**Сопроводительный лист на издание в авторской редакции**

Название работы Работа с указателями Методические указания к выполнению лабораторной работы № 18 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем»

Актуальность и соответствующий   
научно-методический уровень подтверждаю \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись научного редактора)

Рукопись сверена и проверена автором \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись автора)

Рекомендуется к изданию \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись заведующего кафедрой)



**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Брянский государственный технический университет

Утверждаю

Ректор университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Н. Федонин

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019

ПРОГРАММИРОВАНИЕ

РАБОТА С УКАЗАТЕЛЯМИ

**Методические указания**

**к выполнению лабораторной работы № 18**

**для студентов очной формы обучения**

**по направлениям подготовки**

**09.03.04 – «Программная инженерия»,**

**09.03.01 «Информатика и вычислительная техника»,**

**02.03.03 «Математическое обеспечение и администрирование информационных систем»**

**Брянск 2019**

**УДК 004.**

Работа с указателями [Текст] + [Электронный ресурс]: методические указания к выполнению лабораторной работы № 18 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем».– Брянск, 2019. – 7 с.

Разработали:

Д. И. Булатицкий

канд. техн. наук, доц.,

Е..В. Коптенок

ассистент

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ

**Методические издания публикуются в авторской редакции**

# Цель работы

Цель работы – научиться указатели для решения задач.

Продолжительность работы – 2 часа.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

**Указатель** — переменная, содержащая адрес объекта. Указатель не несет информации о содержимом объекта, а содержит сведения о том, где размещен объект. Указатели широко используются в программировании на языке Си.

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 ячейки, где она расположена, а также свое значение. Указатель — это тоже переменная, которая размещается в памяти.

Указатель, как и любая переменная, должен быть объявлен.

Общая форма объявления указателя

тип \*имя\_объекта;

**Тип указателя**— это тип переменной, адрес которой он содержит.

При описании указателей в качестве имени типа данных можно использовать ключевое слово void, например

void \*vd;

При таком описании с указателем не связывается никакой тип данных, то есть получаем указатель на данное произвольного типа.

Для указателей одного и того же типа допустимой является операция присваивания, кроме того указателю типа void может быть присвоено значение адреса данного любого типа, но не наоборот, например

int \*a, \*b;

double \*d;

void \*v;

...

a = b; /\* Правильно \*/

v = a; /\* Правильно \*/

v = d; /\* Правильно \*/

b = v; /\* Неправильно \*/

d = a; /\* Неправильно \*/

В случае неправильного присваивания указателей компиляторы обычно выдают предупреждающие сообщения, которыми никогда не следует пренебрегать. Например, компилятор фирмы Borland выдает сообщение:

"Suspicious pointer conversion", которое переводится как "Подозрительное преобразование указателей".

Если по какой-либо причине необходимо выполнить операцию присваивания между указателями разного типа, то следует использовать явное преобразование типов, например для указателей из предыдущего примера

b = (int \*) v;

d = (double \*) a;

## Арифметика указателей

Для работы с указателями в Си определены две операции:

* операция \* (звездочка) — позволяет получить значение объекта по его адресу - определяет значение переменной, которое содержится по адресу, содержащемуся в указателе;
* операция & (амперсанд) — позволяет определить адрес переменной.

Например,

сhar c; // переменная

char \*p; // указатель

p = &c; // p = адрес c

Для указанного примера обращение к одним и тем же значениям переменной и адреса представлено в Таблице1.

Указателю нужен тип для того, чтобы корректно работала операция **разыменования** (получения содержимого по адресу). Если указатель хранит адрес переменной, необходимо знать, сколько байт нужно взять, начиная от этого адреса, чтобы получить всю переменную.

Таблица 1. Обращение к значениям переменной и адреса

|  |  |  |
| --- | --- | --- |
|  | Переменная | Указатель |
| Адрес | &c | p |
| Значение | c | \*p |

Указатели поддерживают арифметические операции. Для их выполнения необходимо знать размер.

