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
«6» марта 2023 г. _____ /Стрельников.А.М./

Проверил:
Ст. преп. Кафедры ВС
«27» июня 2018 г. _____ /Фульман В.О./

Оценка «_____»

Новосибирск 2023

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

Необходимо локализовать и исправить ошибки, используя GDB.

Задание 1

```c
#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;
}
```

Задание 2

```c
#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;
}
```

Задание 3

```c
#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;
}
```

Задание 4

```c
#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;
}
```

ВЫПОЛНЕНИЕ РАБОТЫ

Задание 1

Дана программа, скомпилируем ее. Она должна заполнить массив “arr” элементами, значения которых равны индексу элемента.

Скомпилировав программу и запустив файл через отладчик GDB.

```
PS D:\VS.main()> gcc -Wall -g -O0 -o app Kp1.c
PS D:\VS.main()> ./app
PS D:\VS.main()> gdb ./app
GNU gdb (GDB) 13.1
```

Обнаруживаем проблему в 20 строке кода “Segmentation fault”.

```
(gdb) r
Starting program: D:\VS.main()\app.exe
[New Thread 30884.0x2448]

Thread 1 received signal SIGSEGV, Segmentation fault.
0x00007ff7f3bb1546 in main () at Kp1.c:20
20      printf("%d\n", arr[i]);
(gdb)
```

Поставим точки останова в отладчике. Первая в функции “init”, вторая перед вызовом функции “init” в функции “main” и третья после вызова “init”.

```
(gdb) b 6
Breakpoint 1 at 0x7ff7f3bb14c8: file Kp1.c, line 7.
(gdb) b 16
Breakpoint 2 at 0x7ff7f3bb151a: file Kp1.c, line 16.
(gdb) b 17
Breakpoint 3 at 0x7ff7f3bb1529: file Kp1.c, line 18.
```

Значение “arr” внутри функции “main”, перед вызовом “init”

```
Thread 1 hit Breakpoint 2, main () at Kp1.c:16
16      init(arr, n);
(gdb) print arr
$1 = (int *) 0x0
(gdb) n
```

Значение “arr” внутри функции “init”

```
Thread 1 hit Breakpoint 1, init (arr=0x1e1570, n=10) at Kp1.c:7
7       for (i = 0; i < n; ++i)
(gdb) print arr
$2 = (int *) 0x1e1570
```

И наконец, значение “arr” в функции “main” после вызова “init”

```
Thread 1 hit Breakpoint 3, main () at Kp1.c:18
18      for (i = 0; i < n; ++i)
(gdb) print arr
$3 = (int *) 0x0
```

Отсюда видим следующее: до вызова функции “init” в “main”, значение “arr” было равно NULL, когда как внутри “init” значение “arr” было другое. После вызова “init”, “arr” снова стал равен NULL.

Становится очевидно, что “arr” в функции “init” работает только в области этой функции. Для исправления нужно передавать в функцию двойной указатель (int**).

Скомпилируем и запустим уже исправленную программу.

```
PS D:\VS.main()> gcc -Wall -g -O0 -o Kp1 Kp1.c
PS D:\VS.main()> ./Kp1
0
1
2
3
4
5
6
7
8
9
PS D:\VS.main()>
```

Задание 2

У нас есть программа, которая должна записывать строковое представление числа “1025”, которое хранится в поле “num” структуры “NumberRepr”. Но на выходе значения отличаются.

```
PS D:\VS.main()> gcc -Wall -g -O0 -o Kp2 Kp2.c
PS D:\VS.main()> ./Kp2
str: 1025
num: 1024
PS D:\VS.main()>
```

С помощью отладчика GDB узнаем, какие значения хранятся в полях структуры “NumberRepr” перед их изменением функцией “format”. Поставив breakpoint на 16 строке (“NumberRepr number = { .num = 1025 };”) видим, что всего структура занимает в памяти компьютера 8 байт, 3 из которых уходят на массив “str”, 4 на тип “int” и один выравнивающий байт.

```
Breakpoint 1 at 0x7ff6fa861524: file Kp2.c, line 16.
(gdb) r
Starting program: D:\VS.main()\Kp2.exe
[New Thread 2400.0x86dc]

Thread 1 hit Breakpoint 1, main () at Kp2.c:16
16          NumberRepr number = { .num = 1025 };
(gdb) ptype /o number
type = struct {
/*      0      |       3 */    char str[3];
/* XXX  1-byte hole      */
/*      4      |       4 */    int num;

                               /* total size (bytes):    8 */
                           }
(gdb)
```

