**Практическая работа №1**

**Оглавление**

[Практическая работа № 1. Основы языка C# 3](#_Toc161928756)

[Содержание пояснительной записки 10](#_Toc161928757)

[Используемое программное обеспечение 11](#_Toc161928758)

[Список литературы 12](#_Toc161928759)

Практическая работа № 1.  
Основы языка C#

**Цель практической работы**

Закрепление теоретических знаний по разработке модульных тестов к программам.

**Постановка задачи**

**Форматированный вывод**

**Рассмотрим на примерах**.

У нас есть некоторая переменная int value = 5;

и мы хотим получить строку вида: **«Васе value лет» («Васе 5 лет»),**

Для этого есть ряд способов:

string str = «Васе » + value + « лет»; **Конкатенация строк**

**Недостатки:** при каждом вызове оператора конкатенации «+», в памяти выделяется место под новую строку, новая строка записывается в новую ячейку памяти, а старая строка уничтожается. Много «+» => очень медленная работа программы.

int value = 5;

string str = string.Format("Васе {0} лет", value);

Примечание:

Если строку нужно вывести в консоль, то string.Format можно не писать:

Console.WriteLine("Васе {0} лет", value);

Современный способ форматированного вывода:

string str = $"Васе {value} лет"; // Интерполяция строк (знак доллара)

**Пример 2:**

Мы хотим получить строку вида: **«Васе age лет, у него money рублей»**

int age = 25;

int money = 20000;

string str = string.Format(«Васе {0} лет, у него {1} руб-лей», age, money);

string str = $"Васе {age} лет, у него {money} рублей";

**Пример 3:**

Мы хотим получить строку вида: **«Васе age лет, у него money рублей»,** но тип **money** вещественный и нам нужно только 2 знака.

int age = 25;

double money = 2.15367521E4; //21536.7521

string str = string.Format(«Васе {0} лет, у него {1 : f2} рублей», age, money);

string str = $"Васе {age} лет, у него {money : f2} рублей";

d – целые числа; f – дробные; c – денежный формат;

e – экспоненциальная форма записи;

g – выбирает наиболее короткий из f и e.

ИЛИ

string str = $"Васе {age} лет, у него {money : 0.00} рублей";

Получим строку вида: **«Васе 25 лет, у него 21536.75 рублей»**

**Перечислимые константы (перечисления)** – это тип значения, определенный набором именованных констант применяемого целочисленного типа. Чтобы определить тип перечисления, используйте ключевое слово **enum** и укажите имена:

enum Transport { Car, Airplane, Bicycle, Boat, Spaceship }

Transport myTransport = Transport.Car;

enum DayTime { Morning, Afternoon, Evening, Night}

DayTime dayTime = DayTime.Evening;

**Строки.**

Со строками можно работать как с массивом символов, т.е:

string str = "Это строка";

Можно обратиться к символу строки как к элементу массива:

Console.WriteLine(str[0]); //Ответ: ‘Э’

Console.WriteLine(str[3]); //Ответ: ‘ ’

Можно узнать длину строки с помощью свойства Length:

Console.WriteLine(str.Length); //Ответ: 10

Однако, строки – неизменяемый тип данных, заменить отдельный символ на другой нельзя!

Существуют методы «замены» символа на другой, но они создают новую строку:

str = str.Replace(‘о’, ‘а’); //Ответ: “Эта страка”

**Сортировка** – процесс упорядочивания элементов по некоторому признаку.

var rnd = new Random();

double[] array = new double[10];

for (int i = 0; i < array.Length; i++)

{

array[i] = rnd.Next(0, 100);

}

Array.Sort(array);

for (int i = 0; i < array.Length; i++)

{

Console.WriteLine($"{array[i]}");

}

**Сортировка пузырьком**

void BubbleSort(double[] arr)

{

for (int i = 0; i < arr.Length; i++)

{

for (int j = 0; j < arr.Length – i – 1; j++)

{

if (arr[j] > arr[j+1])

{

var temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

Алгоритм «**Сортировка выбором**»

Его смысл заключается в том, чтобы, чтобы идти по массиву, например слева направо, каждый раз искать минимальный элемент массива и обменивать его с первым элементом неотсортированной части массива.

1. Находим минимальный элемент в массиве.
2. Меняем местами минимальный и *первый* элемент местами.
3. Ищем минимальный элемент в неотсортированной части массива, т.е., начиная уже со *второго* элемента массива.
4. Меняем местами второй элемент массива и найденный минимальный.
5. Ищем минимальный элемент в массиве, начиная с *третьего*, меняем местами третий и минимальный элементы.
6. Продолжаем алгоритм до тех пор пока не дойдем то конца массива.

