Задание 1

Скомпилируйте программу из методических указаний с отладочной информацией и без нее. Приведите выдачу утилиты valgrind о любой из ошибок с отладочной информацией и без нее.

Выдача с отладочной информацией

```
==19618== Invalid write of size 4
==19618==    at 0x10916B: f (example.c:6)
==19618==    by 0x109180: main (example.c:11)
==19618==  Address 0x4a98068 is 0 bytes after a block of size 40 alloc'd
==19618==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==19618==    by 0x10915E: f (example.c:5)
==19618==    by 0x109180: main (example.c:11)
```

Выдача без отладочной информации

```
==19587== Invalid write of size 4
==19587==    at 0x10916B: f (in
/home/daniil/work/C_labs/Valgrind_task/src/app.exe)
==19587==    by 0x109180: main (in
/home/daniil/work/C_labs/Valgrind_task/src/app.exe)
==19587==  Address 0x4a98068 is 0 bytes after a block of size 40 alloc'd
==19587==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==19587==    by 0x10915E: f (in
/home/daniil/work/C_labs/Valgrind_task/src/app.exe)
==19587==    by 0x109180: main (in
/home/daniil/work/C_labs/Valgrind_task/src/app.exe)
```

Объясните, чем отличается выдача утилиты valgrind в этих случаях.

Как можно заметить по двум листингам работы программы valgrind, при дебажной сборке valgrind находит ошибку в файле самой программы, а при релизной сборке — в исполняемом файле. Соответственно использовать релизную сборку при обработке программы с помощью valgrind нерационально, потому что она не позволяет точно локализовать ошибку в написанном коде. В дальнейшем будет использоваться именно дебажная версия сборки.

Задание 2

Проанализируйте текст программы task_02.c.

Какая ошибка допущена в этой программе?

Если now != 0, то в функции foo возникнет ошибка, связанная с использованием неинициализированных данных

Скомпилируйте программу task_02.c. Запустите программу под управлением утилиты valgrind. Приведите выдачу утилиты valgrind.

```
==19814== Conditional jump or move depends on uninitialised value(s)
==19814==    at 0x10917C: foo (task_02.c:6)
==19814==    by 0x1091C1: main (task_02.c:17)
```

О какой ошибке сообщает утилита valgrind? Что за номер строки приведен в выдаче утилиты?

Valgrind сообщает, что в 6 строке в функции foo возможно сравнение с неинициализированными данными

Подтвердил ли вывод утилиты Ваш ответ?

Вывод утилиты подтвердил мой ответ

Какой ключ утилиты valgrind можно использовать для прояснения ситуации? Помогает ли он в данной ситуации?

Для прояснения ситуации можно использовать ключ —track-origins=yes, но в этом примере он не помогает, так как говорит, что неинициализированные данные появляются в 11 строке, в которой ничего не происходит

Задание 3

Проанализируйте текст программ task_03_1.c и task_03_2.c.

В представленных программах содержится ошибка выхода за границу массива

Скомпилируйте программы. Запустите полученные исполняемые файлы сначала без использования утилиты valgrind, а потом под управлением утилиты valgrind.

В чем заключается разница запуска программ без утилиты valgrind?

Первая программа аварийно завершается в отличии от второй

Помогает ли утилита valgrind в поиске ошибок в программе task_03_1.c? Почему? (Подсказку можно найти в разделе 5 Quick Start Guide.)

Не помогает, так как valgrind не обнаруживает ошибки выхода за границы статического массива

Помогает ли утилита valgrind в поиске ошибок в программе task_03_3.c? Приведите выдачу для каждой из ошибок.

Помогает. Утилита говорит, что не удалось считать элемент, вышедший за границу массива, его размер 0 байт, то есть память под него не выделена

Ошибка 1

```
 ==20253== Process terminating with default action of signal 6 (SIGABRT)
==20253==    at 0x4903A7C: __pthread_kill_implementation (pthread_kill.c:44)
==20253==    by 0x4903A7C: __pthread_kill_internal (pthread_kill.c:78)
==20253==    by 0x4903A7C: pthread_kill@@GLIBC_2.34 (pthread_kill.c:89)
==20253==    by 0x48AF475: raise (raise.c:26)
==20253==    by 0x48957F2: abort (abort.c:79)
==20253==    by 0x48F66F5: __libc_message (libc_fatal.c:155)
==20253==    by 0x49A3769: __fortify_fail (fortify_fail.c:26)
```

```
==20253==    by 0x49A3735: __stack_chk_fail (stack_chk_fail.c:24)
==20253==    by 0x1091C8: main (task_03_1.c:12)
```

Ошибка 2

```
 ==20764== Invalid read of size 1
==20764==    at 0x1091EC: main (task_03_2.c:17)
==20764==  Address 0x4a98045 is 0 bytes after a block of size 5 alloc'd
==20764==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==20764==    by 0x10919E: main (task_03_2.c:6)
==20764==

==20764== Invalid write of size 1
==20764==    at 0x109210: main (task_03_2.c:19)
==20764==  Address 0x4a98045 is 0 bytes after a block of size 5 alloc'd
==20764==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==20764==    by 0x10919E: main (task_03_2.c:6)
```

Задание 4

Проанализируйте текст программы task_04.c.

