Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Смоленский государственный университет»
Кафедра прикладной математики и информатики

«Утверждаю»
Проректор по учебно-
методической работе
_____ Ю.А. Устименко
«23» июня 2022 г.

Рабочая программа дисциплины
Б1.О.20 Разработка и стандартизация программных средств и информационных технологий

Направление подготовки: **09.03.03 Прикладная информатика**
Направленность (профиль): **Информационные системы организаций и предприятий**
Форма обучения: заочная
Курс – 3
Семестр – 5
Всего зачетных единиц – 3, часов – 108
Форма отчетности: зачет – 5 семестр

Программу разработал
кандидат педагогических наук, доцент Козлов С.В.

Одобрена на заседании кафедры
«16» июня 2022 г., протокол № 10

Заведующий кафедрой _____ С.В. Козлов

Смоленск
2022

1. Место дисциплины в структуре ОП

Дисциплина «Разработка и стандартизация программных средств и информационных технологий» относится к дисциплинам обязательной части учебного плана данного направления подготовки. Она изучается на 2 курсе в 4 семестре. При ее изучении необходимы компетенции студентов, сформированные при изучении таких дисциплин, как «Основы информатики», «Операционные системы», «Структуры и алгоритмы компьютерной обработки данных», «Вычислительные системы, сети и телекоммуникации» и др.

В современных условиях процесс разработки программного обеспечения, как сложный и дорогостоящий, регламентируется большим количеством стандартов. Будущему специалисту важно глубоко разбираться во множестве современных стандартов и информационных технологиях, применяемых при разработке программного обеспечения информационных систем. Поэтому компетенции, сформированные при изучении дисциплины, необходимы для последующего изучения курсов «Проектирование информационных систем», «Программная инженерия», «Администрирование информационных систем», написания выпускной квалификационной работы бакалавра и его дальнейшей профессиональной деятельности.

В связи с этим курс «Разработка и стандартизация программных средств и информационных технологий» занимает важное место в предметной подготовке бакалавров по направлению подготовки **09.03.03 «Прикладная информатика»** (профиль «Информационные системы организаций и предприятий»).

Изучение курса основано на традиционных методах высшей школы, тесной взаимосвязи со смежными курсами, обобщающими методологию исследований и проектирования социально-экономических информационных систем.

2. Планируемые результаты обучения по дисциплине

Компетенция	Индикаторы достижения
ОПК-2. Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности	**Знать:** современные информационные технологии и программные средства, в том числе отечественного производства, применяемые при решении задач профессиональной деятельности; **Уметь**: выбирать современные информационные технологии и программные средства, необходимые для решения задач профессиональной деятельности; **Владеть**: навыками применения современных информационных технологий и программных средств при решении задач профессиональной деятельности.
ОПК-3. Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	**Знать:** основные типы стандартных задач профессиональной деятельности и методы их решения с учетом требования информационной безопасности и применяя современные информационно-коммуникационные технологии на базе информационной и библиографической культуры; **Уметь**: решать задачи профессиональной деятельности на основе информационной и библиографической культуры с применение информационно-коммуникационных технологий и с четом основных требований информационной безопасности; **Владеть**: приемами решения задач профессиональной деятельности на основе информационной и библиографической культуры с применение информационно-коммуникационных

	технологий и с четом основных требований информационной безопасности.
ОПК-4. Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью	**Знать**: базовые принципы порядка документирования процессов, связанных со всеми стадиями жизненного цикла информационных систем, основные нормы, правила и стандарты, регламентирующие процесс создания информационных систем, основные этапы разработки информационных систем и нормативную сопроводительную документацию; **Уметь**: разрабатывать стандарты, нормы, правила и прочую сопроводительную техническую документацию, связанную с профессиональной деятельностью; ориентироваться в системе стандартов и других нормативно-правовых актов, регламентирующих сферу разработки программных средств и проектирования информационных технологий; **Владеть**: нормативными требованиями по разработке и сопровождению процессов создания информационных систем по стадиям жизненного цикла, навыками документирования программных средств в соответствии с требованиями.
ОПК-7. Способен разрабатывать алгоритмы и программы, пригодные для практического применения	**Знать**: основные принципы и методики создания алгоритмов и программ для решения прикладных задач, основные среды для разработки программного обеспечения; **Уметь**: внедрять и адаптировать прикладное программное обеспечение; **Владеть**: современными языками программирования и методиками разработки и внедрения прикладного программного обеспечения.
ОПК-8. Способен принимать участие в управлении проектами создания информационных систем на стадиях жизненного цикла	**Знать**: основы проектной деятельности, особенности управления проектами создания информационных систем на стадиях жизненного цикла, основные этапы разработки информационных систем; **Уметь**: организовывать процесс управления проектами создания информационных систем на стадиях жизненного цикла, разрабатывать управленческие решения в соответствии с поставленными целями; **Владеть**: навыками по управлению и сопровождению процессов создания информационных систем по стадиям жизненного цикла.

3. Содержание дисциплины

1. **Характеристика программных средств и информационных технологий как объекта разработки и стандартизации.** Информационные технологии. Основные понятия, терминология и классификация. Сущность, значение и закономерности развития информационных технологий в современной экономике. Экономические законы развития

информационных технологий. Технология и методы обработки экономической информации. Основные классы технологий Структура базовой информационной технологии. Программное обеспечение ЭВМ. Пакеты прикладных программ. Программные средства (ПС).

2. **Особенности разработки программного обеспечения.** Технические особенности разработки программных средств. Принципы модульности и адаптируемости. Экономические особенности разработки программных средств. Способы формального представления знаний, основы устройства и использование экспертных систем в разработке адаптируемого программного обеспечения. Вопросы оценки трудоёмкости разработки программных средств в свете требований стандартизации.
3. **Основные положения технологии и организация проектирования программных средств.** Проблемы и задачи проектирования программных средств. Этапы жизненного цикла программных средств. Виды поддержки и стадии этапа проектирования. Основные понятия и определения статического анализа программных средств. Эффективность технологий проектирования программных средств. Организация проектирования программных средств. Основные процессы жизненного цикла ПС. Вспомогательные процессы жизненного цикла ПС. Организационные процессы жизненного цикла ПС. Базовый стандарт ГОСТ Р ИСО/МЭК 12207-2010.
4. **Стандартизация и метрология в разработке программного обеспечения. Действующие стандарты и проблемы программных интерфейсов.** Нормативные документы по стандартизации и виды стандартов. Стандарты в области программного обеспечения. ИСО/МЭК. Государственный комитет РФ по стандартизации. Проектирование и разработка интерфейса ПС.
5. **Стандарты документирования программных средств**. Общая характеристика состояния стандартизации в области документирования программных средств. Единая система программной документации. Государственные стандарты Российской Федерации (ГОСТ Р).
6. **Оценка качественных и количественных характеристик программных средств.** Понятие качественного ПС и связанные с ним характеристики. Стандартизация показателей качества ПС. Характеристики качества базового международного стандарта ISO 9126:1991. Основные понятия и показатели надежности ПС. Методы обеспечения надежности ПС. Тестирование ПС. Методы оценки технико-экономических показателей программных средств на различных этапах их жизненного цикла
7. **Основы построения системы стандартов информационных технологий. Инструменты функциональной стандартизации**. Понятие открытых систем Международные структуры в области стандартизации и информационных технологий. Методологический базис открытых систем. Архитектурные спецификации (эталонные модели). Эталонная модель взаимосвязи открытых систем. Эталонная модель среды открытых систем (модель OSE). Базовая эталонная модель взаимосвязи открытых систем (модель OSI). Базовые спецификации. Понятие профиля открытой системы. Классификация профилей. Основные свойства и назначение профилей.
8. **Общие сведения о сертификации информационных систем и их программных средств.** Основные положения закона «О техническом регулировании» (ТР). Особенности сертификации программного обеспечения.

4. Тематический план

№ п/п	Разделы и темы	Всего часов	Формы занятий			
			лекции	практические занятия	лабораторные занятия	самостоятельная работа
1	Характеристика программных средств и информационных технологий как объекта разработки и	16	2	–	2	12

	стандартизации					
2	Особенности разработки программного обеспечения	16	2	–	2	12
3	Основные положения технологии и организация проектирования программных средств	16	2	–	2	12
4	Стандартизация и метрология в разработке программного обеспечения. Действующие стандарты и проблемы программных интерфейсов	14	–	–	2	12
5	Стандарты документирования программных средств	12	–	–	–	12
6	Оценка качественных и количественных характеристик программных средств	10	–	–	–	10
7	Основы построения системы стандартов информационных технологий. Инструменты функциональной стандартизации	10	–	–	–	10
8	Общие сведения о сертификации информационных систем и их программных средств	10	–	–	–	10
	Зачет (зачетная конрольная работа)	4	–	–	–	4
ИТОГО		**108**	**6**	**–**	**8**	**90+4**

5. Виды образовательной деятельности

Занятия лекционного типа

1. **Характеристика программных средств и информационных технологий как объекта разработки и стандартизации.**

Информационные технологии. Основные понятия, терминология и классификация. Сущность, значение и закономерности развития информационных технологий в современной экономике. Экономические законы развития информационных технологий.

Технология и методы обработки экономической информации. Основные классы технологий Структура базовой информационной технологии. Программное обеспечение ЭВМ. Пакеты прикладных программ. Программные средства (ПС).

2. **Особенности разработки программного обеспечения.**

Технические особенности разработки программных средств. Принципы модульности и адаптируемости. Экономические особенности разработки программных средств.

Способы формального представления знаний, основы устройства и использование экспертных систем в разработке адаптируемого программного обеспечения. Вопросы оценки трудоёмкости разработки программных средств в свете требований стандартизации.

3. **Основные положения техноло-гии и организация проектирова-ния программных средств.**

Проблемы и задачи проектирования программных средств. Этапы жизненного цикла программных средств. Виды поддержки и стадии этапа проектирования. Основные понятия и определения статического анализа программных средств.

Эффективность технологий проектирования программных средств. Организация проектирования программных средств. Основные процессы жизненного цикла ПС.

Основные процессы жизненного цикла ПС. Вспомогательные процессы жизненного цикла ПС. Организационные процессы жизненного цикла ПС. Базовый стандарт ГОСТ Р ИСО/МЭК 12207-2010.

Лабораторные занятия

При выполнении лабораторных работ в большей степени упор делается не на сложность алгоритмов решения предлагаемых задач, а на технологичность исполнения (максимальное использование возможностей среды разработки), соответствие программного кода структурным требованиям, принципам модульности, ООП, адаптируемости, переносимости ПО.

При разработке студенческих программ закладываются основы проектирования программных средств, разработки систем с открытым программным интерфейсом, уделяется внимание пользовательскому интерфейсу, устойчивости работы программного кода, обработке исключительных ситуаций, оценке качества и тестированию.

В материалах лабораторных работ номера задач для каждого студента указывает преподаватель.

Лабораторное занятие №1.

Цель работы: освоить особенности работы с методами класса, перегруженными методами в среде MS Visual Studio.

Программное обеспечение и материалы: актуальные версии Microsoft Windows, MS Visual Studio.

