МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ГАГАРИНА Ю.А.»

«СОГЛАСОВАНО»	«УТВЕРЖДАЮ»
Решением П(Ц)МК	Заместитель директора по УР
Протокол № ____	_____ С.В. Клюквина
от «__» _____20__г.	«___» _____ 20__ г.

Методические рекомендации для студентов по проведению лабораторных работ по дисциплине

«ОП. 16 Технологии программирования» для специальности 09.02.06 «Сетевое и системное администрирование»

Саратов 2021

Разработал преподаватель СКМ и Э ФГБОУ ВО «СГТУ имени Гагарина Ю.А» Пояркова Т.А.

Данные методические рекомендации предназначены для студентов 3 курса, специальности 09.02.06 «Сетевое и системное администрирование»

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....3
Введение.....4
Цель практических занятий.....4
Организация практического занятия.....4
Структура практического занятия.....8

Введение

Практические занятия, как виды учебных занятий, направлены на экспериментальное подтверждение теоретических положений и формирование учебных и профессиональных практических умений и составляют важную часть теоретической и профессиональной практической подготовки. Семинар является видом практических занятий.

В процессе практического занятия обучающиеся выполняют одно или несколько практических заданий под руководством преподавателя в соответствии с изучаемым содержанием учебного материала.

Выполнение обучающимися практических занятий проводится с целью:

- формирования практических умений в соответствии с требованиями к уровню подготовки обучающихся, установленными рабочей программой дисциплины по конкретным разделам/ темам дисциплины;
- обобщения, систематизации, углубления, закрепления полученных теоретических знаний;
- совершенствования умений применять полученные знания на практике, реализации единства интеллектуальной и практической деятельности;
- развития интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработки таких профессионально значимых качеств, как самостоятельность, ответственность, точность, творческая инициатива при решении поставленных задач при освоении общих компетенций.

Цель практических занятий

В результате изучения дисциплины студент должен знать:

- Особенности современных методологий и технологий создания программных средств;
- Организацию проектирования ПС и содержание различных этапов процесса проектирования;
- Задачи и методы тестирования и отладки программных средств, классификационную схему программных ошибок;
- Типовые средства и методы разработки надежного программного обеспечения;
- Принципы и методы создания программных средств на основе концепции и стандартов открытых систем, CASE – систем;
- Международные стандарты на разработку программного обеспечения;
- Государственные стандарты на документирование программного обеспечения

В результате изучения дисциплины студент должен уметь:

– Проектировать, конструировать и отлаживать программные средства в соответствии с заданными критериями качества и стандартами;

– Выявлять основные факторы, определяющие качество и надежность программных средств;

– Осуществлять тестирование программных средств с целью повышения их качества и надежности;

– Оформлять документацию на программные средства.

1.4. Требования к результатам освоения дисциплины

Изучение дисциплины направлено на формирование следующих компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОП 02.Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языках.

ОК 11. Планировать предпринимательскую деятельность в профессиональной сфере.

ПК 1.1. Выполнять проектирование кабельной структуры компьютерной сети.

ПК 1.4. Принимать участие в приемо-сдаточных испытаниях компьютерных сетей и сетевого оборудования различного уровня и в оценке качества и экономической эффективности сетевой топологии.

ПК 2.3. Обеспечивать сбор данных для анализа использования и функционирования программно-технических средств компьютерных сетей

1.5 Организация практического занятия

Практическое занятие должно проводиться в учебной лаборатории «Технологии программирования» оснащенная необходимым для реализации программы учебной дисциплины оборудованием.

В соответствии с требованиям ФГОС 09.02.06 «Сетевое и системное администрирование» реализация ППССЗ должна обеспечивать выполнение обучающимися и практических занятий, включая как обязательный компонент *практические занятия (с использованием персональных компьютеров).*

Выполнению практических занятий предшествует проверка знаний обучающихся - их теоретической готовности к выполнению задания.

Практические занятия могут носить репродуктивный, частично-поисковый и поисковый характер.

Работы, носящие *репродуктивный характер*, отличаются тем, что при их проведении обучающиеся пользуются подробными инструкциями, в которых указаны: цель работы, пояснения (теория, основные характеристики), оборудование, аппаратура, материалы и их характеристики, порядок выполнения работы, таблицы, выводы (без формулировки), контрольные вопросы, учебная и специальная литература.

Работы, носящие *частично-поисковый характер*, отличаются тем, что при их проведении обучающиеся не пользуются подробными инструкциями, им не дан порядок выполнения необходимых действий, и они требуют от обучающихся самостоятельного подбора оборудования, выбора способов выполнения работы в инструктивной и справочной литературе и др.

Работы, носящие *поисковый характер*, характеризуются тем, что обучающиеся, опираясь на имеющиеся у них теоретические знания, должны решить новую для них проблему.

При планировании практических занятий необходимо находить оптимальное соотношение репродуктивных, частично-поисковых и поисковых работ, чтобы обеспечить высокий уровень интеллектуальной деятельности.

Формы организации обучающихся при проведении практических занятий - фронтальная, групповая и индивидуальная.

При *фронтальной форме* организации занятий все обучающиеся выполняют одновременно одну и ту же работу.

При *групповой форме* организации занятий одна и та же работа выполняется бригадами по 2 - 5 человек.

При *индивидуальной форме* организации занятий каждый обучающийся выполняет индивидуальное задание.

Для повышения эффективности проведения практических занятий рекомендуется:

- разработка сборников задач, заданий и упражнений;
- разработка контрольно-диагностических материалов для контроля за подготовленностью обучающихся к практическим занятиям, в том числе в форме педагогических тестовых материалов для автоматизированного контроля;
- подчинение методики проведения практических занятий ведущим дидактическим целям с соответствующими установками обучающимся;

- применение коллективных и групповых форм работы, максимальное использование индивидуальных форм с целью повышения ответственности каждого обучающегося за самостоятельное выполнение полного объема работ;
- проведение практических занятий на повышенном уровне трудности с включением в них заданий, связанных с выбором обучающимися условий выполнения работы, конкретизацией целей, самостоятельным отбором необходимого оборудования;
- подбор дополнительных задач и заданий для обучающихся, работающих в более быстром темпе, для эффективного использования времени, отводимого на практические занятия.

Структура лабораторных работ

Лабораторная работа № 1

Тема: Алгоритмизация линейных вычислительных процессов

Цели и задачи урока: Научиться писать программы линейной структуры

Тип урока – лабораторное занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. 1. Семакин И.Г, Основы алгоритмизации и программирования : учебное пособие / И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 390 с. :.

