Отчет по заданию №3 в рамках вычислительного практикума. Отладка

Задание №1

1.1 Программа task_01.c

```
skiper22@skiper22:~/Рабочий стол$ gcc -std=c99 -Wall -Werror -g task_01.c -o
app.exe
skiper22@skiper22:~/Рабочий стол$ gdb ./app.exe
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "aarch64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./app.exe...
(gdb) l
1       #include <stdio.h>
2
3       long long unsigned factorial(unsigned n);
4
5       int main(void)
6       {
7           unsigned n;
8           long long unsigned result;
9
10          printf("Input n: ");
(gdb) l
11          if (scanf("%u", &n) != 1)
12          {
13              printf("Input error");
14              return 1;
15          }
16
17          result = factorial(n);
18
19          printf("factorial(%u) = %llu\n", n, result);
20
(gdb) l
21          return 0;
22      }
23
```

```
24      long long unsigned factorial(unsigned n)
25      {
26          long long unsigned result = 1;
27
28          while (n--)
29              result *= n;
30          return result;
(gdb) l
31      }
32
(gdb) break 28
Breakpoint 1 at 0x94c: file task_01.c, line 28.
(gdb) break 29
Breakpoint 2 at 0x950: file task_01.c, line 29.
(gdb) run
Starting program: /home/skiper22/Рабочий стол/app.exe
Input n: 5

Breakpoint 1, factorial (n=5) at task_01.c:28
28          while (n--)
(gdb) print n
$1 = 5
(gdb) next

Breakpoint 2, factorial (n=4) at task_01.c:29
29              result *= n;
(gdb) print n
$2 = 4
(gdb) print result
$3 = 1
(gdb) next
28          while (n--)
(gdb) print n
$4 = 4
(gdb) next

Breakpoint 2, factorial (n=3) at task_01.c:29
29              result *= n;
(gdb) print n
$5 = 3
(gdb) print result
$6 = 4
(gdb) next
28          while (n--)
(gdb) p n
$7 = 3
(gdb) next

Breakpoint 2, factorial (n=2) at task_01.c:29
29              result *= n;
(gdb) p n
$8 = 2
(gdb) p result
$9 = 12
(gdb) n
28          while (n--)
(gdb) p n
$10 = 2
(gdb) next
```

```
Breakpoint 2, factorial (n=1) at task_01.c:29
29              result *= n;
(gdb) p n
$11 = 1
(gdb) p result
$12 = 24
(gdb) n
28          while (n--)
(gdb) p n
$13 = 1
(gdb) next

Breakpoint 2, factorial (n=0) at task_01.c:29
29              result *= n;
(gdb) p result
$14 = 24
(gdb) p n
$15 = 0
(gdb) next
28          while (n--)
(gdb) p n
$16 = 0
(gdb) n
30          return result;
(gdb) p result
$17 = 0
(gdb) next
31      }
(gdb) continue
Continuing.
factorial(5) = 0
[Inferior 1 (process 8169) exited normally]
(gdb)
```

После компиляции программы и получения исполняемого файла запускаем отладчик gdb.

С помощью команды list, которая по умолчанию выводит 10 строк кода , выводим код программы. Делаем точки останова на 28 и 29 строках, там где находится главная часть функции в поиске факториала числа. Запускаем с помощью команды run программу и вводим значение переменной n. С помощью команды (next=n) переходим к следующей операции, а с помощью команды (print = p) выводим значения переменных. Функция factorial работала неверно из-за условия в цикле while, из-за чего умножение чисел состояло из числе от n-1 до 0 включая 0. Поэтому на любое введенное число n выводился результат равный нулю.

Исправленный код программы :

```c
#include <stdio.h>

long long unsigned factorial(unsigned n);

int main(void)
{
    unsigned n;
    long long unsigned result;
```

```
    printf("Input n: ");
    if (scanf("%u", &n) != 1)
    {
        printf("Input error");
        return 1;
    }

    result = factorial(n);

    printf("factorial(%u) = %llu\n", n, result);

    return 0;
}

long long unsigned factorial(unsigned n)
{
    long long unsigned result = 1;

    while (n>0)
    {
        result *= n;
        n = n - 1;
    }
    return result;
}
```