Задания

Часть 1. Решение тривиальных задач

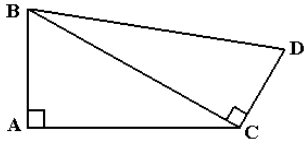
1. Разработать метод min(a,b) для нахождения минимального из двух чисел. Вычислить с помощью него значение выражения z=min(3x,2y)+min(x-y,x+y).

Пример.

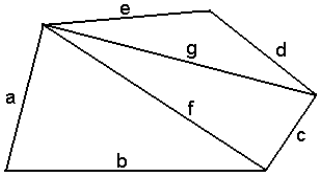
```
using System;
namespace Hello
{
   class Program
   {
      static double min(double a, double b)
      {
         return (a < b) ? a : b;
      }
      static void Main(string[] args)
      {
         Console.Write("x=");
         double x = double.Parse(Console.ReadLine());
```

```
            Console.Write("y=");
            double y = double.Parse(Console.ReadLine());
            double z = min(3 * x, 2 * y) + min(x - y, x + y);
            Console.WriteLine("z=" + z);
        }
    }
}
```

2. Разработать метод min(a,b) для нахождения минимального из двух чисел. Вычислить с помощью него минимальное значение из четырех чисел x, y, z, v.
3. Разработать метод max(a,b) для нахождения максимального из двух чисел. Вычислить с помощью него значение выражения z=max(x,2y-x)+max(5x+3y,y).
4. Разработать метод f(x), который вычисляет значение по следующей формуле: $f(x)=x^3-\sin x$. Определить, в какой из точек a или b, функция принимает наибольшее значение.
5. Разработать метод f(x), который вычисляет значение по следующей формуле: f(x)=cos(2x)+sin(x-3). Определить, в какой из точек a или b, функция принимает наименьшее значение.
6. Разработать метод f(x), который возвращает младшую цифру натурального числа x. Вычислить с помощью него значение выражения z=f(a)+f(b).
7. Разработать метод f(x), который возвращает вторую справа цифру натурального числа x. Вычислить с помощью него значение выражения z=f(a)+f(b)-f(c).
8. Разработать метод f(n), который для заданного натурального числа n находит значение $\sqrt{n}+n$. Вычислить с помощью него значение выражения $\frac{\sqrt{6}+6}{2}+\frac{\sqrt{13}+13}{2}+\frac{\sqrt{21}+21}{2}$.
9. Разработать метод f(n, x), которая для заданного натурального числа n и вещественного x находит значение выражения $\frac{x^n}{n}$. Вычислить с помощью данного метода значение выражения $\frac{x^2}{2}+\frac{x^4}{4}+\frac{x^6}{6}$.
10. Разработать метод f(x), который нечетное число заменяет на 0, а четное число уменьшает в два раза. Продемонстрировать работу данного метода на примере.
11. Разработать метод f(x), который число, кратное 5, уменьшает в 5 раз, а остальные числа увеличивает на 1. Продемонстрировать работу данного метода на примере.
12. Разработать метод f(x), который в двузначном числе меняет цифры местами, а остальные числа оставляет без изменения. Продемонстрировать работу данного метода на примере.
13. Разработать метод f(x), который в трехзначном числе меняет местами первую с последней цифрой, а остальные числа оставляет без изменения. Продемонстрировать работу данного метода на примере.
14. Разработать метод f(a, b), который по катетам a и b вычисляет гипотенузу. С помощью данного метода найти периметр фигуры ABCD по заданным сторонам AB, AC и DC.



15. Разработать метод f(x, y, z), который по длинам сторон треугольника x, y, z вычисляет его площадь. С помощью данного метода по заданным вещественным числам a, b, c, d, e, f, g найти площадь пятиугольника, изображенного на рисунке.



16. Разработать метод f(x_1, y_1, x_2, y_2), который вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и метод d(a, b, c), который вычисляет периметр треугольника по длинам сторон a, b, c. С помощью данных методов найти периметр треугольника, заданного координатами своих вершин.
17. Разработать метод f(x_1, y_1, x_2, y_2), который вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и метод max(a, b), который вычисляет максимальное из чисел a, b.

С помощью данных методов определить, какая из трех точек на плоскости наиболее удалена от начала координат.

18. Разработать метод f(x_1, y_1, x_2, y_2), который вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и метод min(a, b), который вычисляет минимальное из чисел a, b. С помощью данных методов найти две из трех заданных точек на плоскости, расстояние между которыми минимально.
19. Разработать метод f(x_1, y_1, x_2, y_2), который вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и метод t(a, b, c), который проверяет, существует ли треугольник с длинами сторон a, b, c. С помощью данных методов проверить, можно ли построить треугольник по трем заданным точкам на плоскости.
20. Разработать метод f(x_1, y_1, x_2, y_2), который вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и метод t(a, b, c), который проверяет, существует ли треугольник с длинами сторон a, b, c. С помощью данных методов проверить, сколько различных треугольников можно построить по четырем заданным точкам на плоскости.

Часть 2. Построить таблицу значений функции y=f(x) для $x \in [a, b]$ с шагом h.

Замечание. Для решения задачи использовать вспомогательный метод.

1. $$y = \begin{cases} \dfrac{1}{(0.1+x)^2}, & \text{если } x \ge 0.9; \\ 0.2x + 0.1, & \text{если } 0 \le x < 0.9; \\ x^2 + 0.2, & \text{если } x < 0 \end{cases}$$

Пример:

```
using System;
namespace Hello
{
   class Program
   {
      static double f (double x)
      {
         double y;
         if (x >= 0.9) y = 1 / Math.Pow(1 + x, 2);
         else if (x >= 0) y = 0.2 * x + 0.1;
         else y = x * x + 0.2;
         return y;
      }
      static void Main(string[] args)
      {
         Console.Write("a=");
         double a = double.Parse(Console.ReadLine());
         Console.Write("b=");
         double b = double.Parse(Console.ReadLine());
         Console.Write("h=");
         double h = double.Parse(Console.ReadLine());
         for (double i = a; i <= b; i += h)
        Console.WriteLine("f({0:f2})={1:f4}", i, f(i));
      }
   }
}
```

1. $y=\begin{cases}\dfrac{1}{(0.1+x)^2},\ \text{если}\ x\ge 0.9;\\ 0.2x+0.1,\ \text{если} 0\le x<0.9;\\ x^2+0.2,\ \text{если}\ x<0\end{cases}$

2. $y=\begin{cases}\sin(x),\text{если}\,|x|<3;\\ \dfrac{\sqrt{x^2+1}}{\sqrt{x^2+5}},\text{если}\,3\le|x|<9;\\ \sqrt{x^2+1}-\sqrt{x^2+5},\ \text{если}\,|x|\ge 9.\end{cases}$

3. $y=\begin{cases}0,\ \text{если}\,x<a;\\ \dfrac{x-a}{x+a},\ \text{если}\,x>a;\\ 1,\ \text{если}\,x=a.\end{cases}$

4. $y=\begin{cases}x^3-0.1,\ \text{если}\ |x|\le 0.1;\\ 0.2x-0.1,\ \text{если}\ 0.1<|x|\le 0.2;\\ x^3+0.1,\ \text{если}\ |x|>0.2.\end{cases}$

5. $y=\begin{cases}a+b,\ \text{если}\ x^2-5x<0;\\ a-b,\ \text{если}\ 0\le(x^2-5x)<10;\\ ab,\ \textit{если}\ x^2-5x\ge 10.\end{cases}$

6. $y=\begin{cases}x^2,\ \text{если}\left(x^2+2x+1\right)<2;\\ \dfrac{1}{x^2-1},\text{если}\ 2\le\left(x^2+2x+1\right)<3;\\ 0,\ \text{если}\left(x^2+2x+1\right)\ge 3.\end{cases}$

7. $y=\begin{cases}-4,\text{если}\,x<0;\\ x^2+3x+4,\text{если}\,0\le x<1;\\ 2,\ \text{если}\ x\ge 1.\end{cases}$

8. $y=\begin{cases}x^2-1,\ \text{если}\,|x|\le 1;\\ 2x-1,\ \text{если}\,1<|x|\le 2;\\ x^5-1,\ \text{если}\,|x|>2.\end{cases}$

9. $y=\begin{cases}\left(x^2-1\right)^2,\ \text{если}\,x<1;\\ \dfrac{1}{(1+x)^2},\ \ \text{если}\,x>1;\\ 0,\ \text{если}\,x=1.\end{cases}$

10. $y=\begin{cases}x^2,\ \text{если}\ (x+2)\le 1;\\ \dfrac{1}{x+2},\text{если}\ 1<(x+2)<10;\\ x+2,\ \text{если}\ (x+2)\ge 10;\end{cases}$

11. $y=\begin{cases}x^2+5,\ \text{если}\,x\le 5;\\ 0,\text{если}\,5<x<20;\\ 1,\ \text{если}\,x\ge 20.\end{cases}$

12. $y=\begin{cases}0,\ \text{если}\,x<0;\\ x^2+1,\text{если}\,x\ge 0\ \text{и}\ x\ne 1;\\ 1,\text{если}\,x=1.\end{cases}$

13. $y=\begin{cases}1,\text{если}\,x=1\,\text{или}\,x=-1;\\ \dfrac{-1}{1-x},\text{если}\,x\ge 0\ \text{и}\ x\ne 1;\\ \dfrac{1}{1+x},\text{если}\,x<0\ \text{и}\,x\ne -1.\end{cases}$

14. $y=\begin{cases}0.2x^2-x-0.1,\ \text{если}\ x<0;\\ \dfrac{x^2}{x-0.1},\text{если}\,x>0\ \text{и}\,x\ne 0.1;\\ 0,\ \ \text{если}\,x=0.1.\end{cases}$

15. $y=\begin{cases}1,\ \ \text{если}\ (x\text{-}1)<1;\\ 0,\ \text{если}\ (x\text{-}1)=1;\\ -1,\text{если}\ (x\text{-}1)>1.\end{cases}$

16. $y=\begin{cases}x,\ \text{если}\ x>0;\\ 0,\ \text{если}\ -1\le x\le 0;\\ x^2,\ \text{если}\ x<-1.\end{cases}$

17. $y=\begin{cases} a+bx, \text{ если } x<93; \\ \text{b - ac, если } 93\le x\le 120; \\ \text{abx, если } x>120. \end{cases}$

18. $y=\begin{cases} x^2-0.3, \text{ если } y<3; \\ 0, \text{ если } 3\le x\le 5; \\ x^2+1, \text{ если } x>5. \end{cases}$

19. $y=\begin{cases} \sqrt{5x^2+5}, e\text{если} |x|<2; \\ \dfrac{|x|}{\sqrt{5x^2+5}}, \text{ если } 2\le |x|<10; \\ 0, \text{ если } |x|\ge 10. \end{cases}$

20. $y=\begin{cases} \sin(x), \text{ если } |x|<\dfrac{\pi}{2}; \\ \cos(x), \text{ если } \dfrac{\pi}{2}\le |x|\le \pi; \\ 0, \text{ если } |x|>\pi. \end{cases}$

Часть 3. Перегрузите метод f из предыдущего раздела так, чтобы его сигнатура (заголовок) соответствовала виду static void f (double x, out double y). Продемонстрируйте работу перегруженных методов.

