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

```
==7273== Memcheck, a memory error detector
==7273== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7273== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==7273== Command: ./app.exe
==7273==
==7273== Invalid write of size 4
==7273==    at 0x10916B: f (example.c:6)
==7273==    by 0x109180: main (example.c:11)
==7273==  Address 0x4aa0068 is 0 bytes after a block of size 40 alloc'd
==7273==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==7273==    by 0x10915E: f (example.c:5)
==7273==    by 0x109180: main (example.c:11)
==7273==
==7273==
==7273== HEAP SUMMARY:
==7273==     in use at exit: 40 bytes in 1 blocks
==7273==   total heap usage: 1 allocs, 0 frees, 40 bytes allocated
==7273==
==7273== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
==7273==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==7273==    by 0x10915E: f (example.c:5)
==7273==    by 0x109180: main (example.c:11)
==7273==
==7273== LEAK SUMMARY:
==7273==    definitely lost: 40 bytes in 1 blocks
==7273==    indirectly lost: 0 bytes in 0 blocks
==7273==      possibly lost: 0 bytes in 0 blocks
==7273==    still reachable: 0 bytes in 0 blocks
==7273==         suppressed: 0 bytes in 0 blocks
==7273==
==7273== For lists of detected and suppressed errors, rerun with: -s
```

Выдача без отладочной информации

```
==7309== Memcheck, a memory error detector
==7309== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7309== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==7309== Command: ./app.exe
==7309==
==7309== Invalid write of size 4
==7309==    at 0x10916B: f (in /home/george/Music/src(2)/src/app.exe)
==7309==    by 0x109180: main (in /home/george/Music/src(2)/src/app.exe)
==7309==  Address 0x4aa0068 is 0 bytes after a block of size 40 alloc'd
==7309==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==7309==    by 0x10915E: f (in /home/george/Music/src(2)/src/app.exe)
==7309==    by 0x109180: main (in /home/george/Music/src(2)/src/app.exe)
==7309==
==7309==
==7309== HEAP SUMMARY:
```

```
==7309==     in use at exit: 40 bytes in 1 blocks
==7309==   total heap usage: 1 allocs, 0 frees, 40 bytes allocated
==7309==
==7309== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
==7309==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==7309==    by 0x10915E: f (in /home/george/Music/src(2)/src/app.exe)
==7309==    by 0x109180: main (in /home/george/Music/src(2)/src/app.exe)
==7309==
==7309== LEAK SUMMARY:
==7309==    definitely lost: 40 bytes in 1 blocks
==7309==    indirectly lost: 0 bytes in 0 blocks
==7309==      possibly lost: 0 bytes in 0 blocks
==7309==    still reachable: 0 bytes in 0 blocks
==7309==         suppressed: 0 bytes in 0 blocks
==7309==
==7309== For lists of detected and suppressed errors, rerun with: -s
==7309== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

Объясните, чем отличается выдача утилиты valgrind в этих случаях.

Без отладочной информации valgrind выводит информацию об ошибках без указания, где именно произошли ошибки. А с отладочной информацией, valgrind указывает конкретный файл и строку, где произошла ошибка.

Задание 2

Проанализируйте текст программы task_02.c.

Какая ошибка допущена в этой программе?

Переменная y не проинициализирована, поскольку, если now не будет равно 0, y так и останется не проинициализированной и будет передана в функцию, где производится сравнение этой переменной с целым числом, что приведет к неопределенному поведению.

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

```
==7585== Memcheck, a memory error detector
==7585== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7585== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==7585== Command: ./app.exe
==7585==
==7585== Conditional jump or move depends on uninitialised value(s)
==7585==    at 0x10917C: foo (task_02.c:6)
==7585==    by 0x1091C1: main (task_02.c:17)
==7585==
x is less than 10
==7585==
==7585== HEAP SUMMARY:
==7585==     in use at exit: 0 bytes in 0 blocks
==7585==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==7585==
==7585== All heap blocks were freed -- no leaks are possible
==7585==
```

```
==7585== Use --track-origins=yes to see where uninitialised values come from
==7585== For lists of detected and suppressed errors, rerun with: -s
==7585== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

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

Ошибка сообщает, что в 6 строке функции foo, работа условного оператора if зависит от непроинициализированной переменной x.

Подтвердил ли вывод утилиты Ваш ответ?

Да, вывод утилиты valgrind подтвердил мой ответ.

