Балтийский государственный технический университет «ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра О7 «Информационные системы и программная инженерия»

Практическая работа №3

по дисциплине «Информатика: Основы программирования» на тему «Указатели»

Выполнил: Студент Смирнов И.Д. Группа О719Б

Преподаватель: Назарова М.А.

Санкт-Петербург 2021 г.

Задание 1.

Проанализировать текст представленной программы и выдаваемые программой результаты. Объяснить, почему результаты именно такие.

Результаты работы программы:

```
/home/ivan/Practice/practice3/task1
       int: start address 0x7ffcf3387c54 extent 4
     float: start address 0x7ffcf3387c50 extent 4
    double: start address 0x7ffcf3387c48 extent 8
p1: pointer: start address 0x7ffcf3387c40 extent 8
p2: pointer: start address 0x7ffcf3387c38 extent 8
p3: pointer: start address 0x7ffcf3387c30 extent 8
p1: 0x7ffcf3387c54 related value 1
p2: 0x7ffcf3387c50 related value 2.000000
p3: 0x7ffcf3387c48 related value 3.000000
a=1
       b=2.000000
                       c=3.000000
a=5
       b=10.000000 c=1.732051
*p1=5 *p2=10.000000 *p3=1.732051
p1=0x7ffcf3387c50
                       p2=0x7ffcf3387c50
                                                p3=0x7ffcf3387c50
                                                                        p4 = 0 \times 7ff
cf3387c50
*p1=1092616192 *p2=10.000000 *p3=0.000000
                                                *(float*)p4=10.000000
p1=0x7ffcf3387c54
                        p2=0x7ffcf3387c50
                                                p3=0x7ffcf3387c48
*p1=5
               *p2=10.000000 *p3=1.732051
p1=0x7ffcf3387c44
                       p2=0x7ffcf3387c50
                                               p3=0x7ffcf3387c4c
*p1=32764
               *p2=10.000000 *p3=524288.124967
Process returned 0 (0x0) execution time : 0.006 s
Press ENTER to continue.
```

Текст программы:

```
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
int main()
```

```
/* «Обычные» переменные */
int a = 1;
float b = 2;
double c = 3;
/* Указатели */
int *p1 = &a;
float *p2 = &b;
double *p3 = &c;
void *p4;
/* Адреса «обычных» переменных и размер выделяемой памяти */
printf("a:
                 int: start address %p extent %d\n", &a, sizeof(a));
               float: start address %p extent %d\n", &b, sizeof(b));
printf("b:
             double: start address %p extent %d\n\n",&c,sizeof(c));
printf("c:
/* Адреса указателей и размер выделяемой памяти */
printf("p1: pointer: start address %p extent %d\n", &p1, sizeof(p1));
printf("p2: pointer: start address %p extent %d\n", &p2, sizeof(p2));
printf("p3: pointer: start address %p extent %d\n\n",&p3,sizeof(p3));
/* Значения, на которые ссылаются указатели */
printf("p1: %p related value %d\n",p1,*p1);
printf("p2: %p related value %f\n",p2,*p2);
printf("p3: %p related value %lf\n\n",p3,*p3);
```

	переменные		адреса
		00	7c28
		00	7c29
		00	7c2a
	p4	00	7c2b
	1	00	7c2c
		00	7c2d
		00	7c2e
		00	7c2f
		48	7c30
		7c	7c31
		38	7c32
	p3	f3	7c33
	3	fc	7c34
		7f	7c35
		00	7c36
		00	7c37
		50	7c38
		7c	7c39
		38	7c3a
	p2	f3	7c3b
		fc	7c3c
		7f	7c3d
		00	7c3e
		00	7c3f
		54	7c40
		7c	7c41
		38	7c42
	p1	f3	7c43
		fc	7c44
		7f	7c45
		00	7c46
		00	7c47
		00	7c48
		00	7c49
		00	7с4а
1 p	С	00	7c4b
3		00	7c4c
		00	7c4d
		08	7c4e
		40	7c4f
		00	7c50
1 p	b	00	7c51
2		00	7c52
		40	7c53
		01	7c54
1 p	a	00	7c55
01		00	7c56
		00	7c57

Вывод: размещение элементов памяти зависит от операционной системы. В данном случае в ОС <u>Fedora Linux 34</u> используется обратный порядок записи.

```
/* Использование указателей в выражениях */
printf("a=%d\tb=%f\tc=%lf\n",a,b,c);
*p1 = 5;
*p2 = *p2 * *p1;
*p3 = sqrt(*p3);
printf("a=%d\tb=%f\tc=%lf\n",a,b,c);
printf("a=%d\tb=%f\tc=%lf\n",a,b,c);
printf("*p1=%d\t*p2=%f\t*p3=%lf\n\n",*p1,*p2,*p3);
/* Присваивание указателей */
p1 = (int*)p2;
p3 = (double*)p2;
p4 = p2;
printf("p1=%p\tp2=%p\tp3=%p\tp4=%p\n",p1,p2,p3,p4);
printf("*p1=%d\t*p2=%f\t*p3=%f\t*(float*)p4=%f\n\n",*p1,*p2,*p3,*(float*)p4);
```

