ПРАКТИЧЕСКАЯ РАБОТА № 1. УКАЗАТЕЛИ

Цель работы – познакомиться с адресацией памяти, научиться правильно использовать указатели различных типов.

Операции с указателями

Получение адреса, разыменование, присваивание значения указателя другому указателю того же типа, сложение и вычитание указателя и целого.

Постановка задачи

Наберите текст программы, представленный ниже.

Имена переменных измените на свои.

Запустите программу:

- посмотрите, по каким адресам располагаются переменные, какой объем памяти они занимают и начертите схематичное расположение элементов в памяти;

- отметьте стрелками, куда указывают указатели;

- отметьте, куда стали указывать указатели после изменения их значений, обведите (подчеркните, раскрасьте или как-то еще) адресуемые области для каждого указателя, объясните, как получаются выведенные программой значения;

- отметьте, куда стали указывать указатели p1 и p2 после применения к ним операций инкремента и декремента, обведите (подчеркните, раскрасьте или как-то еще) адресуемые области указателей p1 и p2, объясните, как получаются выведенные программой значения;

- объясните, как получаются выведенные программой значения;

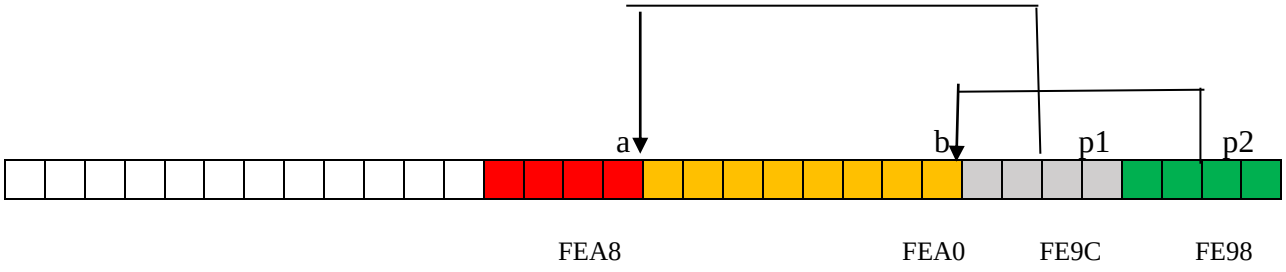
- сформулируйте общие выводы, к которым вы пришли в процессе выполнения задания.

```
#include<stdlib.h>
```

```
#include<stdio.h>
#include <locale.h>
int main()
{
   // Обычные переменные
   int a = 1;
      double b = 3;
   // Указатели
   int *p1 = &a;
      double *p2 = &b;
   void *p4;
   setlocale(LC_ALL, "Russian ");
   // Адреса «обычных» переменных и размер выделяемой памяти
   printf("  a:      int: start address %p extent = %d байт\n",&a, sizeof(a));
   printf("  b:   double: start address %p extent %d\n\n",&b,sizeof(b));
   // 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\n",&p2,sizeof(p2));
    //  Значение указателя и связанное с ним значение
   printf("  p1: %p related  value %d\n",p1,*p1);
      printf("  p2: %p related value %lf\n\n",p2,*p2);
   // Значения обычных переменных
   printf("  a= %d\t \tb= %lf\n",a,b);
   // Изменение значения переменных, используя указатель
   *p1 += 4;
      *p2 = 10;
// Значения обычных переменных после изменения
   printf("  обращение по имени a= %d \t b= %lf\n",a,b);
   printf("  обращение через указатель *p1= %d \t *p2= %lf \n\n",*p1,*p2);
   // Изменение значений указателей
```

```
    p4 = p1;
        p2 = (double*)p1;
        printf(" Значения указателей после их изменения\n”);
        printf("  p1= %p \t p2= %p \t p4= %p \n",p1,p2,p4);
            printf("  *p1= %d \t *p2= %lf \t *(int*) p4= %d \n\n",*p1,*p2,*(int*) p4);
    //     Изменение значений указателей
    p1++;
    p2--;
    printf("  p1= %p \t p2= %p \n",p1,p2);
    printf("  *p1= %d \t *p2= %lf \n",*p1,*p2);
    *p1 = 4;
    p2++;
    printf("p1 =%p\t p2=%p \n",p1,p2);
    printf("*p1=%d \t *p2= %lf \n",*p1,*p2);

    return 0;
}
```



```
  a:      int: start address 0028FEA8 extent = 4
  b:   double: start address 0028FEA0 extent 8

  p1: pointer: start address 0028FE9C extent 4
  p2: pointer: start address 0028FE98 extent 4

  p1: 0028FEA8 related  value 1
  p2: 0028FEA0 related value 3.000000

  a= 1          b= 3.000000
  a= 5   b= 10.000000
  *p1= 5         *p2= 10.000000

  p1= 0028FEA8   p2= 0028FEA8    p4= 0028FEA8
  *p1= 5         *p2= 0.000000   *(int*) p4= 5

  p1= 0028FEAC   p2= 0028FEA0
  *p1= 2686632   *p2= 10.000000
p1 =0028FEAC     p2=0028FEA8
*p1=4    *p2= 0.000000

--------------------------------
```

Контрольные вопросы

1. Что такое указатель?
2. Какой объем памяти занимает указатель?
3. Что является значением переменной-указателя?
4. Как проинициализировать указатель?
5. Что такое NULL?
6. Что такое указатель на void? Зачем нужны такие указатели?
7. Какие операции допустимы при работе с указателями?
8. Чем отличается унарная операция "&" от унарной "*" ?
9. Совместимость типов указателей.
10. Можно ли получить адрес указателя?
11. Почему к указателю на void нельзя применить операцию разыменования?

12. Как работают операции инкремента и декремента, примененные к указателям?

13. Какой спецификатор типа используется при выводе адреса на экран с помощью функции printf()?

14. В чем отличие записи *(double *) a* от *(double) * a*, если *a* – указатель на целое число?

15. В чем отличие записи **a++* от (**a*)++, если *a* – некоторый указатель, отличный от void*?

16. Как описать указатель на начало массива?

17. Как описать указатель на указатель?