==7943== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
[1]    7943 segmentation fault (core dumped)  valgrind --leak-check=full --show-reachable=yes --track-origins=yes
```

Вывод свидетельствует о том, что на тестовых данных в функции **eraseFixup** (149 строка файла **main.cpp**) происходит ошибка **Invalid read of size 4**, которая приводит к **segmentation fault** и экстренному завершению программы.
Если ввести команду **valgrind --vgdb=yes --vgdb-error=1 ./main,** после этого запустить **gdb ./main** и ввести **target remote | vgdb**, можно выполнить отладку программы с того момента, в котором обнаружена ошибка. Это может быть полезно, так как **Valgrind** выводит только строчку, в которой происходит ошибка, а в **gdb** можно проверить значения всех переменных и понять, что именно приводит к ошибке.
В процесе отладки было выявлено, что ошибка была вызвана неправильной работой функции **erase**, которая передавала в функцию **eraseFixup** неправильный указатель на узел дерева, который содержал **nullptr**.
После устранения ошибок вывод Valgrind сообщает о том, что вся выделенная память освобождается, утечек и ошибок нет.

```
❯ valgrind --leak-check=full --show-reachable=yes --track-origins=yes ./lw2_exe
==12624== Memcheck, a memory error detector
==12624== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==12624== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==12624== Command: ./lw2_exe
==12624==
OK
...
Exist
==12624==
==12624== HEAP SUMMARY:
```

```
==12624==     in use at exit: 0 bytes in 0 blocks
==12624==   total heap usage: 444 allocs, 444 frees, 145,906 bytes allocated
==12624==
==12624== All heap blocks were freed -- no leaks are possible
==12624==
==12624== For lists of detected and suppressed errors, rerun with: -s
==12624== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Для анализа производительности в работе применялась утилита **Gprof**. При компиляции необходимо указать флаг **-pg**. После компиляции необходимо запустить программу, чтобы вся информация для профилирования сохранилась в файл **gmon.out**. Далее, выполнив команду **gprof main > profile.log**, можно получить файл, содержащий информацию о времени, затраченном на каждую функцию.

Первые строки таблицы, содержащей функции, упорядоченные по убыванию времени их работы:

```
% cumulative self self total
time seconds seconds calls ms/call ms/call name
34.51 0.39 0.39 94237832 0.00 0.00 std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >::size() const
16.81 0.58 0.19 127844256 0.00 0.00 std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >::operator[](unsigned long)
6.19 0.65 0.07 156036860 0.00 0.00 std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >::_M_data() const
5.31 0.71 0.06 499998 0.00 0.00 toLowerCase(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >&)
4.87 0.77 0.06 11876362 0.00 0.00 unsigned long const& std::min<unsigned
long>(unsigned long const&, unsigned long const&)
4.42 0.81 0.05 11876362 0.00 0.00 std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>
>::compare(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&) const
3.54 0.85 0.04 144171113 0.00 0.00 std::__is_constant_evaluated()
3.54 0.90 0.04 299685 0.00 0.00
RedBlackTree::insert(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&, unsigned long const&)
```

Больше всего времени занимают методы **string** и функция **toLowerCase**, так как они используются при каждом вводе команды для перевода ключей в нижний регистр и также могут применяться внутри других функций.
Из функций обработки дерева больше всего времени занимает функция **insert**, что может быть связано с тем, что в ней два раза выполняется поиск по дереву: сначала, чтобы узнать существует ли узел, который необходимо вставить, а после этого, если узел не существует, нужно найти ему родителя. Скорость работы **insert** можно уменьшить, если написать специальную функцию, которая будет выполнять поиск так, что если узел не существует, возвращается не **nullNode**, как в обычном поиске, а потенциальный родитель этого узла. Так как обычный поиск более универсальный и даже с ним программа смогла завершить работу на тестах за выделенное время, я решил не переделывать функцию **insert**.

Вывод

В ходе работы я исследовал производительность своей программы, а также проверил ее на наличие ошибок и утечек памяти. Инструменты которыми я воспользовался являются очень полезными, поэтому они используются мной регулярно при написании программ. **Valgrind** следует всегда использовать при ручном выделении памяти или при возникновении ошибок, таких как **segmentation fault**, так как с его помощью можно точно установить причину и место возникновения ошибок и утечек. **Gprof** позволяет на основе времени работы функций выявить узкое место в программе, которое больше всего нуждается в доработке.