

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-242

«25» февраля 2023 г.

/Пеалкиви Д.Я./

Проверил:

Ст. преподаватель Кафедры ВС

«__» марта 2023 г.

/Фульман В.О./

Оценка «_____»

Новосибирск 2023

ОГЛАВЛЕНИЕ

ЗАДАНИЕ.....	3
ВЫПОЛНЕНИЕ РАБОТЫ.....	5
ПРИЛОЖЕНИЕ.....	10

ЗАДАНИЕ

Найти и исправить с помощью отладчика ошибки в программах :

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

При попытке запуска программы получаем следующую ошибку:

```
theavangard@DESKTOP-H548M8Q:~/prog2023$ ./task
Segmentation fault
```

Segmentation fault (ошибка сегментации) - ошибка программного обеспечения, возникающая при попытке обращения к недоступным для записи участкам памяти либо при попытке изменить память запрещенным способом.

Для определения причины неисправности перекомпилируем программу без оптимизации и с отладочной информацией:

```
theavangard@DESKTOP-H548M8Q:~/prog2023$ gcc -Wall -g -O0 -o task task.c
```

Запускаем её в отладчике gdb и получаем ошибку в 20 строчке в функции printf:

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

При попытке вывода массива arr[i] получаем ошибку: Cannot access memory at address 0x0 (Не удастся получить доступ к памяти по адресу 0x0):

```
(gdb) print arr[i]
Cannot access memory at address 0x0
```

Из чего можно сделать вывод, что адрес массива arr[i] в функции init и в функции main отличается.

Для решения данной проблемы необходимо добавить указатель на функцию init и в конце вернуть переменную arr, чтобы не получить ошибку при запуске программы:

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

Также, в функции main необходимо изменить вызов функции init, приписав arr =, т.к. был добавлен указатель:

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

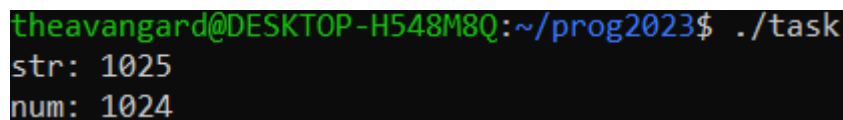
Теперь программа работает исправно и выводит массив:



```
theavangard@DESKTOP-H548M8Q:~/prog2023$ ./task
0
1
2
3
4
5
6
7
8
9
```

Задание 2

Запускаем программу и на выходе получаем значения строки и числа, 1025 и 1024 соответственно:



```
theavangard@DESKTOP-H548M8Q:~/prog2023$ ./task
str: 1025
num: 1024
```

По данным значениям сразу можно понять, что программа содержит ошибку, т.к. в числе она должна выводить то же значение, что и в строке, а именно 1025.

С помощью команды `break` отладчика `gdb`, ставим breakpoint (точку останова) на 17 строчку, где содержится вывод значения строки и запускаем программу командой `run`:

```
(gdb) break 17
Breakpoint 1 at 0x11f4: file task.c, line 17.
(gdb) run
Starting program: /home/theavangard/prog2023/task
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at task.c:17
17          printf("str: %s\n", number.str);
```

Далее выводим значение `number.str` и замечаем, что оно выводится не полностью:

```
(gdb) print number.str
$1 = "102"
```

Соответственно, в структуре необходимо увеличить массив `char str[]` со значения 3 до 5, т.к. одна ячейка пойдет под значение числа, а ещё одна ячейка нужна для нулевого символа, который всегда завершает строку:

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

Теперь программа работает корректно и выводит правильные значения:

```
theavangard@DESKTOP-H548M8Q:~/prog2023$ ./task
str: 1025
num: 1025
```

Задание 3

Запускаем программу и на выходе получаем значение переменной `z`:

```
theavangard@DESKTOP-H548M8Q:~/prog2023$ ./task
z = 11
```

Что уже указывает на ошибку в программе, т.к. значение переменной `z` некорректно, потому что в ней выводится значение $(y+1)$, а переменная `y` в программе равна 5, соответственно, вместо полученных 11, должно получиться $(5 + 1)^2 = 36$.

Проверим, правильно ли работает написанный макрос, для этого перекомпилируем программу со включением дополнительной информации, используя опцию `-g3` и `-gdwarf-2`:

```
theavangard@DESKTOP-H548M8Q:~/prog2023$ gcc -Wall -g3 -O0 -o task task.c
```

Запускаем отладчик gdb, ставим breakpoint на функцию main, запускаем программу командой run и вводим команду macro expand, затем название нашего макроса и используемую в программе формулу:

```
(gdb) break main
Breakpoint 1 at 0x1155: file task.c, line 6.
(gdb) run
Starting program: /home/theavangard/prog2023/task
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at task.c:6
6      int y = 5;
(gdb) macro expand SQR(5+1)
expands to: 5+1 * 5+1
```

Что и следовало ожидать, сам по себе макрос работает, но отсутствие в нём скобок приводит к тому, что сперва выполняется умножение $1 * 5 = 5$, а затем идёт и сумма $5 + 6 = 11$.

Добавляем скобки в макрос:

```
#define SQR(x) (x) * (x)
```

Теперь макрос работает корректно и программа выводит правильное число:

```
theavangard@DESKTOP-H548M8Q:~/prog2023$ ./task
z = 36
```

Задание 4

Запускаем программу и на выходе получаем что-то вроде отсортированного массива, но, во-первых, его сортировка не до конца успешна, а во-вторых, отсутствует один из его элементов:

```
theavangard@DESKTOP-H548M8Q:~/prog2023$ ./task
4 0 5 7 10 15
```


Запускаем отладчик gdb и ставим breakpoint на сортировочный цикл, для того чтобы посмотреть на массив:

```
(gdb) break 11
Breakpoint 1 at 0x11cd: file task.c, line 11.
(gdb) run
Starting program: /home/theavangard/prog2023/task
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, bubble_sort (array=0x7fffffffdfa0, size=6) at task.c:11
11      for (i = 0; i < size - 1; ++i) {
(gdb) display array[0]
1: array[0] = 10
(gdb) display array[1]
2: array[1] = 15
(gdb) display array[2]
3: array[2] = 5
(gdb) display array[3]
4: array[3] = 4
(gdb) display array[4]
5: array[4] = 21
(gdb) display array[5]
6: array[5] = 7
```

Возможно, что число 10, расположенное в нулевой ячейке массива, является не воспринимаемым для указателя в функции, поэтому сортировка выполняется некорректно. Стоит задать переменной значение $i = 1$ и убрать -1 у $size$, чтобы не потерять элемент массива:

```
for (i = 1; i < size; ++i)
```

Теперь программа работает исправно, последний элемент не пропадает и массив сортируется полностью:

```
theavangard@DESKTOP-H548M8Q:~/prog2023$ ./task
4 5 7 10 15 21
```

ПРИЛОЖЕНИЕ

Задание 1

```
#include <stdio.h>
#include <stdlib.h>
int *init(int* arr, int n)
{
    arr = malloc(n * sizeof(int));
    int i;
    for (i = 0; i < n; ++i)
    {
        arr[i] = i;
    }
    return arr;
}
int main()
{
    int* arr = NULL;
    int n = 10;
    arr = 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[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;
}
```

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 = 1; i < size ; ++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;
}
```