

Министерство образования Республики Беларусь
«ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. ЕВФРОСИНИИ
ПОЛОЦКОЙ»

Факультет информационных технологий
Кафедра технологий программирования

**Методические указания для выполнения
лабораторной работы №7
по курсу «Конструирование программного
обеспечения»**

«Работа с массивами в языке высокого уровня»

Полоцк, 2022 г.

ЦЕЛЬ РАБОТЫ

Познакомится с такими понятиями как: массив, ранг, длина измерения, длина массива. Разобрать такие виды массивов как одномерный и многомерный. На основе примеров, приведенных в данной лабораторной работе, выполнить свой вариант практического задания.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Массивы в C#

Массив представляет набор однотипных данных. Объявление массива похоже на объявление переменной за тем исключением, что после указания типа ставятся квадратные скобки:

```
тип_переменной[] название_массива;
```

Например, определим массив целых чисел:

```
int[] numbers;
```

После определения переменной массива мы можем присвоить ей определенное значение:

```
int[] nums = new int[4];
```

Здесь вначале мы объявили массив `nums`, который будет хранить данные типа `int`. Далее используя операцию `new`, мы выделили память для 4 элементов массива: `new int[4]`. Число 4 еще называется длиной массива. При таком определении все элементы получают значение по умолчанию, которое предусмотрено для их типа. Для типа `int` значение по умолчанию - 0.

Также мы сразу можем указать значения для этих элементов:

```
int[] nums2 = new int[4] { 1, 2, 3, 5 };
```

```
int[] nums3 = new int[] { 1, 2, 3, 5 };
```

```
int[] nums4 = new[] { 1, 2, 3, 5 };
```

```
int[] nums5 = { 1, 2, 3, 5 };
```

Все перечисленные выше способы будут равноценны.

Подобным образом можно определять массивы и других типов, например, массив значений типа `string`:

```
string[] people = { "Tom", "Sam", "Bob" };
```

Индексы и получение элементов массива

Для обращения к элементам массива используются индексы. Индекс представляет номер элемента в массиве, при этом нумерация начинается с нуля, поэтому индекс первого элемента будет равен 0, индекс четвертого элемента - 3.

```
int[] numbers = { 1, 2, 3, 5 };

// получение элемента массива
Console.WriteLine(numbers[3]); // 5

// получение элемента массива в переменную
var n = numbers[1]; // 2
Console.WriteLine(n); // 2
```

Также мы можем изменить элемент массива по индексу:

```
int[] numbers = { 1, 2, 3, 5 };

// изменим второй элемент массива
numbers[1] = 505;

Console.WriteLine(numbers[1]); // 505
```

И так как у нас массив определен только для 4 элементов, то мы не можем обратиться, например, к шестому элементу. Если мы так попытаемся сделать, то мы получим ошибку во время выполнения:

```
int[] numbers = { 1, 2, 3, 5 };

Console.WriteLine(numbers[6]); // ! Ошибка - в массиве только 4
элемента
```

Свойство `Length` и длина массива

Каждый массив имеет свойство `Length`, которое хранит длину массива. Например, получим длину выше созданного массива `numbers`:

```
int[] numbers = { 1, 2, 3, 5 };

Console.WriteLine(numbers.Length); // 4
```

Для получения длины массива после названия массива через точку указывается свойство `Length`: `numbers.Length`.

Получение элементов с конца массива

Благодаря наличию свойства `Length`, мы можем вычислить индекс последнего элемента массива - это длина массива - 1. Например, если длина массива - 4 (то есть массив имеет 4 элемента), то индекс последнего элемента будет равен 3. И, используя свойство `Length`, мы можем легко получить элементы с конца массива:

```
int[] numbers = { 1, 2, 3, 5 };

Console.WriteLine(numbers[numbers.Length - 1]); // 5 - первый с
конца или последний элемент

Console.WriteLine(numbers[numbers.Length - 2]); // 3 - второй с
конца или предпоследний элемент

Console.WriteLine(numbers[numbers.Length - 3]); // 2 - третий
элемент с конца
```

Однако при подобном подходе выражения типа `numbers.Length - 1`, смысл которых состоит в том, чтобы получить какой-то определенный элемент с конца массива, утяжеляют код. И, начиная, с версии C# 8.0 в язык был добавлен специальный оператор `^`, с помощью которого можно задать индекс относительно конца коллекции.