адреса	7c28	7c29	7c2a	7c2b	7c2c	7c2d	7c2e	7c2f	7c30	7c31	7c32	7c33	7c34	7c35	7c36	7c37	7c38	7.539	7c3a 7c3h	1030	7534	/c3d	7c3e	7c3f	7c40	7c41	7c42	7c43	7c44	7c45	7c46	7c47	7c48	7c49	7c4a	7c4b	7c4c	7c4d	7c4e	7c4f	7c50	7c51	7c52	7c53	7c54	7c55	7c56	7c57
	50	7c	38	f3	fc	7f	00	00	50	7c	38	f3	fc	7f	00	00	50 7	c 3	38 f3	3 f	c 7	7f (00 0	00 !	50	7c	38 1	f3	fc	7f	00	00	83	c0	ca	a1	45	b6	fb	3f	00	00	20	41	05	00	00	00
переменные				р	4							р	3							p2								p1	1							c	:					k)			á	a	
																																										1	p2					
																																										1	p1					
																																												1	рЗ			

Вывод: указателям можно присвоить адреса на переменные других типов, однако стоит понимать, что значение может представляться по-другому.

Именно поэтому при получении значения указателя (*p1) мы получили не 10, а $1092616192_{10} = 41200000_{16}$, так как представление чисел типа double отличается от типа int. Разыменование позволяет выполнять операции над значениями, находящимися по адресам указателей.

```
/* Изменение значений указателей */
p1++;
p3--;
printf("p1=%p\tp2=%p\tp3=%p\n",p1,p2,p3);
printf("*p1=%d\t\t*p2=%f\t*p3=%lf\n",*p1,*p2,*p3);
```

50 7c 38 f3 fc 7f 00 00 48 7c 38 f3 fc 7f 00 00 48 7c 38 f3 fc 7f 00 00 50 7c 38 f3 fc 7f 00 00 83 c0 ca at 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 00 00 00 00 00 00 00 00		переменные		адреса
The content of the			50	7c28
p4 p3 p2 p1 c p1 c p1			7c	7c29
p4 p3 p3 p2 p1 c p1			38	7c2a
fc 7f 00 00 48 7c 38 f3 fc 7f 00 00 50 50 7c 38 f3 fc 7f 00 00 50 50 7c 38 f3 fc 7f 00 00 50 50 7c 38 f3 fc 7f 00 00 50 50 7c 38 f3 fc 7f 00 00 60 60 60 60 60 60 60 60 60 60 60		p4	f3	7c2b
7f 00 00 48 7c 38 f3 fc 7f 00 00 50 7c 38 f3 fc 7f 00 00 50 7c 38 f3 fc 7f 00 p2 F2		4	fc	7c2c
A			7f	7c2d
00 48 7c 38 f3 fc 7f 00 00 50 7c 38 f3 fc 7f 00 00 50 7c 38 f3 fc 7f 00 p2 p1 c b a			00	7c2e
48 7c 38 f3 fc 7f 00 00 50 7c 38 f3 fc 7f 00 00 50 p2 p2 p1 c b a			00	7c2f
p3			48	7c30
p3			7c	7c31
f3 fc 7f 00 00 50 7c 38 f3 fc 7f 00 00 50 7c 38 f3 fc 7f 00 00 83 co 7f 00 00 50 7c 38 f3 fc 7f 00 00 83 co 7f 00 00 50 7c 38 f3 fc 7f 00 00 83 co ca a1 45 b6 fb 3f co 41 05 00 00 <t< th=""><th></th><th></th><th>38</th><th>7c32</th></t<>			38	7c32
Second S		р	f3	7c33
7f 00 00 50 7c 38 f3 fc 7f 00 00 54 7c 38 f3 fc 7f 00 00 54 7c 38 f3 fc 7f 00 00 54 7c 38 f3 fc 7f 00 00 83 c0 ca at 45 b6 fb 3f 00 00 20 4t 05 00 00 00 00 00 00 00 00 00 00 00 00		o3	fc	7c34
00 00 50 7c 38 f3 fc 7f 00 00 54 7c 38 f3 fc 7f 00 00 83 c0 ca at 45 b6 fb 3f 00 00 20 4t 05 00 00 00 prints at the second secon			7f	7c35
Column C			00	7c36
Second S			00	7c37
The content of the			50	7c38
92 p1 c b a			7c	7c39
f3 fc 7f 00 00 54 7c 38 f3 fc 7f 00 00 83 c0 ca al 45 b6 fb 3f 00 00 20 41 05 00 00 00 p2 al 54 p2 b a			38	7c3a
fc 7f 00 00 54 7c 38 f3 fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 20 41 25 da a		p	f3	7c3b
7f 00 00 54 7c 38 f3 fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 00 00 00 00 00 00 00 00		2	fc	7c3c
00 00 54 7c 38 f3 fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 00 00 00 00 00 00 00 00			7f	7c3d
00 54 7c 38 f3 fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 00 00 00 00 00 00 00 00			00	7c3e
54 7c 38 f3 fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 p1 c			00	7c3f
7c 38 f3 fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 00 00 00 00 00 00 00 00			54	7c40
38 f3 fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 p1 a			7c	7c41
f3 fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 p1			38	7c42
fc 7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 1		p	f3	7c43
7f 00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 co		1	fc	7c44
00 00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 ca a			7f	7c45
00 83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 00 00 00 00 00 00 00 00			00	7c46
83 c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 ca a			00	7c47
c0 ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 c a			83	7c48
ca a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 00 c			c0	7c49
a1 45 b6 fb 3f 00 00 20 41 05 00 00 00 c			ca	7с4а
45 b6 fb 3f 00 00 20 41 05 00 00 00 b	1 p	C	a1	7c4b
b6 fb 3f 00 00 20 41 05 00 00 00 b	03		45	7c4c
fb 3f 00 00 20 41 05 00 00 00 b a			b6	7c4d
3f 00 00 20 41 05 00 00 00 b a			fb	7c4e
00 00 20 41 05 00 00 00 b a			3f	7c4f
00 20 41 05 00 00 00 b a			00	7c50
20 41 05 00 00 00 a	1	k	00	7c51
41 05 00 00 00 a	p2)	20	7c52
05 00 00 00 a			41	7c53
00 00 00 a		·	05	7c54
00 00	1 p	a	00	7c55
	p1	à	00	7c56
			00	7c57