Какая ошибка работы с динамической памятью в ней допущена?

Освобождение памяти, которая не выделена функциями malloc, calloc, realloc, в строке 38 функцией free

К какому виду в классификации, приведенной на лекции, она относится? (Слайды 18, 19 презентации «Указатели и одномерные динамические массивы».)

Освобождение невыделенной или нединамической памяти

Скомпилируйте программу task_04.c. Запустите полученный исполняемый файл под управлением утилиты valgrind. Приведите выдачу утилиты, содержащую описание ошибки.

Замечание
Если ключ "-Werror" препятствует получению исполняемого файла, то в учебных целях в виде исключения его можно опустить.

```
==19066== Invalid write of size 4
==19066==    at 0x1091D5: create_date (task_04.c:23)
==19066==    by 0x109205: main (task_04.c:33)
==19066==  Address 0x4a98048 is 0 bytes after a block of size 8 alloc'd
==19066==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==19066==    by 0x1091AF: create_date (task_04.c:18)
==19066==    by 0x109205: main (task_04.c:33)
==19066==
==19066== Invalid read of size 4
==19066==    at 0x109215: main (task_04.c:36)
==19066==  Address 0x4a98048 is 0 bytes after a block of size 8 alloc'd
==19066==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
```

```
==19066==    by 0x1091AF: create_date (task_04.c:18)
==19066==    by 0x109205: main (task_04.c:33)
```

Подтверждает ли утилиты valgrind правильность Вашего ответа? О каких строках сообщает утилита valgrind?

Нет, не подтверждает. Valgrind ругается на строку, в которой выделяется память, а не на строку, в которой освобождается

К какой категории ошибок относит valgrind эту ошибку? (Подраздел 4.2 User Manual.)

4.2.1. Illegal read/write erros

Задание 5

Проанализируйте текст программы task_05.c.

Какая ошибка работы с динамической памятью в ней допущена?

Повторно выделяется память в строке 18

К какому виду в классификации, приведенной на лекции, она относится? (Слайды 18, 19 презентации «Указатели и одномерные динамические массивы».)

Логическая ошибка (Изменение указателя, который вернула функция выделения памяти)

Скомпилируйте программу task_05.c. Запустите полученный исполняемый файл под управлением утилиты valgrind. Приведите выдачу утилиты, содержащую описание ошибки.

Замечание
Если ключ "-Werror" препятствует получению исполняемого файла, то в учебных целях в виде исключения его можно опустить.

```
==6954== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==6954==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==6954==    by 0x1091A6: main (task_05.c:11)
```

Подтверждает ли утилиты valgrind правильность Вашего ответа? О каких строках сообщает утилита valgrind?

Подтверждает. Valgrind сообщает нам, что было потеряно 4 байта памяти в результате перезаписи указателя

К какой категории ошибок относит valgrind эту ошибку? (Подраздел 4.2 User Manual.)

4.2.9. Memory leak detection

Задание 6

Проанализируйте текст программы task_06.c.

Какая ошибка работы с динамической памятью в ней допущена?

Не очищается выделенная динамически память

К какому виду в классификации, приведенной на лекции, она относится? (Слайды 18, 19 презентации «Указатели и одномерные динамические массивы».)

Утечка памяти

Скомпилируйте программу task_06.c. Запустите полученный исполняемый файл под управлением утилиты valgrind. Приведите выдачу утилиты, содержащую описание ошибки.

Замечание
Если ключ "-Werror" препятствует получению исполняемого файла, то в учебных целях в виде исключения его можно опустить.

```
==7494== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==7494==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==7494==    by 0x109189: process (task_06.c:14)
==7494==    by 0x1091DD: main (task_06.c:31)
```

Подтверждает ли утилиты valgrind правильность Вашего ответа? О каких строках сообщает утилита valgrind?

Подтверждает. Valgrind сообщает, что было потеряно 4 байта памяти из-за отсутствия функции free

К какой категории ошибок относит valgrind эту ошибку? (Подраздел 4.2 User Manual.)

4.2.9. Memory leak detection

Задание 7

Проанализируйте текст программы task_07.c.

Какая ошибка работы с динамической памятью в ней допущена?

Ошибка сегментации при работе с непроинициализироанным указателем

К какому виду в классификации, приведенной на лекции, она относится? (Слайды 18, 19 презентации «Указатели и одномерные динамические массивы».)