Перепишем предыдущий пример, применяя оператор `^`:

```
int[] numbers = { 1, 2, 3, 5 };

Console.WriteLine(numbers[^1]); // 5 - первый с конца или
последний элемент

Console.WriteLine(numbers[^2]); // 3 - второй с конца или
предпоследний элемент

Console.WriteLine(numbers[^3]); // 2 - третий элемент с конца
```

Перебор массивов

Для перебора массивов мы можем использовать различные типы циклов. Например, цикл `foreach`:

```
int[] numbers = { 1, 2, 3, 4, 5 };
foreach (int i in numbers)
{
    Console.WriteLine(i);
}
```

Здесь в качестве контейнера выступает массив данных типа `int`. Поэтому мы объявляем переменную с типом `int`.

Подобные действия мы можем сделать и с помощью цикл `for`:

```
int[] numbers = { 1, 2, 3, 4, 5 };
for (int i = 0; i < numbers.Length; i++)
{
    Console.WriteLine(numbers[i]);
}
```

В то же время цикл `for` более гибкий по сравнению с `foreach`. Если `foreach` последовательно извлекает элементы контейнера и только для чтения, то в цикле `for` мы можем перескакивать на несколько элементов вперед в зависимости от приращения счетчика, а также можем изменять элементы:

```
int[] numbers = { 1, 2, 3, 4, 5 };
for (int i = 0; i < numbers.Length; i++)
{
    numbers[i] = numbers[i] * 2;
    Console.WriteLine(numbers[i]);
}
```

Также можно использовать и другие виды циклов, например, `while`:

```
int[] numbers = { 1, 2, 3, 4, 5 };
int i = 0;
while(i < numbers.Length)
{
    Console.WriteLine(numbers[i]);
    i++;
}
```

Многомерные массивы

Массивы характеризуются таким понятием как ранг или количество измерений. Выше мы рассматривали массивы, которые имеют одно измерение (то есть их ранг равен 1) - такие массивы можно представлять в виде ряда (строки или столбца) элемента. Но массивы также бывают многомерными. У таких массивов количество измерений (то есть ранг) больше 1.

Массивы которые имеют два измерения (ранг равен 2) называют двухмерными. Например, создадим одномерный и двухмерный массивы, которые имеют одинаковые элементы:

```
int[] nums1 = new int[] { 0, 1, 2, 3, 4, 5 };
```

```
int[,] nums2 = { { 0, 1, 2 }, { 3, 4, 5 } };
```

Визуально оба массива можно представить следующим образом:



Поскольку массив `nums2` двухмерный, он представляет собой простую таблицу. Все возможные способы определения двухмерных массивов:

```
int[,] nums1;  
int[,] nums2 = new int[2, 3];  
int[,] nums3 = new int[2, 3] { { 0, 1, 2 }, { 3, 4, 5 } };  
int[,] nums4 = new int[,] { { 0, 1, 2 }, { 3, 4, 5 } };  
int[,] nums5 = new [,]{ { 0, 1, 2 }, { 3, 4, 5 } };  
int[,] nums6 = { { 0, 1, 2 }, { 3, 4, 5 } };
```

Массивы могут иметь и большее количество измерений. Объявление трехмерного массива могло бы выглядеть так:

```
int[, ,] nums3 = new int[2, 3, 4];
```

Соответственно могут быть и четырехмерные массивы и массивы с большим количеством измерений. Но на практике обычно используются одномерные и двумерные массивы.

Определенную сложность может представлять перебор многомерного массива. Прежде всего надо учитывать, что длина такого массива - это совокупное количество элементов.

```
int[,] numbers = { { 1, 2, 3 }, { 4, 5, 6 } };  
foreach (int i in numbers)  
    Console.Write($"{i} ");
```

В данном случае длина массива numbers равна 6. И цикл foreach выводит все элементы массива в строку:

```
1 2 3 4 5 6
```

