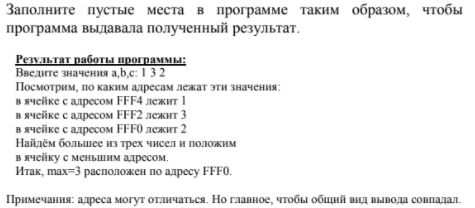
Лабораторная работа №7

Указатели на простейшие типы.

Операции над указателями.

Задание 1.



Код:

#include<stdio.h>

#include<conio.h>

#include<windows.h>

#include <locale.h>

**void** main()

{

SetConsoleCP(CP\_UTF8);

SetConsoleOutputCP(CP\_UTF8);

**int** a,b,c,max;

**int** \*pa=&a, */\* Переменная pa содержит адрес переменной a \*/*

\*pb=&b, */\* Переменная pb содержит адрес переменной b \*/*

\*pc=&c, */\* Переменная pc содержит адрес переменной c \*/*

\*min;

printf("Введите значения a,b,c: ");

scanf("%d %d %d",&a,&b,&c);

printf("Посмотрим, по каким адресам лежат эти значения:**\n**");

printf(" в ячейке с адресом %p лежит %d**\n**",pa,a);

printf(" в ячейке с адресом %p лежит %d**\n**",pb,b);

printf(" в ячейке с адресом %p лежит %d**\n**",pc,c);

printf("Найдем большее из чисел a, b, c и положим**\n**");

printf("в ячейку с меньшим адресом.**\n**");

*/\* Поиск наименьшего адреса \*/*

**if** (pa<=pb)

min=pa;

**else** min=pb;

**if** (pc<=min)

min=pc;

*/\*Допишите блок кода, чтобы был полностью реализован поиск наименьшего*

*адреса (не рассмотрен еще один адрес при сравнении)\*/*

*/\* Поиск наибольшего значения \*/*

**if** (\*pa<=\*pb)

max=\*pb; *// Объясните, почему перед ―max‖ не стоит ―\*‖*

**else** max=\*pa; *// Запись \*max означает обращение к ячейке памяти по адресу max. Нам же нужно обычное присвоение.*

**if** (max<=\*pc)

max=\*pc;

\*min=max;

*/\* Требуемое размещение \*/*

*/\*Положите значение наибольшего числа в ячейку с меньшим адресом*

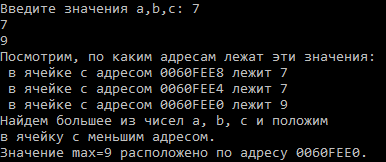
*(Операция включает в себя два операнда и знак равенства между операндами) \*/*

printf("Значение max=%d расположено по адресу %p.**\n\n**", max,min);

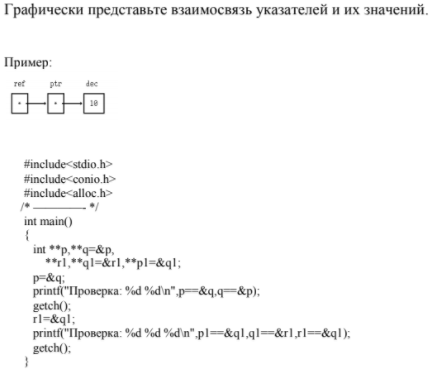
getch();

}

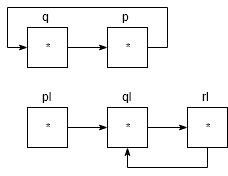
Результат:



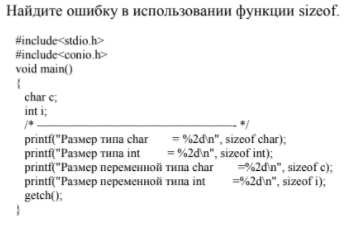
Задание 2.



Графическое предстваление:



Задание 3.



Код:

#include<stdio.h>

#include<conio.h>

**void** main()

{

**char** c;

**int** i;

*/\* Синтаксис функции sizeof предполагает заключение типа в скобки. \*/*

printf("Размер типа char = %2d**\n**", **sizeof** (**char**));

printf("Размер типа int = %2d**\n**", **sizeof** (**int**));

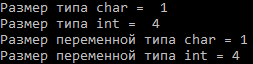
printf("Размер переменной типа char =%2d**\n**", **sizeof** c);

printf("Размер переменной типа int =%2d**\n**", **sizeof** i);

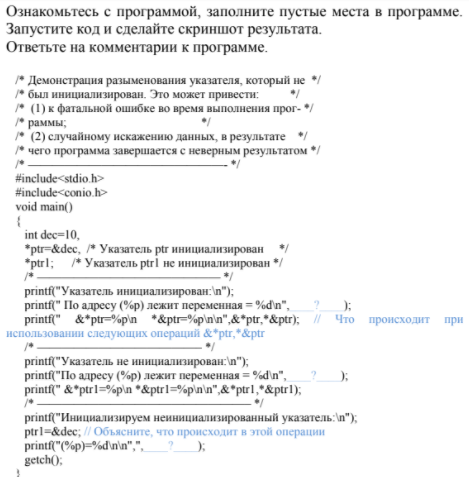
getch();

}

Результат:



Задание 4.



Код:

*/\* Демонстрация разыменования указателя, который не \*/*

*/\* был инициализирован. Это может привести: \*/*

*/\* (1) к фатальной ошибке во время выполнения прог- \*/*

*/\* раммы; \*/*

*/\* (2) случайному искажению данных, в результате \*/*

*/\* чего программа завершается с неверным результатом \*/*

*/\* ————————————————- \*/*

#include<stdio.h>

#include<conio.h>

#include<windows.h>

**void** main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

**int** dec=10,

\*ptr=&dec, */\* Указатель ptr инициализирован \*/*

\*ptr1; */\* Указатель ptr1 не инициализирован \*/*

*/\* ——————————————— \*/*

printf("Указатель инициализирован:**\n**");

printf(" По адресу (%p) лежит переменная = %d**\n**",ptr,\*ptr);

printf(" &\*ptr=%p**\n** \*&ptr=%p**\n\n**",&\*ptr,\*&ptr); *// Что происходит прииспользовании следующих операций &\*ptr,\*&ptr*

*/\* Операция &\*ptr берёт адрес объекта, на который ссылается ptr, т.е &\*ptr == ptr,*

*\*&ptr берёт объект, размещаемый под адресу ptr, т.е. \*&ptr==ptr \*/*

printf("Указатель не инициализирован:**\n**");

printf("По адресу (%p) лежит переменная = %d**\n**",ptr1,\*ptr1);

printf(" &\*ptr1=%p**\n** \*&ptr1=%p**\n\n**",&\*ptr1,\*&ptr1);

*/\* —————————————————– \*/*

printf("Инициализируем неинициализированный указатель:**\n**");

ptr1=&dec; *// Объясните, что происходит в этой операции*

*/\* В указатель записывается адрес переменной dec \*/*

printf("(%p)=%d**\n\n**",ptr1,\*ptr1);

getch();

}

Результат:

