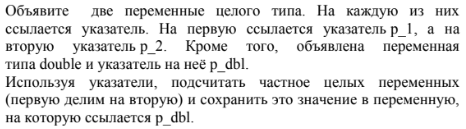
Лабораторная работа №9.

Задание 1.



Код:

#include <stdio.h>

#include <locale.h>

#include <windows.h>

**int** main(){

**int** frst,scnd,\*p\_1=&frst,\*p\_2=&scnd;

**double** thrd,\*p\_dbl=&thrd;

setlocale(LC\_ALL,"Russian");

printf("Введите два целых числа:**\n**");

scanf("%d%d",&frst,&scnd);

\*p\_dbl=(**double**)\*p\_1/(\*p\_2);

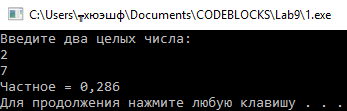
printf("Частное = %0.3f**\n**",\*p\_dbl);

system("pause");

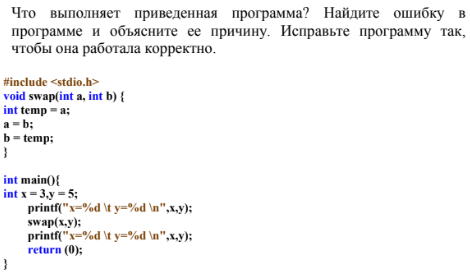
**return** 0;

}

Результат:



Задание 2.



Код:

#include <stdio.h>

**void** swap(**int** \*a, **int** \*b) {

**int** temp = \*a;

\*a = \*b;

\*b = temp;

}

**int** main(){

**int** x = 3,y = 5;

printf("x=%d **\t** y=%d **\n**",x,y);

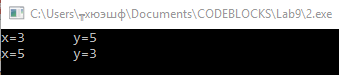
swap(&x,&y);

printf("x=%d **\t** y=%d **\n**",x,y);

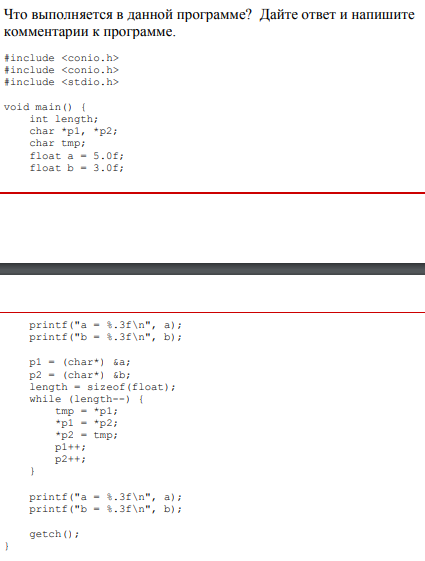
**return** (0);

}

Результат:



Задание 3.



Код:

#include <conio.h>

#include <stdio.h>

**int** main(**void**) {

**int** length;

**char** \*p1, \*p2;

**char** tmp;

**float** a = 5.0f;

**float** b = 3.0f;

printf("a = %.9f**\n**", a);

printf("b = %.9f**\n**", b);

p1 = (**char**\*) &a;

p2 = (**char**\*) &b;

*/\* "(char\*)" - формальная запись, говорящая о том, что данный адрес*

*&a(&b для p2) будет записан в указатель типа char. Запись*

*необязательна и предназначена для устранения предупреждений*

*со стороны компилятора о различии в типах объекта и указателя.*

*Т.к. в языке C размер типа char равен одному байту, то и данные*

*указатели p1 и p2 будут возвращать лишь один первый байт от значений*

*типа float. Для того, чтобы получить доступ ко всему значению типа*

*float в данном случае, необходимо последовательно обратиться*

*к последующим трём байтам памяти, т.к. размер типа float - 4 байта\*/*

length = **sizeof**(**float**); *// вычисление размера типа float*

*/\* Организация цикла, в котором последовательно прогоняют*

*4 байта в соответствии с размером типа float. Кроме того,*

*на каждой итерации переменные a и b, доступ к которым*

*осуществлен через разыменование указателей, меняются значениями \*/*

printf("%p**\n**",&a);

printf("%p**\n\n**",&b);

**while** (length--) {

tmp = \*p1;

\*p1 = \*p2;

\*p2 = tmp;

p1++;

p2++;

}

printf("a = %.9f**\n**", a);

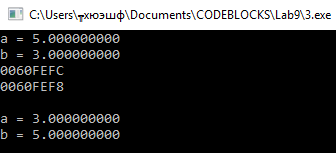
printf("b = %.9f**\n**", b);

getch();

**return** 0;

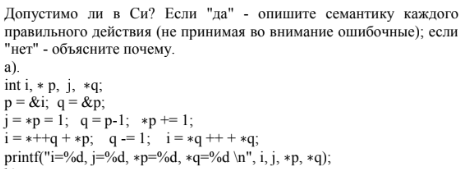
}

Результат:



Задание 4.

a)



Код:

#include <stdio.h>

**int** main(){

**int** i, \*p, j, \*q;

*/\* объявление двух переменных типа int, двух указателей типа int \*/*

p = &i; *// Допустимо, присвоение указателю адреса переменной того же типа*

q = &p; *// Допустимо, присвоение указателю адреса переменной того же типа*

j = \*p = 1; */\* Допустимо, т.к. присвоения происходят справа налево.*

*Переменной i, на которую ссылается указатель p присваивается 1, затем*

*то же значение присваивается переменной j \*/*

q = p-1; */\* Допустимое действие, но в контексте данной программы*

*бессмысленно. Смещение указателя на 4 бита назад относительно p\*/*

\*p += 1; *// Допустимо. Увеличение значения переменной по адресу p на 1.*

i = \*++q + \*p; */\* Допустимое действие, но в данной программе не имеет*

*смысла. "\*++q" - получение значения переменной, следующей за p.*

*"+ \*p" - прибавление значения переменной, содержащейся по адресу p. \*/*

q -= 1; *// Допустимо. Смещение указателя на одну переменную назад.*

i = \*q ++ + \*q; */\* Допустимо. "\*q++" - Разыменование указателя и*

*последующее его перемещение к следующей переменной.*

*"+ \*q" - прибавление значения уже следующей переменной. \*/*

printf("i=%d, j=%d, \*p=%d, \*q=%d **\n**", i, j, \*p, \*q);

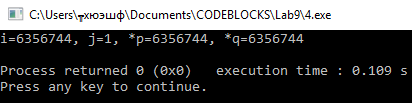
*/\* Использование printf с данными аргументами допустимо, однако при*

*выводе данных значений целесообразнее использовать модификаторы \*/*

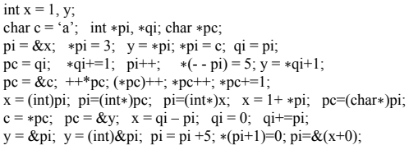
**return** 0;

}

Результат:



b)



Код:

#include <stdio.h>

**int** main(){

**int** x = 1, y; *// Допустимо. Инициализация одной и объявление другой переменных.*

**char** c = 'a'; *// Допустимо. Присвоение переменной типа char символа 'a'.*

**int** \*pi, \*qi; *// Допустимо. Объявление двух указателей на тип int.*

**char** \*pc; *// Допустимо. Объявление указателя на тип char.*

pi = &x; *// Допустимо. Присвоение указателю pi адреса переменной x.*

\*pi = 3; *// Допустимо. Присвоение переменной x, содержащейся по адресу pi значения 3.*

y = \*pi; *// Допустимо. Присвоение значения переменной x содержащейся по адресу pi переменной y.*

\*pi = c; *// Допустимо. Присвоение переменной x, содержащейся по адресу pi значения переменной c.*

qi = pi; *// Допустимо. Присвоение указателю qi адреса, хранящегося в указателе pi.*

pc = qi; *// Допустимо. Несмотря на разные типы указателей, операция присвоения допустима.*

\*qi++; *// Допустимо. Получение значения по адресу qi и смещение указателя к следующей переменной.*

pi++; *// Допустимо. Перемещение указателя к следующей переменной по отношению к текущей.*

\*(--pi) = 5; *// Допустимо. Присвоение предыдущей по отношению к текущей переменной, на которую указывает pi значения 5.*

y = \*qi+1; *// Допустимо. Увеличение переменной по адресу qi на 1.*

pc = &c; */\* Допустимо. Присвоение указателю типа char адреса*

*переменной типа char \*/*

++\*pc; *// Допустимо. Инкрементация переменной по адресу pc.*

(\*pc)++; *// Допустимо. Инкрементация переменной по адресу pc.*

x = (**int**)pi; *// Допустимо. Приведение типа значения переменной pi к int и присвоение переменной x.*

pi=(**int**\*)pc; *// Допустимо. Присвоение адреса хранящегося в переменной pc переменной pi.*

pi=(**int**\*)x; *// Допустимо. Присвоение значения переменной x указателю pi.*

x = 1+ \*pi; *// Допустимо. Присвоение переменной x суммы 1 и значения по адреса pi.*

pc=(**char**\*)pi; *// Допустимо. Присвоение значения указателя pi указателю pc.*

c = \*pc; *// Допустимая операция, присвоение переменной c значения по адресу в pc.*

pc = &y; *//Допустимо. Присвоение указателю pc адреса переменной y.*

x = qi - pi; *// Допустимая операция. Возвращает разность между значениями qi и pi деленную на размер типа данных int.*

qi = 0; *// Допустимая операция. Присвоение указателю qi значения 0.*

*//qi+=pi; // Недопустимо. Данная операция смещает указатель на i-ое кол-во переменных, в данном случае на pi. В качестве аргумента не может выступать второй указатель.*

y = &pi; *// Допустимо. Присвоение переменной y адреса указателя pi.*

y = (**int**)&pi; *// Допустимо. Операция эквивалента предыдущей.*

pi = pi +5; *// Допустимо. Смещение указателя pi на 5 переменных.*

printf("pi=%p**\n**",pi);

printf("x=%d**\n**",x);

printf("y=%p**\n**",y);

printf("c=%d**\n**",c);

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

printf("qi=%p**\n**",qi);

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

printf("pi=%p**\n**",pi);

**return** 0;

}

Результат:

