

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики»
(СибГУТИ)

Кафедра вычислительных систем

ОТЧЕТ
по практической работе №1
по дисциплине «**Программирование**»

Выполнил:

Гердележов Д.Д.

студент гр. ИВ-122

«8» февраля 2022 г.

Проверил:

Фульман В.О.

Старший преподаватель
кафедры ВС.

«__» февраля 2022 г.

Оценка «_____»

Новосибирск 2022

Оглавление:

ЗАДАНИЕ.....	3
ВЫПОЛНЕНИЕ РАБОТЫ.....	5
ПРИЛОЖЕНИЕ.....	8

Задание:

В приведенных программах содержатся ошибки. Необходимо с помощью отладчика локализовать и исправить их.

№1

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void init(int *arr, int n)
5 {
6     arr = malloc(n * sizeof(int));
7     int i;
8     for (i = 0; i < n; ++i)
9     {
10         arr[i] = i;
11     }
12 }
13
14 int main()
15 {
16     int *arr = NULL;
17     int n = 10;
18     init(arr, n);
19
20     int i;
21     for (i = 0; i < n; ++i)
22     {
23         printf("%d\n", arr[i]);
24     }
25     return 0;
26 }

```

№2

```

1 #include <stdio.h>
2
3 typedef struct
4 {
5     char str[3];
6     int num;
7 } NumberRepr;
8
9 void format(NumberRepr *number)
10 {
11     sprintf(number->str, "%3d", number->num);
12 }
13
14 int main()
15 {
16     NumberRepr number = {.num = 1025};
17
18     format(&number);
19
20     printf("str: %s\n", number.str);
21     printf("num: %d\n", number.num);
22
23     return 0;
24 }

```

№3

```

1 #include <stdio.h>
2
3 #define SQR(x) x *x
4
5 int main()
6 {
7     int y = 5;
8     int z = SQR(y + 1);
9     printf("z = %d\n", z);
10    return 0;
11 }

```

№4

```

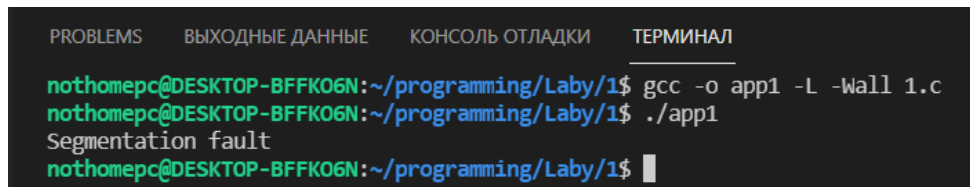
1 #include <stdio.h>
2
3 void swap(int *a, int *b)
4 {
5     int tmp = *a;
6     *a = *b;
7     *b = tmp;
8 }
9
10 void bubble_sort(int *array, int size)
11 {
12     int i, j;
13     for (i = 0; i < size - 1; ++i)
14     {
15         for (j = 0; j < size - i; ++j)
16         {
17             if (array[j] > array[j + 1])
18             {
19                 swap(&array[j], &array[j + 1]);
20             }
21         }
22     }
23 }
24
25 int main()
26 {
27     int array[100] = {10, 15, 5, 4, 21, 7};
28     bubble_sort(array, 6);
29
30     int i;
31     for (i = 0; i < 6; ++i)
32     {
33         printf("%d ", array[i]);
34     }
35     printf("\n");
36
37     return 0;
38 }

```

Выполнение работы:

№1

При запуске команды, возникала ошибка: “Segmentation fault” (Рис. 1).



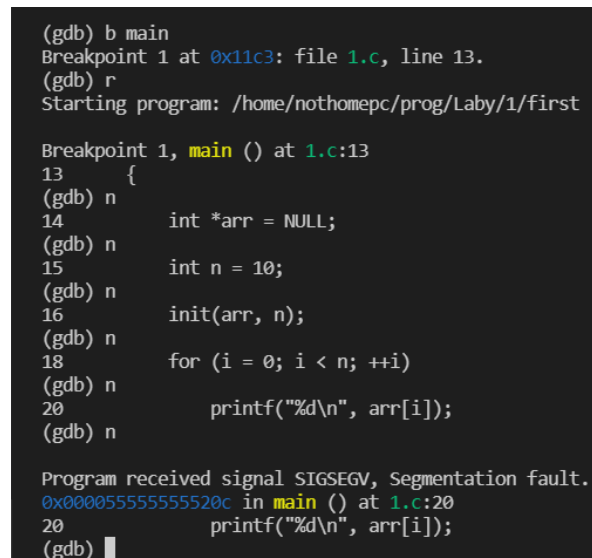
```