Но что если мы хотим отдельно пробежаться по каждой строке в таблице? В этом случае надо получить количество элементов в размерности. В частности, у каждого массива есть метод `GetUpperBound(номер_размерности)`, который возвращает индекс последнего элемента в определенной размерности. И если мы говорим непосредственно о двумерном массиве, то первая размерность (с индексом 0) по сути это и есть таблица. И с помощью выражения

```
numbers.GetUpperBound(0) + 1
```

можно получить количество строк таблицы, представленной двумерным массивом. А через

```
numbers.Length / количество_строк
```

можно получить количество элементов в каждой строке:

```
int[,] numbers = { { 1, 2, 3 }, { 4, 5, 6 } };
```

```
int rows = numbers.GetUpperBound(0) + 1;    // количество строк  
int columns = numbers.Length / rows;        // количество столбцов
```

```
// или так
// int columns = numbers.GetUpperBound(1) + 1;

for (int i = 0; i < rows; i++)
{
    for (int j = 0; j < columns; j++)
    {
        Console.Write($"{numbers[i, j]} \t");
    }
    Console.WriteLine();
}
```

1	2	3
4	5	6

Массив массивов

От многомерных массивов надо отличать **массив массивов** или так называемый "зубчатый массив":

```
int[][] nums = new int[3][];
nums[0] = new int[2] { 1, 2 };           // выделяем память для 1-
ого подмассива
nums[1] = new int[3] { 1, 2, 3 };        // выделяем память для 2-
ого подмассива
nums[2] = new int[5] { 1, 2, 3, 4, 5 };  // выделяем память для 3-
ого подмассива
```

Здесь две группы квадратных скобок указывают, что это массив массивов, то есть такой массив, который в свою очередь содержит в себе другие массивы. Причем длина массива указывается только в первых квадратных скобках, все последующие квадратные скобки должны быть пусты: `new int[3][]`. В данном случае у нас массив `nums` содержит три массива. Причем размерность каждого из этих массивов может не совпадать.

Альтернативное определение массива массивов:

```
int[][] numbers = {
    new int[] { 1, 2 },
    new int[] { 1, 2, 3 },
    new int[] { 1, 2, 3, 4, 5 }
};
```

Используя вложенные циклы, можно перебирать зубчатые массивы. Например:


```

int[][] numbers = new int[3][];
numbers[0] = new int[] { 1, 2 };
numbers[1] = new int[] { 1, 2, 3 };
numbers[2] = new int[] { 1, 2, 3, 4, 5 };
foreach(int[] row in numbers)
{
    foreach(int number in row)
    {
        Console.Write($"{number} \t");
    }
    Console.WriteLine();
}

// перебор с помощью цикла for
for (int i = 0; i<numbers.Length;i++)
{
    for (int j =0; j<numbers[i].Length; j++)
    {
        Console.Write($"{numbers[i][j]} \t");
    }
    Console.WriteLine();
}

```

Основные понятия массивов

Суммируем основные понятия массивов:

- Ранг (rank): количество измерений массива
- Длина измерения (dimension length): длина отдельного измерения массива
- Длина массива (array length): количество всех элементов массива

Например, возьмем массив

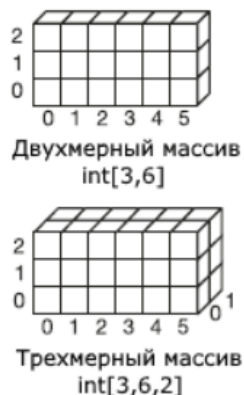
```
int[, ] numbers = new int[3, 4];
```

Массив numbers двухмерный, то есть он имеет два измерения, поэтому его ранг равен 2. Длина первого измерения - 3, длина второго измерения - 4. Длина массива (то есть общее количество элементов) - 12.

Одномерный массив



Многомерные массивы



Зубчатый массив



Массивы в C++

Массив представляет набор однотипных данных. Формальное определение массива выглядит следующим образом:

```
тип_переменной название_массива [длина_массива]
```

После типа переменной идет название массива, а затем в квадратных скобках его размер. Например, определим массив из 4 чисел:

```
int numbers[4];
```

Данный массив имеет четыре числа, но все эти числа имеют неопределенное значение. Однако мы можем выполнить инициализацию и присвоить этим числам некоторые начальные значения через фигурные скобки:

```
int numbers[4] = {1, 2, 3, 4};
```

Значения в фигурных скобках еще называют инициализаторами. Если инициализаторов меньше, чем элементов в массиве, то инициализаторы используются для первых элементов. Если инициализаторов больше, чем элементов в массиве, то при компиляции возникнет ошибка:

```
int numbers[4] = {1, 2, 3, 4, 5, 6};
```

Здесь массив имеет размер 4, однако ему передается 6 значений.