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

При помощи ключа —track-origins=yes можно узнать о проблеме более подробно, поскольку он укажет на строку, где была объявлена непроинициализированная переменная.

Задание 3

Проанализируйте текст программ task_03_1.c и task_03_2.c.

Какие ошибки допущены в этих программах?

В обеих этих программах допущена одна и та же ошибка, чтение и запись элемента за пределами массива. В обоих случаях массив состоит из 5 элементов, а запись и чтение происходит над шестым элементом.

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?

*При запуске программы task_03_1 вылетает ошибка переполнения стека: «**** stack smashing detected ***: terminated Aborted (core dumped)». А при запуске программы task_03_2 ошибки не появляются.

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

Нет, утилита valgrind не помогает в поиске ошибок, поскольку утилита не может определять ошибки связанные с чтением или записью за пределами статического массива.

Помогает ли утилита valgrind в поиске ошибок в программе task_03_2.c? Приведите выдачу для каждой из ошибок.

Да, утилита valgrind помогла обнаружить обе ошибки: чтение и запись за пределами массива.

Ошибка 1

```
==8863== Invalid read of size 1
==8863==    at 0x1091EC: main (task_03_2.c:17)
==8863==  Address 0x4aa0045 is 0 bytes after a block of size 5 alloc'd
==8863==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==8863==    by 0x10919E: main (task_03_2.c:6)
```

Ошибка 2

```
==8863== Invalid write of size 1
==8863==    at 0x109210: main (task_03_2.c:19)
==8863==  Address 0x4aa0045 is 0 bytes after a block of size 5 alloc'd
==8863==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==8863==    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?

В функции create_date некорректно выделяется память для структуры, поскольку размер указателя на структуру возвращает размер самого указателя, а не всей структуры. При попытке заполнения или чтения будут происходить ошибки.

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

Неверный расчет количества выделяемой 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.

```
==9261== Memcheck, a memory error detector
==9261== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==9261== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==9261== Command: ./app.exe
==9261==
==9261== Invalid write of size 4
==9261==    at 0x1091D5: create_date (task_04.c:23)
==9261==    by 0x109205: main (task_04.c:33)
==9261==  Address 0x4aa0048 is 0 bytes after a block of size 8 alloc'd
==9261==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==9261==    by 0x1091AF: create_date (task_04.c:18)
==9261==    by 0x109205: main (task_04.c:33)
==9261==
```

```
==9261== Invalid read of size 4
==9261==    at 0x109215: main (task_04.c:36)
==9261==  Address 0x4aa0048 is 0 bytes after a block of size 8 alloc'd
==9261==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==9261==    by 0x1091AF: create_date (task_04.c:18)
==9261==    by 0x109205: main (task_04.c:33)
==9261==
01.09.2020==9261==
==9261== HEAP SUMMARY:
==9261==     in use at exit: 0 bytes in 0 blocks
==9261==   total heap usage: 2 allocs, 2 frees, 1,032 bytes allocated
==9261==
==9261== All heap blocks were freed -- no leaks are possible
==9261==
==9261== For lists of detected and suppressed errors, rerun with: -s
==9261== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

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

Да, подтверждает, так как утилита сообщает, что в строках 23 и 36 происходят попытки записи и чтения за выделенной памятью, а в строке 18 происходит некорректное выделение памяти.

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

Утилита valgrind относит данные ошибки к категории некорректного чтения или записи.

Задание 5

Проанализируйте текст программы task_05.c.

Какая ошибка работы с динамической памятью в ней допущена?

В программе допущена утечка памяти, поскольку на место адреса первого выделенного блока записывается адрес нового блока, таким образом, первый блок не освобождается в конце программы.

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

Утечка 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.

```
==9828== Memcheck, a memory error detector
==9828== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==9828== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==9828== Command: ./app.exe
==9828==
5
10
==9828==
==9828== HEAP SUMMARY:
==9828==     in use at exit: 4 bytes in 1 blocks
==9828==   total heap usage: 3 allocs, 2 frees, 1,032 bytes allocated
==9828==
==9828== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==9828==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==9828==    by 0x1091A6: main (task_05.c:11)
==9828==
==9828== LEAK SUMMARY:
==9828==    definitely lost: 4 bytes in 1 blocks
==9828==    indirectly lost: 0 bytes in 0 blocks
==9828==      possibly lost: 0 bytes in 0 blocks
==9828==    still reachable: 0 bytes in 0 blocks
==9828==         suppressed: 0 bytes in 0 blocks
==9828==
==9828== For lists of detected and suppressed errors, rerun with: -s
==9828== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

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

Да, подтверждает, так как утилита сообщает, что в строке 11 была выделена память, которая в будущем была не очищена.

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

Утилита valgrind относит данные ошибки к категории утечек памяти.

Задание 6

Проанализируйте текст программы task_06.c.

Какая ошибка работы с динамической памятью в ней допущена?

В программе допущена утечка памяти, поскольку выделенная память не освобождается перед выходом из функции.

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

Утечка памяти.

Скомпилируйте программу task_06c. 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.