Вывод: значения вернулись к прежним, так как мы сместили указатели на изначальные адреса (указатель p1 сместился на 4 байт, так как это указатель на int, p3 — на 8 байт, так как это указатель на double).

```
p1 -= 4;
p3 = (double*)&a - 1;
printf("p1=%p\tp2=%p\tp3=%p\n",p1,p2,p3);
printf("*p1=%d\t*p2=%f\t*p3=%lf\n",*p1,*p2,*p3);
return 0;
```

		переменные		адреса
			50	7c28
			7c	7c29
			38	7c2a
		p²	f3	7c2b
		1	fc	7c2c
			7f	7c2d
			00	7c2e
			00	7c2f
			4c	7c30
			7c	7c31
			38	7c32
		р	f3	7c33
		3	fc	7c34
			7f	7c35
			00	7c36
			00	7c37
			50	7c38
			7c	7c39
			38	7c3a
		р	f3	7c3b
		2	fc	7c3c
			7f	7c3d
			00	7c3e
			00	7c3f
			44	7c40
			7c	7c41
			38	7c42
		р	f3	7c43
		1	fc	7c44
	1		7f	7c45
	р1		00	7c46
			00	7c47
			83	7c48
			c0	7c49
			са	7c4a
		С	a1	7c4b
			45	7c4c
			b6	7c4d
			fb	7c4e
	1 p		3f	7c4f
	3		00	7c50
1 p		b	00	7c51
2		1	20	7c52
			41	7c53
			05	7c54
		a	00	7c55
			00	7c56
			00	7c57

Вывод: при смещении указателя p1 влево на 16 байт (4*sizeof(int)) значение под ним является частью значения p1. Это значение при выводе будет выглядеть так: 32764₁₀. p3 является указателем на тип double, поэтому мы после присваивания к адресу а сместились на 8 байт влево (1*sizeof(double)).

Bыводы: Порядок записи переменных в памяти зависит от ОС. От типа указателя зависит смещение, указателю можно присвоить указатель другого типа или вывести значение указателя на void с помощью явного приведения.

Задание 2.

Операционная система Windows 10, среда разработки Code::Blocks.

Проанализировать текст представленной программы, найти в нем синтаксические ошибки и исправить их, в начало программы добавить вывод на экран адресов всех переменных, а в конец – значений всех переменных, проанализировать полученные результаты и объяснить, почему они именно такие. Заменить инструкцию «m+=2;» инструкцией «m++;», проанализировать результат.

