

Пензенский государственный университет

Кафедра «Вычислительная техника»

## ОТЧЕТ

по лабораторной работе №3

по дисциплине: «Арифметические и логические основы  
вычислительной техники»

на тему: «Формат представления чисел с ПТ в цифровых процессорах»

Выполнил:

студент группы \_\_\_\_\_

\_\_\_\_\_

Принял:

\_\_\_\_\_

Пенза, 2023

## Ход работы

### Целые числа

1. Представил числа **a=32** и **b=-77** в формате короткое вещественное(KB).

$$a = 32_{10} = 10\ 0000_2 = 1,000\ 0000\ 0000\ 0000\ 0000\ 0000 * 10^{101}$$

$$\text{порядок: } 111\ 1111 + 101 = 1000\ 0100$$

$$0\ 100\ 0010\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 = 4200\ 0000_{16}$$

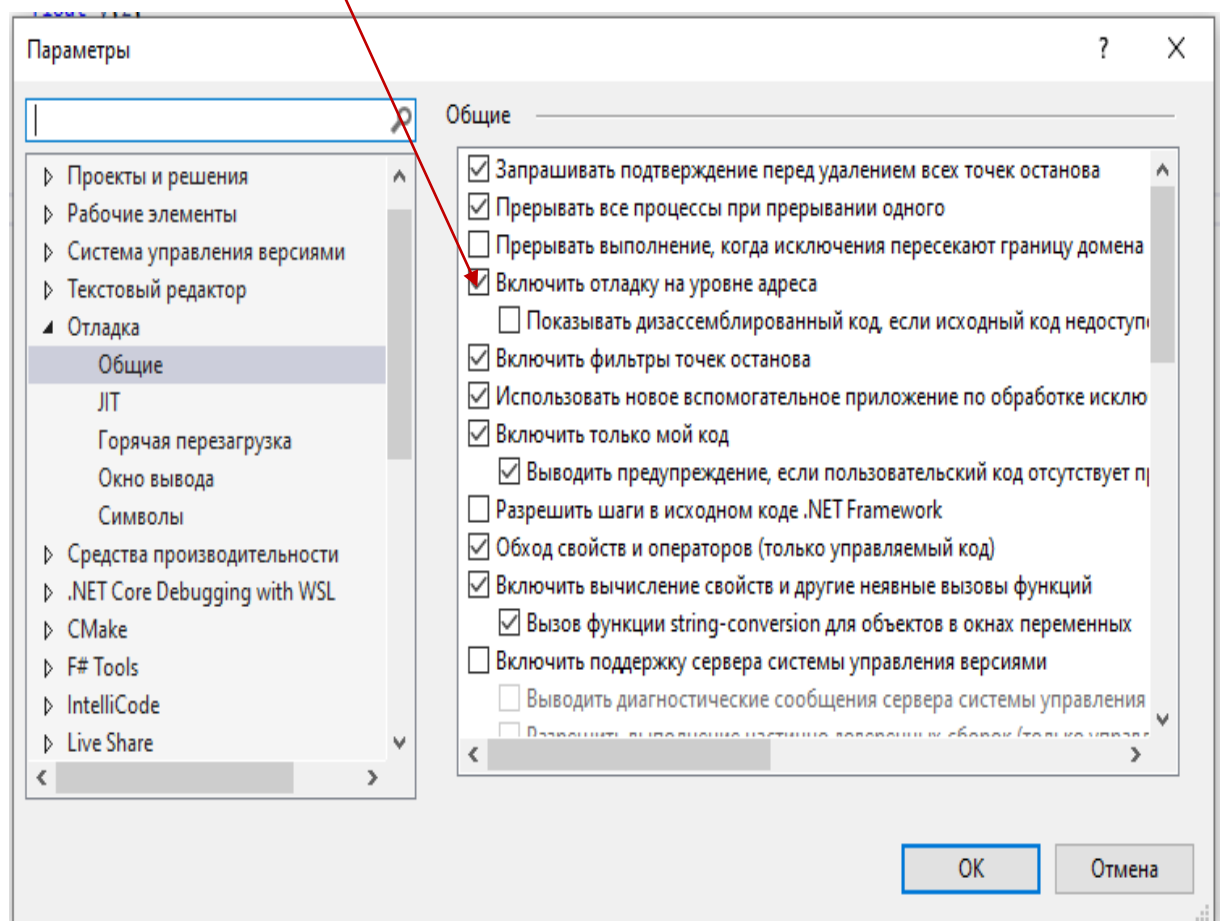
$$b = -77_{10} = -100\ 1101_2 = -1,001\ 1010\ 0000\ 0000\ 0000\ 0000 * 10^{110}$$

$$\text{порядок: } 111\ 1111 + 110 = 1000\ 0101$$

$$1\ 100\ 0010\ 1001\ 1010\ 0000\ 0000\ 0000\ 0000 = c29a\ 0000_{16}$$

2. Для проверки правильности использую отладчик Visual Studio.