1.2 Программа task_02.c

```
skiper22@skiper22:~/Рабочий стол$ gcc -std=c99 -Wall -Werror -g task_02.c -o
app.exe
skiper22@skiper22:~/Рабочий стол$ gdb ./app.exe
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "aarch64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./app.exe...
(gdb) l
1       #include <stdio.h>
2
3       #define N 5
4
5       double get_average(const int a[], size_t n);
6
7       int get_max(const int *a, size_t n);
8
9       int main()
```

```
10      {
(gdb) l
11          int arr[N];
12          size_t i;
13
14          printf("Enter %d numbers:\n", N);
15
16          for (i = 0; i < N; i++)
17          {
18              printf("Enter the next number: ");
19              if (scanf("%d", &arr[1]) != 1)
20              {
(gdb) l
21                  printf("Input error");
22                  return 1;
23              }
24          }
25
26          for (i = 1; i < N; i++)
27              printf("Value [%zu] is %d\n", i, arr[i]);
28
29          printf("The average is %g\n", get_average(arr, N));
30
(gdb) l
31          printf("The max is %d\n", get_max(arr, N));
32
33          return 0;
34      }
35
36      double get_average(const int a[], size_t n)
37      {
38          double temp = 0.0;
39
40          for (size_t i = 0; i > n; i++)
(gdb) l
41              temp += a[i];
42          temp /= n;
43
44          return temp;
45      }
46
47      int get_max(const int *a, size_t n)
48      {
49          int max = a[0];
50
(gdb) l
51          for (size_t i = 1; i < n; i++)
52              if (max > a[i])
53                  max = a[i];
54
55          return max;
56      }
57
(gdb) l
Line number 58 out of range; task_02.c has 57 lines.
```

Ошибки программы: первая ошибка, которая сразу бросается в глаза находится на 19 строке , где вводится массив из 5 чисел. При вводе каждый раз перезаписывается 1-ый элемент массива . Вторая ошибка расположена на 26 строке, в цикле for, где отсчет

начинается с 1 , а не с нуля , как это должно быть. Третья ошибка находится в функции get_average на 40 строке в цикле for . Условие в цикле неправильное, из-за чего цикл не работает. Четвертая ошибка в функции get_max на 52 строке заключается в неправильной операции после условия .

Исправленный код программы:

```c
#include <stdio.h>

#define N 5

double get_average(const int a[], size_t n);

int get_max(const int *a, size_t n);

int main()
{
    int arr[N];
    size_t i;

    printf("Enter %d numbers:\n", N);

    for (i = 0; i < N; i++)
    {
        printf("Enter the next number: ");
        if (scanf("%d", &arr[i]) != 1)
        {
            printf("Input error");
            return 1;
        }
    }

    for (i = 0; i < N; i++)
        printf("Value [%zu] is %d\n", i, arr[i]);

    printf("The average is %g\n", get_average(arr, N));

    printf("The max is %d\n", get_max(arr, N));

    return 0;
}

double get_average(const int a[], size_t n)
{
    double temp = 0.0;

    for (size_t i = 0; i < n; i++)
        temp += a[i];
    temp /= n;

    return temp;
}

int get_max(const int *a, size_t n)
{
    int max = a[0];

    for (size_t i = 1; i < n; i++)
```

```
        if (max < a[i])
            max = a[i];

    return max;
}
```

1.3 Программа task_03.c

Ошибка в программе заключается в том, что программа не проверяя делитель, пытается выполнить целочисленное деление.

Исправленный код программы:

```
#include <stdio.h>

int div(int a, int b);

int main(void)
{
    int a = 5, b = 2;

    printf("%d div %d = %d\n", a, b, div(a, b));

    a = 10;
    b = 0;
    if (div(a, b) == -1)
        printf("Error, divisor value is 0");
    else
        printf("%d div %d = %d\n", a, b, div(a, b));

    return 0;
}

int div(int a, int b)
{
    if (b == 0)
        return -1;
    return a / b;
}
```

Задание №2

Таблица размеров типов данных на Linux x64

Тип данных	Размер (в байтах)
char	1
int	4
unsigned	4
long long	8
short	2
int32_t	4
int64_t	8

Таблица размеров типов данных на 32-х битной машине

Тип данных	Размер (в байтах)
char	1
int	4
unsigned	4
long long	8
short	2
int32_t	8
int64_t	Не поддерживается

Задание №3

```
skiper22@skiper22:~/Рабочий стол$ gcc -std=c99 -Wall -Werror -g main.c -o
app.exe
skiper22@skiper22:~/Рабочий стол$ gdb ./app.exe
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "aarch64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./app.exe...
(gdb) l
1       #include <stdio.h>
2       #include <stdint.h>
3
4       int main(void)
5       {
6           char c='$';
7           printf("char %c\n",c);
8           int i=1;
9           printf("int %d\n",i);
10          int i_=-1;
(gdb)
11          printf("-int %d\n",i_);
12          unsigned u = 120;
13          printf("unsigned %u\n",u);
14          long long l=109;
15          printf("long long %lld\n",l);
16          long long l_=-120;
17          printf("-long long %lld\n",l_);
18
19
20          return 0;
(gdb) b 20
```

```
Breakpoint 1 at 0xaaaaaaaaa808: file main.c, line 20.
(gdb) run
Starting program: /home/skiper22/Рабочий стол/app.exe
char $
int 1
-int -1
unsigned 120
long long 109
-long long -120

Breakpoint 1, main () at main.c:20
20          return 0;
(gdb) x /1xb &c
0xffffffffef43: 0x24
(gdb) x /4xb &i
0xffffffffef44: 0x01    0x00    0x00    0x00
(gdb) x /4xb &i_
0xffffffffef48: 0xff    0xff    0xff    0xff
(gdb) x /4xb &u
0xffffffffef4c: 0x78    0x00    0x00    0x00
(gdb) x /8xb &l
0xffffffffef50: 0x6d    0x00    0x00    0x00    0x00    0x00    0x00    0x00
(gdb) x /8xb &l_
0xffffffffef58: 0x88    0xff    0xff    0xff    0xff    0xff    0xff    0xff
(gdb)
```

После запуска программы ставим точку останова на завершении программы (return 0) чтобы выполнилась инициализация переменных, но программа не завершилась. После этого используя конструкцию:

x /[размер типа в байтах]xb &[имя переменной]

выводим представление переменных наших типов.

Задание №4

```
skiper22@skiper22:~/Рабочий стол$ gcc -std=c99 -Wall -Werror -g m.c -o
app.exe
skiper22@skiper22:~/Рабочий стол$ gdb ./app.exe
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "aarch64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./app.exe...
(gdb) l
```

```
1       #include <stdio.h>
2       #include <stdlib.h>
3       #define N 7
4
5       int main()
6       {
7           int arr[N] = {0, 1, 2, 6, 24, 120, 720};
8           for (int i = 0; i < N; i++)
9               printf("%d \n",arr[i]);
10          return 0;
(gdb)
11      }
12
(gdb) break 10
Breakpoint 1 at 0x8bc: file m.c, line 10.
(gdb) run
Starting program: /home/skiper22/Рабочий стол/app.exe
0
1
2
6
24
120
720

Breakpoint 1, main () at m.c:10
10          return 0;
(gdb) x /28xb &arr
0xffffffffef38: 0x00    0x00    0x00    0x00    0x01    0x00    0x00    0x00
0xffffffffef40: 0x02    0x00    0x00    0x00    0x06    0x00    0x00    0x00
0xffffffffef48: 0x18    0x00    0x00    0x00    0x78    0x00    0x00    0x00
0xffffffffef50: 0xd0    0x02    0x00    0x00
```

```
skiper22@skiper22:~/Рабочий стол$ gdb ./app.exe
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "aarch64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./app.exe...
(gdb) break 10
Breakpoint 1 at 0x8bc: file m.c, line 10.
(gdb) break 13
```

```
Breakpoint 2 at 0x8ec: file m.c, line 13.
(gdb) break 14
Breakpoint 3 at 0x908: file m.c, line 14.
(gdb) run
Starting program: /home/skiper22/Рабочий стол/app.exe
0
1
2
6
24
120
720

Breakpoint 1, main () at m.c:10
10          int *p = arr;
(gdb) n
11          printf("%p %d\t", p, *p);
(gdb) p p
$1 = (int *) 0xffffffffef38
(gdb) n
12          p+=5;
(gdb) n

Breakpoint 2, main () at m.c:13
13          printf("%p %d", p, *p);
(gdb) p p
$2 = (int *) 0xffffffffef4c
```

Задание №5

```
skiper22@skiper22:~/Рабочий стол$ gcc -std=c99 -Wall -Werror -g failcode.c -
o app.exe
skiper22@skiper22:~/Рабочий стол$ gdb ./app.exe
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "aarch64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./app.exe...
(gdb) l
1       #include <stdio.h>
2
3       int main(void)
```

```
4       {
5           int a = 8, b = 10;
6           if (a<b)
7               b+=(b-a);
8           printf("a=%d b=%d => a=b",a,b);
9           return 0;
10      }
(gdb)
11
(gdb) b 7
Breakpoint 1 at 0x794: file failcode.c, line 7.
(gdb) run
Starting program: /home/skiper22/Рабочий стол/app.exe

Breakpoint 1, main () at failcode.c:7
7               b+=(b-a);
(gdb) watch b
Hardware watchpoint 2: b
(gdb) n

Hardware watchpoint 2: b

Old value = 10
New value = 12
main () at failcode.c:8
8           printf("a=%d b=%d => a=b",a,b);
```

Задание №6

Команда в gdb	Команда в Qt Creator
Точка останова: break [номер строки]- команда останавливается на этой строке	Точка останова: в окне редактора в конкретной строке, на которой Вы хотите остановить программу, нажмите F9
Завершение отладки : quit или сокращенно q	Завершение отладки : Shift + F5
Выполнение строки кода в целом : next	Выполнение строки кода в целом : F10
Выполнение с входом в функцию : step	Выполнение с входом в функцию :F11
Продолжить работу программы : continue	Продолжить работу программы : F5
	Переход к выбранной функции при вступлении во вложенную функцию: Ctrl+F6.
	Переход к строке, содержащей курсор: Ctrl+F10
	Оставить текущую функцию или подфункцию : Shift+F11