МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
имени академика С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Массивы и строки

Электронные методические указания
к лабораторной работе № 1

Самара
2011

Составители: МЯСНИКОВ Евгений Валерьевич
ПОПОВ Артем Борисович

В лабораторной работе № 1 по дисциплине "Языки и методы программирования" изучаются принципы работы с массивами и строками на языке C++. Приводятся краткие теоретические и справочные сведения, необходимые для выполнения лабораторных работ. Дан пример выполнения лабораторной работы.

Методические указания предназначены для студентов факультета информатики, направление 010400 – Прикладная математика и информатика, бакалавриат (010400.62)/магистратура (010400.68, магистерская программа – Технологии параллельного программирования и суперкомпьютинг).

Содержание

Цель работы: Приобретение базовых навыков программирования на языке C++. Изучение основных принципов работы с массивами и строками. Знакомство с функциями ввода/вывода и функциями работы со строками стандартной библиотеки языка.

1 Теоретические основы лабораторной работы

Массив - это упорядоченное множество располагающихся в смежных областях памяти однотипных элементов, каждый из которых характеризуется индексом (или набором индексов). Использование массивов позволяет организовать хранение и обработку однотипных данных сравнительно большого объема в оперативной памяти.

Обычно изучение массивов начинают с одномерных массивов (векторов), а затем переходят к многомерным, в частности, двумерным массивам (матрицам).

1.1 Одномерные массивы

Объявление одномерного массива в языке C++ выполняется следующим образом:

```
<тип> <имя> [<размер>] [ = <инициализатор>];
```

Здесь *размер* - значение (или константное выражение), задающее количество элементов в одномерном массиве, *<инициализатор>* - заключенный в фигурные скобки список значений элементов массива, перечисленных через запятую.

При определении массива обязательно должен быть задан либо размер массива, либо инициализатор. Если указано и то и другое, то количество элементов в списке инициализации не должно превышать объявленный размер. Оставшиеся элементы для базовых типов инициализируются нулевыми значениями, а для объектов - с использованием конструктора по умолчанию. В том случае, если инициализатор не указан вообще, инициализация массива будет выполнена, только если массив является статическим или внешним.

Если размер массива не указан, то он считается равным количеству элементов в списке инициализации. Ниже приводится несколько корректных объявлений одномерных массивов.

Объявление массива из 3 элементов с явной инициализацией всех элементов:

```
char ar1[3] = {‘A’, ‘B’, 0};
```

Объявление массива из 3 элементов с явной инициализацией первого элемента, остальные элементы инициализируются нулями:

```
char ar2[3] = {‘A’};
```

Объявление массива из 5 элементов без инициализации:

```
int ar3[5];
```

Объявление массива из 5 элементов в соответствии с инициализатором:

```
int ar4[] = {10, 200, 3, 404, 55};
```

Доступ к элементам массива осуществляется с использованием операции «[]». Внутри квадратных скобок указывается индекс элемента, к которому необходимо получить доступ. При этом следует учитывать, что элементы массивов в C++ **всегда номеруются с нуля**. Таким образом, если в массиве N элементов, элементы массива имеют индексы с нулевого по (N-1). Ниже приводится пример заполнения элементов целочисленного массива квадратами индексов элементов.

```
int ar[10];
for (int i = 0; i<10; i++)  ar[i] = i*i;
```

На практике при работе с массивами часто нужно знать, сколько элементов содержится в массиве. Для этого применяется способ, основанный на использовании операции **sizeof**, возвращающей размер памяти, отведенной под хранение ее аргумента в байтах. Очевидно, во всех приведенных выше примерах, для хранения массива резервируется непрерывный участок памяти, размером достаточным для хранения всех его элементов, и этот размер равен sizeof(<имя массива>). С другой стороны, узнать размер, отводимый под хранение одного элемента можно, узнав размер первого элемента: sizeof(<имя массива>[0]). Разделив размер всего массива на размер первого элемента, мы получим количество элементов в массиве. С использованием операции sizeof приведенный выше пример заполнения массива мог бы выглядеть так:

```
for (int i = 0; i<sizeof(ar)/sizeof(ar[0]); i++)
    ar[i] = i*i;
```

Рассмотрим, как же происходит обращение к конкретному элементу массива. Пусть объявлен некоторый массив: T a[N]. Так как элементы в массиве располагаются последовательно, начиная с нулевого элемента, то доступ к i-му элементу можно представить в виде *(p+i). Здесь p – типизированный указатель на первый элемент массива: T *p=&a[0]. Таким образом, приведенный выше код мог быть записан с использованием указателей в следующем виде:

```
int *p = &ar[0];
for (int i = 0; i<sizeof(ar)/sizeof(ar[0]); i++)
    *(p+i) = i*i;
```

