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. ИВ-222
«6» февраля 2023 г. _____ /Городилов.П.А./

Проверил:
Старший преподаватель Кафедры
ВС
«» февраль 2023 г. _____ /Фульман.В.О./

Оценка «_____»

Новосибирск 2023

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

Получение навыков отладки программ на примере использования отладчика GDB

Задание 1

```
#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

```
#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

```
#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

```
#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: Вывод массива с числами от 0 до 9.
Компиляция-без ошибок. Ошибки при исполнении-ошибка сегментирования.

```
[pavel@fedora Proga 2s]$ gcc -O0 -g main2.c -o main2
[pavel@fedora Proga 2s]$ ./main2
Ошибка сегментирования (образ памяти сброшен на диск)
```

При проверке переменной arr на правильность действий в функции ошибок не замечаем, но при проверки адреса переменной arr мы видим что адрес в функции init и в main у переменной arr разный. Из этого можно сделать вывод что поскольку выделение памяти через malloc происходит в функции где переменная локальная, то при выходе из функции arr не изменяется.

```
6           for (i = 0; i < n; ++i) {
1: *arr@10 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
2: &arr = (int **) 0x7fffffffdf48
(gdb) n
9       }
1: *arr@10 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
2: &arr = (int **) 0x7fffffffdf48
(gdb) n
main () at main2.c:15
15          for (i = 0; i < n; ++i) {
(gdb) display &arr
3: &arr = (int **) 0x7fffffffdf80
(gdb) display *arr@10
4: *arr@10 = <error: Cannot access memory at address 0x0>
(gdb) n
16              printf("%d\n", arr[i]);
3: &arr = (int **) 0x7fffffffdf80
4: *arr@10 = <error: Cannot access memory at address 0x0>
(gdb)

Program received signal SIGSEGV, Segmentation fault.
0x00000000004011d1 in main () at main2.c:16
16              printf("%d\n", arr[i]);
3: &arr = (int **) 0x7fffffffdf80
4: *arr@10 = <error: Cannot access memory at address 0x0>
(gdb)

Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
```

Для того чтобы решить данную проблему можно изначально переменную arr в main задавать через malloc.

```
 4 void init(int* arr, int n)
 5 {
 6     int i;
 7     for (i = 0; i < n; ++i)
 8     {
 9         arr[i] = i;
10     }
11 }
12
13 int main()
14 {
15     int n=10;
16     int* arr = malloc(n*sizeof(int));
17     init(arr, n);
18     int i;
19     for (i = 0; i < n; ++i)
20     {
21         printf("%d\n", arr[i]);
22     }
23     return 0;
24 }
```

```
[pavel@fedora Proga 2s]$ ./main
0
1
2
3
4
5
6
7
8
9
```

Задание 2: Инициализация полей структуры одним числом в символьном и целочисленном формате

Компиляция-без ошибок. Ошибки при исполнении-неправильный вывод.

```
[pavel@fedora Proga 2s]$ gcc -O0 -g tipedef2.c -o tipedef2
[pavel@fedora Proga 2s]$ ./tipedef2
str: 1025
num: 1024
```

Переменные в памяти располагаются друг за другом без пробелов,

```
Breakpoint 1, main () at tipedef2.c:11
11          NumberRepr number = {.num = 1025};
(gdb) n
12          format(&number);
(gdb) n
13          printf("str: %s\n", number.str);
(gdb) p &number.num
$1 = (int *) 0x7fffffffdf8c
(gdb) p&number.str[4]
$2 = 0x7fffffffdf8c ""
```

Чтобы исправить ошибки меняем переменные в структуре местами и увеличиваем массив до 5. При отработке format символы, не вместившиеся в str записываются в num.

```
typedef struct
{
    int num;
    char str[5];
} NumberRepr;
```

```
[pavel@fedora Proga 2s]$ ./tipedef
str: 1025
num: 1025
```

Задание 3: Возведение числа в квадрат

Компиляция-без ошибок. Ошибки при исполнении-Возведение в квадрат работает не правильно, получаем какое то странное число при выводе

```
[pavel@fedora Proga 2s]$ gcc -gdwarf-2 -g3 sqr.c -o sqr
[pavel@fedora Proga 2s]$ ./sqr
z = 11
```