2. Основы алгоритмизации и программирования : лабораторный практикум / сост. И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 144 с.

Теоритические сведения

Программы с линейной структурой являются простейшими и используются для реализации обычных вычислений по формулам (рис. 1). В программах с линейной структурой инструкции выполняются последовательно,одна за другой.

Пример1. Написать программу вычисления функции **Y(a,c,d).** Значения **a, c, d** вводятся с клавиатуры.

$$y = \frac{tgc - d * 23}{a^{d-2} - 1}$$

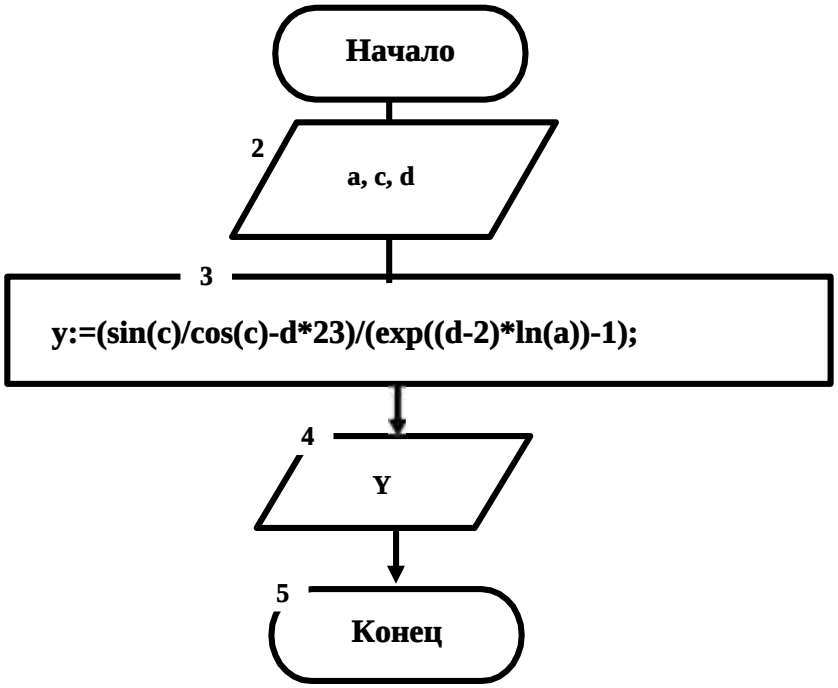

Рисунок 1

Содержание отчета

1. Название, цель работы и задание.
2. 2. Блок-схема алгоритма
3. Текст программы.

Варианты заданий
(2 задания) в табл. 1, 2.
Таблица 1 - Задание 1

№	Функция	Проверочное значение y(x) при x = 2 (или y(x1, x2) при x1 = 2 и x2 = 2)
1	$y = \frac{5x + \sin x^2}{2x + tgx} + \lvert \sin x^2 \rvert$	5,8496
2	$y - \frac{\log_3 \lvert x - 10}{2x \quad 7} - \cos(3x + 5)$	-0,6354
3	$y = \frac{\cos(x_1 + 5) + x_2}{3x_1 + 6x_2} * \lvert \sin x_1^2 \rvert$	0,1158
4	$y = \frac{\cos(3x - 5) - 5^x}{4.35x} + \cos x^2 \rvert + \frac{5}{x}$	6,027716

5	$y=\frac{3x_1+2^{x_2}}{\sin(x_1-x_2)}+\frac{5}{\lvert\sin x2\rvert}+6$	-1,7147
6	$y=\frac{\sin^2(x+3.5)+\lg x}{3x+e^{x+1.35}}$	0,0232
7	$y=\frac{x_1+5.5x_2}{\cos x_1+\ln x_2}+tg(x_1+4.3)$	46,9482
8	$y=\frac{3^{x-1.4}+e^x}{4.5+x}+tg3x$	1,1432
9	$y=\frac{x_1^{x_2}+3.5\lg x_1}{3x_1+5x_2}+\frac{5x_1}{x_2+6}+5$	6,022
10	$y=\frac{\cos^3(x+2.5)+1.5x}{x^2+3.5}+\frac{3x^2}{8x}+8.5$	9,6488
11	$y=\frac{\ln(x+1.3)+5}{1.35x^2+6}+\sqrt{\frac{x+1.3}{(35x)^2+6}}$	0,5693
12	$y=\frac{2x_1+1.4^{x_2}}{tgx_1+2x_2}+\sqrt[3]{6x_1+8^{x2}}$	7,5196
13	$y=\frac{3x+tgx}{2.36x+6}+2x+1.4^4$	6,3159

№	Функция	Проверочное значение y(x) при x = 2 (или y(x1, x2) при x1 = 2 и x2 = 2)
14	$y=\frac{e^{x-1.3}+\sin x}{x+3.5}+2x+1.4\sin x$	5,8045
15	$y=\frac{x+3\cos(x^2+1.5)}{tgx+4.56}+\sin x\frac{2-x}{\sqrt{5^x-56x}}$	2,0480
16	$y=\frac{x+3\cos(x^2-1.5)}{tgx+4.56}+\sin(2x)+\cos(x^3+5)$	0,0694
17	$y=\frac{\cos^2(x_1+1.3)+x_2}{2x_1+e^{x_2}}+\cos(x_1^3+1.5)-\frac{x_2}{\cos(x_1^2+1.5)}$	3,7921
18	$y=\frac{5x_1+1.3^{x_2}}{\cos(x_1+x_2)}+\cos(x_2^2+1.5)+\frac{5+x_2}{\sqrt{x_1}}$	-12,2259

