**DrMemory**

DrMemory - инструментальное программное обеспечение, предназначенное для отладки использования памяти, обнаружения утечек памяти.

**1. Установка.**

Загрузите DrMemory с сайта http://drmemory.org (ссылка Download).

В аудитории 508л установлена версия 1.8.0-8. Сейчас уже доступна версия 1.9.0. Шаги установки могут отличаться в зависимости от версии.

Запустите инсталлятор; согласитесь с лицензией; укажите необходимость изменения переменной PATH для всех пользователей (рекомендуется) или для текущего пользователя; запустите процесс установки.

**2. Подготовка исполняемого файла.**

DrMemory поддерживает 32-х битные приложения, скомпилированные MinGW gcc или g++. При компиляции приложения необходимо указать опцию –ggdb, которая генерирует отладочную информацию в формате, с которым работает DrMemory.

**3. Запуск DrMemory.**

Ваше приложение запускается под управление DrMemory точно так же, как и из командной строки.

Например, Ваше приложение называется lab_01.exe, а для его корректной работы в командной строке необходимо указать имя текстового файла с исходными данными. Т.е. для запуска приложения из командной строки необходимо написать:

```
lab_01.exe test_01.txt
```

Тогда если Вы запускаете DrMemory из каталога, в котором расположен исполняемый файл, командная строка запуска DrMemory будет выглядеть следующим образом:

```
drmemory -- lab_01.exe test_01.txt
```

При необходимости можно указать полный путь к исполняемому файлу.

“--” отделяет аргументы DrMemory от описания запускаемого приложения.

Результаты работы DrMemory размещаются в нескольких текстовых файлах и составляют так называемый отчет. Один из этих файлов, который называется results.txt и содержит основную массу информации, автоматически открывается с помощью программы notepad после завершения работы DrMemory. Местоположение этого файла указывается в нем самом.

По умолчания DrMemory создает отчеты в каталоге «C:\Documents and Settings\user name\Application Data» (для Windows XP). При каждом запуске в этом каталоге DrMemory создает подкаталог с именем DrMemory-<appname>.<pid>.NNN.

С помощью опции –logdir «путь к каталогу» можно указать каталог, в котором DrMemory будет создавать отчеты. Например,

```
drmemory –logdir c:\temp -- lab_01.exe test_01.txt
```

### 4. Ошибки, которые обнаруживает DrMemory.

### 4.1. Обращение к неадресуемой памяти (UNADDRESSABLE ACCESS).

DrMemory рассматривает чтение или запись за пределами выделенной области памяти как обращение к неадресуемой памяти. При этом анализируется только память, выделенная с помощью malloc.
В эту же категорию попадают ошибки обращения к уже освобожденной памяти.

```
01.#include <stdlib.h>
02.
03.#define N 5
04.
05.int main(void)
06.{
07.    int *p = malloc(N * sizeof(int));
08.
09.    if (p)
10.    {
11.       // Выход за границу выделенной области памяти
12.       for (int i = 0; i < N + 1; i++)
13.          p[i] = i;
14.
15.       free(p);
16.
17.       // Выделенная память уже освобождена
18.       p[0] = 5;
19.    }
20.
21.    return 0;
21.}
```

Строка компиляции:

```
gcc -std=c99 -Wall -Werror -ggdb -o test_01.exe test_01.c
```

Строка запуска DrMemory:

```
drmemory -logdir c:\temp -- test_01.exe
```

Фрагменты файла results.txt (на Вашем компьютере выдача может отличаться):

```
Error #1: UNADDRESSABLE ACCESS: writing 0x003f250c-0x003f2510 4 byte(s)
# 0 main                [C:\Homework\C\DrMemory/test_01.c:13]
Note: @0:00:01.047 in thread 3712
Note: next higher malloc: 0x003f2880-0x003f2aa0
Note: refers to 1 byte(s) beyond last valid byte in prior malloc
Note: prev lower malloc:  0x003f24f8-0x003f250c
Note: instruction: mov    %edx -> (%eax)

Error #2: UNADDRESSABLE ACCESS: writing 0x003f24f8-0x003f24fc 4 byte(s)
# 0 main                [C:\Homework\C\DrMemory/test_01.c:18]
Note: @0:00:01.063 in thread 3712
Note: next higher malloc: 0x003f2880-0x003f2aa0
```

```
Note: prev lower malloc:  0x003f24c8-0x003f24d0
Note: 0x003f24f8-0x003f24fc overlaps memory 0x003f24f8-0x003f2510 that was freed
Note: instruction: mov    $0x00000005 -> (%eax)
```

Опция –brief предоставляет меньше информации, но, возможно, полученный результат легче читается.