Задания для самостоятельной работы к лабораторной работе №1

Часть 1. Изучить теоретический материал

Пусть $a_1, a_2, \ldots, a_n$ - произвольная числовая последовательность. Рекуррентным соотношением называется такое соотношение между членами последовательности, в котором каждый следующий член выражается через несколько предыдущих, т.е $a_k = f(a_{k-1}, a_{k-2}, \ldots)$ (1).

Последовательность задана рекуррентно, если для нее определено рекуррентное соотношение вида (1) и заданы первые *l* ее членов.

Самым простым примером рекуррентной последовательности является арифметическая прогрессия. Рекуррентное соотношение для нее записывается в виде: $a_k = a_{k-1} + d$, где *d* – разность прогрессии. Зная первый элемент и разность прогрессии, и, используя данное рекуррентное соотношение, можно последовательно вычислить все остальные члены прогрессии.

Рассмотрим пример программы, в которой вычисляются первые n членов арифметической прогрессии при условии, что $a_1 = \frac{1}{2}$ и $d = \frac{1}{4}$.

```
static void Main()
    {
       Console.Write("a=");
       double a = double.Parse(Console.ReadLine());
       Console.Write("h=");
       double d = double.Parse(Console.ReadLine());
       Console.Write("n=");
       int d = int.Parse(Console.ReadLine());
       Console.WriteLine("a1="+ a);          //вывели первый член последовательности
      //организуем вычисление 2, 3, … ,n члена последовательности
        for (int i = 2; i <= n; ++i)
       {
          a += d;                  //для этого прибавляем к предыдущему члену значение d
          Console.WriteLine("a{0}={1}", i, a);      //и выводим новое значение a на экран
       }
    }
```

Результат работы программы: n состояние экрана

5	a1: 0.5 a2: 0.75 a3: 1. a4: 1.25 a5: 1.5

Более сложная зависимость представлена в последовательности Фибоначчи: $a_1 = a_2 = 1$, $a_i = a_{i-1} + a_{i-2}$. В этом случае каждый член последовательности зависит от значений двух предыдущих членов. Рассмотрим пример программы, в которой вычисляются первые n членов последовательности Фибоначчи.

```
static void Main()
 {
  int a1=1, a2=1, a3; //задали известные члены последовательности
  Console.Write("n=");
  int n = int.Parse(Console.ReadLine());
  //вывели известные члены  последовательности
   Console.WriteLine("a1={0}\na2={1}",a1,a2);
   /*Организуем цикл для вычисления членов последовательности с номерами 3, 4,…, n. При
   этом в переменной a1 будет храниться значение члена последовательности с номером i-2, в
   переменной a2 -  члена с номером i-1, переменная a будет использоваться для вычисления
   члена с номером i. */
  for (int i = 3; i <= n; ++i)
  {
   a3=a1+a2; //по рекуррентному соотношению вычисляем член последовательности
   Console.WriteLine("a{0}={1}", i, a3); //с номером i и выводим его значение на экран
   //выполняем рекуррентный пересчет для следующего шага цикла
   a1 = a2;       //в элемент с номером i-2 записываем значение элемента с номером i-1
   a2 = a3;      //в элемент с номером i-1 записываем значение элемента с номером i
  }
 }
```

Результат работы программы:

n	состояние экрана
5	a1: 1 a2: 1 a3: 2 a4: 3 a5: 5

Часть 2. Выполнить практическое задание

I. Написать программу, вычисляющую первые n элементов заданной последовательности:

1. $b_1 = 1, \quad b_i = b_{i-1} + 1$
2. $b_1 = 5, \quad b_i = \frac{b_{i-1}}{i+1}$
3. $b_1 = 1, \quad b_2 = 1, \quad b_i = b_{i-1} - b_{i-2}$
4. $b_1 = 1, \quad b_i = \frac{i \cdot b_{i-1}}{i+1}$

II. Вычислить и вывести на экран значение n члена последовательности для каждого $x \in [a, b]$ с шагом $h=0.1$ Результат работы программы представить в виде следующей таблицы:

№	Значение x	Значение функции $b_n(x)$
1		
2		

…		

Замечание. Для решения задачи разработать метод, в который передаются значения x и n, и которым возвращается значение b_n.

1) $b_1 = x, b_n = x + 2b_{n-1}$;
2) $b_1 = x, b_n = \sin(b_{n-1}) + \pi$;
3) $b_1 = 0, \quad b_{2n} = b_{2n-1} + x, \quad b_{2n+1} = 2b_{2n}$;
4) $b_1 = x, \quad b_2 = 2x, \quad b_n = \frac{b_{n-2}}{4} + \frac{5}{b_{n-1}^2}$.

Лабораторное занятие №2.

Цель работы: освоить особенности работы с прямой и косвенной рекурсией, рекурсивными методами, возвращающими и не возвращающими значение.

Программное обеспечение и материалы: актуальные версии Microsoft Windows, MS Visual Studio.

Задания

I. Разработать рекурсивный метод (возвращающий значение):

1. для вычисления n-го члена следующей последовательности $b_1 = -10, \quad b_2 = 2, \quad b_{n+2} = |b_n| - 6b_{n+1}$.
2. для вычисления n-го члена следующей последовательности $b_1 = 5, \quad b_{n+1} = \frac{b_n}{n^2 + n + 1}$.
3. для нахождения наибольшего общего делителя методом Евклида:
$$НОД(a,b) = \begin{cases} a, если\ a = b; \\ НОД(a-b,b), если\ a > b; \\ НОД(a,b-a), если\ b > a. \end{cases}$$
4. для вычисления значения функции Аккермана для неотрицательных чисел n и m. Функция Аккермана определяется следующим образом:
$$A(n,m) = \begin{cases} m+1, \text{ если } n = 0; \\ A(n-1,1), \text{ если } n \neq 0, m = 0; \\ A(n-1, A(n,m-1)), \text{ если } n > 0, m > 0. \end{cases}$$
5. для вычисления числа сочетаний *C(n, m)* где $0 \le m \le n$, используя следующие свойства $C_n^0 = C_n^n = 1; \ C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ при *0<m<n*.
6. вычисляющий число *a*, для которого выполняется неравенство $2^{a-1} \le n \le 2^a$, где n – натуральное число. Для подсчета числа a использовать формулу: $a(n) = \begin{cases} 1, n = 1; \\ a(n/2) + 1, n > 1. \end{cases}$
7. для вычисления x^n (*x*–вещественное, $x \neq 0$, а *n*–целое) по формуле:
$$x^n = \begin{cases} 1 & при\ n = 0, \\ 1/x^{|n|} & при\ n < 0, \\ x \cdot x^{n-1} & при\ n > 0. \end{cases}$$
. Вычислить значение x^n для различных x и n.
8. для вычисления $\sum_{i=1}^{n} i$, где n – натуральное число. Для заданных натуральных чисел m и k вычислить с помощью разработанного метода значение выражения $\sum_{i=1}^{m} i + \sum_{i=1}^{2k} i$.

9. для вычисления значения функции $F(N)=\frac{N}{\sqrt{1+\sqrt{2+\sqrt{3+...\sqrt{N}}}}}$.

   Найти ее значение при заданном натуральном N.

10. для вычисления цепной дроби: $\cfrac{x}{1+\cfrac{x}{2+\cfrac{x}{3+...\cfrac{x}{n+x}}}}$. Найти значение данной дроби при заданном натуральном n.

II. Разработать рекурсивный метод (не возвращающий значения):

1. Даны первый член и разность арифметической прогрессии. Написать рекурсивный метод для нахождения n-го члена и суммы n первых членов прогрессии.
2. Даны первый член и знаменатель геометрической прогрессии. Написать рекурсивный метод для нахождения n-го члена и суммы n первых членов прогрессии.
3. Разработать рекурсивный метод, который по заданному натуральному числу N (N≥1000) выведет на экран все натуральные числа не больше N в порядке возрастания. Например, для N=8, на экран выводится 1 2 3 4 5 6 7 8.
4. Разработать рекурсивный метод, который по заданному натуральному числу N (N≥1000) выведет на экран все натуральные числа не больше N в порядке убывания. Например, для N=8, на экран выводится 8 7 6 5 4 3 2 1.
5. Разработать рекурсивный метод для вывода на экран стихотворения:

   10 лунатиков жили на луне
   10 лунатиков ворочались во сне
   Один из лунатиков упал с луны во сне
   9 лунатиков осталось на луне
   9 лунатиков жили на луне
   9 лунатиков ворочались во сне
   Один из лунатиков упал с луны во сне
   8 лунатиков осталось на луне
   …...
   И больше лунатиков не стало на луне

6. Дано натуральное число n. Разработать рекурсивный метод для вывода на экран следующей последовательности чисел:

   1
   2 2
   3 3 3
   …
   n n n … n

7. Дано натуральное число n. Разработать рекурсивный метод для вывода на экран следующей последовательности чисел:

   1
   2 1
   3 2 1
   …
   n n-1 n-2 … 1

8. Разработать рекурсивный метод для вывода на экран цифр натурального числа в прямом порядке. Применить эту процедуру ко всем числам из интервала от A до B.
9. Разработать рекурсивный метод для перевода числа из десятичной системы счисления в двоичную.
10. Разработать рекурсивный метод для перевода числа из двоичной системы счисления в десятичную.
11. Разработать рекурсивный метод для вывода на экран всех делителей заданного натурального числа n.

12. Дано натуральное четное число n. Разработать рекурсивный метод для вывода на экран следующей картинки:

*********	(0 пробелов, n звездочек)
********	(1 пробел, n-1 звездочка)
*******	(2 пробела, n-2 звездочки)
…	
*	(n-1 пробел, 1 звездочка)

13. Дано натуральное четное число n. Разработать рекурсивный метод для вывода на экран следующей картинки:

* *	(n пробелов между звездочками)
** **	(n-2 пробела)
*** ***	(n-4 пробела)
…	…
***** *****	(2 пробела)
**********	(0 пробелов)
***** *****	(2 пробела)
…	…
*** ***	(n-4 пробела)
** **	(n-2 пробела)
* *	(n пробелов

14. Дано натуральное число n. Разработать рекурсивный метод для вывода на экран следующей картинки:

1	(1 раз)
222	(3 раза)
33333	(5 раз)
…	(n раз)
33333	(5 раз)
222	(3 раза)
1	(1 раз)

15. Разработать рекурсивный метод для вывода на экран следующей картинки:

AAAAAAAAAA…AAAAAAAAAA	(80 раз)
BBBBBBBBB…BBBBBBBB	(78 раз)
CCCCCCCC…CCCCCCCC	(76 раз)
…	…
YYY…YYY	(32 раза)
ZZ…ZZ	(30 раз)
YYY…YYY	(32 раза)
…	…
CCCCCCCC…CCCCCCCC	(76 раз)
BBBBBBBBB…BBBBBBBB	(78 раз)
AAAAAAAAAA…AAAAAAAAAA	(80 раз)

Задания для самостоятельной работы к лабораторной работе №2

Задача 1. Разработать рекурсивный метод для вывода на экран всех возможных разложений натурального числа n на множители (без повторений). Например, для n=12 на экран должно быть выведено:

2*2*3=12

2*6=12

3*4=12

Задача 2. Разработать рекурсивный метод для вывода на экран всех возможных разложений натурального числа n на слагаемые (без повторений). Например, для n=5 на экран должно быть выведено:

1+1+1+1+1=5

1+1+1+2=5

1+1+3=5

1+4=5
2+1+2=5
2+3=5

Лабораторное занятие №3.