Таблица 2 - Задание 2

№	Функция и проверочные данные
1	$t=\dfrac{2\cos\left(x-\dfrac{\pi}{6}\right)}{0.5+\sin^2 y}\left(1+\dfrac{z^2}{3-z^2/5}\right).$
	При x=14.26, y=-1.22, z=0.035 t=0.564849
2	$u=\dfrac{\sqrt[3]{8+\lvert x-y\rvert^2+1}}{x^2+y^2+2}-e^{\lvert x-y\rvert}\left(\operatorname{tg}^2 z+1\right)^y.$
	При x=-4.5, y=0.000075, z=84.5 u=-55.6848
3	$v=\dfrac{1+\sin^2(x+y)}{\left\lvert x-\dfrac{2y}{1+x^2y^2}\right\rvert}x^{\lvert y\rvert}+\cos^2\left(arctg\dfrac{1}{z}\right).$
	При x=0.0374, y=-0.825, z=16, v=1.0553
4	$w=\lvert\cos x-\cos y\rvert^{\left(1+2\sin^2 y\right)}\left(1+z+\dfrac{z^2}{2}+\dfrac{z^3}{3}+\dfrac{z^4}{4}\right).$
	При x=4000, y=-0.875, z=-0.000475 w=1.9873
5	$\alpha=\ln\left(y^{\sqrt{\lvert x\rvert}}\right)\left(x-\dfrac{y}{2}\right)+\sin^2 arctg(z).$
	При x=-15.246, y=0.04642, z=2000.1 α=-182.036
6	$\beta=\sqrt{10\left(\sqrt[3]{x}+x^{y+2}\right)\left(\arcsin^2 z-\lvert x-y\rvert\right)}$
	При x=0.01655, y=-2.75, z=0.15 β=-40.6307
7	$\gamma=5arctg(x)-\dfrac{1}{4}\arccos(x)\dfrac{x+3\lvert x-y\rvert+x^2}{\lvert x-y\rvert z-x^2}.$
	При x=-0.1722, y=6.33, z=-0.000325 γ=-205.3056
8	$\varphi=\dfrac{e^{\lvert x-y\rvert}\lvert x-y\rvert^{\cos y}}{arctg(x)+arctg(z)}+\sqrt[3]{x^x+\ln^2 y}.$
	При x=-0.02235, y=2.23, z=15.221 φ=39.374
9	$\psi=\left\lvert\dfrac{y}{x^{\sin x}}-\sqrt[3]{\dfrac{y}{x}}\right\rvert+(y-x)\dfrac{\cos y-\dfrac{z}{(y-x)}}{1+(y-x)^2}.$
	При x=182.5, y=18.225, z=-0.03.298 ψ=1.2131
10	$a=2^{-x}\sqrt{x+\sqrt[4]{\lvert y\rvert}\sqrt[3]{e^{x-1/\sin z}}}.$
	При x=0.03981, y=-1625, z=0.512 a=1.26185.
11	$b=y^{\sqrt[3]{\lvert y\rvert}}+\cos^2(y)\dfrac{\lvert x-y\left(1+\dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{\lvert x-y\rvert}-\dfrac{x}{2}}.$
	При x=5.251, y=0.827, z=25.001 b=0.7121
12	$c=2^{\lvert y\rvert}+\left(3^x\right)^y-\dfrac{y\left(arctg(z)-\dfrac{\pi}{6}\right)}{\lvert x\rvert+\dfrac{1}{y^2+1}}.$
	При x=3.251, y=0.325, z=0.0000466 c=4.2514

13	$f=\frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{\lvert x \quad y\rvert\left(\sin^2 z+tgz\right)}$
	При x=17.421, y=0.010365, z=82800 f=0.33056
14	$g=\frac{y^{x-1}}{\sqrt[3]{\lvert y-2\rvert}-3}+\frac{x+\frac{y}{2}}{2\lvert x+y\rvert}(x+1)^{-1/\sin z}$
	При x=1.23, y=15.4, z=252 g=82.8257.
15	$h=\frac{x^{y+1}+e^{y-1}}{1+x\lvert y-tgz\rvert}\left(1+\lvert y-x\rvert\right)+\frac{\lvert y-x\rvert^2}{2}-\frac{\lvert y-x\rvert^3}{3}$
	При x=2.444, y=0.00869, z=-130 h=-0.49871

Контрольные вопросы:

1. Из каких разделов состоит программа?
2. Что такое оператор?
3. Какие операторы языка вам известны?
4. Зачем нужен оператор присваивания? Какой вид он имеет?
5. Что может быть записано в правой части оператора присваивания?
6. Что такое переменная?
7. Что такое константа?
8. Какие правила применяются для создания имен переменных?
9. Что такое идентификатор?
10. Почему знак умножения всегда выписывают явно (например, пишут a*t,а не at)?
11. Как описываются переменные?
12. Какие стандартные числовые типы вам известны?
13. Что вам известно о соответствии типов переменных?
14. Какие арифметические операции можно выполнять?
15. Что вам известно о приоритете арифметических действий?
16. Какие математические функции есть?
17. Какая команда служит для ввода данных?
18. Какой формат записи имеет команда ввода?
19. Какая команда служит для вывода данных?
20. Какой формат записи имеет команда вывода?

Лабораторная работа № 2

Тема: Алгоритмизация разветвляющихся вычислительных процессов

Цели и задачи урока: Изучить возможность использования условных операторов

Тип урока – лабораторное занятие.
Методы обучения – *частично-поисковый*
Обеспечение занятия – персональный компьютер.
Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. 1. 1. Семакин И.Г, Основы алгоритмизации и программирования : учебное пособие / И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 390 с. :.
2. Основы алгоритмизации и программирования : лабораторный практикум / сост. И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 144 с.

Теоритические сведения

Теория

1. Логические переменные и операции над ними

Переменные логического типа описываются посредством служеб- ного слова bool. Они могут принимать только два значения – False (ложь) и True (истина). Результат False (ложь) и True(истина) возника- ет при использовании операций сравнения > (больше), < (меньше),

!=(не равно), >=(больше или равно), <=(меньше или равно), ==(равно).

Описываются логические переменные следующим образом:

```
bool b;
```

В языке C# имеются логические операции, применяемые к пере- менным логического типа. Это операции *логического отрицания* (!), *ло- гическое И* (&&) и *логическое ИЛИ* (||). Операция логического отрица- ния является *унарной операцией*. Результат операции ! есть False, если операнд истинен, и True, если операнд имеет значение «ложь». Так,

!True → False(неправда есть ложь),

!False → True(не ложь есть правда).

Результат операции *логическое И* (&&) есть истина, только если оба ее операнда истинны, и ложь во всех других случаях. Результат опера- ции *логическое ИЛИ* (||) есть истина, если какой-либо из ее операндов истинен, и ложен только тогда, когда оба операнда ложны.

2. Условные операторы

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор ifи оператор выбора switch.

Условный оператор if используется для разветвления процесса об- работки данных на два направления. Он может иметь одну из форм: со- кращенную или полную.

Форма сокращенного оператора if: if (B) S;

где B – логическое или арифметическое выражение, истинность которо- го проверяется; S– оператор.