В C++ имя массива может использоваться в качестве указателя на его первый элемент (ar ==&ar[0]). Другими словами, везде, где необходим указатель на первый элемент, можно использовать имя массива. С учетом сказанного рассмотренный выше фрагмент примет вид:

```
for (int i = 0; i<sizeof(ar)/sizeof(ar[0]); i++)
    *(ar+i) = i*i;
```

Однако из того, что имя массива можно использовать как указатель, вовсе не следует, что всегда можно делать наоборот. Так, описанная выше операция sizeof будет возвращать размер указателя (4 байта для 32 разрядной архитектуры) независимо от того, на что он указывает. Поэтому, например, определить размер массива с использованием указателей не удастся. В остальных же случаях указатель может использоваться как имя некоторого массива, состоящего из элементов того типа, с которым связан указатель. В частности, к указателям применима операция обращения по индексу «[]».

1.2 Строки

Строка - это последовательность символов, завершающаяся символом с нулевым кодом. В C++ существует два базовых типа символов: char и wchar_t, но мы будем рассматривать только первый из них. Объявление строки может выглядеть как объявление массива символов:

```
char str[]={‘A’,’B’,’C’,0};
```

Здесь мы не только объявили массив, но и инициализировали его так, что он содержит строку из трех символов и завершающий ноль. Инициализировать массив символов можно с использованием строковой константы:

```
char str[] = “ABC”;
```

Несмотря на то, что в константе указано только три символа, завершающий ноль будет добавлен автоматически. Можно легко проверить это, вызвав операцию sizeof(“ABC”), которая вернет 4.

Строковую константу можно присвоить указателю:

```
char *psz = “ABC”;
```

Хотя последние два выражения на первый взгляд кажутся идентичными, нужно ясно представлять себе разницу между ними. В первом случае происходит объявление массива и копирование в него константы. Во втором - объявляется указатель, и в него записывается адрес константы. Существенным моментом здесь является то, что изменение символа строки в первом случае (str[0]=’X’) будет успешным, а во втором случае (*psz=’X’) может привести к ошибке, связанной с попыткой присваивания константе. Чтобы обезопасить себя от таких ошибок объявляйте указатели на строковые константы с модификатором const:

```
const char *psz = “ABC”;
```

Следует отметить, что при работе со строками указатели используются повсеместно. Например, приведем фрагмент кода, вычисляющий длину строки:

```
const char *pcsz = “Example”;
const char *p = pcsz;
while (*p) p++;
int len = p-pcsz;
```

Естественно, стандартная библиотека языка включает в себя целый ряд функций работы со строками (заголовочный файл <string.h>), и Вы, наверняка, найдете в ней все, что Вам нужно. Некоторые из функций стандартной библиотеки приведены в приложении 1.

1.3 Многомерные массивы

Многомерные массивы на практике используются очень часто. Многомерный массив фиксированного размера в C++ объявляется подобно одномерному с той лишь разницей, что после имени массива указываются не одни квадратные скобки с размером массива, а последовательность таких скобок:

```
<тип> <имя> [<размер>]⟦[<размер>]…⟧ ⟦ = <инициализатор>⟧;
```

Здесь каждая размерность массива по-прежнему задается константным выражением, а вот формат инициализатора меняется. В инициализаторе многомерного массива могут быть использованы вложенные фигурные скобки, например, так, как это показано в следующих примерах:

```
int ar[3][4]={{11,12,13,14},{21,22,23,24},{31,32,33,34}};
int ar[2][2][3]={ { {111,112,113},{121,122,123} },
                  { {211,212,213},{221,222,223} } };
```

Однако, учитывая, что многомерные массивы хранятся как непрерывная последовательность элементов, инициализация приведенных выше массивов может быть выполнена следующим образом:

```
int ar[3][4]={11,12,13,14,21,22,23,24,31,32,33,34};
int ar[2][2][3]=
    {111,112,113,121,122,123,211,212,213,221,222,223};
```

2 Пример выполнения лабораторной работы

Задание 1. Написать программу, которая вводит с клавиатуры массив из 10 целых чисел, считает среднее арифметическое элементов массива и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 20 символов, считает длину введенной строки и выводит ее пользователю.