Цель работы: освоить особенности работы с исключениями: операторы try, checked и unchecked, научиться генерировать собственные исключения.

Программное обеспечение и материалы: актуальные версии Microsoft Windows, MS Visual Studio.

Задания

Постройте таблицу значений функции y=f(x) для $x \in [a, b]$ с шагом h. Если в некоторой точке *x* функция не определена, то выведите на экран сообщение об этом.

Замечание. При решении данной задачи использовать вспомогательный метод f(x), реализующий заданную функцию, а также проводить обработку возможных исключений.

1. $y = \frac{1}{(1+x)^2}$

Пример:

```
using System;
namespace Hello
{
   class Program
   {
      static double f(double x)
      {
         try
         {
            //если x не попадает в область определения, то генерируется исключение
           if (x == -1) throw new Exception();
            else return 1 / Math.Pow(1 + x, 2);
         }
         catch
         {
            throw;
         }
      }
      static void Main(string[] args)
      {
         try
         {
            Console.Write("a=");
            double a = double.Parse(Console.ReadLine());
            Console.Write("b=");
            double b = double.Parse(Console.ReadLine());
            Console.Write("h=");
            double h = double.Parse(Console.ReadLine());
            for (double i = a; i <= b; i += h)
               try
               {
                  Console.WriteLine("y({0})={1:f4}", i, f(i));
```

```
                }
                catch
                {
                   Console.WriteLine("y({0})=error", i);
                }
            }
            catch (FormatException)
            {
               Console.WriteLine("Неверный формат ввода данных");
            }
            catch
            {
               Console.WriteLine("Неизвестная ошибка");
            }
        }
    }
}
```

2. $y=\frac{1}{x^2-1}$;
3. $y=\sqrt{x^2-1}$;
4. $y=\sqrt{5-x^3}$;
5. $y=\ln(x-1)$;
6. $y=\ln(4-x^2)$;
7. $y=\frac{x}{\sqrt{2x-1}}$;
8. $y=\frac{3x+4}{\sqrt{x^2+2x+1}}$;
9. $y=\frac{1}{x-1}+\frac{2}{1-4}$;
10. $y=\ln|x-2|$;
11. $y=\ln\frac{x}{x-2}$;
12. $y=\ln(x^4-1)(\ln x)$;
13. $y=\frac{\ln(x-2)}{\sqrt{5x+1}}$;
14. $y=\frac{\sqrt{x^2-2x+1}}{\ln(4-2x)}$;
15. $y=\ln|5+\sqrt{2x^5-3}|$;
16. $y=\frac{3}{|x^3+8|}$;
17. $y=\frac{x+4}{x^2-2}+\sqrt{x^3-3}$;
18. $y=\sqrt{x^2+1}+\sqrt{x^2+5}$;
19. $y=\frac{\sqrt{x^3-1}}{\sqrt{x^2-1}}$;
20. $y=\frac{1}{x+7}+\ln(1+x)$.

Задания для самостоятельной работы к лабораторной работе №3

Часть 1. Изучить теоретический материал

Вычисление конечных сумм и произведений

Решение многих задач связано с нахождением суммы или произведения элементов заданной последовательности. В данном разделе мы рассмотрим основные приемы вычисления конечных сумм и произведений.

Пусть $u_1(x), u_2(x), \ldots, u_n(x)$ - произвольная последовательность n функций. Будем рассматривать конечную сумму вида $u_1(x)+u_2(x)+\ldots+u_n(x)$. Такую сумму можно записать

более компактно, используя следующее обозначение:  . При $n \le 0$ значение суммы равно 0.

В дальнейшем будем также использовать сокращенную запись для конечного произведения данной последовательности, которая выглядит следующим образом: $u_1(x) \cdot u_2(x) \cdot \ldots \cdot u_n(x) = \prod_{i=1}^{n} u_i(x)$.

5) Написать программу, которая подсчитывает сумму натуральных чисел от 1 до n (n≥1).

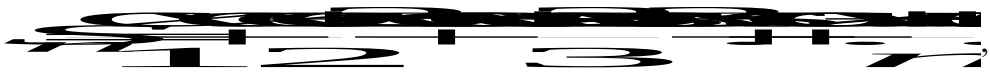
Указания по решению задачи. Пусть s_n - сумма натуральных чисел от 1 до n. Тогда $s_n=1+2+\ldots+(n-1)+n=(1+2+\ldots+(n-1))+n=s_{n-1}+n$, $s_0=0$. Мы пришли к рекуррентному соотношению $s_0=0$, $s_n=s_{n-1}+n$, которым мы можем воспользоваться для подсчета суммы. Соотношение $s_n=s_{n-1}+n$ говорит о том, что сумма на n-ном шаге равна сумме, полученной на предыдущем шаге, плюс очередное слагаемое.

```
static void Main()
        {
                Console.Write("Ввведите значение n: ");
                int n=int.Parse(Console.ReadLine());
                int s=0;
                for (int i=1; i<=n; ++i)
                        s+=i;
                Console.WriteLine("s="+s);
        }
```

6) Написать программу, которая подсчитывает n! для вещественного x и натурального n.

Указание по решению задачи. Из свойства факториала 0!=1!=1, n!=1*2*3*…*n, n!=(n-1)!n. Следовательно, факториал можно вычислять, используя рекуррентное соотношение $b_0=1$, $b_n=b_{n-1}*n$.

```
static void Main()
        {
                Console.Write("Ввведите значение n: ");
                int n=int.Parse(Console.ReadLine());
                int f=1;
                for (int i=1; i<=n; ++i)
                        f*=i;
                Console.WriteLine("{0}!={1}", n, f);
        }
```

7) Написать программу для подсчета суммы  , где x – вещественное число, n – натуральное число.

Указания по решению задачи. Если пронумеровать слагаемые, начиная с 1, то мы увидим, что номер слагаемого совпадает со значением знаменателя. Рассмотрим каждый числитель отдельно: $b_1=\cos x$,  ,  … Эту последовательность можно представить рекуррентным соотношением $b_0=0$, $b_n=b_{n-1}+\cos nx$ *(1)*. Теперь сумму можно представить следующим образом , а для нее справедливо рекуррентное соотношение $S_0=0$, $S_n=S_{n-1}+\frac{b_n}{n}$ *(2)*. При составлении программы будем использовать формулы (1-2).

```
static void Main()
```

```
        {
            Console.Write("Ввведите значение n: ");
            int n=int.Parse(Console.ReadLine());
            Console.Write("Ввведите значение x: ");
            double x=double.Parse(Console.ReadLine());
            double b=0, s=0;
            for (int i=1; i<=n; ++i)
            {
                b+=Math.Cos(i*x);
                s+=b/i;
            }
            Console.WriteLine("s={0:f2}",s);
        }
```

8) Написать программу для подсчета суммы $S_n=\sum_{i=1}^{n}\frac{(-1)^{i+1}x^i}{i!}$, где x – вещественное число, n – натуральное число.

Указания по решению задачи. Перейдем от сокращенной формы записи к развернутой, получим $S_n=\frac{x}{1}-\frac{x^2}{2}+\frac{x^3}{3}-\ldots+\frac{(-1)^{n+1}x^n}{n}$. Каждое слагаемое формируется по формуле $q_n=\frac{(-1)^{n+1}x^n}{n}$. Если в эту формулу подставить n=0, то получим $q_0=\frac{(-1)^1x^0}{0}=-1$.

Чтобы не вводить несколько рекуррентных соотношений (отдельно для числителя, отдельно для знаменателя), представим общий член последовательности слагаемых с помощью рекуррентного соотношением вида $a_n=a_{n-1}q$, где q для нас пока не известно. Найти его можно из выражения $q=\frac{a_n}{a_{n-1}}$. Произведя необходимые расчеты, получим, что $q=-\frac{x}{i}$. Следовательно, для последовательности слагаемых мы получили рекуррентное соотношение $a_0=-1$, $a_i=-a_{i-1}\cdot\frac{x}{i}$ *(3).* А всю сумму, по аналогии с предыдущими примерами, можно представить рекуррентным соотношением: $S_0=0$, $S_n=S_{n-1}+a_i$ *(4).* Таким образом, при составлении программы будем пользоваться формулами (3-4).

```
using System;
namespace Hello
{
      class Program
      {
       static void Main()
            {
                Console.Write("Ввведите значение n: ");
                int n=int.Parse(Console.ReadLine());
                Console.Write("Ввведите значение x: ");
                double x=double.Parse(Console.ReadLine());
                double a=-1, s=0;
                for (int i=1; i<=n; ++i)
                {
                    a*=-x/i;  s+=a;
                }
                Console.WriteLine("s={0:f2}",s);
            }
    }
}
```

Вычисление бесконечных сумм

Будем теперь рассматривать бесконечную сумму вида  . Это выражение называется функциональным рядом. При различных значениях *x* из функционального ряда получаются различные числовые ряды  . Числовой ряд может быть сходящимся или расходящимся. Совокупность значений *x*, при которой функциональный ряд сходится, называется его областью сходимости.

Числовой ряд называется сходящимся, если сумма n первых его членов  при $n \to \infty$ имеет предел, в противном случае, ряд называется расходящимся. Ряд может сходиться лишь при условии, что общий член ряда a_n при неограниченном увеличении его номера стремится к нулю: $\lim_{n\to\infty} a_n = 0$. Это необходимый признак сходимости для всякого ряда.

В случае бесконечной суммы будем вычислять ее с заданной точностью *e*. Считается, что требуемая точность достигается, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше чем *e*, то есть это слагаемое на результат практически не влияет. Тогда его и все последующие слагаемые можно не учитывать.

Пример. Написать программу для подсчета суммы $\sum_{i=1}^{\infty} \frac{(-1)^i}{i!}$ с заданной точностью e (e>0).

Указание по решению задачи. Рассмотрим, что представляет из себя заданный ряд:  . Как видим, общий член ряда с увеличением значения i стремится к нулю. Поэтому данную сумму можно вычислить, но только с определенной точностью *e*. Заметим также, что последовательность слагаемых можно выразить с помощью рекуррентного соотношения $a_1=-1$, $a_i = \frac{-a_{i-1}}{i}$, а всю сумму - с помощью рекуррентного соотношения $S_0=0$, $S_n=S_{n-1}+a_n$. (Данные рекуррентные соотношения выведите самостоятельно.)

```
using System;
namespace Hello
{
   class Program
        {
                static void Main()
                {
                        Console.Write("Задайте точность вычислений e: ");
                        double e=double.Parse(Console.ReadLine());
                        double a=-1, s=0;
                        for (int i=2; Math.Abs(a)>=e; ++i)
                        {
                                s+=a;   a/=-i;
                        }
                        Console.WriteLine("s={0:f2}",s);
                }
        }
}
```

Часть 2. Выполнить практическое задание

Замечание. При решении задач производить обработку следующих исключительных ситуаций: ввода пользователем недопустимых значений и переполнения при вычислении математических выражений.