При выполнении сокращенной формы оператора if сначала вычис- ляется выражение B, затем проводится анализ его результата: если B ис- тинно, то выполняется оператор S; если B ложно, то оператор S пропус- кается. Таким образом, с помощью сокращенной формы оператора if можно либо выполнить оператор S, либо пропустить его.

Форма полного оператора if:

```
if (B) S1; else S2;
```

где B – логическое или арифметическое выражение, истинность которо- го проверяется; S1, S2– операторы.

При выполнении полной формы оператора if сначала вычисляется выражение B, затем анализируется его результат: если B истинно, то вы- полняется оператор S1, а оператор S2

пропускается; если В ложно, то выполняется оператор S2, а S1 – пропускается. Таким образом, с по- мощью полной формы оператора if можно выбрать одно из двух альтернативных действий процесса обработки данных.

Для примера вычислим значение функции:

$$y(x) = \begin{cases} \sin(x), \text{ если } x \le a \\ \cos(x), \text{ если } a < x < b \\ \operatorname{tg}(x), \text{ если } x \ge b \end{cases}$$

Указанное выражение может быть запрограммировано в виде

```
if (x <= a)
      y = Math.Sin(x);
if ((x > a) && (x < b)) y =
        Math.Cos(x);
if (x >= b)
        y = Math.Sin(x) / Math.Cos(x);
```

или с помощью оператора else:

```
if (x <= a)
      y = Math.Sin(x);
else
    if (x < b)
        y = Math.Cos(x);
    else
        y = Math.Sin(x) / Math.Cos(x);
```

Важное примечание! В С-подобных языках программирования, , операция сравнения представляется двумя знаками равенства, например:

```
if (a == b)
```

Задание

Ход работы

1. **Составить блок-схему алгоритма при помощи сервиса dia**
2. **Реализовать программу**
3. **Провести тестирование работоспособности программы**
4. **Составить отчет о проделанной работе**

Задание1. Составить программу для вычисления значений функции y. Вычислить значения функции для указанных значений x.

№ варианта	задание	№ варианта	задание
1	$y = \begin{cases} 1-2x^2; & x < -1 \\ 2+x^2; & -1 \le x \le 2 \\ \lvert 2x \rvert + 1; & 2 \le x \le 3 \\ \ln(1+x); & x > 3 \end{cases}$ $x = -1.5; 0.7; 2.4; 4$	8	$y = \begin{cases} 3x^2; & x < -6 \\ 2x^2 - 4; & -6 \le x \le 6 \\ \cos x + 1; & 6 < x \le 8 \\ [x-2]; & x > 8 \end{cases}$ $x = -6.1; -5.1; 6.1; 8.4$

2	$y=\begin{cases}8x; & x\le 0\\ 5\ln(x); & 0<x\le 1\\ x^2+x; & 1<x\le 4\\ \sin x; & x>4\end{cases}$ $x=-3.1;0.5;1.9;4.2$	9	$y=\begin{cases}x^2+4; & x<1\\ \ln x+3; & 1\le x\le 3\\ 2x+1; & 3<x\le 5\\ [x]; & x>3\end{cases}$ $x=0.4;2;4.1;5.8$
3	$y=\begin{cases}0.1x^2-0.6; & x<0\\ 4x+x^2; & 0\le x\le 2\\ tg(x-3); & 2<x\le 4\\ [x+2]; & x>4\end{cases}$ $x=-1.2;1.1;3.2;4.8$	10	$\begin{cases}1{,}5\cos^2 x; & x<1\\ 1{,}8x; & x=1\\ (x-2)^2+6; & 1<x<2\\ 3tg(x); & x\ge 2\end{cases}$ $x=0\quad x=1\quad x=1{,}5\quad x=2$
4	$y=\begin{cases}2x^2-7x+3; & x<4\\ [x+1]; & 4\le x\le 5\\ 6-x; & 5<x\le 7\\ \cos x; & x>7\end{cases}$ $x=3.5;4.5;5.5;7.8$	11	$y=\begin{cases}arctg(2x); & x<-1\\ [x+2]; & -1\le x<1\\ 2x^2+14x; & 1\le x\le 3\\ Ln(2x); & x>3\end{cases}$ x= -2.5, 0.5, 2.5, 4.6
5	$y=\begin{cases}x^2-7; & x\le 0\\ 5x; & 0<x<2\\ [x+3]; & 2\le x\le 4\\ \sin x; & x>4\end{cases}$ $x=-1.5;1.4;2.5;4.7$	12	$y=\begin{cases}x^2+2x; & x<3\\ \cos(x); & 3<x\le 4\\ [2x]; & 4<x\le 6\\ arctg(x); & x>6\end{cases}$ x=2.5, 3.5, 4.5, 6.8
6	$y=\begin{cases}0.5x^2+0.2; & x<-1\\ tgx; & -1\le x\le 1\\ [4x]; & 1<x\le 2\\ 5-x; & x>2\end{cases}$ $x=-2;0.5;1.5;3.1$	13	$y=\begin{cases}x^2-x-1; & x<0\\ 2x-1; & 0\le x<1\\ \sin(x-1); & 1\le x\le 3\\ arctg[x]; & x>3\end{cases}$ x=-1.4, 0.5, 1.5, 4
7	$y=\begin{cases}\ln(1-x); & x<0\\ x^2+3x-4; & 0\le x<4\\ [2x]; & 4\le x<6\\ x-7; & x\ge 6\end{cases}$ $x=-1.5;2.5;4.5;7.1$	14	$y=\begin{cases}3x^2-1; & x<0.58\\ 2x+3; & 0.58\le x\le 0.68\\ \cos(x-1); & 0.68<x\le 2\\ [1+x]; & x>2\end{cases}$ x=-0.6, 0.6, 0.7, 2.9

2) Составить программу для выполнения указанного ниже задания, используя оператор множественного ветвления.

1. По номеру месяца выдать его название.
2. По номеру дня недели выдать название дня недели.
3. Выдать последнюю цифру куба натурального числа от 1 до 9.
4. Выдать последнюю цифру квадрата натурального числа от 1 до 9.
5. По числу “ног” выдать название животного: птица, паук, жук, черепаха.