Если размер массива не указан явно, то он выводится из количества инициализаторов:

```
int numbers[] = {1, 2, 3, 4, 5, 6};
```

В данном случае в массиве есть 6 элементов.

Свои особенности имеет инициализация символьных массивов. Мы можем передать символьному массиву как набор инициализаторов, так и строку:

```
char s1[] = {'h', 'e', 'l', 'l', 'o'};  
char s2[] = "world";
```

Причем во втором случае массив s2 будет иметь не 5 элементов, а 6, поскольку при инициализации строкой в символьный массив автоматически добавляется нулевой символ '\0'.

При этом не допускается присвоение одному массиву другого массива:

```
int nums1[] = {1,2,3,4,5};
int nums2[] = nums1;      // ошибка
nums2 = nums1;           // ошибка
```

После определения массива мы можем обратиться к его отдельным элементам по индексу. Индексы начинаются с нуля, поэтому для обращения к первому элементу необходимо использовать индекс 0. Обратившись к элементу по индексу, мы можем получить его значение, либо изменить его:

```
#include <iostream>

int main()
{
    int numbers[4] = {1,2,3,4};
    int first_number = numbers[0];
    std::cout << first_number << std::endl; // 1
    numbers[0] = 34;                        // изменяем элемент
    std::cout << numbers[0] << std::endl; // 34

    return 0;
}
```

Число элементов массива также можно определять через константу:

```
const int n = 4;
int numbers[n] = {1,2,3,4};
```

Перебор массивов

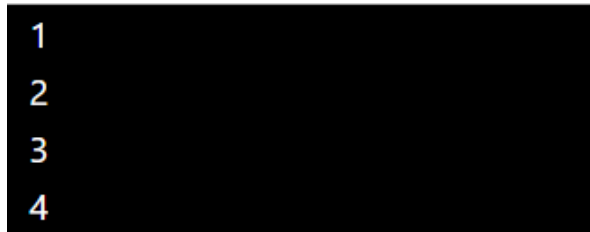
Используя циклы, можно пробежаться по всему массиву и через индексы обратиться к его элементам:

```
#include <iostream>
int main()
{
    int numbers[4] = {1,2,3,4};
    int size = sizeof(numbers) / sizeof(numbers[0]);
    for(int i=0; i < size; i++)
        std::cout << numbers[i] << std::endl;

    return 0;
}
```

Чтобы пройти по массиву в цикле, вначале надо найти длину массива. Для нахождения длины применяется оператор `sizeof`. По сути длина массива равна

совокупной длине его элементов. Все элементы представляют один и тот же тип и занимают один и тот же размер в памяти. Поэтому с помощью выражения `sizeof(numbers)` находим длину всего массива в байтах, а с помощью выражения `sizeof(numbers[0])` - длину одного элемента в байтах. Разделив два значения, можно получить количество элементов в массиве. А далее с помощью цикла `for` перебираем все элементы, пока счетчик `i` не станет равным длине массива. В итоге на консоль будут выведены все элементы массива:



```
1
2
3
4
```

Но также есть и еще одна форма цикла `for`, которая предназначена специально для работы с коллекциями, в том числе с массивами. Эта форма имеет следующее формальное определение:

```
for(тип переменная : коллекция)
{
    инструкции;
}
```

Используем эту форму для перебора массива:

```
#include <iostream>

int main()
{
    int numbers[4] = {1,2,3,4};
    for(int number : numbers)
        std::cout << number << std::endl;

    return 0;
}
```

При переборе массива каждый перебираемый элемент будет помещаться в переменную `number`, значение которой в цикле выводится на консоль.