Включаю окно «Памяти». Чтобы включить окна **Память**, необходимо выбрать параметр **Отладка>Параметры >Отладка>Общие>Включить отладку на уровне адреса**.

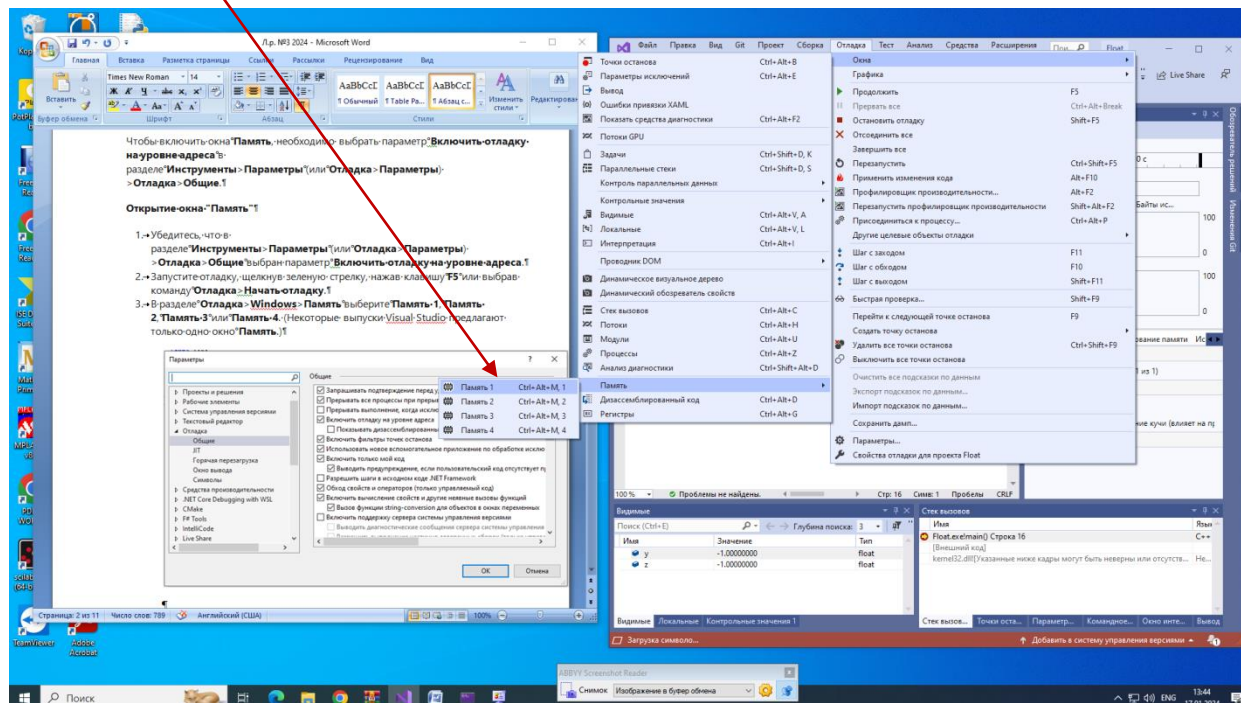


Проект для проверки:

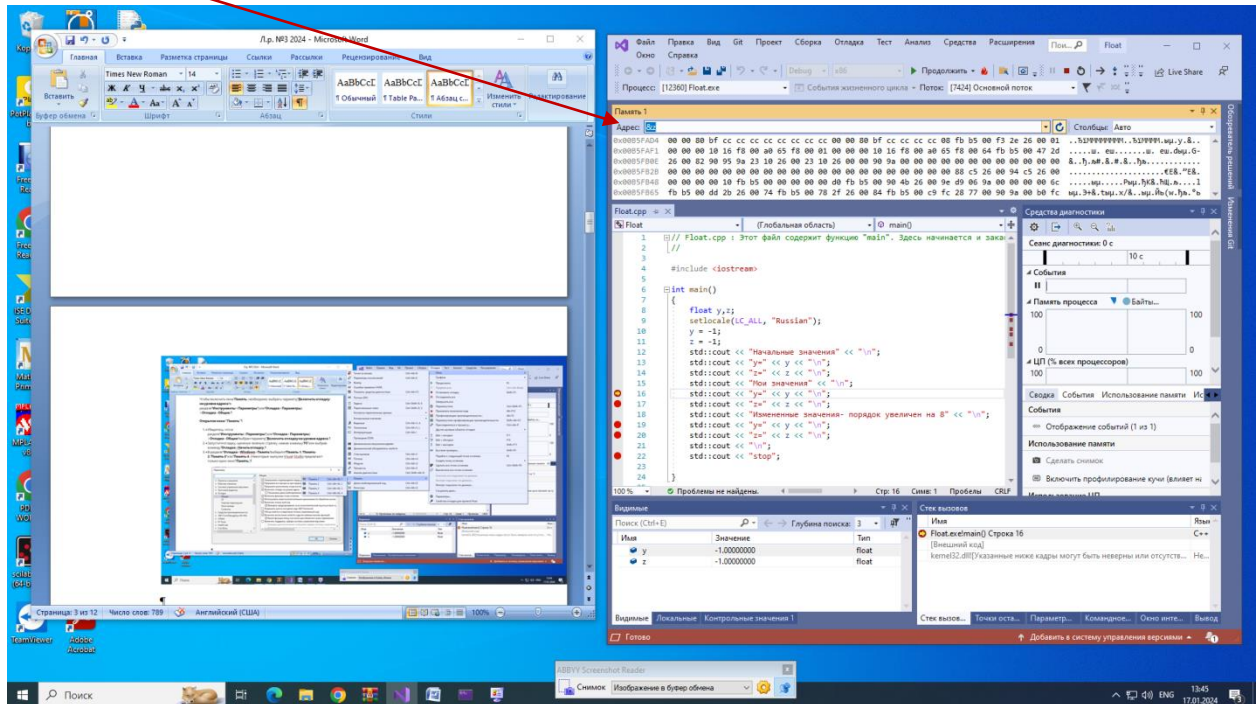
```
1 // Float.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается код программы.
2 //
3
4 #include <iostream>
5
6 int main()
7 {
8     float y,z;
9     setlocale(LC_ALL, "Russian");
10    y = -1;
11    z = -1;
12    std::cout << "Начальные значения" << "\n";
13    std::cout << "y=" << y << "\n";
14    std::cout << "z=" << z << "\n";
15    std::cout << "Мои значения" << "\n";
16    std::cout << "y=" << y << "\n";
17    std::cout << "z=" << z << "\n";
18    std::cout << "Измененные значения- порядок увеличен на 8" << "\n";
19    std::cout << "y=" << y << "\n";
20    std::cout << "z=" << z << "\n";
21    std::cout << "\n";
22    std::cout << "stop";
23 }
24
```

Задаю точки останова в строках кода, где будут проверяться значения переменных. Запускаю отладку, щелкнув зеленую стрелку, нажав клавишу **F5** или **Отладка>Начать отладку**.

Показываю окно «Память». Для этого **Отладка>Windows>Память** выбираю **Память 1**.

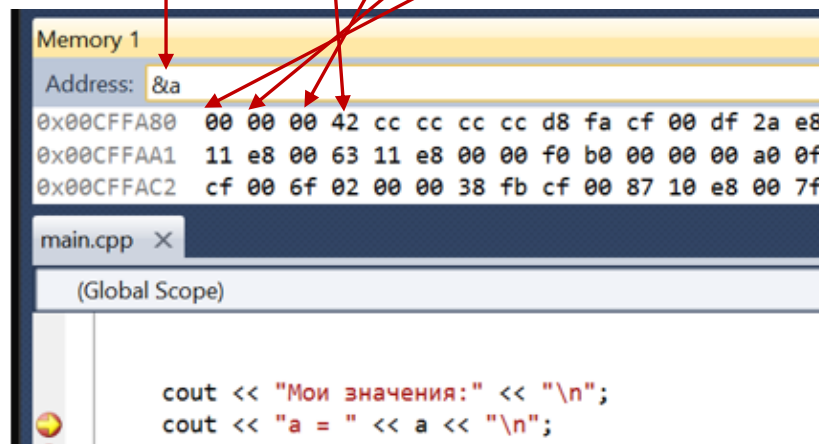


Получил вывод окна памяти:

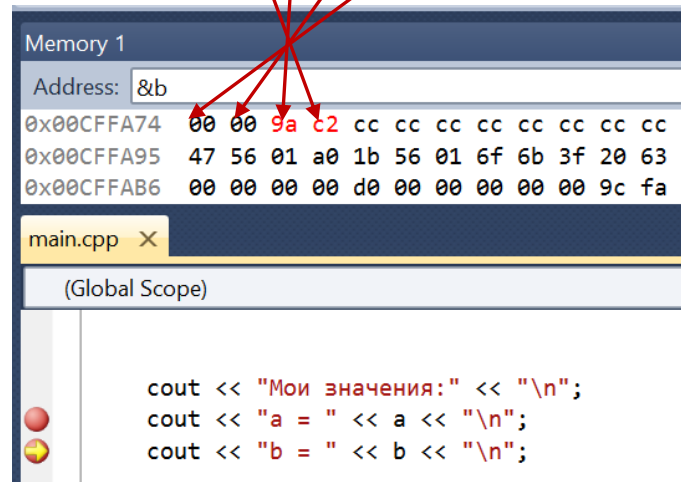


В точке выполненного останова с использованием отладчика, выбрав в памяти адрес переменной **a**, разместил полученные представления чисел в памяти.

