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

Кафедра И5 «Информационные системы и программная инженерия»

Практическая работа №3

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

Выполнил: Студент Альков В.С. Группа И407Б

Преподаватель: Першин Д.В.

Санкт-Петербург 2020 г.

Задание 1.

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

Результаты работы программы:

```
int: start address 000000000061FE04 extent 4
     float: start address 00000000001FE00 extent 4
    double: start address 00000000001FDF8 extent 8
p1: pointer: start address 000000000061FDF0 extent 8
p2: pointer: start address 000000000061FDE8 extent 8
p3: pointer: start address 000000000061FDE0 extent 8
p1: 000000000061FE04 related value 1
p2: 000000000061FE00 related value 2.000000
p3: 000000000061FDF8 related value 3.000000
a=1
       b=2.000000
                       c=3.000000
       b=10.000000
                       c=1.732051
*p1=5 *p2=10.000000 *p3=1.732051
p1=000000000061FE00
                       p2=000000000061FE00
                                               p3=000000000061FE00
                                                                       p4=000000000061FE00
*p1=1092616192 *p2=10.000000 *p3=0.000000
                                               *(float*)p4=10.000000
                       p2=000000000061FE00
p1=000000000061FE04
                                               p3=000000000061FDF8
               *p2=10.000000 *p3=1.732051
*p1=5
p1=000000000061FDF4
                                               p3=000000000061FDFC
                       p2=000000000061FE00
*p1=0 *p2=10.000000 *p3=524288.124967
Process returned 0 (0x0)
                          execution time : 0.020 s
Press any key 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 */
             int: start address %p extent %d\n", &a, sizeof(a));
printf("b: float: start address %p extent %d\n", &b, sizeof(b));
printf("c: double: start address %p extent %d\n\n", &c, sizeof(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);
```

Размещение переменных в памяти

																				61	FD																							61	FE			
D8	B D9	Ð	A D	ЭВ	DC	DD	DE	DF	E	0 E1	E2	2 E	3 E4	4 E	5 E6	E7	E8	E9	EΑ	EB	EC	ED	EE	EF	FO	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	00	01	02	03	04	05	06	07
									F	8 FC) 6:	1 0	0 0	0 0	0 00	00	00	FE	61	00	00	00	00	00	04	FE	61	00	00	00	00	00	00	00	00	00	00	00	08	40	00	00	00	40	01	00	00	00
p4	1								p	3							p2	2							р1	L							С								b				а			
																																	рЗ								р2				р1			

```
/* Использование указателей в выражениях */
    printf("a=%d\tb=%f\tc=%lf\n",a,b,c);
    /*5 присваевается переменной, на которую указывать указатель p1, то есть a*/
    *p1 = 5;
    /*происходить разыменование p2 и p1, в переменную,
    на которую указывает p2, то есть в переменную b, записывается
    произведние значения этой переменной на значение переменной, на которую указывает p1, то есть a */
    *p2 = *p2 * *p1;
```

```
/*в переменную, на которую указывает р3, то есть с, записывается корень этой переменной*/
*p3 = sqrt(*p3);
/*вывод переменных*/
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 */
/*происходит явное приведения адреса, на который указывает указатель p2, к типу int* и присвоение
этого адреса p1, тем самым указатель p1 теперь тоже указывает на переменную b, переменная b типа float,
в ней записано 10, закодированное способом для чисел с плавющей точкой и = 01000001001000000000000000000,
при выводе число раскодируется способом для целых и = 1092616192 */
p1 = (int*)p2;
/*происходит явное приведения адреса, на который указывает указатель p2, к типу double* и присвоение
этого адреса p3, тем самым указатель p3 теперь тоже указывает на переменную b, переменная b типа float,
4 байта, p3 указатель типа double*, ему нужно 8 байт, поэтому он отсчитает от адреса 8 байт захватив переменную а,
в итоге он указывает на значение 000000541200000, что равно 1.11498e-313, но со спецификатором %f выводиться 0,
его можно вывести, если увеличть точность до 329 или если использовать спецификатор %g*/
p3 = (double*)p2;
/*происходит присвоение адреса, на который указывает p2, p4, тем самым указатель p4 теперь
тоже указывает на переменную b, указатель p4 типа void* и явное приведение для этого не требуется,
с типом void* можно указывать на любой тип, но чтобы получить значение, на адрес которого такой указатель указывает
нужно использовать явное приведение к нужному типу*/
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);
```