Если нам неизвестен тип объектов в массиве, то мы можем использовать спецификатор `auto` для определения типа:

```
for(auto number : numbers)
    std::cout << number << std::endl;
```

Многомерные массивы

Кроме одномерных массивов в C++ есть многомерные. Элементы таких массивов сами в свою очередь являются массивами, в которых также элементы могут быть массивами. Например, определим двухмерный массив чисел:

```
int numbers[3][2];
```

Такой массив состоит из трех элементов, при этом каждый элемент представляет массив из двух элементов. Инициализируем подобный массив:

```
int numbers[3][2] = { {1, 2}, {4, 5}, {7, 8} };
```

Вложенные фигурные скобки очерчивают элементы для каждого подмассива. Такой массив еще можно представить в виде таблицы:

1	2
4	5
7	8

Также при инициализации можно опускать фигурные скобки:

```
int numbers[3][2] = { 1, 2, 4, 5, 7, 8 };
```

Возможна также инициализация не всех элементов, а только некоторых:

```
int numbers[3][2] = { {1, 2}, {}, {7} };
```

И чтобы обратиться к элементам вложенного массива, потребуется два индекса:

```
int numbers[3][2] = { {1, 2}, {3, 4}, {5, 6} };  
std::cout << numbers[1][0] << std::endl;    // 3  
numbers[1][0] = 12;                          // изменение элемента  
std::cout << numbers[1][0] << std::endl;    // 12
```

Переберем двухмерный массив:

```
#include <iostream>
int main()
{
    const int rows = 3, columns = 2;
    int numbers[rows][columns] = { {1, 2}, {3, 4}, {5, 6} };
    for(int i=0; i < rows; i++)
    {
        for(int j=0; j < columns; j++)
        {
            std::cout << numbers[i] [j] << "\t";
        }
        std::cout << std::endl;
    }
    return 0;
}
```

Также для перебора элементов многомерного массива можно использовать другую форму цикла for:

```
#include <iostream>
int main()
{
    const int rows = 3, columns = 2;
    int numbers[rows][columns] = { {1, 2}, {3, 4}, {5, 6} };
    for(auto &subnumbers : numbers)
    {
        for(int number : subnumbers)
        {
            std::cout << number << "\t";
        }
        std::cout << std::endl;
    }
    return 0;
}
```

Для перебора массивов, которые входят в массив, применяются ссылки. То есть во внешнем цикле `for(auto &subnumbers : numbers)` `&subnumbers` представляет ссылку на подмассив в массиве. Во внутреннем цикле `for(int number : subnumbers)` из каждого подмассива в `subnumbers` получаем отдельные его элементы в переменную `number` и выводим ее значение на консоль.

Содержание отчета

Отчет должен включать:

- а) титульный лист;
- б) формулировку цели работы;
- в) описание результатов выполнения заданий:
 - листинги программ;
 - результаты выполнения программ;
- г) выводы, согласованные с целью работы.

Варианты

Вариант	Задание 1	Задание 2	Задание 3	Задание 4
1	1	11	21	31
2	2	12	22	32
3	3	13	23	33
4	4	14	24	34
5	5	15	25	35
6	6	16	26	36
7	7	17	27	37
8	8	18	28	38
9	9	19	29	39
10	10	20	30	40

Задания

1. Сформировать массив из 15 целых чисел, выбранных случайным образом из интервала $[-10, 30]$. Найти среднее арифметическое положительных элементов.
2. Сформировать массив из 10 целых чисел, выбранных случайным образом из интервала $[10, 50]$. Найти максимальный среди элементов с четными индексами.
3. Сформировать массив из 15 целых чисел, выбранных случайным образом из интервала $[10, 90]$. Поменять местами первый и минимальный элементы.
4. Задан одномерный массив $A[1..20]$. Найти минимальный элемент среди элементов массива с n -го по k -й (n и k вводятся с клавиатуры)
5. Массив содержит $2n$ чисел. Из суммы первых n его элементов вычесть сумму последних n элементов.
6. Заменить отрицательные элементы в числовом массиве из n чисел ($n > 10$) их квадратами, оставив остальные без изменения.
7. В заданном массиве найти среднее арифметическое положительных чисел, среднее арифметическое отрицательных чисел и число нулей.
8. В массиве из $2n$ чисел найти сумму квадратов элементов с четными индексами и сумму кубов элементов с нечетными индексами.

9. В заданном массиве найти максимальный элемент. Элементы, стоящие после максимального элемента заменить нулями.

10. В заданном массиве поменять местами наибольший и наименьший элементы.

