ПЗ-05: **Указатели** (задания)

**Задание 1**. Определите объем памяти, выделяемой под указатель на переменные типа char, int, double.

Под сам указатель (там, где хранится адрес) также должна быть выделена память. Объем этой памяти можно узнать с помощью функции **sizeof()**:

int \*pi, i=5;

float \*pf, f=2.1f;

printf("%lu\n", sizeof(pi));

printf("%lu\n", sizeof(pf));

Под указатели всех типов выделяется одинаковый объем памяти, т. к. размер адреса не зависит от типа, а зависит от вычислительной системы. Однако по типу данных определяется, сколько ячеек памяти занимает значение, на которое ссылается указатель, и через сколько ячеек начнется следующее значение.

**Задание 2.** Проанализируйте приведенный ниже код. Объясните результат.

int main()

{

float PI=3.14159,\*p1,\*p2;

p1=p2=&PI;

printf("По адресу p1=%p хранится \*p1=%g\n",p1,\*p1);

printf("По адресу p2=%p хранится \*p2=%g\n",p2,\*p2);

}

Обратите внимание на спецификатор формата **%p**. Он используется для вывода значений указателей в функции printf. Но поскольку указатель – это адрес, т. е. целое положительное число, то в зависимости от ситуации можно использовать любые спецификаторы целочисленных переменных. Для наглядности результатов используйте спецификатор **%x** – для вывода шестнадцатеричном коде.

**Задание 3**.Выполните приведенный ниже код и определите, на сколько увеличится значение, на которое указывает ра. Полученные результаты поясните.

int \*pа, x=5;

pа = &x;

printf("По адресу %p хранится \*ptr=%g\n",pa,\*pa);

pа++;

printf("По адресу %p хранится \*ptr=%g\n",pa,\*pa);

pа--;

printf("По адресу %p хранится \*ptr=%g\n",pa,\*pa);

Измените тип данных на **double** и посмотрите, как изменятся значения (на сколько байт смещается указатель после инкремента\декремента)

Измените тип данных на **char** и сравните результаты с предыдущими (на сколько байт смещается указатель после инкремента\декремента)

Исследуйте изменение указателей типа **long double, short unsigned int**.

Перед выполнением операций обязательно выполните инициализацию указателей через разыменование (**&**) инициализированных переменных того же типа.

**Задание 4.** Исследуйте перемещение по массиву с помощью указателя. Объявите и инициализируйте массив на 10 элементов (используйте разные типы данных: **char, short, int, float**):

float array[10]={1.1, 2.2, 3.3, 4.4 ...};

float \*ptr\_a;

**4.1** Убедитесь, что имя массива в действительности является его адресом, т.е. напечатайте значения **array, &array[0], &array;** с помощью спецификатора **%p**.

**4.2** Установите указатель на начало массива **ptr\_a** и выполните операции, одновременно напечатав значение указателя и то, на что он указывает **(\*ptr\_a)**:

ptr\_a++; ptr\_a+4; ptr\_a-2

**4.3** Напишите цикл прохода по массиву с помощью указателя в прямом порядке с увеличением указателя на х (х = кол-во байт в используемом типе).

**4.4** Выполните проход по массиву в обратном порядке, предварительно установив указатель на конец массива с использованием операции декремента.

**Задание 5.** Запустите и исследуйте программу с использованием указателей:

|  |  |
| --- | --- |
| /\* Листинг программы побайтового вывода целого числа: \*/ | Распределение байтов одного слова в памяти  А) с наименьшего адреса ("big-endian" - обратное размещение байтов или MSB – Most Significant Byte) – байт с наибольшей значащей частью  B) с наибольшего адреса ("little-end" - прямое размещение байтов)) |

Сделайте вывод о порядке размещения байтов в текущей системе.

Усовершенствуйте программу так, чтобы она могла побайтно выводить переменные любого типа.