																			61	FD																				61	LFE		
D8 [9	DA	DB	DC	DD	DE	DF	EO	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	ЕВ	EC	ED	EE	EF	FO F	1 F	2 F	3 F4	F F 5	F6	F7	F8	F9 F	A F	B FC	FD	FE	FF	00 0	1 02	03	04	05	06 0
00	E	61	00	00	00	00	00	00	FE	61	00	00	00	00	00	00	FE	61	00	00	00	00	00	00 F	E 6	1 0	0 00	00	00	00	AA	4C 5	8 E	8 7/	A B6	FB	3F	00 0	0 20	41	5	00	00 00
р4								рЗ	3							p2								р1							С							b			а		
																															рЗ							р4			р1		
																																						рЗ					
																																						p2					
																																						p1					

В случае с р3 выдался 0 из-за нехватки точности выведения.

```
/* Изменение значений указателей */
/*сдвиг адреса, на который указывает p1, вправо на байтность типа, на 4 байта*/
p1++;
/*сдвиг адреса, на который указывает p3, влево на байтность типа, на 8 байт*/
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);
```

																					61	FD																						61	FE.			
D	B D	9 [DΑ	DB	D	C D	D [DE D	F E	0 E	1	E2	E3	E4	E5	E6	E7	E8	E9	EΑ	ЕВ	EC	ED	EE	EF	FO	F1	F2	F3	F4	F5	F6 F	7 F	F8 F	9 F	A FE	B FC	FD) FE	FF	00	01	02	03	04	05	06 0	17
00) F	E 6	51	00	00	0 0	0 0	0 0	0 F	8 F	D (61	00	00	00	00	00	00	FE	61	00	00	00	00	00	04	FE	61	00	00	00 (00 0	Ю А	AA 4	IC 5	8 E8	3 7/	А ВЕ	FB	3F	00	00	20	41	05	00	00 0	00
p	4								ŗ	03								p2								р1							(С							b				а			_
																																		рЗ							p4	4			р1			
																																									p2	2						

```
/*сдвиг адреса, на который указывает p1, влево на байтность типа * 4, на 16 байт*/ p1 -= 4;
    /*приведения адреса а к double*, сдвиг влево на 8 байт, и присвоение этого адреса p3 */ 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;
```

																			61	FD																					6:	LFE			
D8	D9	DA	DB	DC	DD	DE	DF	EO	E1	E2	E3	E4	E5	E6	E7	E8	E9	EΑ	ЕВ	EC	ED	EE	EF	FO	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA FB	FC	FD	FE	FF	00 0	1 02	2 03	04	05	06	07
00	FE	61	00	00	00	00	00	FC	FD	61	00	00	00	00	00	00	FE	61	00	00	00	00	00	F4	FD	61	00	00	00	00	00	AA	4C	58 E8	7A	B6	FB	3F	00 0	0 20	41	05	00	00	00
p4								рЗ								p2								р1								С							b			а			
																												р1							рЗ										
																																							p2						
																																							p4						

p1 теперь указывает на FDF4, где записано 00000000, поэтому выводит 0, p3 указывает на FDFC, где записано 412000003FFBB67A, что равно 524288.124967

Выводы: переменные в памяти распределены в порядке объявления, только наоборот, те что раньше объявлены имеют больший адрес, другими словами, когда объявлена переменная а, она одна, после происходит объявление b, и b приписывается слева и имеет меньший адрес. Вообще получается, что в памяти все развернуто. Сами значения развернуты побайтно и расположены они в обратном порядке относительно ввода.

Задание 2.

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()
    /*объявление переменных*/
    char *p, c;
    int *a, b;
    float *x, y = 3.5;
    double *m, n;
    /* Адреса «обычных» переменных и размер выделяемой памяти */
                  char: start address %p extent %d\n",&c,sizeof(c));
    printf("b:
                int: start address %p extent %d\n",&b,sizeof(b));
    printf("y:
               float: start address %p extent %d\n\n",&y,sizeof(y));
    printf("n: double: start address %p extent %d\n\n",&n,sizeof(n));
    /* Адреса указателей и размер выделяемой памяти */
    printf("p: pointer: start address %p extent %d\n", &p, sizeof(p));
    printf("a: pointer: start address %p extent %d\n",&a,sizeof(a));
    printf("x: pointer: start address %p extent %d\n\n",&x,sizeof(x));
    printf("m: pointer: start address %p extent %d\n\n",&m,sizeof(m));
    /*присвоение указателю а адреса b, а указывает на b*/
    a = \&b;
    /*вывод сообщения и ввод b*/
   printf(" Enter b = ");
    scanf("%d", &b);
    /*вывод переменных*/
    printf("a=p\t*a=d\tb=dn", a, *a, b);
    /*приведение типа адреса, на который указывает а, к char* и присвоение его р, р указывает на а*/
    p = (char*)a;
   /* разыменование р и просвоение значения с*/
    /* присвоение переменной, на которую указывает р, значение записанное по адресу смещенному на 3 байта вправо*/
    *p = *(p+3);
    /* присвоение значения с ячейке памяти, смещенной на 3 байта вправо от р^*/
    *(p+3) = c;
    /*вывод переменных*/
    printf("p=%p\tc=%d\ta=%p\tb=%d\n", p, c, a, b);
    /*присвоение адреса у указателю х, х указывает на у*/
    x = &v;
    /*вывод переменыых*/
    printf("x=%p\t*x=%f\ty=%f\n", x, *x, y);
    /*приведение типа адреса, на который указывает x, к int* и присвоение его a, a указывает на y*/
    a = (int*)x;
    /*присвоение знвчения, на которое указывает х, ячейке памяти, на которую указывает а*/
```

```
*a = *x;
/*вывод переменыых*/
printf("a=p\t*a=d\tx=p\t*x=f\ty=f\n", a, *a, x, *x, y);
/*присвоение указателю а адреса b, а указывает на b*/
a = \&b;
/*присвоение у значения 12345.6789*/
v = 12345.6789;
/*вывод переменных*/
printf("x=%p\t*x=%f\ty=%f\n", x, *x, y);
/*приведение типа адреса, на который указывает x, к типу char* и присвоение его p, p указывает на y*/
p = (char*)x;
/*присвоение с значения, на которое указывает р*/
/*присвоение значения ячейки памяти, расположенной по адресу р + 3 байта, ячейке, на которую указывает р*/
*p=*(p+3);
/*присвоение ячейке памяти по адресу адрес, на который указывает р, + 3 байта, значения перемнной с*/
*(p+3)=c;
/*вывод переменных*/
printf("p=%p\tc=%d\tx=%p\ty=%f\n", p, c, x, y);
/*присвоение указателю m адрес n, m указывает на n*/
m = &n;
/*вывол переменных*/
printf("m=%p\t*m=%lf\tn=%lf\n", m, *m, n);
/*присвоение n значение 5.5*/
n = 5.5;
/*вывод переменных*/
printf("m=%p\t*m=%lf\tn=%lf\n", m, *m, n);
/*присвоение b, n, у значение 1.7 */
b = n = v = 1.7;
/*вывод переменных*/
printf("b=%d\ty=%f\tn=%lf\n", b, y, n);
printf("*a=%d\t*x=%f\t*m=%lf\n", *a, *x, *m);
/*сдвиг адреса, на который указывает m, на 8 байт вправо*/
m++;
/*вывод переменных*/
printf("n=%lf\tn=%p\tm=%p\n", n, &n, m);
/*разыменование указателей m,a,x, приведение a к float, x к int, нахождение разницы, нахождение суммы, присвоение значения m*/
*m = (float)*a - n + (int)*x;
/*вывод переменных*/
printf("m=%p\t*m=%lf\n", m, *m);
return 0;
```

Результаты работы программ:

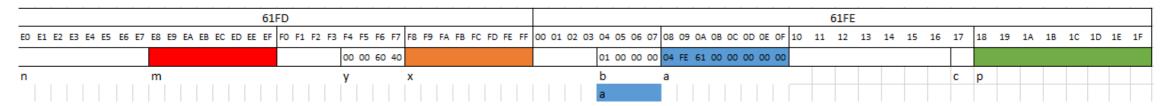
первый вариант (т+=2;)

