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:  студент гр. ИС-241  «\_\_» февраля 2022 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Дмитрюк В.В./ |
|  |  |  |
| Проверил:  ст. преп. Кафедры ВС  «\_\_» февраля 2018 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Фульман В.О./ |

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_»

Новосибирск 2023

ОГЛАВЛЕНИЕ

[ЗАДАНИЕ 3](#__RefHeading___Toc744_4108892630)

[ВЫПОЛНЕНИЕ РАБОТЫ 4](#__RefHeading___Toc748_4108892630)

[ПРИЛОЖЕНИЕ 13](#__RefHeading___Toc746_4108892630)

# **ЗАДАНИЕ**

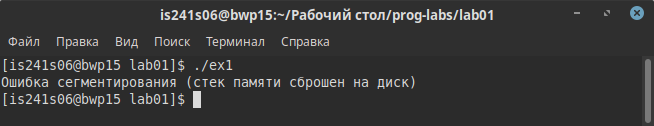
*Получить навыки отладки программ на примере использования отладчика GDB. Локализовать и исправить ошибки в приведённых программах с помощью отладчика.*

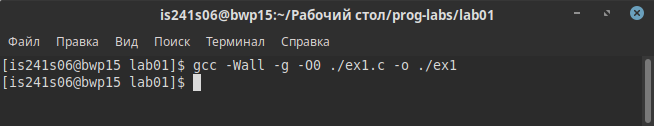
# **ВЫПОЛНЕНИЕ РАБОТЫ**

Рассмотрим задание 1.

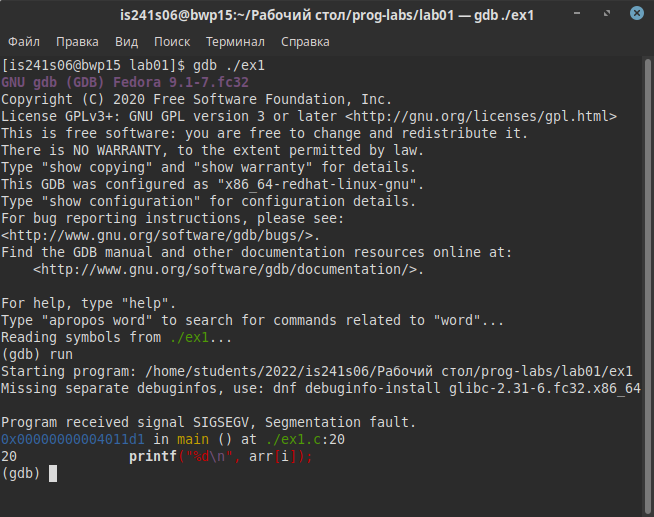
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <stdio.h>  #include <stdlib.h>  **void** **init**(**int**\* arr, **int** n)  {  arr = malloc(n \* **sizeof**(**int**));  **int** i;  **for** (i = **0**; i < n; ++i)  {  arr[i] = i;  }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

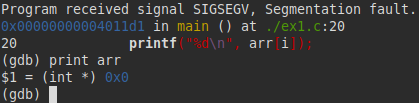
Очевидно, что функция init призвана инициализировать массив определённого размера инкрементально возрастающими числами. Однако, в ней допущена ошибка, из-за чего в результате работы программы происходит ошибка сегментирования.

Компилятор не сообщил о предупреждениях даже с флагом «-Wall».   
Воспользуемся отладчиком для локализации ошибки. Для этого мы должны скомпилировать программу с флагами «-g -O0».



Вызовем отладчик gdb и запустим в нём нашу программу.

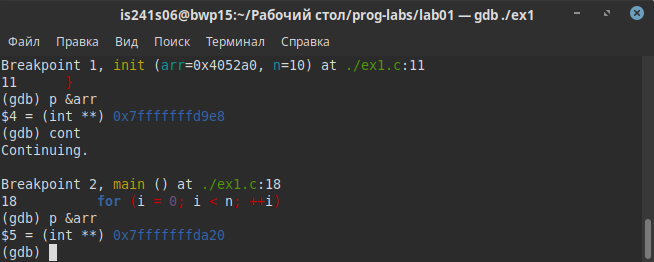
Ошибка произошла на строке 20, при попытке вывести значение массива в стандартный поток ввода-вывода. Ошибка сегментации чаще всего происходит при попытке обратиться к памяти, в которую у нас нет доступа. Выведем на экран значение указателя arr.