Текст измененной программы:

```
#include <stdio.h>
#include <stdlib.h>
int main()
```

```
{
   char *p, c;
   int *a, b;
   float *x, y = 3.5;
   double *m, n;
   a = \&b;
   printf(" Enter b = ");
   scanf("%d", a);
   printf("p=%p\ta=%p\ta=%p\tx=%p\ty=%p\tn=%p\n\n", &p, &c, &a, &b, &x, &y, &m, &n);
   printf("a=%p\t*a=%d\tb=%d\n", a, *a, b);
   p = (char*)a;
   c = *p;
   *p = *(p+3);
   *(p+3) = c;
   printf("p=%p\tc=%d\ta=%p\tb=%d\n", p, c, a, b);
   x = &y;
   printf("x=%p\t*x=%f\ty=%f\n", x, *x, y);
   a = (int*)x;
   *a = *x;
   printf("a=p\t*a=d\tx=p\t*x=f\ty=f\n", a, *a, x, *x, y);
   a = \&b;
   v = 12345.6789;
   printf("x=%p\t*x=%f\ty=%f\n", x, *x, y);
   p = (char*)x;
   c=*p;
   *p=*(p+3);
   *(p+3)=c;
   printf("p=%p\tc=%d\tx=%p\ty=%f\n", p, c, x, y);
   m = &n;
   printf("m=%p\t*m=%lf\tn=%lf\n", m, *m, n);
   n = 5.5;
   printf("m=%p\t*m=%lf\tn=%lf\n", m, *m, n);
   b = n = y = 1.7;
   printf("b=%d\ty=%f\tn=%lf\n", b, y, n);
   printf("*a=%d\t*x=%f\t*m=%lf\n", *a, *x, *m);
   m++;
   printf("n=%lf\tn=%p\tm=%p\n", n, &n, m);
   *m = (float)*a - n + (int)*x;
   printf("m=%p\t*m=%lf\n\n", m, *m);
   printf("p=%p\tc=%d\ta=%p\tb=%d\tx=%p\ty=%f\tm=%p\tn=%f\n\n",p,c,a,b,x,y,m,n);
   return 0;
```

Результаты работы программ:

первый вариант (т+=2;)

```
Enter b = 1
p=000000000061FE18
                        c=000000000061FE17
                                                 a=000000000061FE08
                                                                          b=000000000061FE04
                                                                                                  x=000000000061FDF8
        y=000000000061FDF4
                                m=000000000061FDE8
                                                         n=000000000061FDE0
a=000000000061FE04
                         *a=1
                                b=1
p=000000000061FE04
                        c=1
                                 a=000000000061FE04
                                                         b=16777216
x=000000000061FDF4
                         *x=3.500000
                                         y=3.500000
                                 x=000000000061FDF4
                                                         *x=0.000000
                                                                         y=0.000000
a=000000000061FDF4
                         *a=3
x=000000000061FDF4
                         *x=12345.678711 y=12345.678711
                                                         y=-0.000011
p=000000000061FDF4
                        c=-73
                                x=000000000061FDF4
m=000000000061FDE0
                         *m=-1.#QNAN0
                                         n=-1.#QNAN0
m=000000000061FDE0
                         *m=5.500000
                                         n=5.500000
b=1
        y=1.700000
                        n=1.700000
*a=1
        *x=1.700000
                         *m=1.700000
                n=0000000000061FDE0
n=1.700000
                                         m=000000000061FDF0
m=000000000061FDF0
                         *m=0.300000
p=000000000061FDF4
                                a=000000000061FE04
                                                         b=1
                                                                  x=000000000061FDF4
                                                                                          y=1.650000
                                                                                                                                   n=1.700000
                        c = -73
                                                                                                           m=000000000061FDF0
                           execution time : 0.370 s
Process returned 0 (0x0)
Press any key to continue.
```

второй вариант (т++;)

```
Enter b = 1
p=0000000000061FE18
                        c=000000000061FE17
                                                a=000000000061FE08
                                                                         b=000000000061FE04
                                                                                                 x=000000000061FDF8
       y=000000000061FDF4
                                m=0000000000061FDE8
                                                        n=000000000061FDE0
a=000000000061FE04
                        *a=1
                                b=1
p=000000000061FE04
                        C=1
                                a=000000000061FE04
                                                        b=16777216
x=000000000061FDF4
                        *x=3.500000
                                        y=3.500000
a=000000000061FDF4
                                x=000000000061FDF4
                                                         *x=0.000000
                                                                        y=0.000000
                        *a=3
                        *x=12345.678711 y=12345.678711
x=000000000061FDF4
                               x=0000000000061FDF4
p=000000000061FDF4
                        c=-73
                                                        y = -0.000011
m=000000000061FDE0
                        *m=-1.#QNAN0
                                        n=-1.#QNAN0
                        *m=5.500000
                                        n=5.500000
m=000000000061FDE0
                        n=1.700000
b=1
       y=1.700000
*a=1
        *x=1.700000
                        *m=1.700000
n=1.700000
                n=000000000061FDE0
                                        m=000000000061FDE8
Process returned -1073741819 (0xC0000005)
                                            execution time : 3.099 s
Press any key to continue.
```

адреса	61fde0	61fde1	61fde2	61fde3	61fde4	61fde5	61fde6	61fde7	61fde8	61fde9	61fdea	61fdeb	61fdec	61fded	61fdee	61fdef	61fdf0	61fdf1	61fdf2	61fdf3	61fdf4	61fdf5	61fdf6	61fdf7	61fdf8	61fdf9	61fdfa	61fdfb	61fdfc	61fdfd	61fdfe	61fdff	61fe00	61fe01	61fe02	61fe03	÷
	м	М	М	М	M	М	М	М	м	М	м	М	М	М	М	М					00	00	60	40	М	М	М	М	M	М	м	м					
переменные				r	n							n	m									У	′					>	ζ.								