```
char: start address 000000000061FE17 extent 1
       int: start address 000000000061FE04 extent 4
     float: start address 000000000061FDF4 extent 4
    double: start address 00000000001FDE0 extent 8
p: pointer: start address 000000000061FE18 extent 8
  pointer: start address 000000000061FE08 extent 8
  pointer: start address 000000000061FDF8 extent 8
m: pointer: start address 000000000061FDE8 extent 8
 Enter b = 1
a=000000000061FE04
                        *a=1
                                b=1
                                a=000000000061FE04
p=000000000061FE04
                        C=1
                                                        b=16777216
x=000000000061FDF4
                        *x=3.500000
                                        y=3.500000
a=000000000061FDF4
                                                        *x=0.000000
                        *a=3
                                x=000000000061FDF4
                                                                         y=0.000000
x=000000000061FDF4
                        *x=12345.678711 y=12345.678711
p=000000000061FDF4
                        c=-73 x=0000000000061FDF4
                                                        y = -0.000011
m=000000000061FDE0
                        *m=0.000000
                                        n=0.000000
m=000000000061FDE0
                        *m=5.500000
                                        n=5.500000
       y=1.700000
b=1
                        n=1.700000
*a=1
        *x=1.700000
                        *m=1.700000
n=1.700000
                n=000000000061FDE0
                                        m=000000000061FDF0
m=000000000061FDF0
                        *m=0.300000
Process returned 0 (0x0)
                           execution time : 1.701 s
Press any key to continue.
```

второй вариант (m++;)

```
char: start address 000000000061FE17 extent 1
       int: start address 000000000061FE04 extent 4
     float: start address 00000000001FDF4 extent 4
    double: start address 000000000061FDE0 extent 8
p: pointer: start address 000000000061FE18 extent 8
a: pointer: start address 000000000061FE08 extent 8
x: pointer: start address 000000000061FDF8 extent 8
m: pointer: start address 000000000061FDE8 extent 8
 Enter b = 1
a=000000000061FE04
                                b=1
                        *a=1
p=000000000061FE04
                                a=000000000061FE04
                                                         b=16777216
                        C=1
x=000000000061FDF4
                        *x=3.500000
                                        y=3.500000
                                                         *x=0.000000
                                                                         y=0.000000
a=0000000000061FDF4
                        *a=3
                                x=000000000061FDF4
                        *x=12345.678711 y=12345.678711
x=000000000061FDF4
                               x=0000000000061FDF4
p=000000000061FDF4
                                                        y=-0.000011
                        c = -73
                                        n=0.000000
n=000000000061FDE0
                        *m=0.000000
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=1.700000
               n=000000000061FDE0
                                        m=000000000061FDE8
Process returned -1073741819 (0xC0000005)
                                            execution time : 1.926 s
Press any key to continue.
```

Размещение переменных в памяти



```
/*приведение типа адреса, на который указывает а, к char*, потому что типы адресов разные, и присвоение его р,
    р указывает на последний байт b*/
    p = (char*)a;
    /* разыменование р и присвоение этого значения с*/
    /* присвоение ячейке памяти, на которую указывает р, значение, записанное по адресу смещенному на 3 байта вправо*/
    *p = *(p+3);
    /* присвоение значения с ячейке памяти, смещенной на 3 байта вправо от места, куда указывает р*/
    *(p+3) = c;
    /*вывод переменных*/
   printf("p=%p\tc=%d\ta=%p\tb=%d\n", p, c, a, b);
                                 61FD
                                                                                                               61FE
EO E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF FO F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF OO 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
                                            00 00 60 40
                                                                               00 00 00 01 04 FE 61 00 00 00 00 00
                                                                                                                               01 04 FE 61 00 00 00 00 00
n
                 m
                                                                                                                               с р
```

в итоге, в этом блоке поменяли местами первый и последний байт b, b стало равно 01000000 = 16777216

