Лабораторная работа 1
Тема: Изучение принципов работы с указателями.

Цель работы: изучить организацию хранения данных в памяти и работу с указателями.

Указатели это чрезвычайно мощный инструмент в программировании. С помощью указателей некоторые вещи в программировании можно сделать намного проще и при этом эффективность работы вашей программы значительно повысится. Указатели даже позволяют обрабатывать неограниченное количество данных. Например, с помощью указателей можно изменять значения переменных внутри функции, при этом переменные передаются в функцию в качестве параметров. Кроме того, указатели можно использовать для динамического выделения памяти, что означает, что вы можете писать программы, которые могут обрабатывать практически неограниченные объемы данных на лету — вам не нужно знать, когда вы пишете программу, сколько памяти нужно выделить заранее. Пожалуй, это самая мощная функция указателей.

Память компьютера можно представить в виде последовательности пронумерованных однобайтовых ячеек, с которыми можно работать по отдельности или блоками.

Каждая переменная в памяти имеет свой адрес — номер первой ячейки, где она расположена, а также свое значение. Указатель — это тоже переменная, которая размещается в памяти. Она тоже имеет адрес, а ее значение является адресом некоторой другой переменной.

Указатели представляют собой объекты, значением которых служат адреса других объектов (переменных, констант, указателей) или функций.

Указатель — переменная, содержащая адрес объекта. Указатель не несет информации о содержимом объекта, а содержит сведения о том, где размещен объект.

Идея в том, что зная адрес переменной, вы можете пойти по этому адресу и получить данные, хранящиеся в нем. Если вам нужно передать огромный кусок данных в функцию, намного проще передать адрес в памяти, по которому хранятся эти данные, чем скопировать каждый элемент данных.

Синтаксис указателей

Если у нас есть указатель, значит мы можем получить его адрес в памяти и данные на которые он ссылается, по этой причине указатели имеют несколько необычный синтаксис, отличающийся от объявления простых переменных. Более того, поскольку указатели — это не обычные переменные, то, необходимо сообщить компилятору, что переменная является указателем и сообщить компилятору тип данных, на которые ссылается указатель.

Итак, указатель объявляется следующим образом:

```
data_type *pointerName;
```

где data_type — тип данных, pointerName — имя указателя.

Например, объявим указатель, который хранит адрес ячейки памяти, в которой лежит целое число:

```
int *integerPointer;
```

Работа с указателями

Последовательность действий при работе с указателем включает 3 шага:

1. Определение указуемых переменных и переменной-указателя. Для переменной-указателя это делается особым образом.

```
int  a,x;  // Обычные целые переменнные
int *p;  // указатель на другую  целую переменную
```

В определении указателя присутствует та же самая операция косвенного обращения по указателю. В соответствии с принципами контекстного определения типа переменной эту фразу следует понимать так: переменная *p* при косвенном обращении к ней дает переменную типа int. То есть свойство ее – быть указателем, определяется в контексте возможного применения к ней операции *. Обратите внимание, что в определении присутствует указуемый тип данных. Это значит, что указатель может ссылаться не на любые переменные, а только на переменные заданного типа, то есть указатель в Си типизирован.

2. Связывание указателя с указуемой переменной. Значением указателя является адрес другой переменной. Следующим шагом указатель должен быть настроен, или назначен на переменную, на которую он будет ссылаться.

```
p = &a; // Указатель содержит адрес переменной a
```

Операция & понимается буквально как адрес переменной, стоящей справа от нее. В более широкой интерпретации она «превращает» объект в указатель на него (или производит переход от объекта к указателю на него) и является в этом смысле прямой противоположностью операции *, которая «превращает» указатель в указуемый объект. То же самое касается типов данных. Если переменная a имеет тип int, то выражение &a имеет тип – указатель на int или int*.

3. И наконец, в любом выражении косвенное обращение по указателю интерпретируется как переход от него к указуемой переменной с выполнением над ней всех далее перечисленных в выражении операций.

```
*p=100;              // Эквивалентно a=100
x = x + *p;          // Эквивалентно x=x+a
```

Чтение указателя

Для вывода значения указателя можно использовать специальный спецификатор %p.

Пример 1: вывод значения указателя

```
#include <stdio.h>

int main(void)
{
    int var = 10;
    int *ptrVar;
    ptrVar = &var;
    printf("Address = %p \n", ptrVar);
}
```

Пример 2:

```
#include <stdio.h>
int main()
{
  int a, *b;
  a = 134;
  b = &a;

  printf("\n Значение переменной a равно %d = %x шестн.", a,a);
  printf("\n Адрес переменной a равен %x шестн.", &a);
  printf("\n Данные по адресу указателя b равны %d = %x шестн.", *b,*b);
  printf("\n Значение указателя b равно %x шестн.", b);
  printf("\n Адрес расположения указателя b равен %x шестн.", &b);
}
```

Используя полученное значение в результате операции разыменования мы можем присвоить его другой переменной:

```
int x = 10;
int *p  = &x;
int y = *p;
printf("x = %d \n", y);
```

И также используя указатель, мы можем менять значение по адресу, который хранится в указателе:

```
int x = 10;
int *p = &x;
```

```
*p = 45;
printf("x = %d \n", x);  // 45
```

Задание 1.

- Объявить переменные типа char, short, int, float, double. Инициализировать различными значениями.
- Объявить указатели на данные переменные
- Вывести на экран значения адресов указателей, а также значения, хранящиеся по этим адресам.

Размер указателя

Когда программист разыменовывает указатель, к примеру, на int, то компилятор знает что этот адрес указывает на данные int и при разыменовании (доступу к этим данным) нужно взять 4 байта, если же это, к примеру double, то соответственно это 8 байт.

```
#include <conio.h>
#include <stdio.h>

void main() {
    int A = 100;
    int *a = &A;
    double B = 2.3;
    double *b = &B;

    printf("%d\n", sizeof(A));
    printf("%d\n", sizeof(a));
    printf("%d\n", sizeof(B));
    printf("%d\n", sizeof(b));
}
```

Несмотря на то, что переменные имеют разный тип и размер, указатели на них имеют один размер. Действительно, если указатели хранят адреса, то они должны быть целочисленного типа. Так и есть, указатель сам по себе хранится в переменной типа size_t), это тип, который ведёт себя как целочисленный, однако его размер зависит от разрядности системы.

Задание 2.

- Объявить переменные типа char, short, int, float, double. Инициализировать различными значениями.
- Объявить указатели на данные переменные
- Вывести на экран размеры переменных, размеры указателей, а также отдельно размер типа `size_t`.

- Сделать вывод по работе программы

Пустые ссылки

Обратите внимание, что сначала указатель инициализируется. Это нужно для того, чтобы указатель ссылался на определенный адрес памяти. Если бы мы начали использовать указатель не инициализировав его, он бы ссылался на какой угодно участок памяти. И это могло бы привести к крайне неприятным последствиям. Например, операционная система, вероятно, помешает вашей программе получить доступ к неизвестному участку памяти, так как ОС знает, что в вашей программе не выполняется инициализация указателя. В основном это просто приводит к краху программы.

Задание 3.

- Объявить переменную любого типа
- Объявить указатель на данную переменную
- Вывести на экран значения переменой и адрес указателя
- Присвоить указателю произвольный адрес памяти
- Вывести на экран значение адреса, хранящегося в указателе и значение, хранящее по данному адресу.
- Сделать вывод по работе программе

Арифметика указателей

Так как указатель хранит адрес, можно изменить его до другого типа. Это может понадобиться, например, если мы хотим взять часть переменной, или если мы знаем, что переменная хранит нужный нам тип.

Пример:

```
#include <stdio.h>

int main() {
    int A = 28453;
    int *intPtr;
    char *charPtr;

    intPtr = &A;
    printf("%d\n", *intPtr);
    printf("--------------------\n");
    charPtr = (char*)intPtr;
    printf("%d ", *charPtr);
    charPtr++;
    printf("%d ", *charPtr);
    charPtr++;
    printf("%d ", *charPtr);
```

```
    charPtr++;
    printf("%d ", *charPtr);
}
```

В этом примере мы пользуемся тем, что размер типа `int` равен 4 байта, а `char` 1 байт. За счёт этого, получив адрес первого байта, можно пройти по остальным байтам числа и вывести их содержимое.

Задание 4.

- Объявить переменную типа int. Инициализировать ее значением
- Объявить указатель на данную переменную.
- Объявить указатель на однобайтовый тип данных
- Вывести на экран значения отдельных байт переменной типа int.
- Привести двоичное значение исходной переменной типа int, а а также двоичные представления отдельных ее байт.