:	61fe04	61fe05	61fe06	61fe07	61fe08	61fe09	61fe0a	61fe0b	61fe0c	61fe0d	61fe0e	61fe0f	61fe10	61fe11	61fe12	61fe13	61fe14	61fe15	61fe16	61fe17	61fe18	61fe19	61fe1a	61fe1b	61fe1c	61fe1d	61fe1e	61fe1f
	01	00	00	00	04	fe	61	00	00	00	00	00								М	M	м	м	М	М	M	М	м
		ŀ)					6	3											С				р				
		1	а																									

```
p = (char*)a;
c = *p;
*p = *(p+3);
*(p+3) = c;
printf("p=%p\tc=%d\ta=%p\tb=%d\n", p, c, a, b);
```

Здесь указателю р присваивается значение указателя а, но так как указатель char на 1 байт, то р будет указывать только на этот байт. При этом мы меняем местами значения по адресам 61fe04 и 61fe07 с помощью переменной с, это удалось сделать, так смещение у р равно размеру char (1 байт), в переменной b получилось число $1000000_{16} = 16777216_{10}$.

адреса	61fde0	61fde1	61fde2	61fde3	61fde4	61fde5	61fde6	61fde7	61fde8	61fde9	61fdea	61fdeb	61fdec	61fded	61fdee	61fdef	61fdf0	61fdf1	61fdf2	61fdf3	61fdf4	61fdf5	61fdf6	61fdf7	61fdf8	61fdf9	61fdfa	61fdfb	61fdfc	61fdfd	61fdfe	61fdff	61fe00	61fe01	61fe02	61fe03	:
	м	М	м	м	м	м	М	М	м	м	М	м	м	М	м	м					00	00	60	40	М	M	м	м	М	м	м	м					
переменные				ı	n							r	n										у					;	x								

:	61fe04	61fe05	61fe06	61fe07	61fe08	61fe09	61fe0a	61fe0b	61fe0c	61fe0d	61fe0e	61fe0f	61fe10	61fe11	61fe12	61fe13	61fe14	61fe15	61fe16	61fe17	61fe18	61fe19	61fe1a	61fe1b	61fe1c	61fe1d	61fe1e	61fe1f
	00	00	00	01	04	fe	61	00	00	00	00	00								01	04	fe	61	00	00	00	00	00
		k)					í	a											С				ŗ)			
		1	a																									
	1 p																											

```
x = &y;
printf("x=%p\t*x=%f\ty=%f\n", x, *x, y);
a = (int*)x;
*a = *x;
printf("a=%p\t*a=%d\tx=%p\t*x=%f\ty=%f\n", a, *a, x, *x, y);
```

Здесь указателю х даётся адрес переменной у, а к указателю а присваивается значение указателя х, преобразованное в int. В результате изменения значения переменной у (*a = *x) в ней оказалось число 3, что в стандарте ieee754 будет очень маленьким числом, поэтому при выводе *x (значения под указателем на float) получается 0. Напротив, при выводе *a (значения под указателем на int) получается число, обрезанное до целого (3.5 (int) = 3), так получилось при присваивании *a = *x.

адреса	61fde0	61fde1	61fde2	61fde3	61fde4	61fde5	61fde6	61fde7	61fde8	61fde9	61fdea	61fdeb	61fdec	61fded	61fdee	61fdef	61fdf0	61fdf1	61fdf2	61fdf3	61fdf4	61fdf5	61fdf6	61fdf7	61fdf8	61fdf9	61fdfa	61fdfb	61fdfc	61fdfd	61fdfe	61fdff	61fe00	61fe01	61fe02	61fe03	:
	м	м	М	м	М	м	м	М	м	м	м	м	м	м	м	м					03	00	00	00	f4	fd	61	00	00	00	00	00					
переменные				n	1							n	n									,	у)	κ								
																						1	x														
																						1	`a														

i	61fe04	61fe05	61fe06	61fe07	61fe08	61fe09	61fe0a	61fe0b	61fe0c	61fe0d	61fe0e	61fe0f	61fe10	61fe11	61fe12	61fe13	61fe14	61fe15	61fe16	61fe17	61fe18	61fe19	61fe1a	61fe1b	61fe1c	61fe1d	61fe1e	61fe1f
	00	00	00	01	f4	fd	61	00	00	00	00	00								01	04	fe	61	00	00	00	00	00
		k)					ć	a											С				ŗ)			
	Îр																											

```
a = &b;
y = 12345.6789;
printf("x=%p\t*x=%f\ty=%f\n", x, *x, y);
p = (char*)x;
c=*p;
*p=*(p+3);
*(p+3)=c;
printf("p=%p\tc=%d\tx=%p\ty=%f\n", p, c, x, y);
```

Здесь присваивается указателю р значение указателя x с преобразованием в char, далее идёт обмен значениями по адресам р и p+3 (61fdf4 и 61fdf7 соответственно), в результате при выводе переменной у получается очень маленькое число. т.к. смещенный порядок числа стал меньше (110 < 140).

адреса	61fde0	61fde1	61fde2	61fde3	61fde4	61fde5	61fde6	61fde7	61fde8	61fde9	61fdea	61fdeb	61fdec	61fded	61fdee	61fdef	61fdf0	61fdf1	61fdf2	61fdf3	61fdf4	61fdf5	61fdf6	61fdf7	61fdf8	61fdf9	61fdfa	61fdfb	61fdfc	61fdfd	61fdfe	61fdff	61fe00	61fe01	61fe02	61fe03	
	M	м	м	M	М	М	м	М	м	М	м	м	М	м	М	м					46	e6	40	b7	f4	fd	61	00	00	00	00	00					I
переменные				1	1							n	า									Y	У					;	x								
																						1	`x														T
																																					T
	: ,	61fe04	61fe05	61fe06		61te07	61fe08	61fe09	61fe0a	001100	61fe0b	61fe0c	61fe0d		61te0e	61fe0f	61fe10		61fe11	61fe12	61fe13		61fe14	61fe15		61fe16	61fe17		61te18	61fe19	61fe1a	61fe1h	2	61fe1c	61fe1d	61fe1e	
	(00	00	00	()1	04	fe	6:	1	00	00	00) (00	00											b7	f	4	fd	61	00) (00	00	00	
			k)							a																С						р				
			1	а																																	
	-	t p																																			

```
m = &n;
printf("m=%p\t*m=%lf\tn=%lf\n", m, *m, n);
n = 5.5;
```

```
printf("m=%p\t*m=%lf\tn=%lf\n", m, *m, n);
b = n = y = 1.7;
printf("b=%d\ty=%f\tn=%lf\n", b, y, n);
printf("*a=%d\t*x=%f\t*m=%lf\n", *a, *x, *m);
m+=2;
printf("n=%lf\tn=%p\tm=%p\n", n, &n, m);
*m = (float)*a - n + (int)*x;
printf("m=%p\t*m=%lf\n\n", m, *m);
printf("p=%p\tc=%d\ta=%p\tb=%d\tx=%p\ty=%f\tm=%p\tn=%f\n\n",p,c,a,b,x,y,m,n);
```

при m+=2 указатель был смещён на 16 байт (2*sizeof (double)), первые 4 ячейки не являются какой-либо переменной, а вторые 4 ячейки — это переменная у. Потом вычисляется значение:

При выводе у отображается 1.650000, так как на порядок у float отводится меньше, чем у double, в итоге сам порядок в представлении float стал больше, и число изменилось (0.300000 = > 1.650000).

адреса	61fde0	61fde1	61fde2	61fde3	61fde4	61fde5	61fde6	61fde7	61fde8	61fde9	61fdea	61fdeb	61fdec	61fded	61fdee	61fdef	61fdf0	61fdf1	61fdf2	61fdf3	61fdf4	61fdf5	61fdf6	61fdf7	61fdf8	61fdf9	61fdfa	61fdfb	61fdfc	61fdfd	61fdfe	61fdff	61fe00	61fe01	61fe02	61fe03	:
	33	33	33	33	33	33	fb	3f	f0	fd	61	00	00	00	00	00	33	33	33	33	33	33	d3	3f	f4	fd	61	00	00	00	00	00					
переменные				ı	n							n	n									Ņ	у						x								
																				1	m																
																						1	`x														

÷	61fe04	61fe05	61fe06	61fe07	61fe08	61fe09	61fe0a	61fe0b	61fe0c	61fe0d	61fe0e	61fe0f	61fe10	61fe11	61fe12	61fe13	61fe14	61fe15	61fe16	61fe17	61fe18	61fe19	61fe1a	61fe1b	61fe1c	61fe1d	61fe1e	61fe1f
	01	00	00	00	04	fe	61	00	00	00	00	00								b7	f4	fd	61	00	00	00	00	00
		k						;	a											С				ţ)			
		1	a																									
	1 p																											

После замены инструкции «m+=2;» инструкцией «m++;» указатель стал ссылаться на себя. При попытке изменить значение под указателем выводится ошибка, так как это значение (адрес) будет находиться в другой, недоступной для компилятора, области памяти. (segmentation fault – ошибка доступа к памяти).