**Сложность алгоритма** – оценка количества базовых действий, необходимых для выполнения алгоритма.

Если требуется **n, 2n, … kn**, действий, то говорят, что сложность алгоритма **O(n)**, где k – некоторая константа, не зависящая от числа действий.

**Пример 1.**

var n = int.Parse(Console.ReadLine());

var sum = 0.0;

for (var i = 0; i < n; i++)

{

sum += n;

}

Console.WriteLine(sum);

Сложность алгоритма **O(n)**

**Пример 2.**

var n = int.Parse(Console.ReadLine());

var sum = 0.0;

for (var i = 0; i < n; i++)

{

for (var j = 0; j < n; j++)

{

sum += n;

}

}

Сложность алгоритма **O(n2)**

**Задание на практическую работу**

1. Разработать консольное приложение для решения задачи согласно варианту.
2. Предусмотреть понятный интерфейс ввода и вывода данных на консоль.
3. Защита работы включает демонстрацию работы программы.

**Варианты заданий на практическую работу**

Задания берем с сайта «Школа программиста» <https://acmp.ru/>

**Примечание:**

Ввод и вывод данных осуществлять через графический интерфейс программы (консоль) без использования файлов.

|  |  |
| --- | --- |
| Номер варианта | Ссылка на задание |
| 1 | 35 <https://acmp.ru/index.asp?main=task&id_task=35> |
| 2 | 819 <https://acmp.ru/index.asp?main=task&id_task=819> |
| 3 | 907 <https://acmp.ru/index.asp?main=task&id_task=907> |
| 4 | 149 <https://acmp.ru/index.asp?main=task&id_task=149> |
| 5 | 3 <https://acmp.ru/index.asp?main=solution&id_task=3> |
| 6 | 92 <https://acmp.ru/index.asp?main=task&id_task=92> |
| 7 | 754 <https://acmp.ru/index.asp?main=task&id_task=754> |
| 8 | 948 <https://acmp.ru/asp/do/index.asp?main=task&id_course=3&id_section=22&id_topic=85&id_problem=441> |
| 9 | 52 <https://acmp.ru/index.asp?main=task&id_task=52> |
| 10 | 970 <https://acmp.ru/index.asp?main=task&id_task=970> |
| 11 | 529 <https://acmp.ru/asp/do/index.asp?main=task&id_course=3&id_section=23&id_topic=92&id_problem=471> |
| 12 | 757 <https://acmp.ru/index.asp?main=task&id_task=757> |
| 13 | 892 <https://acmp.ru/index.asp?main=task&id_task=892> |
| 14 | 756 <https://acmp.ru/asp/do/index.asp?main=task&id_course=3&id_section=22&id_topic=78&id_problem=412> |
| 15 | 692 <https://acmp.ru/index.asp?main=task&id_task=692> |
| 16 | 550 <https://acmp.ru/index.asp?main=task&id_task=550> |
| 17 | 933 <https://acmp.ru/index.asp?main=task&id_task=933> |
| 18 | 697 <https://acmp.ru/index.asp?main=task&id_task=697> |
| 19 | 597 <https://acmp.ru/index.asp?main=solution&id_task=597> |
| 20 | 66 <https://acmp.ru/index.asp?main=task&id_task=66> |
| 21 | 106 <https://acmp.ru/index.asp?main=solution&id_task=106> |
| 22 | 685 <https://acmp.ru/index.asp?main=task&id_task=685> |
| 23 | 539 <https://acmp.ru/index.asp?main=task&id_task=539> |
| 24 | 854 <https://acmp.ru/index.asp?main=task&id_task=854> |
| 25 | 23 <https://acmp.ru/index.asp?main=solution&id_task=23> |
| 26 | 940 <https://acmp.ru/asp/do/index.asp?main=task&id_course=3&id_section=26&id_topic=108&id_problem=572> |
| 27 | 324 <https://acmp.ru/index.asp?main=task&id_task=324> |
| 28 | 504 <https://acmp.ru/index.asp?main=task&id_task=504> |
| 29 | 755 <https://acmp.ru/asp/do/index.asp?main=task&id_course=3&id_section=22&id_topic=77&id_problem=408> |
| 30 | 777 <https://acmp.ru/index.asp?main=task&id_task=777> |
| 31 | 606 <https://acmp.ru/index.asp?main=task&id_task=606> |

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»