6. По номеру класса выдать название школьной ступени.
7. По номеру месяца выдать количество праздничных и воскресных дней.
8. По номеру месяца выдать день недели, на который приходится 15-е число.
9. Написать программу, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений: "Рабочий день", "Суббота" или "Воскресенье".
10. По возрасту определить род занятий человека: ясли, школа, работа, пенсия.
11. Для первых двадцати натуральных чисел определить - является ли это число четным в интервале от 2 до 10, нечетным в интервале от 1 до 10, или это число попадает в интервал от 11 до 20.
12. Пусть переменная N принимает значения от 1 до 9. Напечатать значение этой переменной римскими цифрами.
13. Пусть переменная N принимает значения от 1 до 9. Напечатать название хранящейся в переменной цифры (один, два, ...).
14. По величине отметки выдать ее словесное описание (1-плохо, 2-неудовлетворительно, 3-удовлетворительно, 4-хорошо, 5-отлично).

Контрольные вопросы:

1. Условный оператор: общий вид, выполнение, блок-схема.
2. Оператор выбора: общий вид, выполнение.
3. Что понимают под алгоритмом ветвления?
4. Какие операторы используются для программирования разветвлений?
5. В каких случаях используется оператор выбора?
6. Какую из функций: Sin(x), Abs(x), Trunc(x) можно заменить условным оператором if x<0 then x := -x?

Лабораторная работа № 3

Тема: Алгоритмизация циклических вычислительных процессов

Цели и задачи урока: Закрепить навыки работы с цикличекими операторами

Тип урока – лабораторное занятие.
Методы обучения – *частично-поисковый*
Обеспечение занятия – персональный компьютер.
Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Семакин И.Г, Основы алгоритмизации и программирования : учебное пособие / И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 390 с. :.
2. Основы алгоритмизации и программирования : лабораторный практикум / сост. И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 144 с.

Теоритические сведения

1. Цикл с параметром

Цикл с параметром имеет следующую структуру:

for (<инициализация>; <выражение>; <модификация>)
<оператор>;

Инициализация используется для объявления и/или присвоения на- чальных значений величинам, используемым в цикле в качестве пара- метров (счетчиков). В этой части можно записать несколько операторов, разделенных запятой. Областью действия переменных, объявленных в части инициализации цикла, является цикл и вложенные блоки. Ини- циализация выполняется один раз в начале исполнения цикла.

Выражение определяет условие выполнения цикла: если его ре- зультат истинен, цикл выполняется. Истинность выражения проверяется перед каждым выполнением тела цикла, таким образом, цикл с параметром реализован как *цикл с предусловием*. В блоке *выражение* через запятую можно записать несколько логических выражений, тогда запя- тая равносильна операции *логическое И* (&&).

Модификация выполняется после каждой итерации цикла и служит обычно для изменения параметров цикла. В части *модификация* можно записать несколько операторов через запятую.

Оператор (простой или составной) представляет собой тело цикла. Любая из частей оператора for (инициализация, выражение, моди- фикация, оператор) может отсутствовать, но точку с запятой, опреде-
ляющую позицию пропускаемой части, надо оставить.

Пример формирования строки, состоящей из чисел от 0 до 9, разде- ленных пробелами:

```
string s = ""; // Инициализация строки
for (int i = 0; i <= 9; i++) // Перечисление всех чисел
s += i.ToString() + " "; // Добавляем число и пробел
Consol.WriteLine(s.ToString()); // Показываем результат
```

Данный пример работает следующим образом. Сначала вычисляет- ся начальное значение переменной i. Затем, пока значение i меньше или равно 9, выполняется тело цикла, и затем повторно вычисляется значение i. Когда значение i становится больше 9, условие – ложно и управление передается за пределы цикла.

2. Цикл с пред условием

Оператор цикла while организует выполнение одного оператора (про-
стого или составного) неизвестное заранее число раз. Формат цикла while: while (B) S;
где B – выражение, истинность которого проверяется (условие завершения цикла); S– тело цикла – оператор (простой или составной).

Перед каждым выполнением тела цикла анализируется значение выражения B: если оно истинно, то выполняется тело цикла, и управление передается на повторную проверку условия B; если значение B ложно – цикл завершается и управление передается на оператор, следующий за оператором S.

Если результат выражения B окажется ложным при первой проверке, то тело цикла не выполнится ни разу. Отметим, что если условие B во время работы цикла не будет изменяться, то возможна ситуация зацикливания, то есть невозможность выхода из цикла. Поэтому внутри тела должны находиться операторы, приводящие к изменению значения выражения Bтак, чтобы цикл мог корректно завершиться.

```
// Считывание начальных данных
double x0 =  Convert.ToDouble(Console.ReadLine());
double xk =  Convert.ToDouble(Console.ReadLine());
double dx =  Convert.ToDouble(Console.ReadLine());
```

```
            double a = Convert.ToDouble(Console.ReadLine());
            string text = "Работу выполнил ст. Иванов М.А."
            // Цикл для табулирования функции
            double x = x0;
            while (x <= (xk + dx / 2))
            {
                  double y = a * Math.Log(x);
                  text += "x=" + Convert.ToString(x) +"; y=" + Convert.ToString(y) + x = x + dx;
            }
      Console.WriteLine(text);
```

Задание

Задание 1

Составить алгоритм, написать программу, отладить программу на ПК. Написать тест программы. Все результаты предъявить преподавателю. Подготовить ответы на контрольные вопросы.

№ варианта	Задачи
1	Найти натуральное число из интервала от a до b, у которого количество делителей максимально.
2	Составить программу нахождения цифрового корня. Цифровой корень данного числа получается, если сложить все цифры этого числа, затем все цифры найденной суммы и повторять этот процесс пока в результате будет получено однозначное число (цифра), которая и называется цифровым корнем.
3	Дано натуральное число n. Напечатать разложение этого числа на простые множители.
4	Найти все трехзначные простые числа (простым называется число, большее 1, не имеющее других делителей, кроме единицы и самого себя, например, число 7 простое, число 6 не простое).
5	Найти размеры всех прямоугольников, площадь которых равна заданному натуральному числу s и стороны которых выражены натуральными числами.
6	Найти все совершенные числа, меньшие 100 000 (натуральное число называется совершенным, если оно равно сумме своих делителей, включая 1 и исключая это самое число, например, число 6 совершенное 6=1+2+3).
7	Найти все пары натуральных дружественных чисел, меньших 50 000 (два натуральных числа называются дружественными, если каждое из них равно сумме всех делителей другого, само другое число в качестве делителя не рассматривается).
8	Составить программу для нахождения всех натуральных решений уравнения , где лежат в интервале от 1 до 30.
9	Дано натуральное число n (n ≤ 27). Найти все трехзначные числа, сумма цифр которых равна n.