```
==10205== Memcheck, a memory error detector
==10205== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10205== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==10205== Command: ./app.exe
==10205==
5
==10205==
==10205== HEAP SUMMARY:
==10205==     in use at exit: 4 bytes in 1 blocks
==10205==   total heap usage: 2 allocs, 1 frees, 1,028 bytes allocated
==10205==
==10205== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==10205==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==10205==    by 0x109189: process (task_06.c:14)
==10205==    by 0x1091DD: main (task_06.c:31)
==10205==
==10205== LEAK SUMMARY:
==10205==    definitely lost: 4 bytes in 1 blocks
==10205==    indirectly lost: 0 bytes in 0 blocks
==10205==      possibly lost: 0 bytes in 0 blocks
==10205==    still reachable: 0 bytes in 0 blocks
==10205==         suppressed: 0 bytes in 0 blocks
==10205==
==10205== For lists of detected and suppressed errors, rerun with: -s
==10205== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

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

Да, подтверждает, так как утилита сообщает, что в строке 31 была вызвана функция, в которой на 11 строке была выделена память, которая не была очищена в конце.

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

Утилита valgrind относит данные ошибки к категории утечек памяти.

Задание 7

Проанализируйте текст программы task_07.c.

Какая ошибка работы с динамической памятью в ней допущена?

В программе используется непроинициализированный указатель, который приводит к ошибкам чтения значения по данному указателю.

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

Логическая ошибка. Использование непроинициализированного указателя.

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.

```
==10625== Use of uninitialised value of size 8
==10625==    at 0x48DB149: __vfscanf_internal (vfscanf-internal.c:1896)
==10625==    by 0x48D6141: __isoc99_scanf (isoc99_scanf.c:30)
==10625==    by 0x1091D7: main (task_07.c:14)
==10625==
==10625== Invalid write of size 4
==10625==    at 0x48DB149: __vfscanf_internal (vfscanf-internal.c:1896)
==10625==    by 0x48D6141: __isoc99_scanf (isoc99_scanf.c:30)
==10625==    by 0x1091D7: main (task_07.c:14)
==10625==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==10625==
==10625==
==10625== Process terminating with default action of signal 11 (SIGSEGV)
==10625==  Access not within mapped region at address 0x0
==10625==    at 0x48DB149: __vfscanf_internal (vfscanf-internal.c:1896)
==10625==    by 0x48D6141: __isoc99_scanf (isoc99_scanf.c:30)
==10625==    by 0x1091D7: main (task_07.c:14)
==10625==  If you believe this happened as a result of a stack
==10625==  overflow in your program's main thread (unlikely but
==10625==  possible), you can try to increase the size of the
==10625==  main thread stack using the --main-stacksize= flag.
==10625==  The main thread stack size used in this run was 8388608.
==10625==
==10625== HEAP SUMMARY:
==10625==     in use at exit: 1,024 bytes in 1 blocks
==10625==   total heap usage: 1 allocs, 0 frees, 1,024 bytes allocated
==10625==
==10625== LEAK SUMMARY:
==10625==    definitely lost: 0 bytes in 0 blocks
==10625==    indirectly lost: 0 bytes in 0 blocks
==10625==      possibly lost: 0 bytes in 0 blocks
==10625==    still reachable: 1,024 bytes in 1 blocks
==10625==         suppressed: 0 bytes in 0 blocks
==10625== Reachable blocks (those to which a pointer was found) are not shown.
==10625== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==10625==
==10625== Use --track-origins=yes to see where uninitialised values come from
==10625== For lists of detected and suppressed errors, rerun with: -s
==10625== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

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

Да, подтверждает, так как утилита сообщает, что в строке 14 происходит запись по неинициализированному указателю.

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

Утилита valgrind относит данные ошибки к категории некорректного чтения или записи.

Задание 8

Проанализируйте текст программы task_08.c.

Какая ошибка работы с динамической памятью в ней допущена?

В программе, после освобождения памяти, происходит запись в освобожденный блок. То есть ошибка заключается в использовании освобожденного блока памяти.

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.

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.