I. Для заданного натурального *n* и действительного *x* подсчитать следующие суммы:

1) $S=1+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{3}}+\ldots+\frac{1}{\sqrt{i}};$

2) 

3) S = 1!+2!+3!+…+n!; 4) $S=1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}.$

II. Для заданного натурального *k* и действительного *x* подсчитать следующие выражения:

1) $S=\sum_{n=1}^{k}\frac{x^n}{n}$ 2) $S=\sum_{n=1}^{k}\frac{(-1)^{n+1}x^{2n+1}}{(2n+1)!}$

3) $P=\prod_{n=1}^{k}(1+\frac{x^{2n+1}}{n(n+1)})$ 4) $P=\prod_{n=2}^{k}(1+\frac{(-1)^n x^{2n-1}}{n^3-1})$

III. Вычислить бесконечную сумму ряда с заданной точностью e (e>0).

1) $\sum_{i=1}^{\infty}\frac{1}{i^2}$ 2) $\sum_{i=1}^{\infty}\frac{1}{3^i+4^i}$ 3) $\sum\frac{(-1)^i}{(2i-1)!}$ 4) $\sum_{i=1}^{\infty}\frac{(-1)^{i+1}}{3^{2i-1}}$

IV. Вычислить и вывести на экран значение функции F(x) на отрезке *[a,b]* с шагом *h=0.1* с точностью *e*. Результат работы программы представить в виде следующей таблицы:

№	Значение x	Значение функции F(x)	Количество просуммированных слагаемых n
1			
2			
…			

Замечание. При решении задачи использовать вспомогательную функцию.

1. F(x) = $1+\frac{x^2}{4}+\frac{x^3}{4}+\frac{x^4}{4}+\frac{x^5}{4}+\ldots$, $x\in[0.1; 0.9]$.
2. F(x) = $1+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\ldots$, $x\in[0; 0.99]$.
3. F(x) = $1-\frac{x^2}{3}+\frac{x^4}{5}-\frac{x^6}{7}+\frac{x^8}{9}-\ldots$, $x\in[0, 1]$.
4. F(x) = $\frac{x+1}{x}+\frac{(x+2)^2}{2x^2}+\frac{(x+3)^3}{3x^3}+\ldots$, $x\in[1; 2]$.

Лабораторное занятие №4.

Цель работы: освоить особенности работы массивами: одномерными, двумерными, ступенчатыми.

Программное обеспечение и материалы: актуальные версии Microsoft Windows, MS Visual Studio.

Задания

I. Дана последовательность целых чисел.

Замечание. Задачи из данного пункта решить двумя способами, используя одномерный массив, а затем двумерный. Размерность массива вводится с клавиатуры.

1. Заменить все положительные элементы противоположными им числами.

Пример 1: для одномерного массива

```
using System;
namespace ConsoleApplication2
{
	class Class
	{
		static int [] Input ()
		{
			Console.WriteLine("введите размерность массива");
			int n=int.Parse(Console.ReadLine());
			int []a=new int[n];
			for (int i = 0; i < n; ++i)
			{
				Console.Write("a[{0}]= ", i);
				a[i]=int.Parse(Console.ReadLine());
			}
			return a;
		}
		static void Print(int[] a)
		{
			for (int i = 0; i < a.Length; ++i) Console.Write("{0} ", a[i]);
			Console.WriteLine();
		}
		static void Change(int[] a)
		{
			for (int i = 0; i < a.Length; ++i)
				if (a[i] > 0) a[i] = -a[i];
		}
		static void Main()
		{
			int[] myArray=Input();
			Console.WriteLine("Исходный массив:");
			Print(myArray);
			Change(myArray);
			Console.WriteLine("Измененный массив:");
			Print(myArray);
		}
	}
}
```

Пример 2: для двумерного массива

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [,] Input (out int n, out int m)
		{
			Console.WriteLine("введите размерность массива");
			Console.Write("n = ");
			n=int.Parse(Console.ReadLine());
			Console.Write("m = ");
			m=int.Parse(Console.ReadLine());
```

```
            int [,]a=new int[n, m];
            for (int i = 0; i < n; ++i)
                for (int j = 0; j < m; ++j)
                {
                    Console.Write("a[{0},{1}]= ", i, j);
                    a[i, j]=int.Parse(Console.ReadLine());
                }
            return a;
        }
        static void Print(int[,] a)
        {
            for (int i = 0; i < a.GetLength(0); ++i,Console.WriteLine() )
                for (int j = 0; j < a.GetLength(1); ++j)
                    Console.Write("{0,5} ", a[i, j]);
        }
        static void Change(int[,] a)
        {
            for (int i = 0; i < a.GetLength(0); ++i)
                for (int j = 0; j < a.GetLength(1); ++j)
                    if (a[i, j] > 0) a[i, j] = -a[i, j];
        }
        static void Main()
        {
            int n,m;
            int[,] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray);
            Change(myArray);
            Console.WriteLine("Измененный массив:");
            Print(myArray);
        }
    }
}
```

2. Заменить все элементы, меньшие заданного числа, этим числом.
3. Заменить все элементы, попадающие в интервал [a, b], нулем.
4. Заменить все отрицательные элементы, не кратные 3, противоположными им числами.
5. Все элементы, меньшие заданного числа, увеличить в два раза.
6. Подсчитать среднее арифметическое элементов.
7. Подсчитать среднее арифметическое отрицательных элементов.
8. Подсчитать количество нечетных элементов.
9. Подсчитать сумму элементов, попадающих в заданный интервал.
10. Подсчитать сумму элементов, кратных 9.
11. Подсчитать количество элементов, не попадающих в заданный интервал.
12. Подсчитать сумму квадратов четных элементов.
13. Вывести на экран номера всех элементов больших заданного числа.
14. Вывести на экран номера всех нечетных элементов.
15. Вывести на экран номера всех элементов, которые не делятся на 7.
16. Вывести на экран номера всех элементов, не попадающих в заданный интервал.
17. Определить, является ли произведение элементов трехзначным числом.
18. Определить, является ли сумма элементов двухзначным числом.
19. Вывести на экран элементы с четными индексами (для двумерного массива – сумма индексов должна быть четной).

20. Вывести на экран положительные элементы с нечетными индексами (для двумерного массива – первый индекс должен быть нечетным).

II. Дана последовательность из n действительных чисел.

Замечание. Задачи из данного пункта решить, используя одномерный массив.

1. Подсчитать количество максимальных элементов.

Пример.

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [] Input ()
		{
			Console.WriteLine("введите размерность массива");
			int n=int.Parse(Console.ReadLine());
			int []a=new int[n];
			for (int i = 0; i < n; ++i)
			{
				Console.Write("a[{0}]= ", i);
				a[i]=int.Parse(Console.ReadLine());
			}
			return a;
		}
		static int Max(int[] a)
		{
			int max=a[0];
			for (int i = 1; i < a.Length; ++i)
				if (a[i] > max) max=a[i];
			return max;
		}
		static void Main()
		{
			int[] myArray=Input();
			int max=Max(myArray);
			int kol=0;
			for (int i=0; i<myArray.Length;++i)
				if (myArray[i]==max)++kol;
			Console.WriteLine("Количество максимальных элементов = "+kol);
		}
	}
}
```

2. Вывести на экран номера всех минимальных элементов.
3. Заменить все максимальные элементы нулями.
4. Заменить все минимальные элементы на противоположные.
5. Поменять местами максимальный элемент и первый.
6. Вывести на экран номера всех элементов, не совпадающих с максимальным.
7. Найти номер первого минимального элемента.
8. Найти номер последнего максимального элемента.
9. Подсчитать сумму элементов, расположенных между максимальным и минимальным элементами (минимальный и максимальный элементы в массиве единственные). Если максимальный элемент встречается позже минимального, то выдать сообщение об этом.

10. Найти номер первого максимального элемента.
11. Найти номер последнего минимального элемента.
12. Подсчитать сумму элементов, расположенных между первым максимальным и последним минимальными элементами. Если максимальный элемент встречается позже минимального, то выдать сообщение об этом.
13. Поменять местами первый минимальный и последний максимальный элементы.
14. Найти максимум из отрицательных элементов.
15. Найти минимум из положительных элементов.
16. Найти максимум из модулей элементов.
17. Найти количество пар соседних элементов, разность между которыми равна заданному числу.
18. Подсчитать количество элементов, значения которых больше значения предыдущего элемента.
19. Найти количество пар соседних элементов, в которых предыдущий элемент кратен последующему.
20. Найти количество пар соседних элементов, в которых предыдущий элемент меньше последующего.

III. Дан массив размером n×n, элементы которого целые числа.

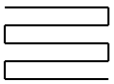
Замечание. При решении задач из данного пункта использовать двумерный массив.

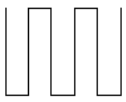
1. Подсчитать среднее арифметическое нечетных элементов, расположенных выше главной диагонали.

Пример.

```
using System;
namespace ConsoleApplication
{
    class Class
    {
        static int [,] Input (out int n)
        {
            Console.WriteLine("введите размерность массива");
            Console.Write("n = ");
            n=int.Parse(Console.ReadLine());
            int [,]a=new int[n, n];
            for (int i = 0; i < n; ++i)
                for (int j = 0; j < n; ++j)
                {
                    Console.Write("a[{0},{1}]= ", i, j);
                    a[i, j]=int.Parse(Console.ReadLine());
                }
            return a;
        }
        static void Print(int[,] a)
        {
            for (int i = 0; i < a.GetLength(0); ++i,Console.WriteLine() )
                for (int j = 0; j < a.GetLength(1); ++j)
                    Console.Write("{0,5} ", a[i, j]);
        }
        static double Rezalt(int[,] a)
        {
            int k=0;
            double s=0;
            for (int i = 0; i < a.GetLength(0); ++i)
```

```
                        for (int j = i+1; j < a.GetLength(1); ++j)
                            if (a[i, j] %2!= 0) {++k; s+=a[i, j];}
                    if (k!=0) return s/k;
                    else return 0;
                }
                static void Main()
                {
                    int n;
                    int[,] myArray=Input(out n);
                    Console.WriteLine("Исходный массив:");
                    Print(myArray);
                    double rez=Rezalt(myArray);
                    Console.WriteLine("Среднее арифметическое ={0:f2}", rez);
                }
            }
}
```

2. Подсчитать среднее арифметическое четных элементов, расположенных ниже главной диагонали.
3. Подсчитать сумму элементов, расположенных на побочной диагонали.
4. Подсчитать среднее арифметическое ненулевых элементов, расположенных над побочной диагональю.
5. Подсчитать среднее арифметическое элементов, расположенных под побочной диагональю.
6. Поменять местами столбцы по правилу: первый с последним, второй с предпоследним и т.д.
7. Поменять местами две средних строки, если количество строк четное, и первую со средней строкой, если количество строк нечетное.
8. Поменять местами два средних столбца, если количество столбцов четное, и первый со средним столбцом, если количество столбцов нечетное.
9. Если количество строк в массиве четное, то поменять строки местами по правилу: первую строку со второй, третью – с четвертой и т.д. Если количество строк в массиве нечетное, то оставить массив без изменений.
10. Если количество столбцов в массиве четное, то поменять столбцы местами по правилу: первый столбец со вторым, третий – с четвертым и т.д. Если количество столбцов в массиве нечетное, то оставить массив без изменений.
11. Вычислить A^n, где n – натуральное число.
12. Подсчитать норму матрицы по формуле $\|A\| = \sum_i \max_j a_{i,j}$.
13. Подсчитать норму матрицы по формуле $\|A\| = \sum_j \max_i a_{i,j}$.
14. Вывести элементы матрицы в следующем порядке: 
15. Выяснить, является ли матрица симметричной относительно главной диагонали.
16. Заполнить матрицу числами от 1 до n (где n=m×k, а m – количесто строк, а k – количество столбцов прямоугольной матрицы) следующим образом:



17. Определить, есть ли в данном массиве строка, состоящая только из положительных элементов.
18. Определить, есть ли в данном массиве столбец, состоящий только из отрицательных

элементов.