Число  $a = 4200\ 0000_{16}$



Число  $b = c29a\ 0000_{16}$



Выполнил программу и получил подтверждение правильности представления чисел **a** и **b** в формате КВ с ПТ.

```
Мои значения:
a = 32
b = -77
```

3. Увеличил порядок каждого операнда на  $1000_2$ .

**a:**  $1000\ 0100 + 1000 = 1000\ 1100$

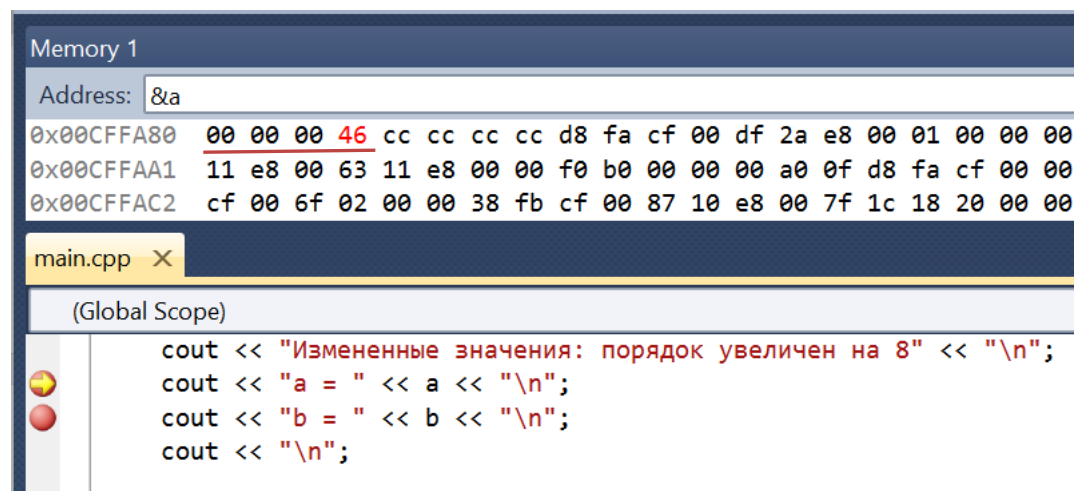
**0**  $100\ 0110\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 = 4600\ 0000_{16}$

**b:**  $1000\ 0101 + 1000 = 1000\ 1101$

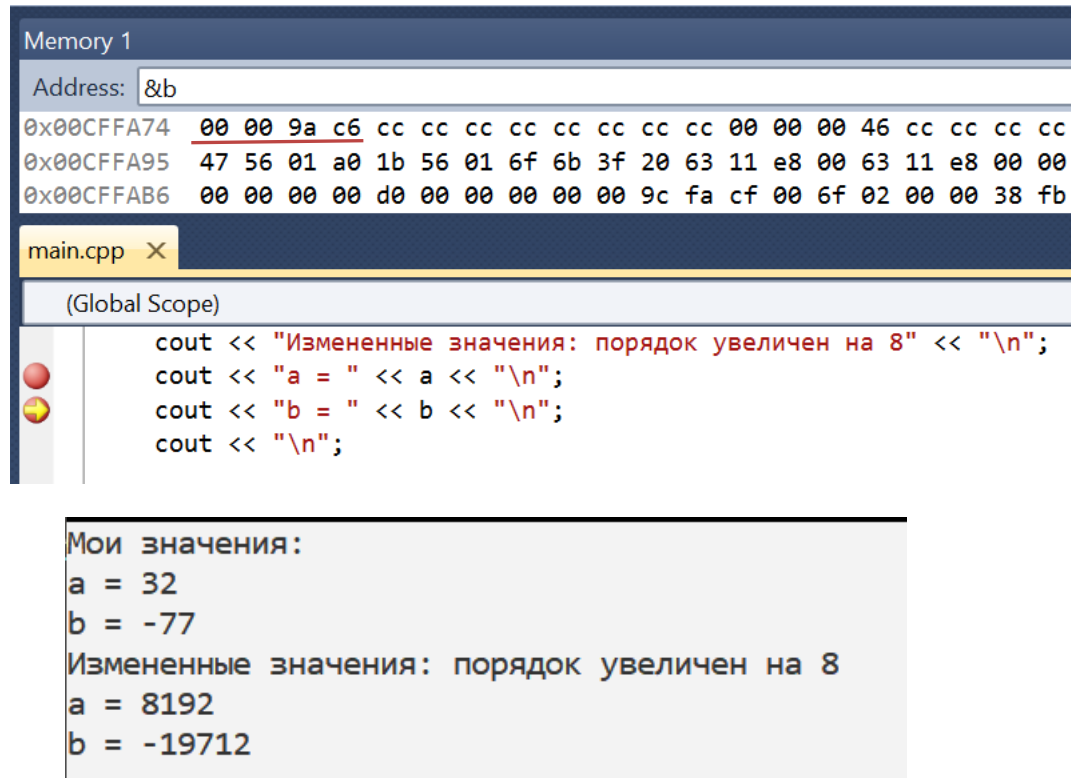
**1**  $100\ 0110\ 1001\ 1010\ 0000\ 0000\ 0000\ 0000 = C69A0000_{16}$