nothomepc@DESKTOP-BFFK06N:~/programming/Laby/1$ gcc -o app1 -L -Wall 1.c
nothomepc@DESKTOP-BFFK06N:~/programming/Laby/1$ ./app1
Segmentation fault
nothomepc@DESKTOP-BFFK06N:~/programming/Laby/1$

```

Рис. 1. Терминал, после запуска программы.

Путем отладки было выявлено, что ошибка возникала в 20 строке, при выводе массива на экран. (Рис. 2)



```

(gdb) b main
Breakpoint 1 at 0x11c3: file 1.c, line 13.
(gdb) r
Starting program: /home/nothomepc/prog/Laby/1/first

Breakpoint 1, main () at 1.c:13
13  {
(gdb) n
14      int *arr = NULL;
(gdb) n
15      int n = 10;
(gdb) n
16      init(arr, n);
(gdb) n
18      for (i = 0; i < n; ++i)
(gdb) n
20          printf("%d\n", arr[i]);
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x000055555555520c in main () at 1.c:20
20      printf("%d\n", arr[i]);
(gdb)

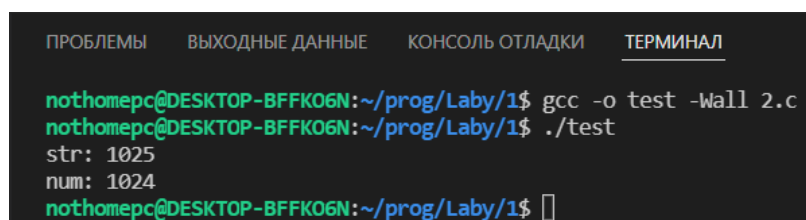
```

Рис. 2. Отладка программы.

Ошибка появилась из-за того, что функция “malloc” (выделяет блок памяти под массив и возвращает указатель на него), записанная в функции “init” выделяла память для копии arr, другой переменной. Чтобы исправить ошибку можно: перенести функцию malloc в main, чтобы инициализация массива происходила там. Или предавать в init адрес указателя на изначальный массив. (исправленный код – Приложение 1).

№2

При запуске программы не появлялось никаких ошибок, программа выполнялась и выводила данные. Однако поле “num” изменило свое значение (Рис. 3.)



```

nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$ gcc -o test -Wall 2.c
nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$ ./test
str: 1025
num: 1024
nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$

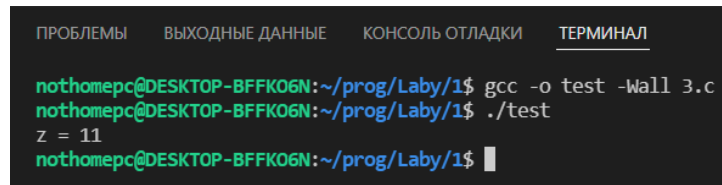
```

Рис. 3. Терминал, после запуска программы.

“sprint” – не безопасная функция, поэтому, при недостаточном количестве места для записи она переходит в следующую ячейку памяти, так как в структуре данные располагаются друг за другом, происходила перепись поля num. Чтобы исправить ошибку, нужно увеличить размерность массива с трех до пяти (4 символа для цифр и 1 для тернарного оператора). (строка 5) (исправленный код – Приложение 2).

№3

Программа скомпилировалась и запустилась без ошибок. Однако вместо вывода “z = 36” мы видим вывод: “z = 11” (Рис. 4).