19. В каждой строке найти максимум и заменить его на противоположный элемент.
20. В каждом столбце найти минимум и заменить его нулем.

IV. Дан массив размером n×n, элементы которого целые числа.

Замечание. Для хранения массив n×n использовать ступенчатый массив.

1. Найти максимальный элемент в каждой строке и записать данные в новый массив.

Пример

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [][] Input ()
		{
			Console.WriteLine("введите размерность массива");
			Console.Write("n = ");
			int n=int.Parse(Console.ReadLine());
			int [][]a=new int[n][];
			for (int i = 0; i < n; ++i)
			{
				a[i]=new int [n];
				for (int j = 0; j < n; ++j)
				{
					Console.Write("a[{0},{1}]= ", i, j);
					a[i][j]=int.Parse(Console.ReadLine());
				}
			}
			return a;
		}
		static void Print1(int[] a)
		{
			for (int i = 0; i < a.Length; ++i)
				Console.Write("{0,5} ", a[i]);
		}
		static void Print2(int[][] a)
		{
			for (int i = 0; i < a.Length; ++i,Console.WriteLine() )
				for (int j = 0; j < a[i].Length; ++j)
					Console.Write("{0,5} ", a[i][j]);
		}
		static int Max(int[] a)
		{
			int max=a[0];
			for (int i = 1; i < a.Length; ++i)

				if (a[i] >max) {max=a[i];}
			return max;
		}
		static void Main()
		{
			int[][] myArray=Input();
```

```
                Console.WriteLine("Исходный массив:");
                Print2(myArray);
                int[]rez=new int [myArray.Length];
                for (int i=0;i<myArray.Length; ++i)
                        rez[i]=Max(myArray[i]);
                Console.WriteLine("Новый массив:");
                Print1(rez);
        }
    }
}
```

2. Найти минимальный элемент в каждом столбце и записать данные в новый массив.
3. Четные столбцы таблицы заменить на вектор X.
4. Нечетные строки таблицы заменить на вектор X.
5. Вычислить A*X, где A – двумерная матрица, X – вектор.
6. Для каждой строки подсчитать количество положительных элементов и записать данные в новый массив.
7. Для каждого столбца подсчитать сумму отрицательных элементов и записать данные в новый массив.
8. Для каждого столбца подсчитать сумму четных положительных элементов и записать данные в новый массив.
9. Для каждой строки подсчитать количество элементов, больших заданного числа, и записать данные в новый массив.
10. Для каждого столбца найти первый положительный элемент и записать данные в новый массив.
11. Для каждой строки найти последний четный элемент и записать данные в новый массив.
12. Для каждого столбца найти номер последнего нечетного элемента и записать данные в новый массив.
13. Для каждой строки найти номер первого отрицательного элемента и записать данные в новый массив.
14. Для каждой строки найти сумму элементов с номерами от k1 до k2 и записать данные в новый массив.
15. Для каждого столбца найти произведение элементов с номерами от k1 до k2 и записать данные в новый массив.
16. Для каждой строки подсчитать сумму элементов, не попадающих в заданный интервал, и записать данные в новый массив.
17. Подсчитать сумму элементов каждой строки и записать данные в новый массив. Найти максимальный элемент нового массива.
18. Подсчитать произведение элементов каждого столбца и записать данные в новый массив. Найти минимальный элемент нового массива.
19. Для каждой строки найти номер первой пары неравных элементов. Данные записать в новый массив.
20. Для каждого столбца найти номер первой пары одинаковых элементов. Данные записать в новый массив.

Задания для самостоятельной работы к лабораторной работе №4

Часть 1. Изучить теоретический материал

При объявлении массива мы определяем его максимальную размерность, которая в дальнейшем изменена быть не может. Однако с помощью вспомогательной переменной можно контролировать текущее количество элементов, которое не может быть больше максимального.

Замечание. В пространстве имен System.Collection реализована коллекция ArrayList –

массив, динамически изменяющий свой размер. Мы будем рассматривать его позже.

Пример. Рассмотрим фрагмент программы:

```
int []a=new int [10];
int n=5;
for (int i=0; i<5;i++) a[i]:=i*i;
```

В этом случае массив можно представить следующим образом:

n=5	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
a	0	1	4	9	16	0	0	0	0	0

Так как во время описания был определен массив из 10 элементов, а заполнено только первые 5, то оставшиеся элементы будут заполнены нулями.

Что значит *удалить из одномерного массива* элемент с номером 3? Удаление должно привести к физическому «уничтожению» элемента с номером 3 из массива, при этом общее количество элементов должно быть уменьшено. В этом понимании удаления элемента итоговый массив должен выглядеть следующем образом

	0	*1*	*2*	*4*	*5*	*6*	*7*	*8*	*9*	*недопустимое*
a	0	1	4	16	0	0	0	0	0	*состояние*

Такое удаление для массивов *невозможно,* поскольку элементы массива располагаются в памяти последовательно друг за другом, что позволяет организовать индексный способ обращения к массиву.

Однако «удаление» можно смоделировать сдвигом элементов влево и уменьшением значения переменной, которая отвечает за текущее количество элементов в массиве, на единицу:

n=4	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
a	0	1	4	16	0	0	0	0	0	0

В общем случае, если мы хотим удалить элемент массива с номером *k* (всего в массиве *n* элементов, а последний элемент имеет индекс *n-1*), то нам необходимо произвести сдвиг элементов, начиная с *k+1*-го на одну позицию влево. Т.е. на *k*-ое место поставить *k+1*-й элемент, на место *k+1 – k+2*-й элемент, …, на место *n-2 – n-1*-й элемент. После чего значение *n* уменьшить на 1. В этом случае размерность массива не изменится, изменится лишь текущее количество элементов, и у нас создастся ощущение, что элемент с номером *k* удален. Рассмотрим данный алгоритм на примере:

```
using System;
namespace ConsoleApplication
{
        class Class
        {
                static int [] Input ()
                {
                        Console.WriteLine("введите размерность массива");
                        int n=int.Parse(Console.ReadLine());
                        int []a=new int[n];
                        for (int i = 0; i < n; ++i)
                        {
                                Console.Write("a[{0}]= ", i);
                                a[i]=int.Parse(Console.ReadLine());
                        }
                        return a;
                }

                static void Print(int[] a, int n)
                {
```

```
            for (int i = 0; i < n; ++i) Console.Write("{0} ", a[i]);
            Console.WriteLine();
        }

        static void DeleteArray(int[] a, ref int n, int m)
        {
            for (int i = m; i < n-1; ++i)
                a[i] = a[i+1];
            --n;
        }

        static void Main()
        {
            int[] myArray=Input();
            int n=myArray.Length;
            Console.WriteLine("Исходный массив:");
            Print(myArray, n);
            Console.WriteLine("Введите номер элемента для удаления:");
            int m=int.Parse(Console.ReadLine());
    DeleteArray(myArray, ref n,m);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n);
        }
    }
}
```

Задание. Подумайте, какие исключительные ситуации могут возникнуть в данной программе и добавьте в нее соответствующие обработки исключительных ситуаций

Рассмотрим теперь операцию *удаления в двумерном массиве*. Размерность двумерного массива также зафиксирована на этапе объявления массива. Однако при необходимости можно «смоделировать» удаление целой строки в массиве, выполняя сдвиг всех строк, начиная с k-той на единицу вверх. В этом случае размерность массива не изменится, а текущее количество строк будет уменьшено на единицу. В качестве примера удалим из двумерного массива, строку с номером k.

```
using System;
namespace ConsoleApplication
{
    class Class
    {
        static int [,] Input (out int n, out int m)
        {
            Console.WriteLine("введите размерность массива");
            Console.Write("n = ");
            n=int.Parse(Console.ReadLine());
            Console.Write("m = ");
            m=int.Parse(Console.ReadLine());
            int [,]a=new int[n, m];
            for (int i = 0; i < n; ++i)
                for (int j = 0; j < m; ++j)
                {
                    Console.Write("a[{0},{1}]= ", i, j);
                    a[i, j]=int.Parse(Console.ReadLine());
```

```
                }
            return a;
        }

        static void Print(int[,] a, int n, int m)
        {
            for (int i = 0; i < n; ++i,Console.WriteLine() )
                for (int j = 0; j < m; ++j)
                    Console.Write("{0,5} ", a[i, j]);
        }

        static void DeleteArray(int[,] a, ref int n, int m, int k)
        {
            for (int i = k; i < n-1; ++i)
                for (int j = 0; j < m; ++j)
                    a[i, j] = a[i+1, j];
            --n;
        }

        static void Main()
        {
            int n,m;
            int[,] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray, n, m);
            Console.WriteLine("Введите номер строки для удаления:");
            int k=int.Parse(Console.ReadLine());
            DeleteArray(myArray, ref n, m, k);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n, m);
        }
    }
```

Задания.

1. Подумайте, какие исключительные ситуации могут возникнуть в данной программе и добавьте в нее соответствующие обработки исключительных ситуаций.
2. Измените программу так, чтобы она удаляла k-тый столбец в двумерном массиве.

Рассмотрим модификацию предыдущей программы, для случая, когда используется ступенчатый массив.