**операция + N сдвигает указатель вперёд на N\*sizeof(тип) байт.**

Например, если указатель int \*p; хранит адрес CC02, то после p += 10; он будет хранить адрес СС02 + sizeof(int)\*10 = CC02 + 28 = CC2A (Все операции выполняются в шестнадцатиричном формате). Пусть мы создали указатель на начало массива. После этого мы можем "двигаться" по этому массиву, получая доступ до отдельных элементов.

#include <conio.h>

#include <stdio.h>

void main() {

int A[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int \*p;

p = A;

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

p++;

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

p = p + 4;

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

getch();

}

Массив, по сути, сам является указателем, поэтому не нужно использовать оператор &. Мы можем переписать пример по-другому

p = &A[0];//Получить адрес первого элемента и относительно него двигаться по

массиву.

Кроме операторов + и - указатели поддерживают операции сравнения. Если у нас есть два указателя a и b, то a > b, если адрес, который хранит a, больше адреса, который хранит b.

void main() {

int A[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int \*a, \*b;

a = &A[0];

b = &A[9];

printf("&A[0] == %p\n", a);

printf("&A[9] == %p\n", b);

if (a < b) {

printf("a < b");

} else {

printf("b > a");

}

getch();

}

Если же указатели равны, то они указывают на одну и ту же область памяти.

# ХОД РАБОТЫ

**Задание 1**. Создайте проект, содержащий листинг, представленный на Рис.3.1. Запустите программу.

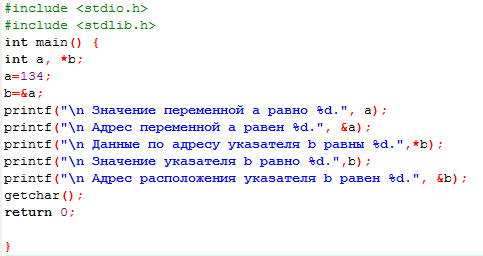


Рис.3.1. Листинг программы к Заданию 1.

Обратите внимание на полученные результаты. Объясните их. Почему адрес расположения указателя b и значение указателя b не равны?

**Задание 2.** Создайте проект, содержащий листинг, представленный на Рис.3.2. Запустите программу.

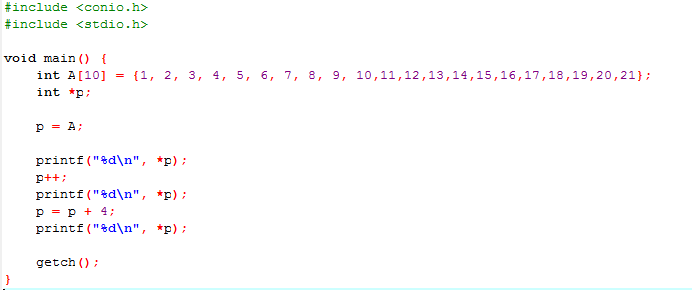


Рис.3.2. Листинг программы к Заданию 2.

Измените код таким образом, чтобы на экран выводилось сначала значение первого элемента массива, а затем значения 1+N и 1+(2N%21) элементов, где N – номер варианта.

**Задание 3.** Поменять переменные местами. На основе объяснений, представленных ниже, требуется составить программу обмена местами двух переменных.

Обычно функция возвращает одно значение. А как вернуть больше одного? Рассмотрим код функции, которая меняет местами две переменные, представлена на Рис.3.3.

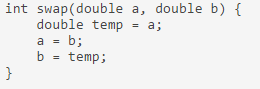


Рис.3.3. Листинг функции

Пусть есть переменные x и y с некоторыми значениями. Если выполнить функцию, передав в нее x и y, окажется, что никакого обмена не произошло. И это правильно.

При вызове этой функции в стеке будут сохранены значения x и y. Далее a и b получат значения x и y. Будет выполнена перестановка. Затем функция завершится и значения x и y будут восстановлены из стека. Все по-честному.

Чтобы заставить функцию работать так, как нужно, следует передавать в нее не значения переменных x и y, а их адреса. Но и саму функцию тогда нужно адаптировать для работы с адресами (Рис.3.4).

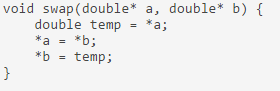


Рис.3.4. Листинг функции, меняющей местами значения переменных.