адреса	61fde0	61fde1	61fde2	61fde3	61fde4	61fde5	61fde6	61fde7	61fde8	61fde9	61fdea	61fdeb	61fdec	61fded	61fdee	61fdef	61fdf0	61fdf1	61fdf2	61fdf3	61fdf4	61fdf5	61fdf6	61fdf7	61fdf8	61fdf9	61fdfa	61fdfb	61fdfc	61fdfd	61fdfe	61fdff	61fe00	61fe01	61fe02	61fe03	:
	33	33	33	33	33	33	fb	3f	e8	fd	61	00	00	00	00	00					9a	99	d9	3f	f4	fd	61	00	00	00	00	00					
переменные				r	1							n	า									,	у					:	x								
												1	m									1	x														

:	61fe04	61fe05	61fe06	61fe07	61fe08	61fe09	61fe0a	61fe0b	61fe0c	61fe0d	61fe0e	61feOf	61fe10	61fe11	61fe12	61fe13	61fe14	61fe15	61fe16	61fe17	61fe18	61fe19	61fe1a	61fe1b	61fe1c	61fe1d	61fe1e	61fe1f
	00	00	00	01	04	fe	61	00	00	00	00	00								b7	f4	fd	61	00	00	00	00	00
		k)					,	a											С				ţ)			
		1	a																									
	↑p																											

Задание 3.

Объявить по две переменные типов *char*, *int* и *double*, а также указатель на *char*. Вывести на экран размеры и адреса всех переменных, начертить схему расположения переменных в памяти. Поменять порядок объявления переменных (например, *int*, *char*, *double*, *char**, *char*, *double*, *int*). Запустить программу повторно, проанализировать, что изменилось. Задать переменной типа *int* такое значение, чтобы значение каждого байта было уникальным, использовать для этого шестнадцатеричную константу. Записать адрес этой переменной в указатель на *char* и с его помощью вывести на экран содержимое каждого байта (тоже в шестнадцатеричной системе счисления). Проанализировать, прямой или обратный порядок расположения байт при записи числа применяется в используемой системе.

Повторить выполнение этого задания на компьютере с другой операционной системой и/или другой IDE (можно использовать домашний компьютер или онлайн-компилятор). Сравнить результаты работы программы на разных платформах, сделать выводы.

1 Операционная система <u>KDE neon (Ubuntu 20.04.2 LTS)</u>, среда разработки <u>Code::Blocks</u>

Текст программы (первый порядок объявления переменных):

```
#include <stdio.h>
int main() {
    char c1,c2;
    int i1,i2;
    double d1,d2;
```

```
char* pC;
printf("c1, address = %p, size = %d\n",&c1,sizeof(c1));
printf("c2, address = %p, size = %d\n",&c2,sizeof(c2));
printf("i1, address = %p, size = %d\n",&i1,sizeof(i1));
printf("i2, address = %p, size = %d\n",&i2,sizeof(i2));
printf("d1, address = %p, size = %d\n",&d1,sizeof(d1));
printf("d2, address = %p, size = %d\n",&d2,sizeof(d2));
printf("pC, address = %p, size = %d\n",&pC,sizeof(pC));
return 0;
```

Результаты работы программы:

```
c1, address = 0x7ffd96dd8876, size = 1
c2, address = 0x7ffd96dd8877, size = 1
i1, address = 0x7ffd96dd8878, size = 4
i2, address = 0x7ffd96dd887c, size = 4
d1, address = 0x7ffd96dd8880, size = 8
d2, address = 0x7ffd96dd8888, size = 8
pC, address = 0x7ffd96dd8890, size = 8
Process returned 0 (0x0) execution time : 0.003 s
Press ENTER to continue.
```

Размещение переменных в памяти:

адреса	dd8876	dd8877	dd8878	dd7879	dd787a	dd787b	dd787c	dd787d	dd787e	dd787f	dd7880	dd7881	dd7882	dd7883	dd7884	dd7885	dd7886	dd7887	dd7888	dd7889	dd788a	dd788b	dd788c	dd788d	dd788e	dd788f	dd7890	dd7891	dd7892	dd7893	dd7894	dd7895	9682pp	dd7897
переменные	c1	c2		i	1			i	2					d	1							d	12							p	С			

Текст программы (второй порядок объявления переменных):

```
#include <stdio.h>
int main(){
   char c1;
   double d1;
   char *pC;
   int i1;
   double d2;
    char c2;
    int i2 = 0x7f5f3a22;
   printf("c1, address = %p, size = %d\n",&c1,sizeof(c1));
   printf("c2, address = %p, size = %d\n",&c2,sizeof(c2));
   printf("i1, address = %p, size = %d\n",&i1,sizeof(i1));
   printf("i2, address = %p, size = %d\n",&i2,sizeof(i2));
   printf("d1, address = %p, size = %d\n",&d1,sizeof(d1));
   printf("d2, address = %p, size = %d\n",&d2,sizeof(d2));
   printf("pC, address = %p, size = %d\n\n",&pC,sizeof(pC));
   printf("i2 value = %x\n", i2);
   printf("i2 byte structure:\n\n");
   for(pC = (char*)&i2; pC < (char*)&i2+sizeof(int); pC++) {</pre>
       printf("%x\n",*pC);
    return 0;
```