```
using System;
namespace ConsoleApplication
{
    class Class
    {

        static int [][] Input (out int n, out int m)
        {
            Console.WriteLine("введите размерность массива");
            Console.Write("n = ");
            n=int.Parse(Console.ReadLine());
            Console.Write("m = ");
```

```
            m=int.Parse(Console.ReadLine());
            int [] []a=new int[n][];
            for (int i = 0; i < n; ++i)
            {
                a[i]=new int[m];
                for (int j = 0; j < m; ++j)
                {
                    Console.Write("a[{0},{1}]= ", i, j);
                    a[i][j]=int.Parse(Console.ReadLine());
                }
            }
            return a;
        }

        static void Print(int[][] a, int n, int m)
        {
            for (int i = 0; i < n; ++i,Console.WriteLine() )
                for (int j = 0; j < m; ++j)
                    Console.Write("{0,5} ", a[i] [j]);
        }

        static void DeleteArray(int[][] a, ref int n, int k)
        {
            for (int i = k; i < n-1; ++i)//производим сдвиг ссылок
                a[i] = a[i+1];
            --n;
        }

        static void Main()
        {
            int n,m;
            int[][] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray, n, m);
            Console.WriteLine("Введите номер строки для удаления:");
            int k=int.Parse(Console.ReadLine());
            DeleteArray(myArray, ref n, k);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n, m);
        }
    }
}
```

Вернемся к массиву, определенному в самом первом примере. И подумаем теперь, что значит *добавить элемент в одномерный массив* в позицию с номером k? В этом случае все элементы, начиная с k-ого, должны быть сдвинуты вправо на одну позицию. Однако сдвиг нужно начинать с конца, т.е. на первом шаге на n-е место поставить n-1-ый элемент, потом на n-1-ое место поставить n-2-й элемент, …, наконец, на k+1 место вставить k-й элемент. Таким образом, копия k-го элемента будет на k+1-м месте и на k-е место можно поставить новый элемент. Затем необходимо увеличить текущее количество элементов на 1.

Рассмотрим массив из примера 1 и в качестве k зададим значение равное 3. В этом случае массив будет выглядеть следующим образом:

k=3	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

a	0	1	4	**9**	9	16	0	0	0	0

Теперь в позицию с номером 3 можно поместить новое значение. А текущее количество элементов в массиве становится равным 6. Подумайте, почему сдвиг нужно выполнять с конца массива, а не с начала, как мы это делали в случае удаления элемента из массива.

Рассмотрим программную реализацию данного алгоритма:

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [] Input (out int n)
		{
			Console.WriteLine("введите размерность массива");
			n=int.Parse(Console.ReadLine());
			int []a=new int[2*n]; //выделяем памяти больше чем требуется
			for (int i = 0; i < n; ++i)
			{
				Console.Write("a[{0}]= ", i);
				a[i]=int.Parse(Console.ReadLine());
			}
			return a;
		}

		static void Print(int[] a, int n)
		{
			for (int i = 0; i < n; ++i) Console.Write("{0} ", a[i]);
			Console.WriteLine();
		}

		static void AddArray(int[] a, ref int n, int m)
		{
			for (int i = n; i >= m; --i)
				a[i] = a[i-1];
			++n;
			Console.WriteLine("Введите значение нового элемента");
			a[m]=int.Parse(Console.ReadLine());
		}

		static void Main()
		{
			int n;
			int[] myArray=Input(out n);
			Console.WriteLine("Исходный массив:");
			Print(myArray, n);
			Console.WriteLine("Введите номер элемента для вставки:");
			int m=int.Parse(Console.ReadLine());
			AddArray(myArray, ref n,m);
			Console.WriteLine("Измененный массив:");
			Print(myArray, n);
		}
	}
}
```

Теперь рассмотрим *добавление строки в двумерный массив*. Для этого все строки после строки с номером k передвигаем на 1 строку вниз. Затем увеличиваем количество строк на 1. После этого копия строки с номером k будет находиться в столбце с номером k+1. И, следовательно, k-тый столбец можно заполнить новыми значениями. Рассмотрим программную реализацию алгоритма:

```
using System;
namespace ConsoleApplication
{
	class Class
	{

		static int [,] Input (out int n, out int m)
		{
			Console.WriteLine("введите размерность массива");
			Console.Write("n = ");
			n=int.Parse(Console.ReadLine());
			Console.Write("m = ");
			m=int.Parse(Console.ReadLine());
			//выделяем памяти больше чем необходимо
			int [,]a=new int[2*n, m];
			for (int i = 0; i < n; ++i)
				for (int j = 0; j < m; ++j)
				{
					Console.Write("a[{0},{1}]= ", i, j);
					a[i, j]=int.Parse(Console.ReadLine());
				}
			return a;
		}


		static void Print(int[,] a, int n, int m)
		{
			for (int i = 0; i < n; ++i,Console.WriteLine() )
				for (int j = 0; j < m; ++j)
					Console.Write("{0,5} ", a[i, j]);
		}

		static void AddArray(int[,] a, ref int n, int m, int k)
		{
			for (int i = n; i >=k; --i)
				for (int j = 0; j < m; ++j)
					a[i+1, j] = a[i, j];
			++n;
			Console.WriteLine("Введите элементы новой строки");
			for (int j=0; j<m;++j)
			{
				Console.Write("a[{0},{1}]=", k, j);
				a[k, j]=int.Parse(Console.ReadLine());
			}
		}

		static void Main()
		{
```

```
            int n,m;
            int[,] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray, n, m);
            Console.WriteLine("Введите номер строки для добавления:");
            int k=int.Parse(Console.ReadLine());
            AddArray(myArray, ref n, m, k);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n, m);
        }
    }
}
```

Задания.

1. Подумайте, какие исключительные ситуации могут возникнуть в данной программе и добавьте в нее соответствующие обработки исключительных ситуаций.
2. Измените программу так, чтобы она добавляла k-тый столбец в двумерном массиве.

Рассмотрим модификацию предыдущей программы для случая, когда используется ступенчатый массив.

```
using System;
namespace ConsoleApplication
{
    class Class
    {

        static int [][] Input (out int n, out int m)
        {
            Console.WriteLine("введите размерность массива");
            Console.Write("n = ");
            n=int.Parse(Console.ReadLine());
            Console.Write("m = ");
            m=int.Parse(Console.ReadLine());
            //выделяем памяти больше чем неообходимо
            int [][]a=new int[2*n][];
            for (int i = 0; i < n; ++i)
            {
                a[i]=new int [m];
                for (int j = 0; j < m; ++j)
                {
                    Console.Write("a[{0}][{1}]= ", i, j);
                    a[i][j]=int.Parse(Console.ReadLine());
                }
            }
            return a;
        }

        static void Print(int[][] a, int n, int m)
        {
            for (int i = 0; i < n; ++i,Console.WriteLine() )
                for (int j = 0; j < m; ++j)
                    Console.Write("{0,5} ", a[i][j]);
        }
```

```
        static void AddArray(int[][] a, ref int n, int m, int k)
        {
            for (int i = n; i >=k; --i)//выполняем сдвиг ссылок
                a[i+1] = a[i];
            ++n;
            a[k]=new int[m]; //создаем новую строку
            Console.WriteLine("Введите элементы новой строки");
            for (int j=0; j<m;++j)
            {
                Console.Write("a[{0}][{1}]=", k, j);
                a[k][j]=int.Parse(Console.ReadLine());
            }
        }

        static void Main()
        {
            int n,m;
            int[][] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray, n, m);
            Console.WriteLine("Введите номер строки для добавления:");
            int k=int.Parse(Console.ReadLine());
            AddArray(myArray, ref n, m, k);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n, m);
        }
    }
}
```

Часть 2. Освоить практические задания

I. В одномерном массиве, элементы которого – целые числа, произвести следующие действия:

1. Удалить из массива все четные числа.
2. Вставить новый элемент после всех элементов, которые заканчиваются на данную цифру.
3. Удалить из массива повторяющиеся элементы, оставив только их первые вхождения.
4. Вставить новый элемент между всеми парами элементов, имеющими разные знаки.
5. Уплотнить массив, удалив из него все нулевые значения.

II. В двумерном массиве, элементы которого – целые числа, произвести следующие действия:

1. Вставить новую строку после строки, в которой находится первый встреченный минимальный элемент.
2. Вставить новый столбец перед всеми столбцами, в которых встречается заданное число.
3. Удалить все строки, в которых нет ни одного четного элемента.
4. Удалить все столбцы, в которых все элементы положительны.
5. Удалить из массива k-тую строку и j-тый столбец, если их значения совпадают.
6. Уплотнить массив, удалив из него все нулевые строки и столбцы.

Задания для лабораторных работ по дисциплине «Разработка и стандартизация программных средств и информационных технологий» предоставляется студентам на занятиях в электронном виде.

Самостоятельная работа

Задания для самостоятельной работы предложены в каждом лабораторном занятии.

Самостоятельная работа студентов направлена на углубление и закрепление знаний, а также развитие практических умений и заключается в:

- работе с лекционным материалом, поиске и анализе литературы и электронных источников информации;
- выполнении домашних заданий (домашние задания представляют из себя перечень задач, с которыми студенты не справились в ходе выполнения лабораторных работ, а также заданий для самостоятельного выполнения);
- изучении теоретического материала к лабораторным занятиям.

Проверка качества самостоятельной работы студентов проводится во время защиты лабораторных работ. Студент должен ориентироваться в теоретической базе, необходимой для выполнения текущей работы, выполнить все задания, уметь отвечать на контрольные вопросы по направлению данной работы.

Консультирование студентов осуществляться в индивидуальном порядке на занятиях и во внеурочное время. Выполнение самостоятельной работы оценивается по электронным материалам, подготовленным студентами. Результаты деятельности накапливаются в индивидуальных портфолио студентов.

6. Критерии оценивания результатов освоения дисциплины (модуля)

6.1. Оценочные средства и критерии оценивания для текущей аттестации

Вопросы для самоконтроля

1.Жизненный цикл программных средств (ПС).

2.Основные, вспомогательные и организационные процессы жизненного цикла.

3.Классическая технология проектирования ПС.

4.Технология прототипного проектирования ПС.

5.RAD – технологии проектирования ПС.

6.Классификации проектов ПС.

7.Стратегии разработки ПС.

8.Модели разработки ПС.

9.Методы адаптации пакетов прикладных программ.

10. Организация проектирования программного обеспечения (ПО); этапы процесса проектирования.

11.Продукционная модель представления знаний.

12.Программное обеспечение экспертных систем.

13.Представление знаний в виде семантических сетей.

14.Представление знаний в виде фреймов.

15.Формальные логические модели представления знаний.

16.Классификация экспертных систем.

17.Стратегии управления выводом в экспертных системах.

18.Основные требования к разработке ПС. Понятия стандарта и стандартизации.

19.Международные организации, разрабатывающие стандарты. Государственный комитет РФ по стандартизации и метрологии.

20.Нормативные документы по стандартизации. Виды стандартов.

21.Государственные стандарты РФ (ГОСТ Р). Дать полную характеристику каждому из них.

22.Основные понятия и показатели надежности ПС.

23.Методы обеспечения надежности ПС.

24.Тестирование ПС. Виды тестирования. Протокол и отчет о тестировании.

25.Принципы тестирования ПО. Стратегии тестирования «черного ящика» и «белого ящика».

26.Стандарты комплекса ГОСТ 34. Стадии и этапы создания автоматизированных систем.

27.Группы стандартов ЕСПД. ГОСТ 19.102-77. Стадии разработки.

28.ГОСТ 19.402-78 ЕСПД. Описание программы.

29.Схема процессов Жизненного цикла

30.ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристика качества и руководство по их применению.

31.ГОСТ Р ИСО/МЭК 8631-94. Информационная технология. Программные конструктивы и условные обозначения для их представления.