Решение.

```
int main(){
        printf("Задание 1. \nввести массив из 10 целых чисел, посчитать и вывести среднее арифметическое элементов; \nввести с клавиатуры строку длиной не более 20 символов, посчитать и вывести ее длину");
    int A[10];
    printf("Введите элементы массива");
    for (int i=0; i<10; i++){
        printf("A[%d] = ", i);
        scanf("%d", &A[i]);
    }
    double sum = 0;
    for (int i=0; i<10; i++){
        sum = sum + A[i];
    }
    printf("Среднее арифметическое равно %lf \n", sum/10);
    char str[21];
    printf("Введите строку (не более 20 символов)\n");
    scanf("%s", str);
    int length;
    for (length=0; (length<20)&&(str[length]!=0); length++);
    printf("Длина строки %s равна %d\n", str, length);
    printf("Для выхода нажмите любую клавишу");
    char c;
    scanf("%c", &c);//считываем <Enter> от предыдущих вводов
    scanf("%c", &c);//собственно считывание символа
    return 0;
}
```

3 Содержание отчета

Отчет по лабораторной работе должен содержать:

1. Титульный лист.

2. Задание на лабораторную работу.

3. Описание основных алгоритмов и структур данных, используемых в программе.

4. Описание интерфейса пользователя программы.

5. Контрольный пример и результаты тестирования.

6. Листинг программы.

4 Контрольные вопросы

1. Каким образом можно объявить и инициализировать одномерный массив?

2. Как связано имя массива с указателем на первый элемент. Какие между ними различия?

3. Что представляет собой строка?

4. Какие способы выделения и освобождения динамической памяти Вы знаете?

5. Что представляют собой многомерные массивы в C++? Как объявить и проинициализировать такой массив? Приведите пример для массива 2x3.

6. Какими способами можно выделить многомерный динамический массив? В чем отличие этих способов?

5 Задания на лабораторную работу

5.1 Начальный уровень сложности

Общие требования: в начале программы вывести задание, в процессе работы выводить подсказки пользователю (что ему нужно ввести, чтобы продолжить выполнение программы).

Варианты заданий:

1. Написать программу, которая вводит с клавиатуры массив из 10 целых чисел, считает **среднее арифметическое** элементов массива и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 20 символов, считает **длину** введенной строки и выводит ее пользователю.
2. Написать программу, которая вводит с клавиатуры массив из 10 целых чисел, считает **количество положительных** элементов массива и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 20 символов, считает количество символов «**а**» и выводит результат пользователю.
3. Написать программу, которая вводит с клавиатуры массив из 10 целых чисел, находит среди элементов массива **минимальный** и выводит результат пользователю; затем вводит с клавиатуры две строки длиной не более 20 символов, сравнивает их на **равенство** и выводит результат пользователю.
4. Написать программу, которая вводит с клавиатуры массив из 10 целых чисел, находит среди элементов массива **индекс максимального** элемента и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 20 символов, проверяет, **является ли строка перевёртышем**, и выводит результат пользователю.
5. Написать программу, которая вводит с клавиатуры массив из 10 целых чисел, **заменяет отрицательные** элементы массива **на нули** и выводит результирующий массив пользователю; затем вводит с клавиатуры строку длиной не более 20 символов, **меняет порядок символов на обратный** и выводит результат пользователю.
6. Написать программу, которая вводит с клавиатуры массив из 10 целых чисел, считает **количество четных** элементов массива и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 20 символов, **копирует введённую строку** в другую переменную и выводит результат пользователю.
7. Написать программу, которая вводит с клавиатуры массив из 10 целых чисел, считает **сумму** элементов массива стоящих **в чётных позициях** и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 20 символов, проверяет, **начинается ли строка** с «abba» и выводит результат пользователю.

5.2 Средний уровень сложности

Общие требования: в начале программы вывести задание, в процессе работы выводить подсказки пользователю (что ему нужно ввести, чтобы продолжить выполнение программы). Предусмотреть возможность повторного прогона алгоритмов без перезапуска программы (хотите начать заново (y/n)).

Варианты заданий:

8. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, находит среди элементов массива **индекс первого минимального и первого максимального** элемента и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 100 символов, считает **количество слов** (слова разделяются одним или несколькими пробелами) и выводит результат пользователю.

9. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, находит среди элементов массива **минимальный и максимальный** элементы **и их количество** и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, еще одну строку не более чем 10 символов и номер позиции, начиная с которой **ищет первую позицию вхождения второй строки в первую**, и выводит эту позицию результат пользователю.

10. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, **упорядочивает** элементы массива по возрастанию и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, еще одну строку не более 10 символов и **распечатывает позиции вхождения второй строки в первую** и их общее количество.

11. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, **упорядочивает** элементы массива по убыванию и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, считает **количество цифр в строке** и выводит результат пользователю.

12. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, **удаляет** из массива **нулевые** элементы, сдвигая при этом оставшиеся влево, и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, считает **количество символов латинского алфавита** и выводит результат пользователю.

13. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, считает **количество непрерывно возрастающих серий** элементов массива и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, находит **последнее слово в строке** и выводит результат пользователю.

14. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, в новый массив **скопировать самую длинную непрерывно убывающую серию** элементов массива и вывести результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, еще одну строку не более 10 символов, ищет **позицию последнего любого символа из второй строки в первой** и выводит результат пользователю.

15. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, преобразовывает массив так, чтобы **сначала шли чётные, затем нечётные** элементы массива (при этом не меняя порядок следования среди чётных и нечетных элементов) и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов и распечатывает **все слова длинной более** четырёх символов.

16. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел преобразовывает массив так, чтобы **сначала шли положительные, затем отрицательные, затем нулевые** элементы массива (при этом, не меняя порядок следования среди положительных и отрицательных элементов) и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, вводит еще два символа, затем **заменят все вхождения первого символа на второй,** и выводит позиции замен и получившуюся строку.

17. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, создает **два новых массива**, в один из которых помещает **положительные** элементы массива, а в другой **отрицательные** и нулевые элементы, и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, ищет в ней **самое короткое слово** и выводит его пользователю.

18. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, находит элемент массива, **встречающийся наибольшее число раз**, выводит этот элемент и его количество пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, ищет в ней **самое длинное слово** и выводит его и выводит результат пользователю.

19. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, создает **новый** массив, в который помещает **элементы исходного массива, игнорируя при этом повторяющиеся**, и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 6 символов, проверяет, что **строка содержит число**, выводит пользователю **сумму цифр** и сообщает, является ли эта сумма нечетным числом.

20. Написать программу, которая вводит с клавиатуры два массива из N вещественных чисел, создает новый массив размером 2N, в который помещает элементы первого и второго массивов в следующем порядке: **1й из первого, 1й из второго, 2й из первого, 2й из второго** и т.д., и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 6

символов, проверяет, что строка содержит число, и выводит пользователю цифры числа прописью (например, **ввод '352', вывод – 'три пять два'**).

5.3 Высокий уровень сложности

Общие требования: в начале программы вывести задание, в процессе работы выводить подсказки пользователю (что ему нужно ввести, чтобы продолжить выполнение программы). Предусмотреть возможность повторного прогона алгоритмов без перезапуска программы («хотите начать заново (y/n)?»). Основные алгоритмы выполнить в виде отдельных функций с необходимыми параметрами.

Варианты заданий:

21. Написать программу, которая вводит с клавиатуры массив из NxM вещественных чисел, **транспонирует его, находит среднеквадратическое отклонение** элементов массива и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, **упорядочивает в строке слова по возрастанию длины** и выводит результат пользователю.

22. Написать программу, которая вводит с клавиатуры массив из NxM вещественных чисел, изменяет массив так, чтобы в строках **остались элементы, которые встречаются более одного раза**, остальные заменяет нулём и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, **исключает повторяющиеся слова**, выводит получившуюся строку, а также повторяющиеся слова и их количество.

23. Написать программу, которая вводит с клавиатуры массив из NxM вещественных чисел, ищет **седловые точки** в массиве и выводит их и их позиции пользователю; затем вводит с клавиатуры число, преобразовывает его **в строку в шестнадцатеричном** представлении и выводит результат пользователю.

24. Написать программу, которая вводит с клавиатуры массив из NxM вещественных чисел, **упорядочивает элементы массива** по возрастанию по строкам и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 8 символов, проверяет, является ли строка **шестнадцатеричным** числом, преобразовывает **в десятеричное число** и выводит результат пользователю.

25. Написать программу, которая вводит с клавиатуры массив из N вещественных чисел, **находит элемент массива, «расстояние»** от которого до других элементов **минимально** и выводит результат пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, **упорядочивает в строке слова по уменьшению длины** и выводит результат пользователю.

26. Написать программу, которая вводит с клавиатуры массив из NxM вещественных чисел, в каждой **строке** ищется **минимальный** элемент, **затем** среди всех найденных ищется **максимальный** и выводит его и его **позицию** пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, находит **слово** диной более четырёх символов, в котором **буква *а* встречается реже** и выводит результат пользователю.

27. Написать программу, которая вводит с клавиатуры массив из NxM вещественных чисел, **умножает каждый элемент первой строки на a[1,1]** (в том числе и элемент a[1,1]) а каждый элемент второй строки на a[2,2] и т.д., выводит получившуюся матрицу пользователю; затем вводит с клавиатуры строку длиной не более 200 символов, **находит все палиндромы** (т.е. слова-перевертыши) и выводит их пользователю.