```
==11008== Memcheck, a memory error detector
==11008== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==11008== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==11008== Command: ./app.exe
==11008==
5
==11008== Invalid write of size 4
==11008==    at 0x1091E0: main (task_08.c:20)
==11008==  Address 0x4aa0040 is 0 bytes inside a block of size 4 free'd
==11008==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==11008==    by 0x1091DB: main (task_08.c:18)
==11008==  Block was alloc'd at
==11008==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==11008==    by 0x10919E: main (task_08.c:11)
==11008==
==11008== Invalid read of size 4
==11008==    at 0x1091EA: main (task_08.c:22)
==11008==  Address 0x4aa0040 is 0 bytes inside a block of size 4 free'd
==11008==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==11008==    by 0x1091DB: main (task_08.c:18)
==11008==  Block was alloc'd at
==11008==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==11008==    by 0x10919E: main (task_08.c:11)
==11008==
7
==11008==
==11008== HEAP SUMMARY:
==11008==     in use at exit: 0 bytes in 0 blocks
==11008==   total heap usage: 2 allocs, 2 frees, 1,028 bytes allocated
==11008==
```

```
==11008== All heap blocks were freed -- no leaks are possible
==11008==
==11008== For lists of detected and suppressed errors, rerun with: -s
==11008== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)


==10625==    still reachable: 1,024 bytes in 1 blocks
==10625==         suppressed: 0 bytes in 0 blocks
==10625== Reachable blocks (those to which a pointer was found) are not shown.
==10625== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==10625==
==10625== Use --track-origins=yes to see where uninitialised values come from
==10625== For lists of detected and suppressed errors, rerun with: -s
==10625== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

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

Да, подтверждает, так как утилита сообщает, что в строке 20 происходит запись в освобожденный блок, в строке 22 происходит чтение из освобожденного блока памяти, в 18 строке происходит освобождение этого блока, а в строке 11 этот блок был изначально выделен.

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

Утилита valgrind относит данные ошибки к категории некорректного чтения или записи.

Задание 9

Проанализируйте текст программы task_09.c.

Какая ошибка работы с динамической памятью в ней допущена?

В программе, после завершения работы цикла будет изменен указатель на начало выделенной памяти, то есть программа в конце не сможет освободить выделенную память, что приведет к утечке памяти.

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

Логическая ошибка, утечка памяти, изменение указателя, указывающего на начало выделенной памяти, а также освобождение не выделенной 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.

```
==11294== Memcheck, a memory error detector
==11294== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==11294== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==11294== Command: ./app.exe
==11294==
==11294== Invalid free() / delete / delete[] / realloc()
==11294==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==11294==    by 0x1091D5: main (task_09.c:23)
==11294==  Address 0x4aa0054 is 0 bytes after a block of size 20 alloc'd
==11294==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==11294==    by 0x10918C: main (task_09.c:12)
==11294==
==11294==
==11294== HEAP SUMMARY:
==11294==     in use at exit: 20 bytes in 1 blocks
==11294==   total heap usage: 1 allocs, 1 frees, 20 bytes allocated
==11294==
==11294== 20 bytes in 1 blocks are definitely lost in loss record 1 of 1
==11294==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==11294==    by 0x10918C: main (task_09.c:12)
==11294==
==11294== LEAK SUMMARY:
==11294==    definitely lost: 20 bytes in 1 blocks
==11294==    indirectly lost: 0 bytes in 0 blocks
==11294==      possibly lost: 0 bytes in 0 blocks
==11294==    still reachable: 0 bytes in 0 blocks
==11294==         suppressed: 0 bytes in 0 blocks
==11294==
==11294== For lists of detected and suppressed errors, rerun with: -s
==11294== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

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

Да, подтверждает, так как утилита сообщает, что в строке 23 происходит освобождение не выделенной памяти, а в 12 строке происходит выделение памяти, указатель на которую в следующих строках менялся.

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

Утилита valgrind относит данные ошибки к категориям утечки памяти, а также освобождение не выделенной памяти.