10	Найти наименьшее натуральное число n, которое можно представить двумя различными способами в виде суммы кубов двух натуральных чисел.
11	Дано натуральное число n. Получить все натуральные числа, меньшие n и взаимно простые с ним (два натуральных числа называются взаимно простыми, если их наибольший общий делитель равен 1).
12	Напечатать в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр.

Задание 2. Составьте программу табулирования функции $y(x)$, где x принимают значения от x0 до xk с приращением dx, выведите на экран значения x и $y(x)$. Нужный вариант задания выберите из нижеприведенного списка по указанию преподавателя.

1) $y = 10^{-2} bc / x + \cos\sqrt{a^3 x}$,
$x_0 = -1.5; x_k = 3.5; dx = 0.5;$
$a = -1.25; b = -1.5; c = 0.75;$

2) $y = 1.2(a-b)^3 e^{x^2} + x$,
$x_0 = -0.75; x_k = -1.5; dx = -0.05;$
$a = 1.5; b = 1.2;$

3) $y = 10^{-1} ax^3 tg(a - bx)$,
$x_0 = -0.5; x_k = 2.5; dx = 0.05;$
$a = 10.2; b = 1.25;$

4) $y = ax^3 + \cos^2(x^3 - b)$,
$x_0 = 5.3; x_k = 10.3; dx = 0.25;$
$a = 1.35; b = -6.25;$

5) $y = x^4 + \cos(2 + x^3 - d)$,
$x_0 = 4.6; x_k = 5.8; dx = 0.2;$
$d = 1.3;$

6) $y = x^2 + tg(5x + b / x)$,
$x_0 = -1.5; x_k = -2.5; dx = -0.5;$
$b = -0.8;$

7) $y = 9(x + 15\sqrt{x^3 + b^3})$,
$x_0 = -2.4; x_k = 1; dx = 0.2;$
$b = 2.5;$

8) $y = 9x^4 + \sin(57.2 + x)$,
$x_0 = 0.75; x_k = 2.05; dx = 0.2;$

9) $y = 0.0025bx^3 + \sqrt{x + e^{0.82}}$,
$x_0 = -1; x_k = 1; dx = 0.5;$
$b = 2.3;$

10) $y = x \cdot \sin(\sqrt{x + b - 0.0084})$,
$x_0 = -2.05; x_k = -3.05; dx = -0.2;$
$b = 3.4;$

11) $y = x + \sqrt{\left|x^3 + a - be^x\right|}$,
$x_0 = -4; x_k = -6.2; dx = -0.2;$
$a = 0.1;$

12) $y = 9(x^3 + b^3)tgx$,
$x_0 = 1; x_k = 2.2; dx = 0.2;$
$b = 3.2;$

13) $y = \left|x - b\right|^{1/2} / \left|b^3 - x^3\right|^{3/2} + \ln\left|x - b\right|$,
$x_0 = -0.73; x_k = -1.73; dx = -0.1;$
$b = -2;$

14) $y = (x^{5/2} - b)\ln(x^2 - 12.7)$,
$x_0 = 0.25; x_k = 5.2; dx = 0.3;$
$b = 0.8;$

15) $y = 10^{-3}\left|x\right|^{5/2} + \ln\left|x + b\right|$,
$x_0 = 1.75, x_k = -2.5, dx = -0.25,$
$b = 35.4;$

16) $y = 15.28\left|x\right|^{-3/2} + \cos(\ln\left|x\right| + b)$,
$x_0 = 1.23, x_k = -2.4, dx = -0.3,$
$b = 12.6;$

17) $y = 0.00081(\ln\left|x\right|^{5/4} + b)/(x^2 + 3.82)$,
$x_0 = -2.35; x_k = -2; dx = 0.05;$
$b = 74.2;$

18) $y = 0.8 \cdot 10^{-5}(x^3 + b^3)^{7/6}$,
$x_0 = -0.05; x_k = 0.15; dx = 0.01;$
$b = 6.74;$

19) $y = (\ln(\sin(x^3 + 0.0025)))^{3/2} + 0.8 \cdot 10^{-3}$,
$x_0 = 0.12; x_k = 0.64; dx = 0.2;$

20) $y = a + x^{2/3}\cos(x + e^x)$,
$x_0 = 5.62; x_k = 15.62; dx = 0.5;$
$a = 0.11$

Контрольные вопросы:

1. Какова структура итерационного цикла?
2. Что такое **цикл**?
3. Как изменяется значение счетчика итерационного цикла?
4. Чем отличается цикл с предусловием от циклов с параметром?
5. Структура цикла с предусловием?

Лабораторная работа № 4
Тема: Алгоритмы сортировки данных

Цели и задачи урока: Изучение возможностей сортировки данных

Тип урока – лабораторное занятие.
Методы обучения – *частично-поисковый*
Обеспечение занятия – персональный компьютер.
Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. 1. Семакин И.Г, Основы алгоритмизации и программирования : учебное пособие / И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 390 с. :.
2. Основы алгоритмизации и программирования : лабораторный практикум / сост. И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 144 с.

Теория

Выполнение лабораторной работы направлено на освоение основных приемов использования массивов, методов доступа к элементам массивов, их реорганизации и модификации. В качестве практической проблемы, требующей решения, рассматривается известная задача сортировки (упорядочивания) массива в порядке возрастания его элементов. При решении этой задачи требуется исходный массив, содержащий произвольные целые числа, преобразовать к виду, когда каждый элемент массива находится перед другим элементом этого массива, если его значение меньше (больше), чем значение сравниваемого элемента.

В данной лабораторной работе необходимо изучить ряд известных алгоритмов сортировки и создать комплекс программ, реализующий:

- метод сортировки выбором;
- метод сортировки пузырьком;
- метод сортировки включением.

Метод сортировки выбором

Исходный массив длиной N разбивается на две части: итог и остаток. Участок массива, называемый **итогом**, располагается с начала массива и должен быть упорядоченным, а участок массива, называемый **остатком**, располагается вплотную за итогом и содержит исходные числа не отсортированной части исходного массива.

Текстуальный алгоритм сортировки выбором:

Шаг 1. Полагается i=0, т.е. считается, что итоговый участок - пуст.

Шаг 2. В остатке массива ищется минимальный элемент и он меняется местом с первым элементом остатка (i-ым элементом массива). После чего значение i увеличивается на единицу, тем самым расширяя итоговый участок массива (отсортированную часть исходного массива).