Строка запуска DrMemory:

```
drmemory –brief -logdir c:\temp -- test_01.exe
```

Фрагменты файла results.txt:

```
Error #1: UNADDRESSABLE ACCESS: writing 4 byte(s)
# 0 main               [test_01.c:13]
Note: refers to 1 byte(s) beyond last valid byte in prior malloc

Error #2: UNADDRESSABLE ACCESS: writing 4 byte(s)
# 0 main               [test_01.c:18]
Note: refers to memory that was freed here:
Note: # 0 main               [test_01.c:18]
```

### 4.2. Работа с неинициализированными переменными (UNINITIALIZED READ).

Компилятор может отследить лишь простые случаи использования неинициализированных переменных. DrMemory выполняет более глубокий анализ.

```
01.#include <stdio.h>
02.
03.void print(const int *a)
04.{
05.    printf("%d", *a);
06.}
07.
08.int main(void)
09.{
10.    int b;
11.
12.    // Переменной не было присвоено значение
13.    print(&b);
14.
15.    return 0;
16.}

Error #1: UNINITIALIZED READ: reading register edi
# 0 __pformat_int                 [../../misc/src/libcrt/stdio/pformat.c:450]
# 1 __mingw_pformat               [../../misc/src/libcrt/stdio/pformat.c:1887]
# 2 __mingw_vprintf               [../../src/libcrt/stdio/vprintf.c:69]
# 3 print                         [test_02.c:5]
# 4 main                          [test_02.c:13]
```

### 4.3. Неверные аргументы при работе с кучей (INVALID HEAP ARGUMENT).

Одна из распространенных ошибок такого рода – освобождение с помощью функции free памяти, которая не была выделена с помощью функции malloc, или освобождение уже освобожденной памяти.

```
01.#include <stdlib.h>
02.
```

```
03.void my_free(int *a)
04.{
05.    free(a);
06.}
07.
08.int main(void)
09.{
10.    int a[5];
11.    int *p = malloc(sizeof(int));
12.
13.    if(p)
14.    {
15.       free(p);
16.
17.       // Случайно освободили память еще раз
18.       free(p);
19.    }
20.
21.    // По ошибке передали статический массив
22.    my_free(a);
23.
24.    return 0;
25.}
```

```
Error #1: INVALID HEAP ARGUMENT to free 0x00f40868
# 0 replace_free               [d:\drmemory_package\common\alloc_replace.c:2503]
# 1 main                       [C:\Homework\reserch\drmemory/test_03.c:18]
Note: @0:00:00.734 in thread 2924
Note: prev lower malloc:  0x00f40808-0x00f40813
Note: memory was previously freed here:
Note: # 0 replace_free         [d:\drmemory_package\common\alloc_replace.c:2503]
Note: # 1 main                 [C:\Homework\reserch\drmemory/test_03.c:15]

Error #2: INVALID HEAP ARGUMENT to free 0x0022ff08
# 0 replace_free               [d:\drmemory_package\common\alloc_replace.c:2503]
# 1 my_free                    [C:\Homework\reserch\drmemory/test_03.c:5]
# 2 main                       [C:\Homework\reserch\drmemory/test_03.c:22]
Note: @0:00:00.750 in thread 2924
Note: 0x0022ff08 refers to -404 byte(s) beyond the top of the stack 0x0022fd74
```

### 4.4. Утечки памяти (LEAK).

Согласно wikipedia, «утечка памяти - процесс неконтролируемого уменьшения объёма свободной оперативной памяти компьютера, связанный с ошибками в работающих программах, вовремя не освобождающих ненужные уже участки памяти, или с ошибками системных служб контроля памяти».

```
01.#include <stdio.h>
02.#include <stdlib.h>
03.
04.int main(void)
05.{
06.    int *a = malloc(sizeof(int));
07.    int *b = malloc(sizeof(int));
08.
09.    *a = 3;
10.    *b = 4;
11.
12.    printf("%d\n", *a);
13.    printf("%d\n", *b);
14.
15.    // Потеряли память, на которую указывает b
```

```
16.    b = a;
17.
18.    free(a);
19.}

Error #1: LEAK 4 direct bytes 0x00f70890-0x00f70894 + 0 indirect bytes
# 0 replace_malloc              [d:\drmemory_package\common\alloc_replace.c:2373]
# 1 main                        [C:\Homework\reserch\drmemory/test_4.c:7]
```

Обратите внимание в сообщение об утечке указывается номер строки, в которой память была выделена!

***Вместо заключения.***

Настоятельно рекомендую самостоятельно скомпилировать все приводимые примеры и посмотреть на выдачу DrMemory.