Действительно, ошибка произошла при попытке обратиться к памяти по нулевому указателю. Значение указателя в функции arr не изменилось, вопреки замыслу автора, поскольку переменные в функцию передаются по значению. Докажем это, сравнив значения указателей arr внутри функций init и main.

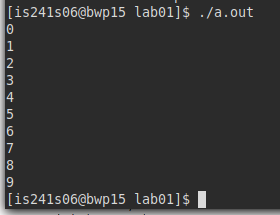
Поставим точку останова на строчках 11 и 17 — завершение работы функции init и строчка, следующая за вызовом функции init внутри функции main соответственно, и посмотрим в них значения переменной arr.

Адреса этих указателей тоже различаются:



Изменим программу таким образом, чтобы избежать проблемы передачи аргументов по значению: будем изменять указатель, определённый в функции main из функции init по указателю на него.  
Добавления выделены жёлтым цветом.

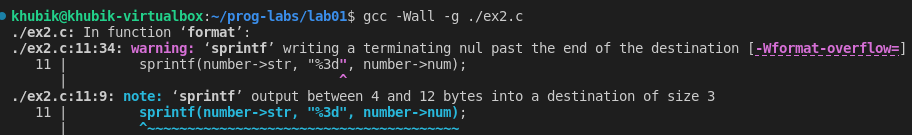
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #include <stdio.h>  #include <stdlib.h>  **void** **init**(**int**\*\* arr, **int** n)  {  \*arr = malloc(n \* **sizeof**(**int**));  **int** i;  **for** (i = **0**; i < n; i++)  {  (\*arr)[i]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

Теперь программа работает так, как и было задумано.

Рассмотрим задание 2.

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

Задумка программиста — представить число 1025 в строковом и целочисленном представлении. При сборке программы, компилятор предупреждает нас о возможной ошибке — признак конца строки записывается за пределами назначения.

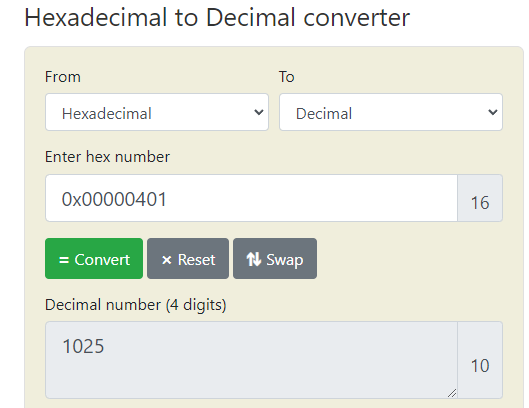
При запуске программы, можно увидеть, что строковое и целочисленное представление числа не совпадают.

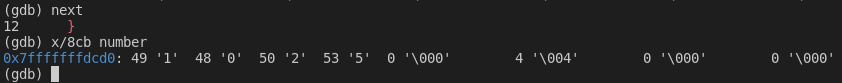
Заметим, что number.str отображается правильно, но number.num — нет, как будто потерян младший разряд.



Выведем память нашей структуры на экран. На первый взгляд, она занимает в памяти только семь байт — но это не правда, поскольку поля структуры выравниваются в памяти.

Выведем восемь байт памяти структуры number.

Как можно видеть, на место str (первые три байта) ещё ничего не записано. Один байт пропущен для выравнивания, а с четвёртого по восьмой записано число целое число 1025 в кодировке Little Endian (начиная с младшего разряда). В Big Endian это число было бы записано как 0x00 00 00 04 01.  
  
Запускаем программу и видим, как функция printf невозбранно перезаписывает младший разряд признаком конца строки.

Это происходит потому, что строка, записанная командой sprintf, состоит из пяти символов — четырёх, соответствующих цифрам числа 1025, и признака конца строки, в то время, как строка str[3] способна разместить только три символа.   
Два лишних символа перезаписали и пустое место памяти, оставленное для выравнивания, и разряд следующего за ним целого числа. Спецификатор %3d только добавляет недостающие пробелы, если разрядов меньше трёх — в нашем случае, он эквивалентен спецификатору %d.