28. Написать программу, которая вводит с клавиатуры массив из NxN вещественных чисел, вычисляет **определитель матрицы** и выводит результат пользователю; затем вводит с клавиатуры три строки длиной не более 200 символов, **заменяет в первой строке все вхождения второй строки на третью** и выводит результат пользователю.

6 Библиографический список

1. Страуструп Б. Язык программирования C++. Специальное издание. М.: Радио и связь, 1991. - 349с.

2. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. - М.: Мир, 1989.

3. Язык программирования Си / Б. Керниган, Д. Ритчи. - 3-е изд., испр. - СПб. : Невский Диалект, 2004. - 351 с.

4. Практикум по программированию на языке СИ / В. В. Подбельский. - М. : Финансы и статистика, 2004

7 Справочная информация

7.1 Некоторые функции работы со строками (заголовочный файл <string.h>)

Функция	Назначение
size_t strlen(const char *str);	Вычисление длины строки
char *strcpy(char *strDestination, const char *strSource);	Копирование строк
int strcmp(const char *string1, const char *string2);	Сравнивание строк
char *strcat(char *strDestination, const char *strSource);	Конкатенация строк
char *strchr(char * str, int c)	Поиск символа в строке
char *strstr(const char *str, const char *strSearch);	Поиск подстроки в строке
char *strtok(char *strToken, const char *strDelimit);	Разбиение строки на лексемы
char *strdup(const char *s); (может не входить в стандартную библиотеку)	Создание копии строки в динамической памяти

7.2 Функции ввода вывода

Функция вывода printf

int printf(const char *format, [, argument...]); – функция форматирует и печатает наборы символов и значений в выходной стандартный поток stdout. Строка формата состоит из обычных символов, escape-последовательностей и, если за строкой формата следуют аргументы, еще и спецификации формата. Обычные символы и escape-последовательности просто копируются в stdout в порядке их появления.

Например, строка кода

```
printf ("Line one value=%d\n\t\tLine two value=%lf\n", 10, 12.6);
```

выработает на выводе

```
Line one value=10
Line two value = 12.6
```

Некоторые часто встречающиеся escape-последовательности:

\n – напечатать перевод строки

\t – напечатать символ табуляции

\\ – напечатать символ ‘\’

Некоторые часто встречающиеся спецификации формата

%d или **%i** – напечатается целочисленный десятичный знаковое значение.

%f – напечатается знаковое значение, имеющее форму [-]dddd.dddd, где dddd - одна или более десятичных цифр. Количество цифр перед десятичной точкой зависит от величины числа, а количество цифр после десятичной точки зависит от требуемой точности.

%8.2lf – в данном случае число типа double напечатается как минимум из 8 символов, дробная часть будет состоять из 2 символов.

%x – напечатается беззнаковый шестнадцатеричный целый.

%s – напечатается строка (char*).

%c – напечатается один символ (char)

Функция ввода scanf

int scanf(const char *format, [, argument...]); – функция читает данные из стандартного потока stdin в место, определяемое аргументами arguments. **Каждый аргумент должен быть указателем на значение** с типом, который соответствует типу, заданному в строке формата. Строка формата управляет преобразованиями полей ввода.

Пример считывания целого числа:

```
int i;
scanf("%d", &i);
```

Строка формата читается слева направо. Символы вне спецификации формата предполагаются согласованными с последовательностью символов в потоке stdin; эти согласованные символы в stdin сканируются, но не запоминаются. Если символ в stdin противоречит строке формата, scanf оканчивает свою работу. Этот конфликтующий символ остается в stdin, так как он не может быть прочитан. Когда встречается первая спецификация формата, тогда значение первого поля ввода преобразовывается в соответствии со спецификацией формата и запоминается в месте, заданном первым аргументом. По второй спецификации формата выполняется преобразование второго поля ввода и запоминание его по второму аргументу; и так до конца строки формата. Поле ввода ограничивается первым "пробельным" символом или первым символом, который не может преобразоваться по заданному формату, или случаем достижения поля width, которое идет первым. Если для выбранной спецификации формата задано больше аргументов, чем требуется, то лишние аргументы игнорируются. Спецификация формата имеет следующую форму аналогичную функции printf.

Пример использования.

```
int A[10];
printf("Введите элементы массива\n");
```

```
for (int i=0; i<10; i++){
     printf("A[%d] = ", i);
     scanf("%d", &A[i]);
}
```