4. Разместил полученные представления чисел в памяти.

**a:**



**b:**



Memory 1

Address: &b

0x00CFFA74	00 00 9a c6 cc cc cc cc cc cc cc cc 00 00 00 46 cc cc cc cc
0x00CFFA95	47 56 01 a0 1b 56 01 6f 6b 3f 20 63 11 e8 00 63 11 e8 00 00
0x00CFFAB6	00 00 00 00 d0 00 00 00 00 00 9c fa cf 00 6f 02 00 00 38 fb

main.cpp X

(Global Scope)

```
cout << "Измененные значения: порядок увеличен на 8" << "\n";  
cout << "a = " << a << "\n";  
cout << "b = " << b << "\n";  
cout << "\n";
```

Мои значения:  
a = 32  
b = -77  
Измененные значения: порядок увеличен на 8  
a = 8192  
b = -19712

5. Проверяем результат переводом из КВ в десятичную систему счисления.

Для **a**:

0 100 0110 0000 0000 0000 0000 0000

мантисса + «скрытый бит» =  $1_2$

порядок =  $10001100_2 - 111\ 1111_2 = 1101_2 = 13_{10}$

$a = 1_2 * 10^{13} = 10\ 0000\ 0000\ 0000_2 = 2000_{16} = 8192_{10}$

Для **b**:

1 100 0110 1001 1010 0000 0000 0000

мантисса + «скрытый бит» =  $-1,0011010_2$

порядок =  $10001101_2 - 111\ 1111_2 = 1110_2 = E_{16} = 14_{10}$

$b = -1,0011010_2 * 10^{14} = 1001101000000000_2 = -4D00_{16} = -19712_{10}$

### Дробные числа

1. Представил числа **c**=0,32 и **d**=-0,77 в формате КВ.

$$c = 0,32_{10} = 0.0101\ 0001\ 1110\ 1011\ 1000\ 0101_2 =$$

$$= 1,010\ 0011\ 1101\ 0111\ 0000\ 1010 * 10^{-10}$$

Порядок:  $111\ 1111 - 10 = 1111101$

$$0\ 011\ 1110\ 1010\ 0011\ 1101\ 0111\ 0000\ 1010 = 3EA3\ D70A_{16}$$

$$d = -0,77_{10} = -0.1100\ 0101\ 0001\ 1110\ 1011\ 1000_2 =$$

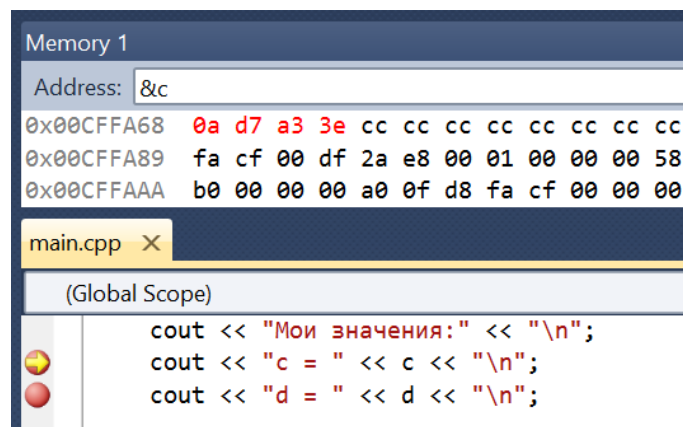
$$= -1,1000\ 1010\ 0011\ 1101\ 0111\ 0000 * 10^{-1}$$

Порядок:  $111\ 1111 - 1 = 111\ 1110$

$$1\ 011\ 1111\ 010001010001111010111000 = BF45\ 1EB8_{16}$$

2. Разместил полученные представления чисел в памяти.

c:



Memory 1

Address: &c

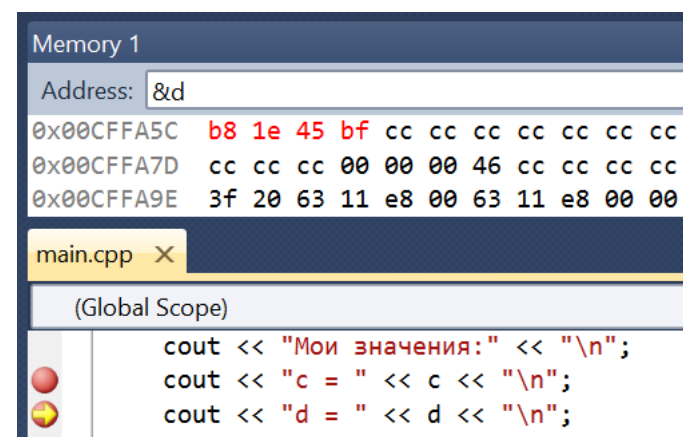
0x00CFFA68	0a d7 a3 3e	cc cc cc cc cc cc cc cc
0x00CFFA89	fa cf 00 df 2a e8 00 01 00 00 00 58	
0x00CFFAAA	b0 00 00 00 a0 0f d8 fa cf 00 00 00	

main.cpp X

(Global Scope)

```
cout << "Мои значения:" << "\n";
cout << "c = " << c << "\n";
cout << "d = " << d << "\n";
```

d:



Memory 1

Address: &d

0x00CFFA5C	b8 1e 45 bf	cc cc cc cc cc cc cc cc
0x00CFFA7D	cc cc cc 00 00 00 46 cc cc cc cc	
0x00CFFA9E	3f 20 63 11 e8 00 63 11 e8 00 00	

main.cpp X

(Global Scope)

```
cout << "Мои значения:" << "\n";
cout << "c = " << c << "\n";
cout << "d = " << d << "\n";
```

```
Мои значения:
c = 0.32
d = -0.77
```

3. Увеличил порядок каждого операнда на  $1000_2$ .

**c:**  $111\ 1101 + 1000 = 10000101$

**0**  $1000010\ 1010\ 0011\ 1101\ 0111\ 0000\ 1010 = 42A3D70A_{16}$

**d:**  $111\ 1110 + 1000 = 10000110$

**1**  $1000011\ 010001010001111010111000 = C3451EB8_{16}$

4. Разместил полученные представления чисел в памяти.

**c:**

Address: &c

0x00CFFA68	0a d7 a3 42 cc cc cc cc cc cc cc cc 00 00	.ЧJBMMMMMMMM..
0x00CFFA76	9a c6 cc cc cc cc cc cc cc cc 00 00 00 46	ьЖMMMMMMMM...F
0x00CFFA84	cc cc cc cc d8 fa cf 00 df 2a e8 00 01 00	MMMMШЪП.Я*и...

main.cpp X

(Global Scope) main()

```
cout << "Измененные значения: порядок увеличен на 8" << "\n"
cout << "c = " << c << "\n";
cout << "d = " << d << "\n";
cout << "\n";
```

**d:**

Memory 1

Address: &d

0x00CFFA5C	b8 1e 45 c3 cc cc cc cc cc cc cc cc 0a d7	ё.EГMMMMMMMM.Ч
0x00CFFA6A	a3 42 cc cc cc cc cc cc cc cc 00 00 9a c6	JBMMMMMMMM...ьЖ
0x00CFFA78	cc cc cc cc cc cc cc cc 00 00 00 46 cc cc	MMMMMMMM...FMM

main.cpp X

(Global Scope) main()

```
cout << "Измененные значения: порядок увеличен на 8" << "\n"
cout << "c = " << c << "\n";
cout << "d = " << d << "\n";
cout << "\n";
```



```

Мои значения:
c = 0.32
d = -0.77
Измененные значения: порядок увеличен на 8
c = 81.919998168945313
d = -197.1199951171875

```

5. Проверяем результат переводом из КВ в десятичную систему счисления.

**c:**

0 1000010 1010 0011 1101 0111 0000 1010

мантисса + «скрытый бит» = 1,010 0011 1101 0111 0000 1010<sub>2</sub>

порядок = 1000 0101<sub>2</sub> – 111 1111<sub>2</sub> = 110<sub>2</sub> = 6<sub>10</sub>

1,010 0011 1101 0111 0000 1010<sub>2</sub> \* 10<sup>6</sup> = 1010 001,1 1101 0111 0000 1010<sub>2</sub>  
 = 51.EB85<sub>16</sub> = 81.9199981689453125<sub>10</sub>

Получилось расхождение в младших разрядах из-за того, что результат выводится в формате ДВ, а не КВ (первые восемь десятичных цифр в этом случае должны совпасть, что и видно).