11. Задан массив положительных вещественных чисел. Вычислить значения функции $y=0,5x$ при значениях аргумента, заданных в исходном массиве, и поместить их в другой массив. Вывести на экран дисплея оба массива в виде двух столбцов.

12. В одномерном массиве целых чисел a_1, a_2, \dots, a_n найти номер первого четного числа. Если четных чисел нет, то ответом может быть число 0.

13. Заданы два массива A и B. Написать программу нахождения элементов, общих для A и B.

14. Определить, какие элементы массива A и сколько раз встречаются в массиве B.

15. Дан массив из n чисел как положительных, так и отрицательных. Нужно сначала записать положительные числа, а затем отрицательные в том же порядке, как они были записаны в исходном массиве.

16. Задан одномерный массив $A[1..15]$. Определить количество четных положительных элементов массива.

17. Заполнить массив $A[1..8]$ числами, вводимыми с клавиатуры. Найти среднее арифметическое положительных элементов.

18. Задан одномерный массив $A[1..20]$. Просуммировать все отрицательные элементы, стоящие на нечетных местах.

19. Задан одномерный массив $A[1..17]$. Определить среднее значение нечетных положительных элементов массива.

20. Задан одномерный массив $A[1..15]$. Определить сумму четных положительных элементов массива с n-го по k-й.

21. Заменить положительные элементы двумерного массива на 1, а отрицательные на 0.

22. В двумерном массиве числа, кратные n, заменить частными от деления на n. Если таких чисел нет, то вывести на экран сообщение об этом.

23. Следом квадратной матрицы называют число, равное сумме элементов главной диагонали. Составить программу нахождения следа квадратной матрицы порядка n.

24. В двумерном массиве найти сумму всех тех элементов, сумма индексов которых равна n.

25. Из данной прямоугольной таблицы вывести на экран строки, содержащие хотя бы один нулевой элемент.

26. Составить программу для определения номера строки и номера столбца прямоугольной матрицы, на пересечении которых находится наибольший по абсолютной величине элемент этой матрицы.

27. Составить программу нахождения максимального элемента в каждом столбце и минимального в каждой строке квадратной матрицы.

28. Составить программу обмена местами максимального и минимального элементов главной диагонали матрицы.

29. Дана матрица $N \times N$. Вывести на экран дисплея элементы той строки, сумма элементов которой максимальна.

30. Дана квадратная матрица порядка n . Составить программу вычисления количества положительных элементов в нижнем левом треугольнике, включая диагональные элементы.

31. Сформировать и вывести на экран в виде таблицы массив $A[1..4, 1..6]$, заполнив его целыми случайными числами из интервала $[30, 75]$. Найти среднее арифметическое каждого столбца.

32. В массиве $B[1..4, 1..4]$ найти сумму элементов главной диагонали.

33. Найти наибольший и наименьший элементы данного двумерного массива. Указать их индексы.

34. В двумерном массиве числа, кратные 2, заменить частными от деления на 2. Если таких чисел нет, то вывести на экран сообщение об этом.

35. Составить программу для определения номера строки и номера столбца прямоугольной матрицы, на пересечении которых находится наименьший по абсолютной величине элемент этой матрицы.

36. Составить программу нахождения минимального элемента в каждом столбце и максимального в каждой строке квадратной матрицы.

37. Составить программу обмена местами максимального и минимального элементов главной диагонали матрицы.

38. Дана матрица $N \times N$. Вывести на экран дисплея элементы той строки, сумма элементов которой максимальна.

39. Вывести на экран матрицу 5×5 . Определить сумму минимальных элементов столбцов матрицы.

40. Вывести на экран матрицу 4×5 . Определить номера столбцов, содержащих более половины положительных элементов

Контрольные вопросы

1. Что такое массив в языке программирования?
2. Какие виды массивов могут быть представлены?
3. Какие преимущества использования массивов в программах?
4. Каким образом организовано представление массивов?
5. Какая общая форма объявления одномерного массива?
6. Какой массив называется многомерным?
7. Какая общая форма объявления многомерного массива?
8. Что такое ступенчатый массив? Какая общая форма объявления ступенчатого массива?
9. В каких случаях целесообразно использовать ступенчатые массивы?