Шаг 3. Если i < N, то повторяется Шаг 2. В противном случае - конец алгоритма, т.к. итог становится равным всему массиву.

Конец алгоритма.

Схема алгоритма методом сортировки выбора представлена на рис. 1.

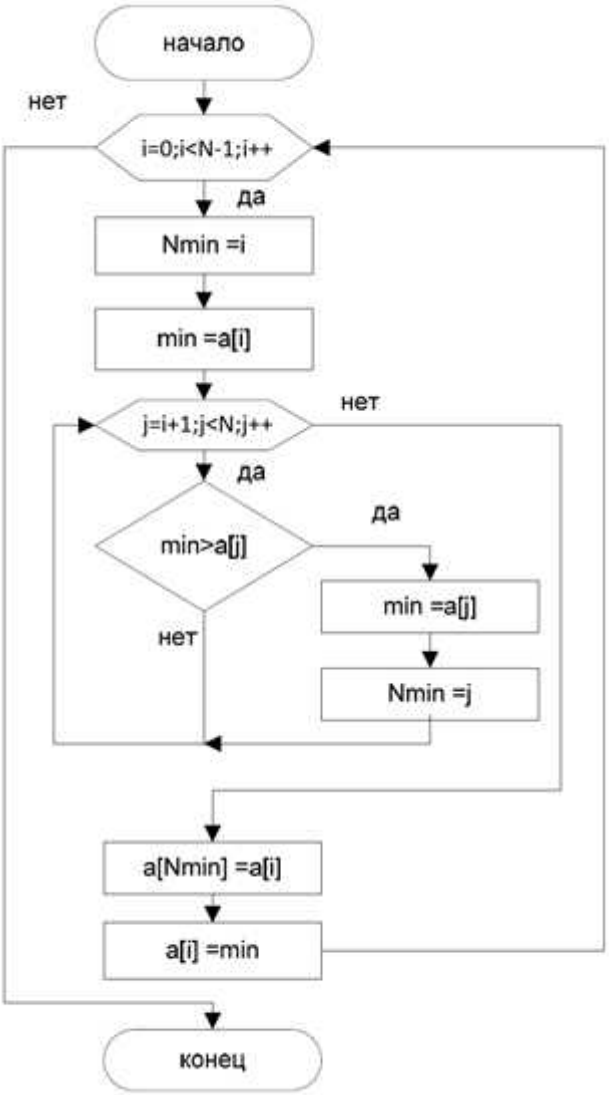

Рисунок 1. Схема алгоритма сортировки методом выбора

Метод сортировки пузырька

Аналогично, как и в методе выбора, исходный массив длиной N разбивается на две части: отсортированную (итог) и не отсортированную (остаток). Первый элемент остатка является i-ым элементом массива.

Текстуальный алгоритм сортировки пузырьком:

Шаг 1. Пусть k=N-1 , т.е. итоговый участок состоит из одного элемента.

Шаг 2. Берется первый элемент остатка и перемещается на место в итоговый участок массива так, чтобы итог остался упорядоченным. Первый элемент остатка назовем перемещаемым. Перемещение выполняется путем сравнения перемещаемого элемента с последующим элементом. Если последующий элемент больше сравниваемого элемента, то процесс перемещения этого элемента закончен.

Шаг 3. После того, как первый элемент остатка переместился в итоговый участок, уменьшается на единицу значение переменной k, тем самым увеличивая отсортированную часть массива. Если k>0, то управление передается на Шаг 2, в противном случае - работа алгоритма завершена.

Конец алгоритма.

Схема алгоритма методома сортировки пузырька представлена на рис. 2.

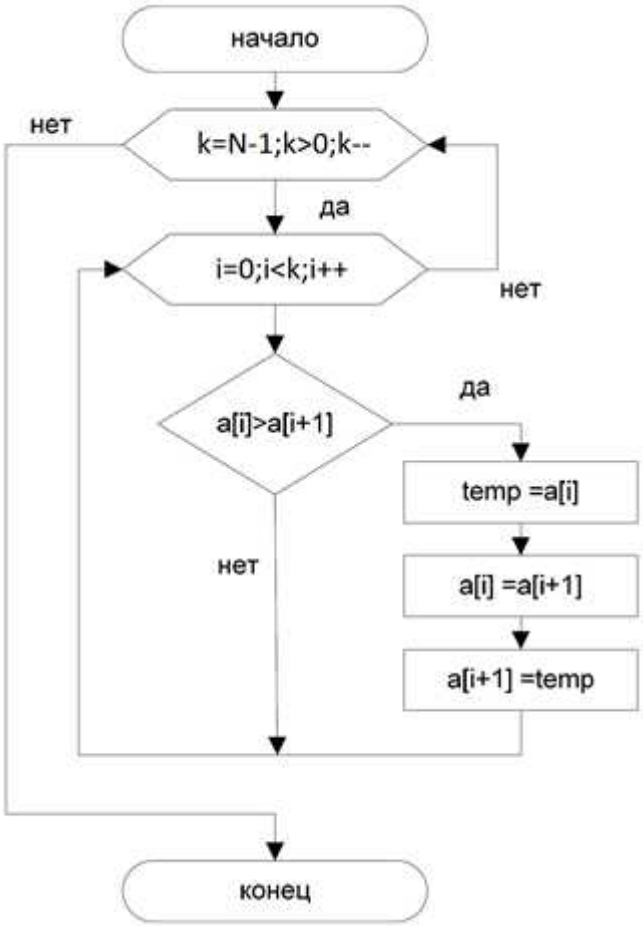

Рисунок 2. Блок-схема алгоритма сортировки методом пузырька

Метод сортировки включением

Этот метод похож на метод пузырька. Происходит такое же разбиение массива на отсортированную и не отсортированную части, но перемещение первого элемента остатка на принадлежащее ему место в итоге делается не сравнением двух соседних элементов, а с помощью метода двоичного поиска, который удобно оформить в виде отдельной процедуры.

Текстуальный алгоритм методом включением:

Шаг 1. Пусть i=1 , т.е. итоговый участок состоит из одного элемента.

Шаг 2. Берется первый элемент остатка и перемещается в отсортированную часть массива так, чтобы итоговый участок остался упорядоченным.

Шаг 3. После того, как первый элемент остатка переместился в итоговый участок, увеличивается на единицу значение переменной i, тем самым увеличивая отсортированную часть массива. Если i < N, то управление передается на Шаг 2, в противном случае - работа алгоритма завершена.

Конец алгоритма.

Схема алгоритма методом сортировки включением представлена на рис. 3.