**d:**

1 100 0011 0100 0101 0001 1110 1011 1000

мантисса = -1,100 0101 0001 1110 1011 1000<sub>2</sub>

порядок = 1000 0110<sub>2</sub> – 111 1111<sub>2</sub> = 111<sub>2</sub> = 7<sub>10</sub>

-1,100 0101 0001 1110 1011 1000<sub>2</sub> \* 10<sup>7</sup> = -1100 0101, 0001 1110 1011 1000<sub>2</sub>  
 = -C5.1EB8<sub>16</sub> = -197.1199951171875<sub>10</sub>

6. Представил числа **f**=32,77 и **g**=-77,32 в формате КВ.

**f** = 32,77<sub>10</sub> = 100000.1100010100011110<sub>2</sub> =  
 1,00000110001010001111011 \* 10<sup>101</sup>

Порядок: 111 1111+101=1000 0100

0100 0010 000000110001010001111011 = 4203 147B<sub>16</sub>

**g** = -77,32<sub>10</sub> = -100 1101.0101 0001 1110 1011<sub>2</sub> = -1.001 1010 1010 0011

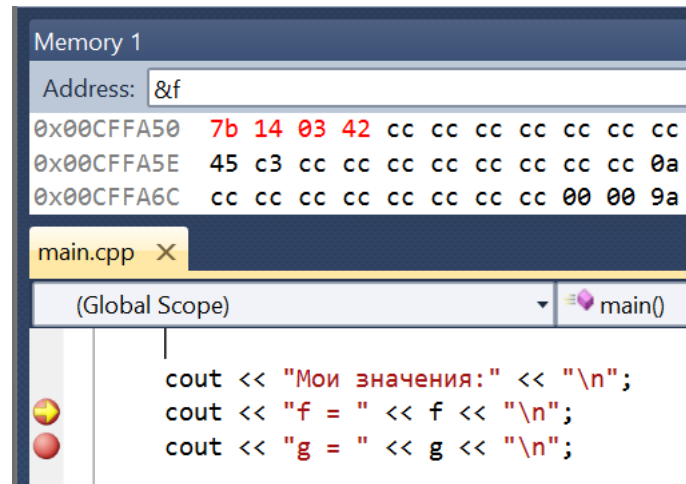
110 1 0111\*10<sup>110</sup>

Порядок: 111 1111+101=1000 0101

11000010100110101010001111010111 = C29A A3D7<sub>16</sub>

7. Разместил полученные представления чисел в памяти.

f:

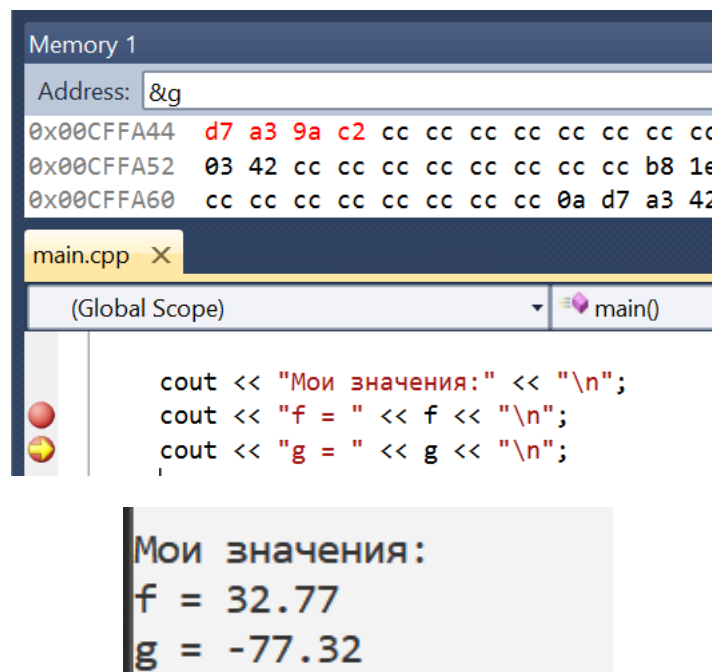


```
Memory 1
Address: &f
0x00CFFA50  7b 14 03 42 cc cc cc cc cc cc cc
0x00CFFA5E  45 c3 cc cc cc cc cc cc cc cc 0a
0x00CFFA6C  cc cc cc cc cc cc cc cc 00 00 9a

main.cpp X
(Global Scope) main()

cout << "Мои значения:" << "\n";
cout << "f = " << f << "\n";
cout << "g = " << g << "\n";
```

g:



```
Memory 1
Address: &g
0x00CFFA44  d7 a3 9a c2 cc cc cc cc cc cc cc
0x00CFFA52  03 42 cc cc cc cc cc cc cc cc b8 1e
0x00CFFA60  cc cc cc cc cc cc cc cc 0a d7 a3 42

main.cpp X
(Global Scope) main()

cout << "Мои значения:" << "\n";
cout << "f = " << f << "\n";
cout << "g = " << g << "\n";
```

```
Мои значения:
f = 32.77
g = -77.32
```

8. Увеличил порядок каждого операнда на 1000<sub>2</sub>.

