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»

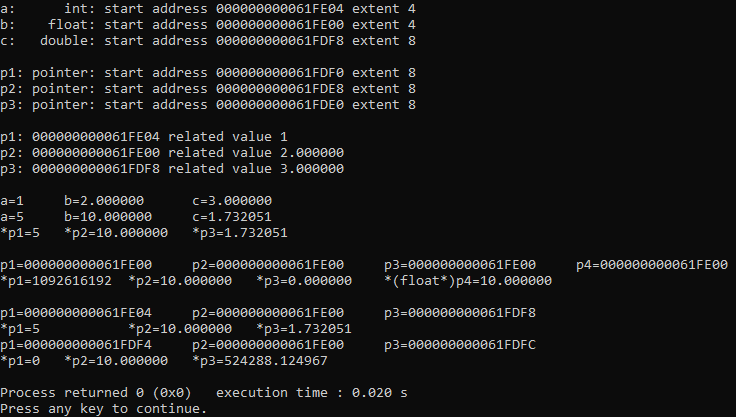
Выполнил:  
Студент Альков В.С.  
Группа И407Б  
  
Преподаватель: Першин Д.В.

Санкт-Петербург  
2020 г.

**Задание 1.**

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

Результаты работы программы:



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));

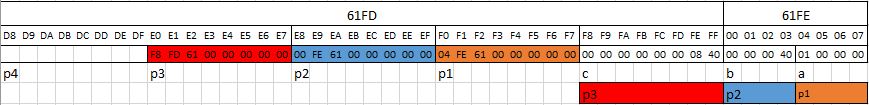
/\* Значения, на которые ссылаются указатели \*/

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**



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

/\* 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 p3, то есть c, 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 = 01000001001000000000000000000000,

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 a,

в итоге он указывает на значение 0000000541200000, 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,

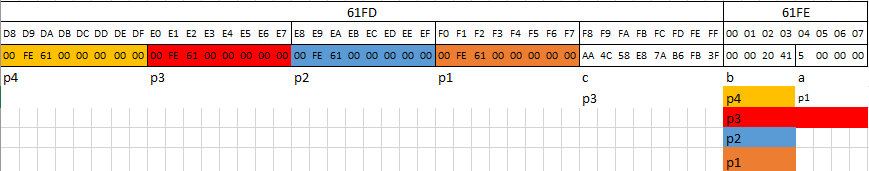
с типом void\* можно указывать на любой тип, но чтобы получить значение, на адрес которого такой указатель указывает

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);



В случае с p3 выдался 0 из-за нехватки точности выведения.

/\* Изменение значений указателей \*/

/\*сдвиг адреса, на который указывает p1, вправо на байтность типа, на 4 байта\*/

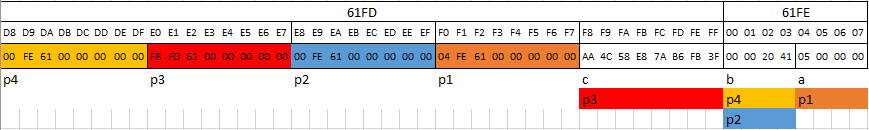
p1++;

/\*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);



/\*сдвиг адреса, на который указывает p1, влево на байтность типа \* 4, на 16 байт\*/

p1 -= 4;

/\*приведения адреса a к double\*, сдвиг влево на 8 байт, и присвоение этого адреса p3 \*/

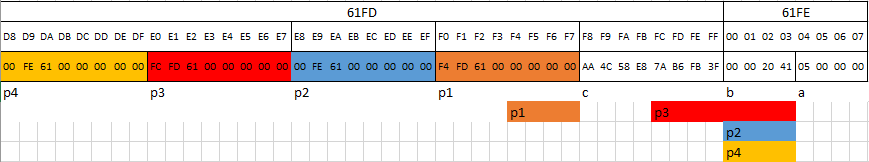
p3 = (double\*)&a - 1;

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;

}



**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 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.**

**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 \*/

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 просвоение 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 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 x, ячейке памяти, на которую указывает a\*/

\*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 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 перемнной 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\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\*/

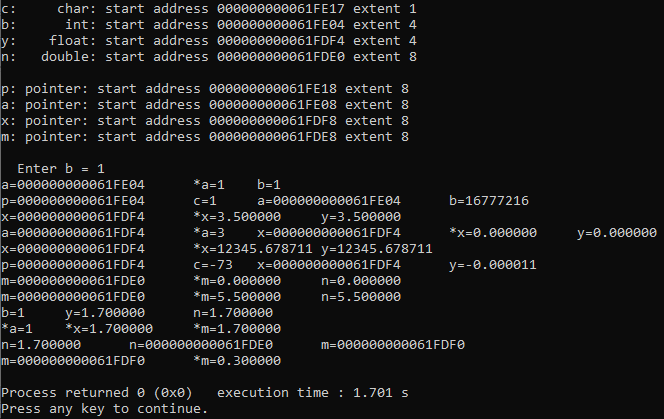
printf("m=%p\t\*m=%lf\n", m, \*m);

return 0;