```
/*присвоение адреса у указателю x, x указывает на y*/
x = &y;
/*вывод переменыых*/
printf("x=%p\t*x=%f\ty=%f\n", x, *x, y);
/*приведение типа адреса, на который указывает x, к int* и присвоение его a, a указывает на y*/
a = (int*)x;
/*присвоение значения, на которое указывает x, ячейке памяти, на которую указывает a,
a указывает на целое, a x на дробное, при присвоении значение, на которое указывает x
неявно приводится к типу int, отбрасывая дробную часть*/
*a = *x;
/*вывод переменыых*/
printf("a=%p\t*a=%d\tx=%p\t*x=%f\ty=%f\n", a, *a, x, *x, y);
```

61F	-D				61FE	
EO E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF	FO F1 F2 F3 F4 F5 F6 F7	F8 F9 FA FB FC FD FE FF	00 01 02 03 04 05 06 07	08 09 0A 0B 0C 0D 0E 0F	10 11 12 13 14 15 16 17	18 19 1A 1B 1C 1D 1E 1F
	03 00 00 00	F4 FD 61 00 00 00 00 00	00 00 00 01	F4 FD 61 00 00 00 00 00	01	04 FE 61 00 00 00 00 00
n m	у	Х	b	a	С	p
	a x		р			

В итоге *a = 3, потому что 00000003 в типе int = 3, *x и y = 0, потому что не хватает точности вывода, на самом деле если раскодировать 00000003 как float, то получится 4.2039E-45

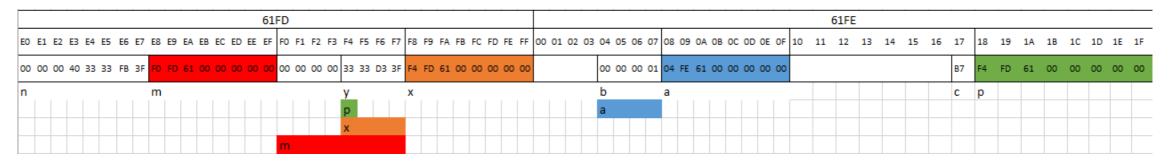
```
/*присвоение указателю а адреса b, а указывает на b*/
а = &b;
/*присвоение у значения 12345.6789*/
у = 12345.6789;
/*вывод переменных*/
printf("x=&p\t*x=&f\ty=&f\n", x, *x, y);
/*приведение типа адреса, на который указывает x, к типу char*, потому что типы указателей разные, и присвоение его p, p указывает на y*/
p = (char*)x;
/*присвоение с значения, на которое указывает p*/
c=*p;
/*присвоение значения ячейки памяти, расположенной по адресу p + 3 байта, ячейке, на которую указывает p*/
*p=*(p+3);
/*присвоение ячейке памяти, расположенной по адресу p + 3 байта, значения переменной c*/
*(p+3)=c;
/*вывод переменных*/
printf("p=%p\tc=%d\tx=%p\ty=%f\n", p, c, x, y);
```

														6	1F	D																																	61F	E																
EO	E1 E	E2 E	3 E	4 E	5 E	6 E	7 E	8 E	9 E	A E	B E	C EI	D E	E E	FF	0 F	1	-2	3	F4 F	5	F6 I	7	F8 F	9 F	A I	FB F	С	FD	FE	FF	00	01	02	03	04 (05 (6 0	7 08	3 09	9 0/	\ ОВ	ОС	OD	0E	OF 1	10	11	12	13	14	1	15	16	17	18	3	19	1A	18	3 :	1C	1D	1	Ε :	1F
																				46 E	6	40 E	37 F	F4 F	D 6	51 (00 (00	00	00	00					00 (00 (0 0	1 04	FE.	61	00	00	00	00	00									В7	F4		FD	61	00) (00	00	0	0 (00
n							r	n												у)	X												b			а																С	р										
												\perp								p x																а																														

В итоге в этом блоке первый байт и последний байт переменной у меняются местами, c = 6ывшему последнему байту y = B7 = 10110111, что в char = -73, y = B740E646 = -0.000011

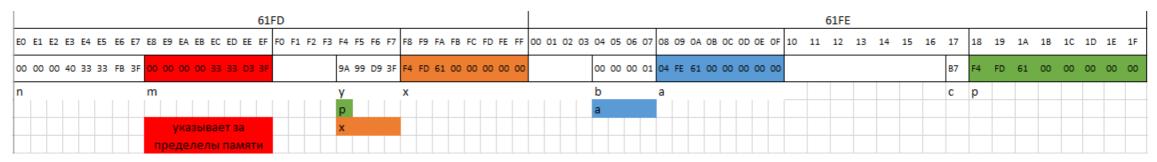
```
/*присвоение указателю m адрес n, m указывает на n*/
m = &n;
 /*вывод переменных*/
printf("m=%p\t*m=%lf\tn=%lf\n", m, *m, n);
 /*присвоение n значение 5.5*/
 n = 5.5;
 /*вывод переменных*/
 printf("m=%p\t*m=%lf\tn=%lf\n", m, *m, n);
 /*присвоение b, n, y значение 1.7, b = 1, так как тип привелся, отбросив дробную часть */
 b = n = y = 1.7;
 /*вывод переменных*/
printf("b=%d\ty=%f\tn=%lf\n", b, y, n);
 printf("*a=%d\t*x=%f\t*m=%lf\n", *a, *x, *m);
 /*сдвиг адреса, на который указывает m, на 16 байт вправо*/
 m+=2;
 /*вывод переменных*/
 printf("n=%lf\tn=%p\tm=%p\n", n, &n, m);
/*разыменование указателей m,a,x, приведение a к float, x к int, нахождение разницы, нахождение суммы, присвоение значения m,
 1-1.7+1=0.3*/
 *m = (float)*a - n + (int)*x;
 /*вывод переменных*/
 printf("m=%p\t*m=%lf\n", m, *m);
```

Поясняем, что происходит в этом блоке, иллюстрируем на схеме



m стал указывать на FDF0, туда записалось 3FD3333300000000 = 0.3

После замены инструкции «m+=2;» инструкцией «m++;» что изменилось и почему. Иллюстрируем на схеме.



Если оставить m++, то указатель начнет указывать сам на себя, ему присвоиться **3FD3333300000000**, и он должен туда указывать, но это выходит за границы выделенной памяти, поэтому система прерывает программу, чтобы ничего не сломалось.

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 Windows 10 LTSC 1809, среда разработки Code::Blocks

Текст программы (первый порядок объявления переменных):