При проверке в отладчике gdb макроса мы сразу же видим ошибку в порядке действий макроса при данном нам условии.

```
Breakpoint 1, main () at sqr.c:4
4           int y = 5;
(gdb) n
5           int z = SQR(y + 1);
(gdb)
6           printf("z = %d\n", z);
(gdb) macro expand SQR(y+1)
expands to: y+1 *y+1
```

Для решения данной проблемы достаточно поставить скобочки на x в макросе, тогда действия будут идти в том порядке, в котором нам нужно.

```
1 #include <stdio.h>
2 #define SQR(x) (x) * (x)
3 int main()
```

```
[pavel@fedora Proga 2s]$ gcc -O0 -g sql.c -o sql
[pavel@fedora Proga 2s]$ ./sql
z = 36
```

Задание 4: Пузырьковая сортировка

Компиляция-без ошибок. Ошибки при исполнении-Максимальное значение массива заменяется 0.

```
[pavel@fedora Proga 2s]$ gcc -O0 -g sort2.c -o sort2
[pavel@fedora Proga 2s]$ ./sort2
4 0 5 7 10 15
```

При проверки в отладчике gdb и выводе переменных мы замечаем ошибку во втором форе функции bubble_sort, а именно выполняется лишнее действие из за того что j равное 5 удовлетворяет условию цикла. Из за чего у нас осуществляется лишнее сравнение, а из за того что массив на сто элементов то сравнение 21 происходит с 0 и они благополучно меняются. При выводе мы видим только 6 элементов, поэтому ошибка не сразу заметна.

```
Breakpoint 1, bubble_sort (array=0x7fffffffddf0, size=6) at sort2.c:9
9         for (i = 0; i < size - 1; ++i) {
(gdb) n
10          for (j = 0; j < size - i; ++j) {
(gdb) n
11            if (array[j] > array[j + 1]) {
(gdb) display i
1: i = 0
(gdb) display j
2: j = 0
(gdb) display *array@7
3: *array@7 = {10, 15, 5, 4, 21, 7, 0}
(gdb) n
10          for (j = 0; j < size - i; ++j) {
1: i = 0
2: j = 0
3: *array@7 = {10, 15, 5, 4, 21, 7, 0}
(gdb)
```

```
3: *array@7 = {10, 5, 4, 15, 7, 21, 0}
(gdb)
11            if (array[j] > array[j + 1]) {
1: i = 0
2: j = 5
3: *array@7 = {10, 5, 4, 15, 7, 21, 0}
(gdb)
12              swap(&array[j], &array[j + 1]);
1: i = 0
2: j = 5
3: *array@7 = {10, 5, 4, 15, 7, 21, 0}
(gdb)
10          for (j = 0; j < size - i; ++j) {
1: i = 0
2: j = 5
3: *array@7 = {10, 5, 4, 15, 7, 0, 21}
(gdb)
9         for (i = 0; i < size - 1; ++i) {
1: i = 0
2: j = 6
3: *array@7 = {10, 5, 4, 15, 7, 0, 21}
```

Чтобы решить данную проблему, достаточно во втором for функции в неравенстве j<size-i отнимать от size-i -1. В таком случае мы не будем делать лишнее сравнение и сможем

```
 9 void bubble_sort(int* array, int size)
10 {
11     int i, j;
12     for (i = 0; i < size - 1; ++i) {
13         for (j = 0; j < size - i-1; ++j) {
14             if (array[j] > array[j + 1]) {
15                 swap(&array[j], &array[j + 1]);
16             }
17         }
18     }
19 }
```

отсортировать всё правильно.

```
[pavel@fedora Proga 2s]$ ./sort
4 5 7 10 15 21
```

ПРИЛОЖЕНИЕ

main.c

```c
#include <stdio.h>
#include <stdlib.h>

void init(int* arr, int n)
{
    int i;
    for (i = 0; i < n; ++i)
    {
        arr[i] = i;
    }
}

int main()
{
    int n=10;
    int* arr = malloc(n*sizeof(int));
    init(arr, n);
    int i;
    for (i = 0; i < n; ++i)
    {
        printf("%d\n", arr[i]);
    }
    return 0;
}
```

tipedef.c

```c
#include <stdio.h>

typedef struct
{
    int num;
    char str[3];
} 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;
}
```

sql.c

```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;
}
```

sort.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;
}
```