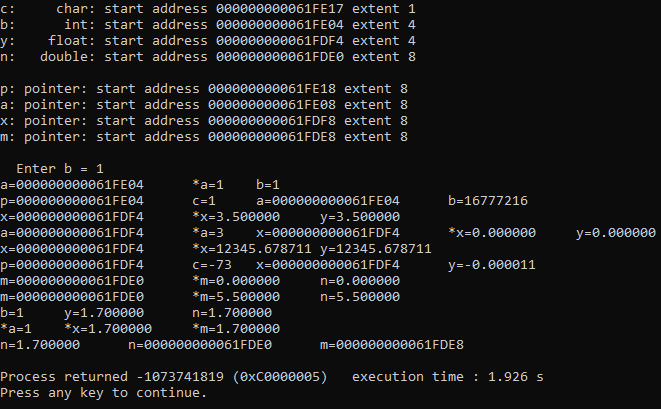
}

Результаты работы программ:

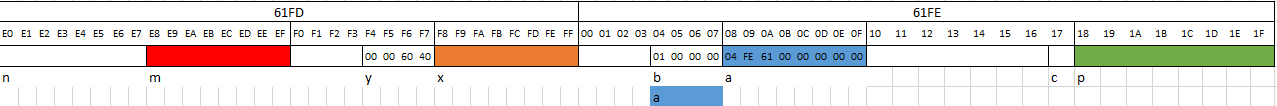
первый вариант (m+=2;)



второй вариант (m++;)



Размещение переменных в памяти



/\*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;

/\* разыменование p и присвоение этого значения c\*/

c = \*p;

/\* присвоение ячейке памяти, на которую указывает p, значение, записанное по адресу смещенному на 3 байта вправо\*/

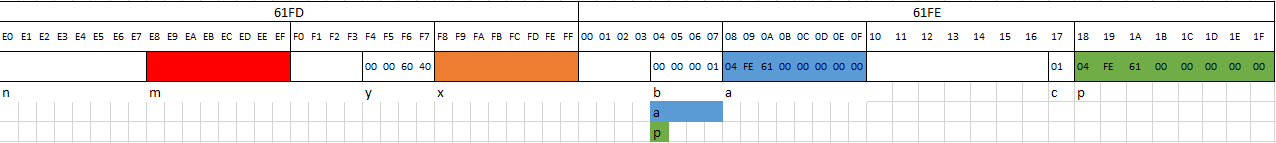
\*p = \*(p+3);

/\* присвоение значения c 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);



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

/\*присвоение адреса y 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,

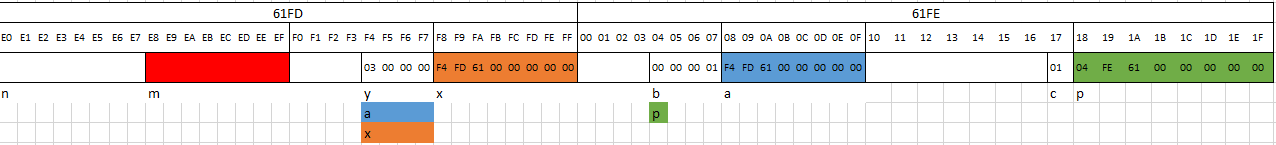
a указывает на целое, а x на дробное, при присвоении значение, на которое указывает x

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);



***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 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\*/

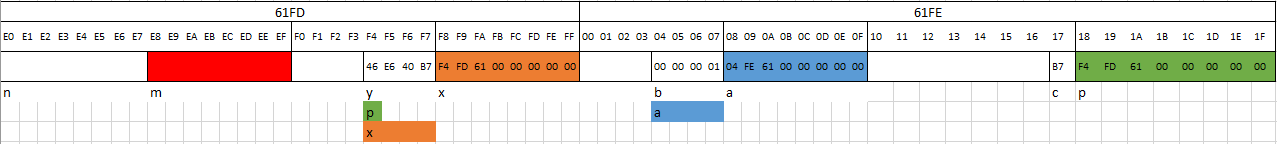
\*p=\*(p+3);

/\*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\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,

