Анализ программного кода

2. Стек программы

доц. Нестеренко В.А.

Отладчик OllyDbg

Введение и краткий обзор отладчика

…

Начнём с простой программы Stack.exe

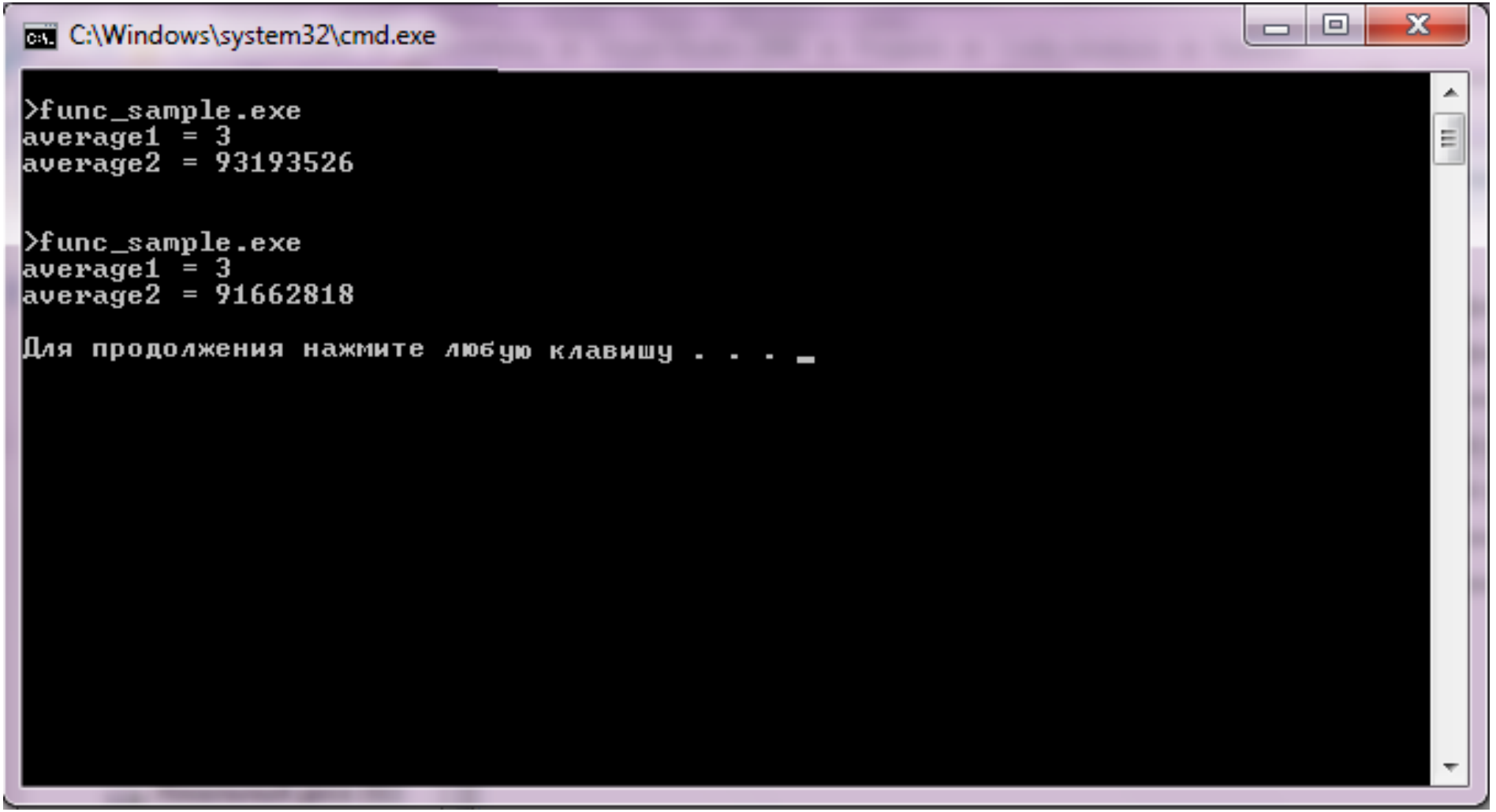
```
#include <stdio.h>

char char_average(char array[], int count) {
    int i;
    char average=0;
    for (i = 0; i < count; i++)
        average += array[i];
    average /= count;
    return average;
}

int int_average(int array[], int count) {
    int i, average=0;
    for (i = 0; i < count; i++)
        average += array[i];
    average /= count;
    return average;
}

void main(void) {
    char chars[] = { 1, 2, 3, 4, 5 };
    int integers[] = { 1, 2, 3, 4, 5 };
    printf("average1 = %d\n", char_average(chars, sizeof(chars)));
    printf("average2 = %d\n", int_average(integers, sizeof(integers)));
}
```

Результат исполнения Func_Sample.exe



Трассировка в отладчике OllyDbg

Использование стека при вызове функции:

1. На стек заносятся аргументы функции.
2. На стеке сохраняется адрес возврата
3. Текущий указатель на вершину стека сохраняется в EBP:

```
PUSH   EBP
MOV    EBP, ESP
```

4. На стеке выделяется память под локальные переменные:

```
SUB    ESP, xxx
```

5. [EBP + xxx] – обращение к к аргументам функции
   [EBP - xxx] – обращение к локальным переменным
6. Перед выходом из функции восстанавливается указатель на вершину стека:

```
MOV    ESP, EBP
POP    EBP
```

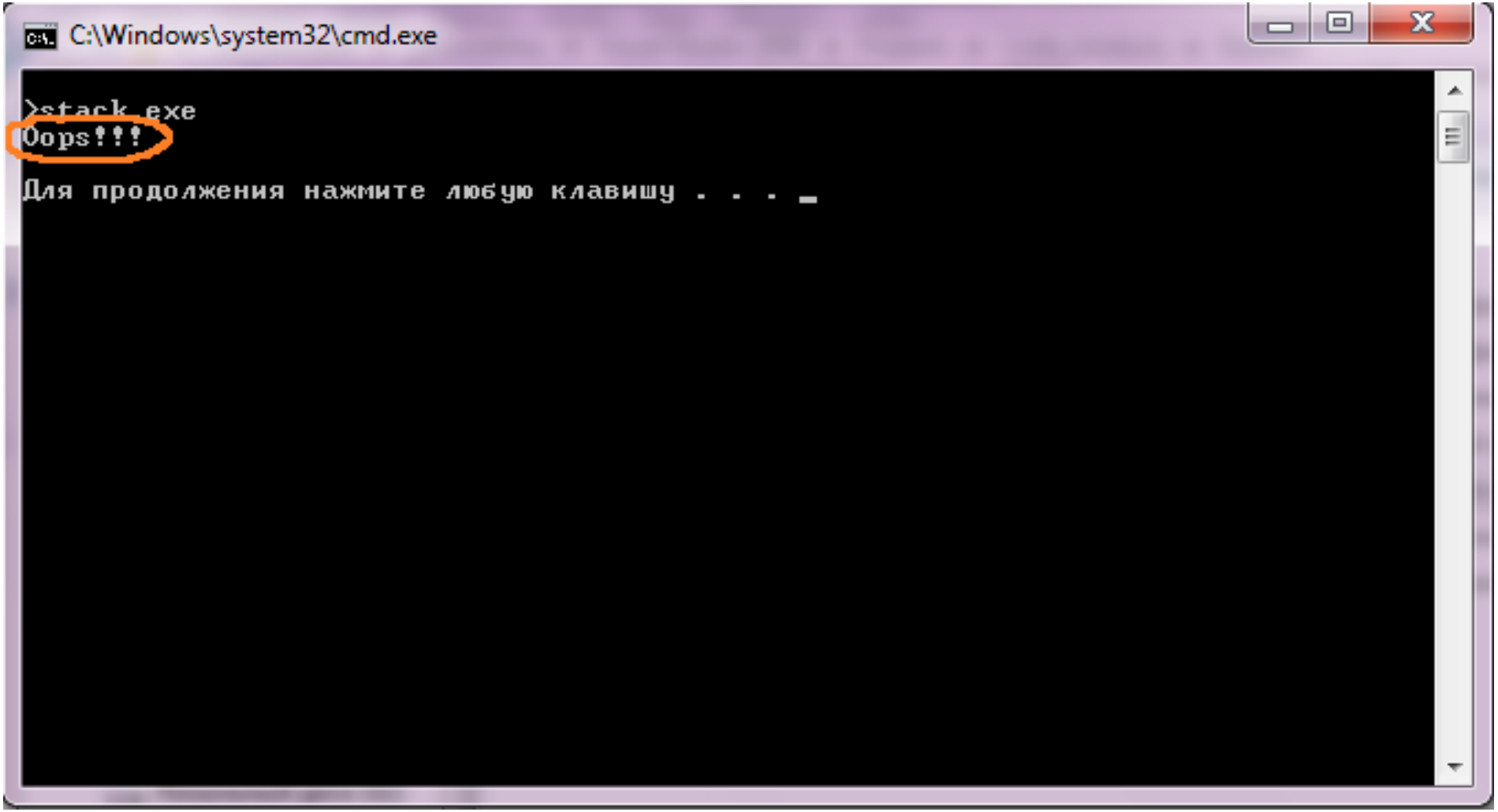
Ещё пример Stack.exe

```
#include <stdio.h>
typedef unsigned DWORD;
DWORD adr_ret;

void Hook () {
   DWORD M[1];
   M[2] = adr_ret;
   printf ("Oops!!!\n");
}

void main () {
   DWORD M[1];
   adr_ret = (DWORD)M[2];
   M[2] = (DWORD)Hook;
}
```

Результат исполнения Stack.exe



Трассировка в отладчике OllyDbg

…