Порядок: 1000 0100+1000=10001100

01000110000000110001010001111011<sub>2</sub>=4603 147B<sub>16</sub>

Порядок: 1000 0101+1000=1000 1101

1100 0110 1001 1010 1010 0011 1101 0111<sub>2</sub>=C69AA3D7<sub>16</sub>

9. Разместил полученные представления чисел в памяти.

**f:**

The screenshot shows a debugger window with two panes. The top pane, titled 'Memory 1', displays memory addresses and their contents. The address 0x003AFCC8 is selected, showing the value 7b 14 03 46 in red. The bottom pane shows the source code for 'main.cpp' at the '(Global Scope)' level, specifically the 'main()' function. The code includes several cout statements and a getch() call. The first cout statement is highlighted with a red arrow, indicating it is the current execution point.

```
Memory 1
Address: 0x003AFCC8
0x003AFCC8  7b 14 03 46 cc cc cc cc cc cc cc cc b8 1e {..FMMMMMMMMë.
0x003AFCD6  45 bf cc cc cc cc cc cc cc cc cc 0a d7 a3 3e EïMMMMMMMM.ЧJ>
0x003AFCE4  cc cc cc cc cc cc cc cc cc 00 00 80 3f cc cc MMMMMMMM..Б?MM

main.cpp X
(Global Scope) main()
cout << "Измененные значения: порядок увеличен на 8" << "\n"
cout << "f = " << f << "\n";
cout << "g = " << g << "\n";
cout << "\n";

getch();
}
```

**g:**

The screenshot shows a debugger window with two panes. The top pane, titled 'Memory 1', displays memory addresses and their contents. The address &g is selected, showing the value d7 a3 9a c6 in red. The bottom pane shows the source code for 'main.cpp' at the '(Global Scope)' level, specifically the 'main()' function. The code includes several cout statements and a getch() call. The first cout statement is highlighted with a red arrow, indicating it is the current execution point.

```
Memory 1
Address: &g
0x0098FDE8  d7 a3 9a c6 cc cc cc cc cc cc cc cc 7b 14 ЧJъЖMMMMMMMM{.
0x0098FDF6  03 46 cc cc cc cc cc cc cc cc cc b8 1e 45 bf .FMMMMMMMMë.Eï
0x0098FE04  cc cc cc cc cc cc cc cc cc 0a d7 a3 3e cc cc MMMMMMMM.ЧJ>MM

main.cpp X
(Global Scope) main()
cout << "Измененные значения: порядок увеличен на 8" << "\n"
cout << "f = " << f << "\n";
cout << "g = " << g << "\n";
cout << "\n";

getch();
}
```

```

Мои значения:
f = 32.77
g = -77.32
Измененные значения: порядок увеличен на 8
f = 8389.1201171875
g = -19793.919921875

```

10. Перевел результаты в десятичную систему счисления.

**f:**

01000110 000000110001010001111011<sub>2</sub>

мантисса + «скрытый бит»: 1,00000110001010001111011<sub>2</sub>

Порядок: 1000 1100<sub>2</sub> – 111 1111<sub>2</sub> = 1101<sub>2</sub> = 13<sub>10</sub>

1,00000110001010001111011<sub>2</sub> \* 10<sup>13</sup> = 10000011000101,0001111011<sub>2</sub> =  
20C5.1EC<sub>16</sub> = 8389.1201171875<sub>10</sub>

**g:**

1100 0110 1001 1010 1010 0011 1101 0111<sub>2</sub>

Мантисса: 1,001 1010 1010 0011 1101 0111<sub>2</sub>

Порядок: 1000 1101<sub>2</sub> – 111 1111<sub>2</sub> = 1100<sub>2</sub> = 14<sub>10</sub>

1,001 1010 1010 0011 1101 0111<sub>2</sub> \* 10<sup>14</sup> = 1001 1010 1010 001,1 1101 0111<sub>2</sub>  
= -4D51.EB8<sub>16</sub> = -19793.919921875<sub>10</sub>

**Вывод:** научился представлять целые, дробные и смешанные числа в формате короткое вещественное и переходить от представления числа с ПТ в памяти компьютера к числу в десятичной системе счисления.