Не стоить забывать о том, что и вызов функции теперь должен выглядеть иначе (Рис.3.5.).



Рис.3.5. Листинг вызова функции.

Теперь в функцию передаются адреса. И работа ведется относительно переданных адресов.

Если функция должна вернуть несколько значений, необходимо передавать в нее адреса.

**Если функция должна менять значение переменной, нужно передавать ей адрес этой переменной.**

**Задание 4**.

Создайте проект, содержащий листинг, представленный на Рис.3.6. Запустите программу. Программа должна вывести на экран значение переменной х.

Определите, какие ошибки были допущены в коде и исправьте их.

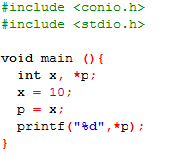


Рис.3.6. Листинг программы к Заданию 4.

**Задание 5.** Создайте проект, содержащий листинг, представленный на Рис.3.7.. Запустите программу. Программа должна вывести на экран числа от одного до десяти

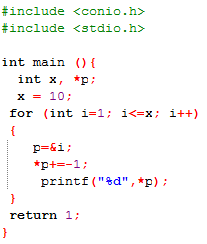


Рис.3.7. Листинг программы к Заданию 5.

Определите, какие ошибки были допущены в коде и исправьте их.

# ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

**Задание 1.** Написать программу на языке СИ для решения задачи согласно варианту. Для решения задачи использовать указатели.

1. Определить максимальное и минимальное значения из двух различных вещественных чисел.
2. Поменять местами значения трех переменных.
3. Найти сумму и произведение всех чисел от А до В.
4. Даны координаты трех вершин треугольника. Найти длины всех его сторон и площадь, если треугольник существует.
5. Найти сумму и произведение цифр трех чисел.
6. Найти наибольшую и наименьшую цифру в записи числа
7. Определить корни квадратного уравнения.
8. Решить систему из двух линейных уравнений (частные случаи можно не рассматривать).
9. Три введенных числа увеличить в А раз и прибавить В.
10. Упорядочить три числа по возрастанию их модулей.

# КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое цикл?
2. Для чего применяются циклы.
3. Что такое вложенный цикл?

# Список рекомендуемой литературы

## Основная литература

1. Павловская Т.А. C#. Программирование на языке высокого уровня:учеб. Для вузов.-м. [и др.]:Питер,2014.-432
2. Давыдова Н.А.,боровская Е.В. Программирование: учеб. Пособие. М.:Бином. Лаб. Знаний,2011.-238 с.
3. Булатицкий Д.И. Объектно-ориентированное программирование: учеб. пособие.- Брянск: изд-во БГТУ,2008.-120 с.

## Дополнительная литература

1. Тюльпинова Н.В. Алгоритмизация и программирование: учеб. Пособие/Брян. Гос. Техн. Ун-т.- Брянск:изд-во БГТУ,2012.-199 с.-[+ электронная копия].
2. Иванова Г.С. Программирование: основы алгоритмизации и процедурное программирование, объектно ориентированное программирование: учеб. Для вузов.-2-е изд., стер.-м.:Кнорус, 2014.-425 с.

Работа с указателями [Текст] + [Электронный ресурс]: методические указания к выполнению лабораторной работы № 18 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем». – Брянск, 2019. – 7 с.

БУЛАТИЦКИЙ ДМИТРИЙ ИВАНОВИЧ

КОПТЕНОК ЕЛИЗАВЕТА ВИКТОРОВНА

Научный редактор В.Я. Израилев

Компьютерный набор Е.В. Коптенок

Иллюстрации Е.В. Коптенок

Подписано в печать 20.02.2017. Формат 60х84 1/15 Бумага офсетная. Офсетная печать. Усл.печ.л. \_\_\_ Уч.-изд.л. \_\_\_ Тираж 1 экз. Бесплатно

Брянский государственный технический университет

Кафедра «Информатика и программное обеспечение», тел. 56-09-84

241035, Брянск, бульвар 50 лет Октября, 7 БГТУ, тел. 58-82-49