

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:

```
a:      int: start address 00000000061FE04 extent 4
b:      float: start address 00000000061FE00 extent 4
c:      double: start address 00000000061FDF8 extent 8

p1: pointer: start address 00000000061FDF0 extent 8
p2: pointer: start address 00000000061FDE8 extent 8
p3: pointer: start address 00000000061FDE0 extent 8

p1: 00000000061FE04 related value 1
p2: 00000000061FE00 related value 2.000000
p3: 00000000061FDF8 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=00000000061FE00      p2=00000000061FE00      p3=00000000061FE00      p4=00000000061FE00
*p1=1092616192  *p2=10.000000  *p3=0.000000  *(float*)p4=10.000000

p1=00000000061FE04      p2=00000000061FE00      p3=00000000061FDF8
*p1=5      *p2=10.000000  *p3=1.732051
p1=00000000061FDF4      p2=00000000061FE00      p3=00000000061FDFC
*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 */
printf("a:      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

61FD																																61FE																
D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	00	01	02	03	04	05	06	07	
								F8	FD	61	00	00	00	00	00	00	FE	61	00	00	00	00	00	04	FE	61	00	00	00	00	00	00	00	00	00	00	00	00	08	40	00	00	00	40	01	00	00	00
p4								p3								p2								p1								c				b				a								
																																p3				p2				p1								

```

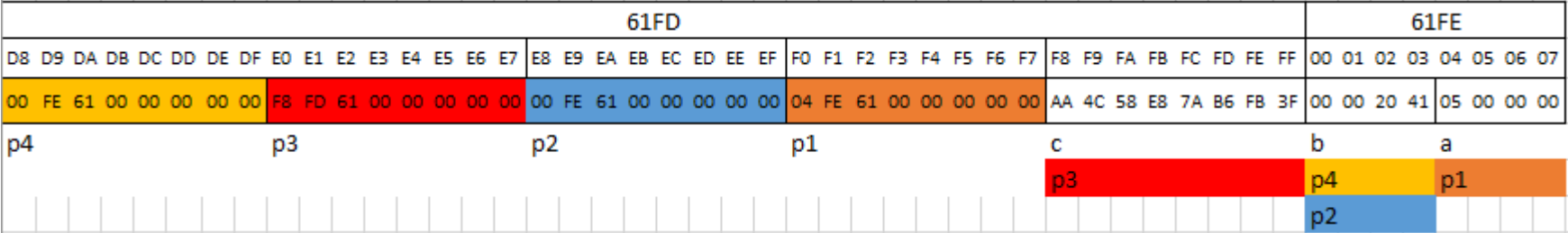
/* Использование указателей в выражениях */
printf("a=%d\tb=%f\tc=%lf\n", a, b, c);
/*5 присваивается переменной, на которую указывать указатель p1, то есть a*/
*p1 = 5;
/*происходить разыменование p2 и p1, в переменную,
на которую указывает p2, то есть в переменную b, записывается
произведение значения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

```

61FD																																61FE															
D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	00	01	02	03	04	05	06	07
00	FE	61	00	00	00	00	00	00	FE	61	00	00	00	00	00	00	FE	61	00	00	00	00	00	00	FE	61	00	00	00	00	00	AA	4C	58	E8	7A	B6	FB	3F	00	00	20	41	5	00	00	00
p4								p3								p2								p1								c								b				a			
																																p3								p4				p1			
																																								p3							
																																								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);
```



61FD																																61FE																															
D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	00	01	02	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	00	20	41	05	00	00	00																
p4								p3								p2								p1								c								b								a															
																p1																								p3																							
																																																p2															
																																																p4															

Выводы: переменные в памяти распределены в порядке объявления, только наоборот, те что раньше объявлены имеют больший адрес, другими словами, когда объявлена переменная *a*, 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.

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.

```
#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 */
    printf("c:      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 a адреса b, a 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=%d\n", a, *a, b);
    /*приведение типа адреса, на который указывает a, к char* и присвоение его p, p указывает на a*/
    p = (char*)a;
    /* разыменование p и присвоение значения c*/
    c = *p;
    /* присвоение переменной, на которую указывает p, значение записанное по адресу смещенному на 3 байта вправо*/
    *p = *(p+3);
    /* присвоение значения c ячейке памяти, смещенной на 3 байта вправо от p*/
    *(p+3) = c;
    /*вывод переменных*/
    printf("p=%p\tc=%d\ta=%p\tb=%d\n", p, c, a, b);
    /*присвоение адреса y указателю x, x указывает на y*/
    x = &y;
    /*вывод переменных*/
    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 = *x;
/*вывод переменных*/
printf("a=%p\t*a=%d\tx=%p\t*x=%f\ty=%f\n", a, *a, x, *x, y);
/*присвоение указателю a адреса b, a указывает на b*/
a = &b;
/*присвоение y значения 12345.6789*/
y = 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 c 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 по адресу адрес, на который указывает 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);
/*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 = 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 8 байт вправо*/
m++;
/*вывод переменных*/
printf("n=%lf\tn=%p\tn=%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 (m+=2;)