```

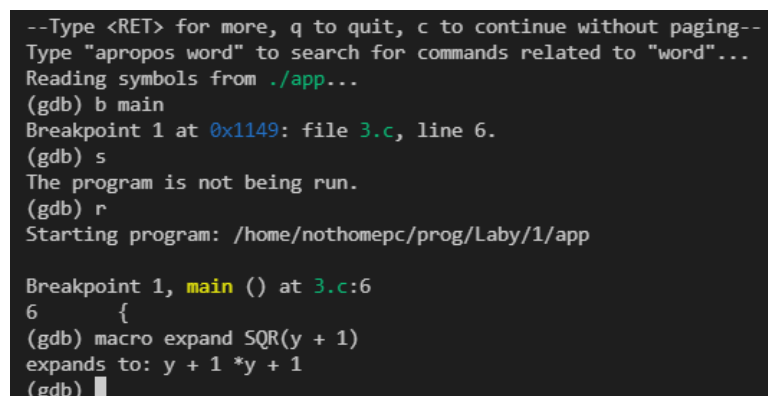
ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ

nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$ gcc -o test -Wall 3.c
nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$ ./test
z = 11
nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$

```

Рис. 4. Терминал, после запуска программы.

С помощью отладки (Рис. 5.) видно, что передающееся выражение $(y + 1)$, разворачивается так: $(y + 1 * y + 1)$, именно отсюда и берется число 11 $(5 + 1 * 5 + 1)$ чтобы исправить эту ошибку можно увеличить “y” до вызова макроса, или поставить дополнительные скобки (в объявлении макроса или его вызове) (исправленный код – Приложение 3).



```

--Type <RET> for more, q to quit, c to continue without paging--
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./app...
(gdb) b main
Breakpoint 1 at 0x1149: file 3.c, line 6.
(gdb) s
The program is not being run.
(gdb) r
Starting program: /home/nothomepc/prog/Laby/1/app

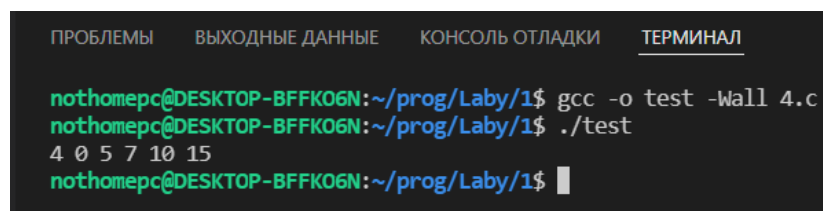
Breakpoint 1, main () at 3.c:6
6      {
(gdb) macro expand SQR(y + 1)
expands to: y + 1 * y + 1
(gdb)

```

Рис. 5. Отладка программы.

№4

Программа скомпилировалась и запустилась без ошибок. Программа представляет из себя сортировку пузырьком, но конечный массив не упорядочен (терминал – рис. 6).



```

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ

nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$ gcc -o test -Wall 4.c
nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$ ./test
4 0 5 7 10 15
nothomepc@DESKTOP-BFFK06N:~/prog/Laby/1$

```

Рис. 6. Терминал, после запуска программы.

После отладки видно, что программа лишний раз заходит во вложенный цикл и вместо того, чтобы сравнивать сортируемые числа массива, сравнивает последний элемент этих чисел с нулём, который заполняет не используемые элементы массива (Рис. 7, 8).