```
#include <stdio.h>
#include <stdlib.h>
int main()
    /*объявление переменных*/
   char c1, c2;
    int i1, i2;
    double d1, d2;
    char* c;
    /* Адреса переменных и размер выделяемой памяти */
   printf("c1: char: start address %p extent %d\n",&c1,sizeof(c1));
   printf("c2: char: start address %p extent %d\n", &c2, sizeof(c2));
   printf("i1: int: start address %p extent %d\n",&i1,sizeof(i1));
   printf("i2:
                   int: start address %p extent %d\n",&i2,sizeof(i2));
   printf("d1: double: start address %p extent %d\n", &d1, sizeof(d1));
   printf("d2: double: start address %p extent %d\n", &d2, sizeof(d2));
   printf("c: pointer: start address %p extent %d\n", &c, sizeof(c));
    i1 = 0x1abcdef2;
    c=&i1;
    for (int i = 0; i < 4; i++)
   printf("%x ",*(c+i));
```

Результаты работы программы:

```
c1: char: start address 0000000000061FE1B extent 1
c2: char: start address 000000000061FE1A extent 1
i1: int: start address 000000000061FE1A extent 4
i2: int: start address 000000000061FE10 extent 4
d1: double: start address 000000000061FE08 extent 8
d2: double: start address 000000000061FE00 extent 8
c: pointer: start address 000000000061FDF8 extent 8
fffffff2 ffffffde ffffffbc 1a
Process returned 0 (0x0) execution time: 0.022 s
Press any key to continue.
```

Размещение переменных в памяти

			61	.FD																		(51F	E														
F8	F9	FA	FB	FC	FD	FE	FF	00 0	1	02 (03	04	05	06	07	08	09	OA	ОВ	oc	OD	0E	OF	10	11	12	13	14	15	16	17	18	19	1A	18	1C	1D	1E
14	FE	61	00	00	00	00	00																					F2	DE	вс	1A							
С								d2								d1								i2				i1						c2	c1			
																												С										

Текст программы (второй порядок объявления переменных):

```
#include <stdio.h>
#include <stdlib.h>
int main()
   /*объявление переменных*/
   int i1;
   char c1;
   double d1;
   char* c;
   char c2;
   double d2;
   int i2;
/* Адреса переменных и размер выделяемой памяти */
   printf("c1: char: start address %p extent %d\n",&c1,sizeof(c1));
   printf("c2: char: start address %p extent %d\n", &c2, sizeof(c2));
   printf("i1: int: start address %p extent %d\n",&i1,sizeof(i1));
   printf("i2: int: start address %p extent %d\n", &i2, sizeof(i2));
   printf("d1: double: start address %p extent %d\n",&d1,sizeof(d1));
   printf("d2: double: start address %p extent %d\n",&d2,sizeof(d2));
   printf("c: pointer: start address %p extent %d\n", &c, sizeof(c));
   i1 = 0x1abcdef2;
   c=&i1;
   for (int i = 0; i < 4; i + +)
   printf("%x ",*(c+i));
```

Результаты работы программы:

```
c1: char: start address 00000000061FE17 extent 1
c2: char: start address 00000000061FDFF extent 1
i1: int: start address 00000000061FE18 extent 4
i2: int: start address 000000000061FDEC extent 4
d1: double: start address 000000000061FDEC extent 8
d2: double: start address 000000000061FDF0 extent 8
c: pointer: start address 000000000061FDF0 extent 8
ffffffff fffffde ffffffbc 1a
Process returned 0 (0x0) execution time: 0.007 s
Press any key to continue.
```

Размещение переменных в памяти

										(51	FD																									6	1FI	E														
EC E	D	EE	EF	F	0	F1	F2	FE	B F	4 1	5	F6	F7	F	8 F	9	FA	FB	FC	F	D	FE	FF	00	0 01	1 0	2 03	04	05	5 06	5 07	08	0	9 0/	А ОВ	oc	00	O 0E	OF	10	11	1 1	12 1	3	14	19	16	17	7 1	8 1	9 1	1A	1B
				T																				18	B FE	: 6	1 00	00	00	0 0	00																		F.	2 0	E E	вс	1A
i2				(d2																		c2	С								d1	1															C.	1 ii	L			

Выводы: порядок записи зависит от объявления, переменные,

что объявлены раньше будут иметь больший адрес, объявленные позже – меньший.

Запись происходит определенным образом, данные как бы приписываются слева,

то есть была в памяти только і1, при объявлении следующей с1 память выделяется слева

Порядок следования байт при записи числа для меня обратный, если рассматривать отдельно

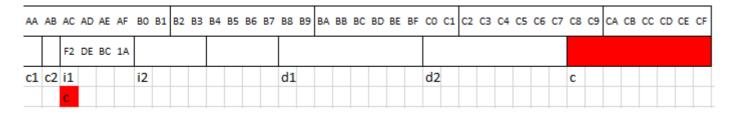
ячейку памяти, то байты расположены наоборот, но для компьютера возможно и прямой, если чтение происходит справа налево.

2. Операционная система GNU, https://www.onlinegdb.com/online_c_compiler

Результаты работы программы при первом порядке объявления переменных:

```
c1: char: start address 0x7ffe78ef7caa extent 1
c2: char: start address 0x7ffe78ef7cab extent 1
i1: int: start address 0x7ffe78ef7cac extent 4
i2: int: start address 0x7ffe78ef7cb0 extent 4
d1: double: start address 0x7ffe78ef7cb8 extent 8
d2: double: start address 0x7ffe78ef7cc0 extent 8
c: pointer: start address 0x7ffe78ef7cc8 extent 8
ffffffff ffffffde ffffffbc 1a
...Program finished with exit code 0
Press ENTER to exit console.
```

Размещение переменных в памяти:



Результаты работы программы при втором порядке объявления переменных:

```
c1: char: start address 0x7ffe855ad42a extent 1
c2: char: start address 0x7ffe855ad42b extent 1
i1: int: start address 0x7ffe855ad42c extent 4
i2: int: start address 0x7ffe855ad430 extent 4
d1: double: start address 0x7ffe855ad438 extent 8
d2: double: start address 0x7ffe855ad448 extent 8
c: pointer: start address 0x7ffe855ad440 extent 8
fffffff2 ffffffde ffffffbc 1a
...Program finished with exit code 0
Press ENTER to exit console.
```

Размещение переменных в памяти:

2A	2B	20	20) 2E	2F	30	31	32	33	34	35	36	37	38	39	за	38	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C 4	4D	4E	4F
		F2	DE	ВС	1A																																
c1	c2	i1				i2								d1								С								d2							
		С																																			

Выводы: на этой системе память распределяется от младшего к старшему, то есть по занимаемой памяти, но также учитывается порядок объявления, если равны по занимаемой памяти, то есть если была объявлена переменная байтностью 8, в след за ней такая же, то она будет после первой.

Порядок следования байт при записи числа такой же, как и на прошлой системе.

Выводы: мне кажется, методы распределения памяти и порядок следования байт в записи числа зависит в основном от операционной системы.