Результаты работы программы:

```
x

cl, address = 0x7ffcel3317d6, size = 1
c2, address = 0x7ffcel3317d7, size = 1
il, address = 0x7ffcel3317d8, size = 4
i2, address = 0x7ffcel3317d8, size = 4
dl, address = 0x7ffcel3317e0, size = 8
d2, address = 0x7ffcel3317f0, size = 8
pC, address = 0x7ffcel3317e8, size = 8
i2 value = 7f5f3a22
i2 byte structure:

22
3a
5f
7f

Process returned 0 (0x0) execution time : 0.004 s
Press ENTER to continue.
```

адреса	3317d6	3317d7	3317d8	3317d9	3317da	3317db	3317dc	3317dd	3317de	3317df	3317e0	3317e1	3317e2	3317e3	3317e4	3317e5	3317e6	3317e7	3317e8	3317e9	3317ea	3317eb	3317ec	3317ed	3317ee	3317ef	3317f0	3317f1	3317f2	3317f3	3317f4	3317f5	3317f6	3317f7
							22	3a	5f	7f									df	17	33	e1	fc	7f	00	00								
переменные	c1	c2		i	1			i	i2					d	1							р	oC							d	2			

Выводы: память выделяется по возрастанию размера переменных в байтах (т.е. char (1 байт), int (4 байт), double и указатели (8 байт)). Порядок следования байт при записи числа - *обратный*.

2 Операционная система <u>KDE neon (Ubuntu 20.04.2 LTS)</u>, среда разработки <u>онлайн-компилятор Replit https://replit.com/languages/с</u> Результаты работы программы при первом порядке объявления переменных:

```
c1, address = 0x7ffd43ccc5fb, size = 1
c2, address = 0x7ffd43ccc5fa, size = 1
i1, address = 0x7ffd43ccc5f4, size = 4
i2, address = 0x7ffd43ccc5f0, size = 4
d1, address = 0x7ffd43ccc5e8, size = 8
d2, address = 0x7ffd43ccc5e0, size = 8
pC, address = 0x7ffd43ccc5d8, size = 8
```

адреса	ccc5d8	6p5ɔɔɔ	ccc5da	cc5db	ccc5dc	cc5dd	ccc5de	ccc5df	cc5e0	ccc5e1	ccc5e2	ccc5e3	ccc5e4	ccc5e5	ccc5e6	ccc5e7	ccc5e8	ccc5e9	ссс5еа	cccSeb	cccSec	ccc5ed	cccSee	ccc5ef	ccc5f0	ccc5f1	ccc5f2	ccc5f3	ccc5f4	ccc5f5	ccc5f6	ccc5f7	ccc5f8	6JS222	ccc5fa	ccc5fb
переменные				p	oC .							d	2							d	1					i2	2			i1					c2	c1

Результаты работы программы при втором порядке объявления переменных:

```
c1, address = 0x7ffdb015e23b, size = 1
c2, address = 0x7ffdb015e217, size = 1
i1, address = 0x7ffdb015e224, size = 4
i2, address = 0x7ffdb015e210, size = 4
d1, address = 0x7ffdb015e230, size = 8
d2, address = 0x7ffdb015e218, size = 8
pC, address = 0x7ffdb015e228, size = 8
i2 value = 7f5f3a22
i2 byte structure:

22
3a
5f
7f_
```

משמנים	15e210	ъ 15e211	15e212	Jf 15e213	1	15e215	15e216	15e217	15e218	15e219	15e21a	15e21b	15e21c	15e21d	15e21e	15e21f	15e220	15e221	15e222	15e223	15e224	15e225	15e226	15e227	15e228	5 15e229	15e22a	පි 15e22b	تع 15e22c	J5e22d	S 15e22e	S 15e22f	15e230	15e231	15e232	15e233	15e234	15e235	15e236	15e237	15e238	15e239	15e23a	
аннымосьы		i2		,,,				c2					d2									i	1		13	CZ	13		C	71	00	00				d	1							

Выводы: память под переменные выделяется в том порядке, в котором они были объявлены, выделение памяти идёт от большего адреса к меньшему.

Порядок следования байт при записи числа также обратный.

Выводы: Распределение памяти под переменные зависит от компилятора (gcc или clang) и ОС (Windows или Linux), но в обоих системах порядок следования байт при записи числа является *обратным*.