32.ГОСТ Р ИСО/МЭК 12119:1994. Информационная технология. Пакеты программных средств. Требования к качеству и испытания.

33.ГОСТ Р ИСО/МЭК 12207-2010. Процессы жизненного цикла программных средств.

34.Основные понятия и показатели надежности ПС.

35.Методы обеспечения надежности ПС.

36.Тестирование ПС. Виды тестирования. Протокол и отчет о тестировании. ГОСТ Р ИСО/МЭК 12119-2000

37.Принципы тестирования программного обеспечения. Стратегии тестирования «черного ящика» и «белого ящика».

38.Сертификация программных средств.

39.Сертификация информационных технологий.

40.Основные процессы жизненного цикла ПО.

41.Вспомогательные процессы жизненного цикла ПО.

42.Организационные процессы жизненного цикла ПО.

Критерии оценивания ответов на вопросы для самоконтроля

Каждому студенту предлагается ответить на 5 произвольных вопросов для самоконтроля. Ответ по каждому вопросу оценивается от 0 до 1 балла (в зависимости от содержательности ответа). Итоговая оценка по теме в разрезе вопросов для самоконтроля складывается по формуле:

$$R = 2 + \frac{3}{5}\sum_{i=1}^{5} Q_i ,$$

где Q_i – баллы за ответ по каждому из вопросов.

Задания для лабораторных работ и задания для самостоятельной работы

Полный список типовых задач и заданий для самостоятельной работы представлен в материалах каждой лабораторной работы.

Задания для лабораторных и самостоятельной работ, образцы решений основных типовых задач практики также размещены в системе дистанционного обучения СмолГУ (www.moodle.smolgu.ru).

Критерии оценивания заданий из лабораторных работ и заданий для самостоятельной работы

Уровень выполнения	Оценка
Задание выполнено в полном объёме, алгоритмические и вычислительные ошибки отсутствуют, проведен анализ полученного решения.	5 (отлично)
Задание выполнено в полном объёме с незначительными техническими ошибками или отсутствует анализ результатов решения задачи.	4 (хорошо)
Задание выполнено не полностью или в решении присутствуют ошибки алгоритмического характера, незначительно влияющие на ход решения задачи.	3 (удовлетворительно)

Задание не выполнено или в решении присутствует значительное количество ошибок алгоритмического характера, существенно влияющих на ход решения задачи.	2 (неудовлетворительно)

Оценка за выполнение заданий по лабораторной работе вычисляется как среднее арифметическое оценок за каждое задание по данной лабораторной работе.

6.2. Оценочные средства и критерии оценивания для промежуточной аттестации

Зачетная контрольная работа

1. Разработать проект «Шифрование данных» методом сдвига.
2. Создать блок-схему алгоритма программы.
3. Подготовить документацию по проекту, содержащую варианты тестирования и отладки программы.

Критерии оценивания зачетной контрольной работы

1. Нормы оценивания работы

№ п/п	Структурная часть контрольной работы	Количество баллов (*)
1	Правильно реализован каждый метод решения	1 балл
2	Анализ результатов	2 балла

(*) Возможна градация в 0,25 балла.

2. Шкала оценивания работы:

п/п	Оценка	Количество баллов
1	Отлично	4,75-5
2	Хорошо	3,75-4,5
3	Удовлетворительно	3-3,5
4	Неудовлетворительно	менее 3

Критерии получения зачета

Зачет выставляется по результатам работы студента в течение семестра согласно Положению о текущем контроле успеваемости и промежуточной аттестации студентов в федеральном государственном бюджетном образовательном учреждении высшего профессионального образования «Смоленский государственный университет» (утверждено приказом ректора от 24 апреля 2014 г. №01-36).

Для получения зачета студент должен:

- выполнить задания лабораторных работ на оценку не ниже «удовлетворительно»;
- выполнить задания для самостоятельной работы на оценку не ниже «удовлетворительно»;
- ответить на вопросы для самоконтроля на оценку не ниже «удовлетворительно».

7. Перечень основной и дополнительной учебной литературы

7.1. Основная литература

1. Гниденко И. Г. Технологии и методы программирования: учебное пособие для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. – Москва: Издательство Юрайт, 2020. – 235 с. – (Высшее образование). – ISBN 978-5-534-02816-4. – URL: https://urait.ru/bcode/450999
2. Зыков С. В. Программирование: учебник и практикум для вузов / С. В. Зыков. – Москва: Издательство Юрайт, 2020. – 320 с. – (Высшее образование). – ISBN 978-5-534-02444-9. – URL: https://urait.ru/bcode/450832
3. Крупский В. Н. Теория алгоритмов. Введение в сложность вычислений: учебное пособие для вузов / В. Н. Крупский. – 2-е изд., испр. и доп. – Москва: Издательство Юрайт, 2020. –

117 с. – (Высшее образование). – ISBN 978-5-534-04817-9. – URL: https://urait.ru/bcode/454121
4. Кувшинов Д. Р. Основы программирования: учебное пособие для вузов / Д. Р. Кувшинов. – Москва: Издательство Юрайт, 2020. – 104 с. – (Высшее образование). – ISBN 978-5-534-07559-5. – URL: https://urait.ru/bcode/454667
5. Лаврищева Е. М. Программная инженерия и технологии программирования сложных систем: учебник для вузов / Е. М. Лаврищева. – 2-е изд., испр. и доп. – Москва: Издательство Юрайт, 2020. – 432 с. – (Высшее образование). – ISBN 978-5-534-07604-2. – URL: https://urait.ru/bcode/452137
6. Лаврищева Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства: учебник для вузов / Е. М. Лаврищева. – 2-е изд., испр. – Москва: Издательство Юрайт, 2020. – 280 с. – (Высшее образование). – ISBN 978-5-534-01056-5. – URL: https://urait.ru/bcode/452156
7. Малявко А. А. Формальные языки и компиляторы: учебное пособие для вузов / А. А. Малявко. – Москва: Издательство Юрайт, 2020. – 429 с. – (Высшее образование). – ISBN 978-5-534-04288-7. – URL: https://urait.ru/bcode/453250
8. Методы оптимизации: теория и алгоритмы: учебное пособие для вузов / А. А. Черняк, Ж. А. Черняк, Ю. М. Метельский, С. А. Богданович. – 2-е изд., испр. и доп. – Москва: Издательство Юрайт, 2020. – 357 с. – (Высшее образование). – ISBN 978-5-534-04103-3. – URL: https://urait.ru/bcode/453567
9. Тузовский А. Ф. Объектно-ориентированное программирование: учебное пособие для вузов / А. Ф. Тузовский. – Москва: Издательство Юрайт, 2020. – 206 с. – (Высшее образование). – ISBN 978-5-534-00849-4. – URL: https://urait.ru/bcode/451429
10. Черткова Е. А. Программная инженерия. Визуальное моделирование программных систем: учебник для вузов / Е. А. Черткова. – 2-е изд., испр. и доп. – Москва: Издательство Юрайт, 2020. – 147 с. – (Высшее образование). – ISBN 978-5-534-09172-4. – URL: https://urait.ru/bcode/452749

7.2. Дополнительная литература

1. Демин А. Ю. Информатика. Лабораторный практикум: учебное пособие для вузов / А. Ю. Демин, В. А. Дорофеев. – Москва: Издательство Юрайт, 2020. – 131 с. – (Высшее образование). – ISBN 978-5-534-08366-8. – URL: https://urait.ru/bcode/451395
2. Зыков С. В. Программирование. Объектно-ориентированный подход: учебник и практикум для вузов / С. В. Зыков. – Москва: Издательство Юрайт, 2020. – 155 с. – (Высшее образование). – ISBN 978-5-534-00850-0. – URL : https://urait.ru/bcode/451488
3. Казанский А. А. Программирование на Visual C#: учебное пособие для вузов / А. А. Казанский. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2020. – 192 с. – (Высшее образование). – ISBN 978-5-534-12338-8. – URL: https://urait.ru/bcode/451467
4. Лобанова Н. М. Эффективность информационных технологий: учебник и практикум для вузов / Н. М. Лобанова, Н. Ф. Алтухова. – Москва: Издательство Юрайт, 2020. – 237 с. – (Высшее образование). – ISBN 978-5-534-00222-5. – URL: https://urait.ru/bcode/450399
5. Мойзес О. Е. Информатика. Углубленный курс: учебное пособие для вузов / О. Е. Мойзес, Е. А. Кузьменко. – Москва: Издательство Юрайт, 2020. – 157 с. – (Высшее образование). – ISBN 978-5-9916-7051-7. – URL: https://urait.ru/bcode/451401
6. Токарев В. В. Методы оптимизации: учебное пособие для вузов / В. В. Токарев. – Москва: Издательство Юрайт, 2020. – 440 с. – (Высшее образование). – ISBN 978-5-534-04712-7. – URL: https://urait.ru/bcode/454017
7. Сысолетин Е. Г. Разработка интернет-приложений: учебное пособие для вузов / Е. Г. Сысолетин, С. Д. Ростунцев. – Москва: Издательство Юрайт, 2020. – 90 с. – (Высшее образование). – ISBN 978-5-9916-9975-4. – URL: https://urait.ru/bcode/453345
8. Трофимов В. В. Алгоритмизация и программирование: учебник для вузов / В. В. Трофимов, Т. А. Павловская; под редакцией В. В. Трофимова. – Москва: Издательство Юрайт, 2020. –

137 с. – (Высшее образование). – ISBN 978-5-534-07834-3. – URL: https://urait.ru/bcode/452333
9. Тузовский А. Ф. Проектирование и разработка web-приложений: учебное пособие для вузов / А. Ф. Тузовский. – Москва: Издательство Юрайт, 2020. – 218 с. – (Высшее образование). – ISBN 978-5-534-00515-8. – URL: https://urait.ru/bcode/451207
10. Черпаков И. В. Основы программирования: учебник и практикум для вузов / И. В. Черпаков. – Москва: Издательство Юрайт, 2020. – 219 с. – (Высшее образование). – ISBN 978-5-9916-9983-9. – URL: https://urait.ru/bcode/450823

7.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Система дистанционного обучения СмолГУ (moodle.smolgu.ru).
2. Национальный открытый университет (intuit.ru).
3. Национальная платформа открытого образования (opened.ru)

8. Материально-техническое обеспечение

Для проведения занятий лекционного типа предлагаются наборы демонстрационного оборудования и учебно-наглядных пособий, обеспечивающие тематические иллюстрации, соответствующие программе дисциплины (модулей), учебная ауд. 224 на 12 посадочных мест.

Перечень материально-технического обеспечения, необходимого для реализации курса, включает в себя лабораторию, оснащенную персональными компьютерами, объединенные в сеть с выходом в Интернет, проектором и интерактивной доской, ауд.224 на 12 посадочных мест и 6 парт (12 посадочных мест).

Помещение для самостоятельной работы обучающихся оснащено компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду университета, ауд.224 на 12 посадочных мест и 6 парт (12 посадочных мест).

9. Программное обеспечение

1. Операционная система MS Windows XP, Linux.
2. Система программирования MS Visual Studio 19 (язык программирования C#).
3. Поисковые системы сети Интернет.