```
c:   char: start address 000000000061FE17 extent 1
b:   int: start address 000000000061FE04 extent 4
y:   float: start address 000000000061FDF4 extent 4
n:   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      *a=1      b=1
p=000000000061FE04      c=1      a=000000000061FE04      b=16777216
x=000000000061FDF4      *x=3.50000      y=3.50000
a=000000000061FDF4      *a=3      x=000000000061FDF4      *x=0.000000      y=0.000000
x=000000000061FDF4      *x=12345.678711 y=12345.678711
p=000000000061FDF4      c=-73      x=000000000061FDF4      y=-0.000011
m=000000000061FDE0      *m=0.000000      n=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=000000000061FDF0
m=000000000061FDF0      *m=0.300000

Process returned 0 (0x0)   execution time : 1.701 s
Press any key to continue.
```

второй вариант (m++;)

```

c:      char: start address 000000000061FE17 extent 1
b:      int:  start address 000000000061FE04 extent 4
y:      float: start address 000000000061FDF4 extent 4
n:      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      *a=1      b=1
p=000000000061FE04      c=1      a=000000000061FE04      b=16777216
x=000000000061FDF4      *x=3.500000      y=3.500000
a=000000000061FDF4      *a=3      x=000000000061FDF4      *x=0.000000      y=0.000000
x=000000000061FDF4      *x=12345.678711 y=12345.678711
p=000000000061FDF4      c=-73 x=000000000061FDF4      y=-0.000011
m=000000000061FDE0      *m=0.000000      n=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=000000000061FDE0

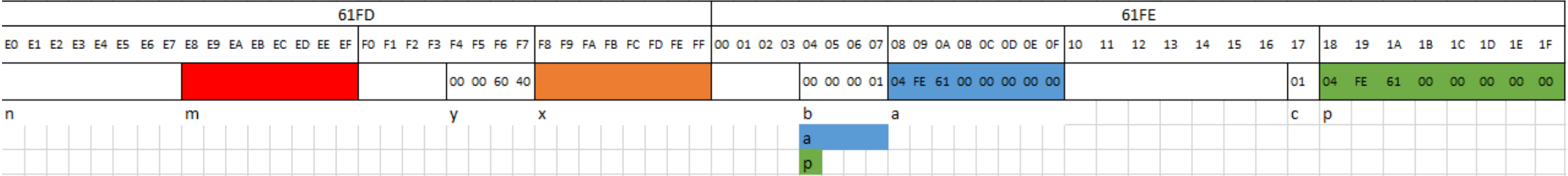
Process returned -1073741819 (0xC0000005)   execution time : 1.926 s
Press any key to continue.

```

Размещение переменных в памяти

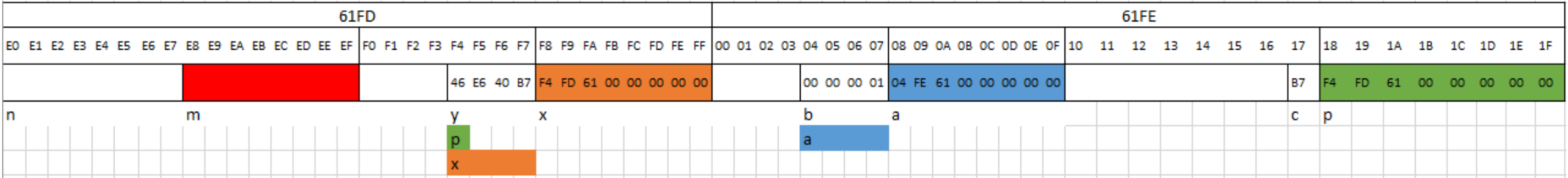
[illegible]