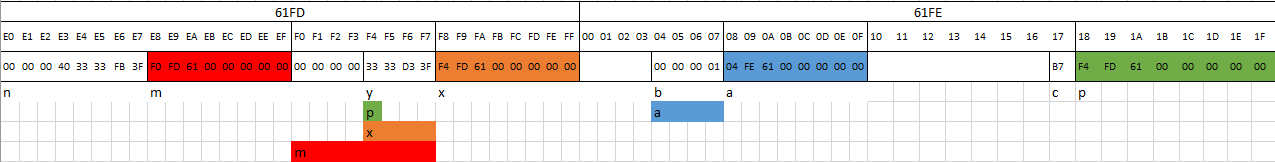
1-1.7+1=0.3\*/

\*m = (float)\*a - n + (int)\*x;

/\*вывод переменных\*/

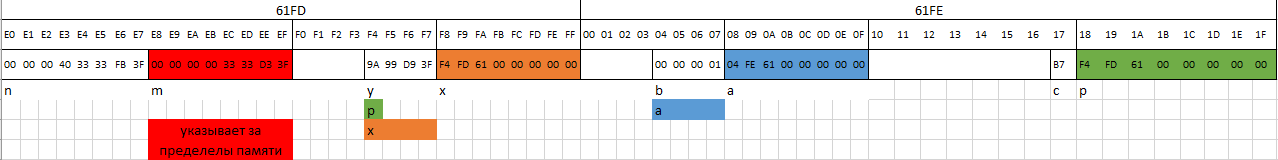
printf("m=%p\t\*m=%lf\n", m, \*m);

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;

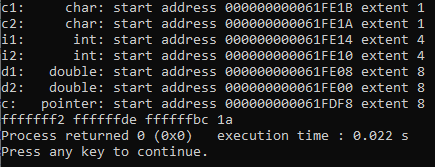
c=&i1;

for (int i = 0; i<4; i++)

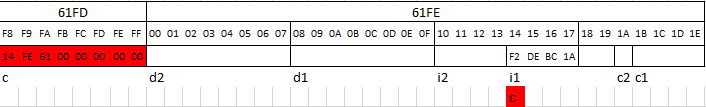
printf("%x ",\*(c+i));

}

**Результаты работы программы:**



**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;

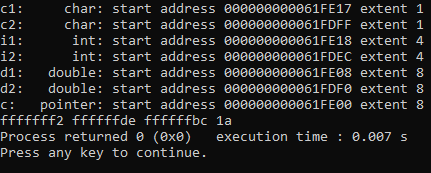
c=&i1;

for (int i = 0; i<4; i++)

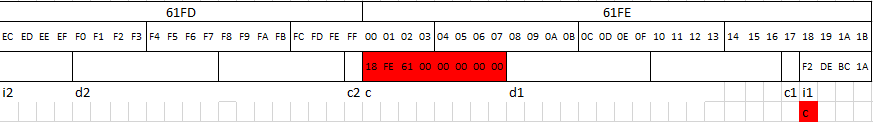
printf("%x ",\*(c+i));

}

**Результаты работы программы:**



Размещение переменных в памяти



**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,

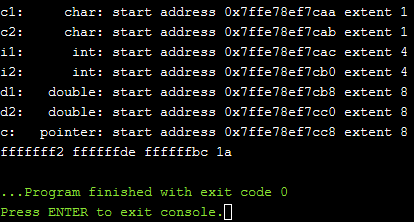
то есть была в памяти только i1, при объявлении следующей c1 память выделяется слева

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

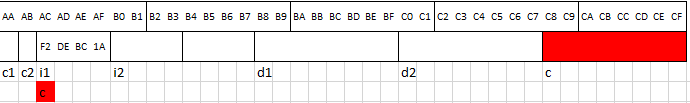
ячейку памяти, то байты расположены наоборот, но для компьютера возможно и прямой, если чтение происходит справа налево.

1. **Операционная система GNU, https://www.onlinegdb.com/online\_c\_compiler**

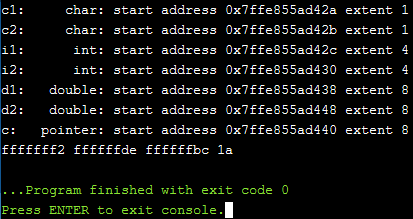
Результаты работы программы при первом порядке объявления переменных:



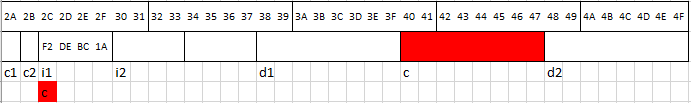
Размещение переменных в памяти:



Результаты работы программы при втором порядке объявления переменных:



Размещение переменных в памяти:



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.