```
(gdb) n
28      bubble_sort(array, 6);
1: array = {10, 15, 5, 4, 21, 7, 0 <repeats 94 times>}
```

Рис. 7. Массив после инициализации

```
(gdb)
17      if (array[j] > array[j + 1])
1: *(array) = 10
2: *(array + 1) = 5
3: *(array + 2) = 4
4: *(array + 3) = 15
5: *(array + 4) = 7
6: *(array + 5) = 21
(gdb) s
19      swap(&array[j], &array[j + 1]);
1: *(array) = 10
2: *(array + 1) = 5
3: *(array + 2) = 4
4: *(array + 3) = 15
5: *(array + 4) = 7
6: *(array + 5) = 21
(gdb) s
swap (a=0x7fffffffde50, b=0x7fffffffde54) at 4.c:4
4      {
(gdb) s
5          int tmp = *a;
(gdb) s
6          *a = *b;
(gdb) s
7          *b = tmp;
(gdb) s
8      }
(gdb) s
bubble_sort (array=0x7fffffffde40, size=6) at 4.c:15
15      for (j = 0; j < size - i - 1; ++j)
1: *(array) = 10
2: *(array + 1) = 5
3: *(array + 2) = 4
4: *(array + 3) = 15
5: *(array + 4) = 7
6: *(array + 5) = 0
```

Рис. 8. Этап отладки, на котором видно выход за пределы сортируемых чисел.
(исправленный код – Приложение 4).

Приложение:

1)

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 void init(int *arr, int n)
4 {
5     int i;
6     for (i = 0; i < n; ++i)
7     {
8         arr[i] = i;
9     }
10 }
11 int main()
12 {
13     int *arr = NULL;
14     int n = 10;
15     arr = malloc(n * sizeof(int));
16     init(arr, n);
17     int i;
18     for (i = 0; i < n; ++i)
19     {
20         printf("%d\n", arr[i]);
21     }
22     return 0;
23 }

```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void init(int **arr, int n)
5 {
6     *arr = malloc(n * sizeof(int));
7     int i;
8     for (i = 0; i < n; ++i)
9     {
10         (*arr)[i] = i;
11     }
12 }
13
14 int main()
15 {
16     int *arr = NULL;
17     int n = 10;
18     init(&arr, n);
19     int i;
20     for (i = 0; i < n; ++i)
21     {
22         printf("%d\n", arr[i]);
23     }
24     return 0;
25 }

```


2)

```

1 #include <stdio.h>
2
3 typedef struct
4 {
5     char str[5];
6     int num;
7 } NumberRepr;
8
9 void format(NumberRepr *number)
10 {
11     sprintf(number->str, "%3d", number->num);
12 }
13
14 int main()
15 {
16     NumberRepr number = {.num = 1025};
17
18     format(&number);
19
20     printf("str: %s\n", number.str);
21     printf("num: %d\n", number.num);
22
23     return 0;
24 }

```

3)

```

1 #include <stdio.h>
2
3 #define SQR(x) x *x
4
5 int main()
6 {
7     int y = 5;
8     int z = SQR((y + 1));
9     printf("z = %d\n", z);
10    return 0;
11
12 }

```

```

1 #include <stdio.h>
2
3 #define SQR(x) (x) *(x)
4
5 int main()
6 {
7     int y = 5;
8     int z = SQR(y + 1);
9     printf("z = %d\n", z);
10    return 0;
11
12 }

```

4)

```
1 #include <stdio.h>
2
3 void swap(int *a, int *b)
4 {
5     int tmp = *a;
6     *a = *b;
7     *b = tmp;
8 }
9
10 void bubble_sort(int *array, int size)
11 {
12     int i, j;
13     for (i = 0; i < size - 1; ++i)
14     {
15         for (j = 0; j < size - i - 1; ++j)
16         {
17             if (array[j] > array[j + 1])
18             {
19                 swap(&array[j], &array[j + 1]);
20             }
21         }
22     }
23 }
24
25 int main()
26 {
27     int array[100] = {10, 15, 5, 4, 21, 7};
28     bubble_sort(array, 6);
29
30     int i;
31     for (i = 0; i < 6; ++i)
32     {
33         printf("%d ", array[i]);
34     }
35     printf("\n");
36
37     return 0;
38 }
```