```
/*приведение типа адреса, на который указывает a, к char*, потому что типы адресов разные, и присвоение его p,
p указывает на последний байт b*/
p = (char*)a;
/* разыменованное p и присвоение этого значения c*/
c = *p;
/* присвоение ячейке памяти, на которую указывает p, 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 p*/
*(p+3) = c;
/*вывод переменных*/
printf("p=%p\tc=%d\ta=%p\tb=%d\n", p, c, a, b);
```



61FD																61FE																																																							
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	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								
n							m													y		x						b		a																c	p																								
																				a								p																																											
																				x																																																			

```
/*присвоение указателю a адреса b, a указывает на b*/
a = &b;
/*присвоение y значения 12345.6789*/
y = 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 c 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);
```



В итоге в этом блоке первый байт и последний байт переменной y меняются местами, c = бывшему 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\tn=%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

61FD																61FE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
E0	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																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
00	00	00	40	33	33	FB	3F	F0	FD	61	00	00	00	00	00	00	00	00	00	33	33	D3	3F	F4	FD	61	00	00	00	00	00					00	00	00	01	04	FE	61	00	00	00	00	00								B7	F4	FD	61	00	00	00	00	00																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
n								m												y				x												b				a																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										

m стал указывать на **FDF0**, туда записалось **3FD3333000000000 = 0.3**

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

[illegible]

Если оставить m++, то указатель начнет указывать сам на себя, ему присвоится **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.

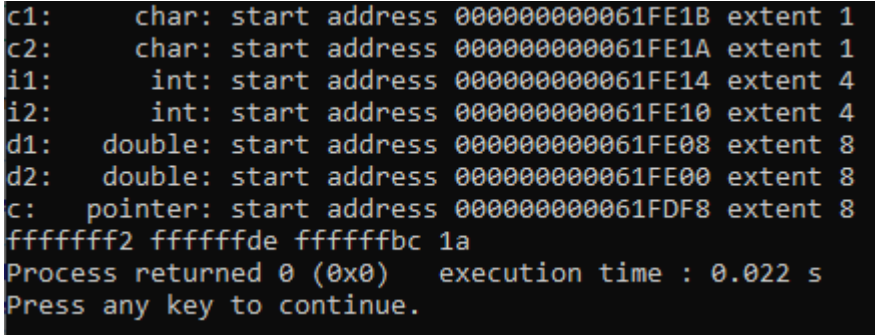
1. Операционная система Windows 10 LTSC 1809, среда разработки Code::Blocks

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 000000000061FE1B extent 1
c2:      char: start address 000000000061FE1A extent 1
i1:      int: start address 000000000061FE14 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
ffffff2 fffffffde fffffffbc 1a
Process returned 0 (0x0)  execution time : 0.022 s
Press any key to continue.
```

Размещение переменных в памяти

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 000000000061FE17 extent 1
c2:      char: start address 000000000061FDFF extent 1
i1:      int: start address 000000000061FE18 extent 4
i2:      int: start address 000000000061FDEC extent 4
d1:     double: start address 000000000061FE08 extent 8
d2:     double: start address 000000000061FDF0 extent 8
c:   pointer: start address 000000000061FE00 extent 8
ffffff2 fffffffde fffffffbc 1a
Process returned 0 (0x0)   execution time : 0.007 s
Press any key to continue.
```

Размещение переменных в памяти

[illegible]

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 $i1$, при объявлении следующей $s1$ 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.

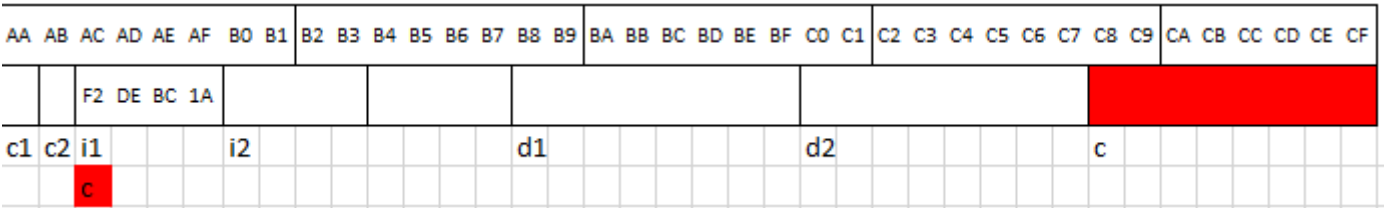
2. Операционная система GNU, https://www.onlinegdb.com/online_c_compiler

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
ffffff2 fffffffde fffffffbc 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
ffffff2 fffffffde fffffffbc 1a

...Program finished with exit code 0
Press ENTER to exit console.
```

Размещение переменных в памяти:

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

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.