Исправим ошибку программиста, увеличив размер строки до 5 символов.

char str[3]; // было

char str[5]; // стало

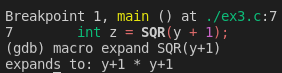
Теперь программа работает правильно!

Рассмотрим задание 3.

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

Перед нами программа, призванная вычислить квадрат числа (5+1).  
6\*6=36, однако программа нами не соглашается.

  
Функция SQR реализована макросом. Посмотрим, до чего он развернётся:

  
Нетрудно заметить, что порядок действий нарушен из-за недостающих скобок. Перепишем макрос следующим образом:

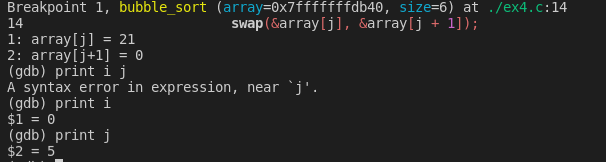
#define SQR(x) (x) \* (x)

Теперь квадрат будет вычислен правильно.  


Рассмотрим задание 4.

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

Программист реализовал сортировку первых 6 элементов массива из 100 элементов пузырьком на месте и вывел отсортированные элементы на экран, но допустил ошибку. Ожидалось увидеть 4, 5, 7, 10, 15, 21.

  
Скорее всего, ноль поднялся в первые шесть элементов из стоящих правее — проверим это.  
Это действительно так — после шестого числа (индекс j=5) берётся ещё одно — седьмое (array[j+1], индекс 6). После шестого числа массив инициализирован нулями. Эту ошибку можно исправить, переписав условие вложенного цикла.

for (j = 0; j < size - i - 1 ; ++j)

Теперь массив сортируется правильно.

# **ПРИЛОЖЕНИЕ**

**ex1.c – исправленный код Задания 1**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | #include <stdio.h>  #include <stdlib.h>  /\* Было: int\* arr, т.е. - принимался NULL по значению.\*/  **void** **init**(**int**\*\* arr, **int** n)  {  /\* Было - arr = malloc... ,  т.е. - менялось значение переменной arr, локальной  для функции init.  \*/  \*arr = malloc(n \* **sizeof**(**int**));  **int** i;  **for** (i = **0**; i < n; i++)  {  /\* Было - arr[i] = i.  Та же ошибка - arr локален для функции init.  \*/  (\*arr)[i] = i;  }  }  **int** **main**(){    **int**\* arr = NULL;  **int** n = **10**;    /\* Исправлена ошибка: попытка изменить значение по адресу,  по которому лежит переменная, после её передачи по значению  Было: init(arr, n)  \*/  init(&arr, n);  **int** i;  **for** (i = **0**; i < n; ++i)  {  printf("%d**\n**", arr[i]);  }  **return** **0**;  } |

**ex2.c — исправленный код Задания 2**

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

**ex3.c — исправленный код Задания 3**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #include <stdio.h>  /\* Исправлена ошибка - неправильный порядок действий  после развёртки макроса  Было: #define SQR(x) x \* x  \*/  #define SQR(x) (x) \* (x)  **int** **main**()  {  **int** y = **5**;  **int** z = SQR(y + **1**);  printf("z = %d**\n**", z);  **return** **0**;  } |

**ex4.c -исправленный код Задания 4**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | #include <stdio.h>  **void** **swap**(**int**\* a, **int**\* b)  {  **int** tmp = \*a;  \*a = \*b;  \*b = tmp;  }  **void** **bubble\_sort**(**int**\* array, **int** size)  {  **int** i, j;  **for** (i = **0**; i < size - **1**; ++i) {  /\* Исправлена ошибка - взятие лишнего элемента с конца  Было: (j = 0; j < size - i ; ++j) \*/  **for** (j = **0**; j < size - i - **1** ; ++j) {  **if** (array[j] > array[j + **1**]) {  swap(&array[j], &array[j + **1**]);  }  }  }  }  **int** **main**()  {  **int** array[**100**] = {**10**, **15**, **5**, **4**, **21**, **7**};  bubble\_sort(array, **6**);  **int** i;  **for** (i = **0**; i < **6** ; ++i) {  printf("%d ", array[i]);  }  printf("**\n**");  **return** **0**;  } |