Логическая ошибка. Использование непроинициализированного указателя (Wild pointer)

Скомпилируйте программу task_07.c. Запустите полученный исполняемый файл под управлением утилиты valgrind. Приведите выдачу утилиты, содержащую описание ошибки.

Замечание

Если ключ "-Werror" препятствует получению исполняемого файла, то в учебных целях в виде исключения его можно опустить.

```
==8019== Use of uninitialised value of size 8
==8019==    at 0x48D41C9: __vfscanf_internal (vfscanf-internal.c:1896)
==8019==    by 0x48CF1C1: __isoc99_scanf (isoc99_scanf.c:30)
==8019==    by 0x1091D7: main (task_07.c:14)
==8019==
==8019== Invalid write of size 4
==8019==    at 0x48D41C9: __vfscanf_internal (vfscanf-internal.c:1896)
==8019==    by 0x48CF1C1: __isoc99_scanf (isoc99_scanf.c:30)
==8019==    by 0x1091D7: main (task_07.c:14)
==8019==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
```

Подтверждает ли утилиты valgrind правильность Вашего ответа? О каких строках сообщает утилита valgrind?

Valgrind подтверждает правильность моего ответа

К какой категории ошибок относит valgrind эту ошибку? (Подраздел 4.2 User Manual.)

4.2.3. Use of uninitialised or unaddressable values in system calls

Задание 8

Проанализируйте текст программы task_08.c.

Какая ошибка работы с динамической памятью в ней допущена?

В строке 20 используется память, которую уже очистили в строке 18

К какому виду в классификации, приведенной на лекции, она относится? (Слайды 18, 19 презентации «Указатели и одномерные динамические массивы».)

Логическая ошибка. Использование указателя сразу после освобождения памяти (Dangling pointer)

Скомпилируйте программу task_08.c. Запустите полученный исполняемый файл под управлением утилиты valgrind. Приведите выдачу утилиты, содержащую описание ошибки.

Замечание
Если ключ "-Werror" препятствует получению исполняемого файла, то в учебных целях в виде исключения его можно опустить.

```
==8207== Invalid write of size 4
==8207==    at 0x1091E0: main (task_08.c:20)
==8207==  Address 0x4a98040 is 0 bytes inside a block of size 4 free'd
==8207==     at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==8207==    by 0x1091DB: main (task_08.c:18)
==8207==  Block was alloc'd at
==8207==     at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
```

```
==8207==    by 0x10919E: main (task_08.c:11)
==8207==
==8207== Invalid read of size 4
==8207==    at 0x1091EA: main (task_08.c:22)
==8207==  Address 0x4a98040 is 0 bytes inside a block of size 4 free'd
==8207==     at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==8207==    by 0x1091DB: main (task_08.c:18)
==8207==  Block was alloc'd at
==8207==     at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==8207==    by 0x10919E: main (task_08.c:11)
```

Подтверждает ли утилиты valgrind правильность Вашего ответа? О каких строках сообщает утилита valgrind?

Подтверждает. Valgrind ругается на то, что из-за очистки памяти в 18 строке нельзя изменять указатель в 20 строке и читать этот указатель в 22

К какой категории ошибок относит valgrind эту ошибку? (Подраздел 4.2 User Manual.)

4.2.1. Illegal read/write erros

Задание 9

Проанализируйте текст программы task_09.c.

Какая ошибка работы с динамической памятью в ней допущена?

В 20 строке нельзя менять объявленный динамически указатель

К какому виду в классификации, приведенной на лекции, она относится? (Слайды 18, 19 презентации «Указатели и одномерные динамические массивы».)

Логическая ошибка. Изменение указателя, который вернула функция выделения памяти

Скомпилируйте программу task_09.c. Запустите полученный исполняемый файл под управлением утилиты valgrind. Приведите выдачу утилиты, содержащую описание ошибки.

Замечание
Если ключ "-Werror" препятствует получению исполняемого файла, то в учебных целях в виде исключения его можно опустить.

```
==8425== Invalid free() / delete / delete[] / realloc()
==8425==     at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==8425==    by 0x1091D5: main (task_09.c:23)
==8425==  Address 0x4a98054 is 0 bytes after a block of size 20 alloc'd
==8425==     at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==8425==    by 0x10918C: main (task_09.c:12)
==8425==
==8425== 20 bytes in 1 blocks are definitely lost in loss record 1 of 1
```

```
==8425==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==8425==    by 0x10918C: main (task_09.c:12)
```

Подтверждает ли утилиты valgrind правильность Вашего ответа? О каких строках сообщает утилита valgrind?

Подтверждает. Valgrind ругается на то, что из-за изменения указателя происходит утечка памяти

К какой категории ошибок относит valgrind эту ошибку? (Подраздел 4.2 User Manual.)

4.2.9. Memory leak detection
4.2.5. When a heap block is freed with an inappropriate deallocation function