Установим breakpoint до изменения полей структуры в функции “format”, и использовав команды “x/8db”, “x/8dc”, где “x” - команда, отображающая содержимое памяти по заданному адресу в указанном формате, “8” - длина значений, “d” и “c” - формат значений, “b” - модификатор, означает вывод в байтах, “&number” - адресное выражение.

```
(gdb) x/8dc &number
0x5ffe98:       0 '\000'        0 '\000'        0 '\000'        0 '\000'        1 '\0
01'     4 '\004'        0 '\000'        0 '\000'
(gdb) x/8db &number
0x5ffe98:       0       0       0       0       1       4       0       0
```

Теперь используем те же самые команды после вызова функции “format”.

```
(gdb) x/8dc &number
0x5ffe98:       49 '1'  48 '0'  50 '2'  53 '5'  0 '\000'        4 '\004'        0 '\0
00'     0 '\000'
(gdb) x/8db &number
0x5ffe98:       49      48      50      53      0       4       0       0
(gdb)
```

Видно, что первые 5 байт структуры изменили значения, а значит, есть пересечение массива “str” с “num”. В результате программы получаем 1024 потому, что завершающий байт был записан в старший байт “num”. Для исправления этой ошибки нужно увеличить размер массива “srt”.

```
PS D:\VS.main()> gcc -Wall -g -O0 -o Kp2 Kp2.c
PS D:\VS.main()> ./Kp2
str: 1025
num: 1025
PS D:\VS.main()>
```

Задание 3

У нас есть программа, которая рассчитана на вывод выражения (y + 1) умноженное на саму себя.

```
PS D:\VS.main()> gcc -Wall -g -O0 -o Kp3 Kp3.c
PS D:\VS.main()> ./Kp3
z = 11
PS D:\VS.main()>
```

Программа работает некорректно, так как ожидаем на выходе 36, а получаем 11. При помощи отладчика раскроем макрос “SQR”.

```
(gdb) macro expand SQR(y + 1)
expands to: y + 1 * y + 1
(gdb)
```

С помощью команды “macro expand SQR(y + 1)” мы раскрываем вызов макроса “SQR”. Наглядно видно, что макрос написан неправильно, для исправления ошибки следует расставить скобки между произведением.

```
(gdb) macro expand SQR(y + 1)
expands to: (y + 1) * (y + 1)
(gdb)
```

Скомпилируем программу с исправлением и проверим результат.

```
PS D:\VS.main()> gcc -Wall -g -O0 -o Kp3 Kp3.c
PS D:\VS.main()> ./Kp3
z = 36
PS D:\VS.main()>
```

Задание 4

Четвертый номер задания подразумевает получить на выходе программы отсортированный массив ”array” из 6 элементов.

```
PS D:\VS.main()> gcc -Wall -g -O0 -o Kp4 Kp4.c
PS D:\VS.main()> ./Kp4
4 0 5 7 10 15
PS D:\VS.main()>
```

Но программа работает неточно, не так, как задумывалось. Локализуем проблему с помощью отладчика GDB. Для этого пройдемся по циклу в функции “bubble_sort”.

```
12      for (j = 0; j < size - i; ++j) {
1: array[j] = 10
(gdb) display array[j + 1]
2: array[j + 1] = 15
(gdb) display array[0]
3: array[0] = 10
(gdb) display array[1]
4: array[1] = 15
(gdb) display array[2]
5: array[2] = 5
(gdb) display array[3]
6: array[3] = 4
(gdb) display array[4]
7: array[4] = 21
(gdb) display array[5]
8: array[5] = 7
(gdb)
```

Используем команду “display”, чтобы смотреть за значениями в цикле. Замечаем, что цикл работает с ошибками; последний элемент сравнивается со следующим.

```
Thread 1 hit Breakpoint 1, bubble_sort (array=0x5ffcf0, size=6) at Kp4.c:13
13      if (array[j] > array[j + 1]) {
1: array [j] = 21
2: array [j + 1] = 0
3: array [0] = 10
4: array [1] = 5
5: array [2] = 4
6: array [3] = 15
7: array [4] = 7
8: array [5] = 21
```

Для исправления нужно в строке 12 (“ for (j = 0; j < size – i; ++j) ”) после “ size - I " дописать “ - 1 ”, чтобы избежать переполнения.

ПРИЛОЖЕНИЕ

Первое задание:

```c
#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;
}
```

Второе задание:

```c
#include <stdio.h>

typedef struct
{
    char str[10];
    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;
}
```

Третье задание:

```c
#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;
}
```

Четвертое задание:

```c
#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 - 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;
}
```