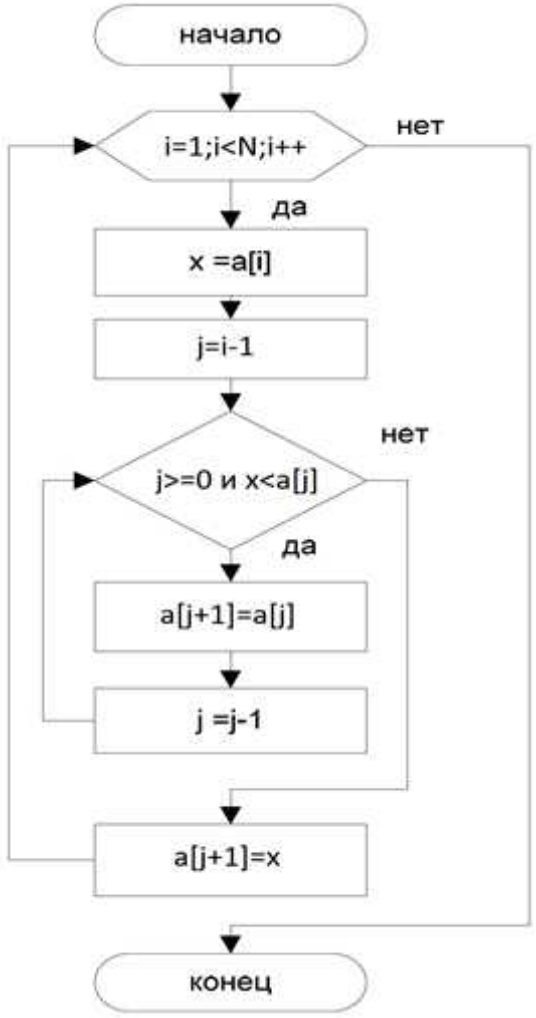

Рисунок 3. Блок-схема алгоритма прямым включением

Задание

Разработать программу сортировки одномерных массивов данных. Программа должна выполнять:

1) заполнение трех одномерных массивов VI, V2, V3 одинакового размера случайными целыми числами в заданном диапазоне (варианты индивидуальных заданий представлены в табл. 1);

2) сортировку полученных массивов VI, V2, V3 (метод и направление сортировки указаны в табл. 2: в таблице обозначение Т соответствует сортировке по возрастанию, Г - сортировке по убыванию);

3) вывод на экран: а) исходных массивов; б) каждой итерации сортировки массивов; в) отсортированных массивов.

Таблица 1

вар.	Диапазон значений элементов одномерных массивов		
	VI	*V2*	*V3*
1	[-90;-10]	[-10;+ 10)	(+10;+ 20)
2	[-80;-20]	[-20;+ 20)	(+20;+ 30)
3	[-70;-30]	[-30;+ 30)	(+30;+ 40)
4	[-60;-40]	[-40;+ 40)	(+ 40;+ 50)
5	[+10;+ 90]	[-50;+ 50)	(-20;-10)
6	[+20; + 80]	[-60;+ 60)	(-30;-20)
7	[+30;+ 70]	[-70;+ 70)	(-40;-30)
8	[+40; + 60]	[-80;+ 80)	(-50;-40)
9	[-90;-10]	[-90;+ 90)	(+ 50; + 60)
10	[-80;-20]	(-10; + 10]	(+60;+ 70)
11	[-70;-30]	(-20;+ 20]	(+70;+ 80)
12	[-60;-40]	(-30; + 30]	(+80;+ 90)
13	[+10; + 90]	(-40;+ 40]	(-60;-50)
14	[+20; + 80]	(-50;+ 50]	(-70;-60)
15	[+30;+ 70]	(-60; + 60]	(-70;-80)
16	[+ 40; + 60]	(-70;+ 70]	(-90;-80)
17	[-90;-10]	(-80;+ 80]	(+10;+ 20)

18	[—80; — 20]	(-90; + 90]	(+20;+ 30)
19	[-70;-30]	[-10;+ 10)	(+30;+ 40)
20	[-60;-40]	[-20;+ 20)	(+40;+ 50)
21	[+10; + 90]	[-30;+ 30)	(-20;-10)

Таблица 2

вар.	Метод и направление сортировки одномерных массивов					
	VI		*V2*		*V3*	
1	метод «пузырька»	т	метод выбора	г	метод вставки	т
2	метод вставки	г	метод «пузырька»	т	метод выбора	г
3	метод выбора	т	метод вставки	г	метод «пузырька»	т
4	метод выбора	г	метод «пузырька»	т	метод вставки	г
5	метод вставки	т	метод выбора	г	метод «пузырька»	т
6	метод «пузырька»	г	метод вставки	т	метод выбора	г
7	метод «пузырька»	г	метод выбора	т	метод вставки	г
8	метод вставки	т	метод «пузырька»	г	метод выбора	т
9	метод выбора	г	метод вставки	т	метод «пузырька»	г
10	метод выбора	т	метод «пузырька»	г	метод вставки	т
И	метод вставки	г	метод выбора	т	метод «пузырька»	г
12	метод «пузырька»	т	метод вставки	г	метод выбора	т
13	метод «пузырька»	т	метод выбора	г	метод вставки	г
14	метод вставки	г	метод «пузырька»	т	метод выбора	т
15	метод выбора	т	метод вставки	г	метод «пузырька»	г
16	метод выбора	г	метод «пузырька»	т	метод вставки	т

17	метод вставки	т	метод выбора	г	метод «пузырька»	г
18	метод «пузырька»	г	метод вставки	т	метод выбора	т
19	метод «пузырька»	г	метод выбора	т	метод вставки	т
20	метод вставки	т	метод «пузырька»	г	метод выбора	г
21	метод выбора	г	метод вставки	т	метод «пузырька»	т

Лабораторная работа № 5
Тема: Обработка массивов данных

Цели и задачи урока: Закрепление возможностей по обработке массивов

Тип урока – лабораторное занятие.

Методы обучения – *частично-поисковый*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. 1. Семакин И.Г, Основы алгоритмизации и программирования : учебное пособие / И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 390 с. :.
2. Основы алгоритмизации и программирования : лабораторный практикум / сост. И.Г. Семакин, А.П. Шестаков. - Академия, 2020. - 144 с.

Теория

Массив - набор элементов одного и того же типа, объединенных общим именем. Массивы в C# можно использовать по аналогии с тем, как они используются в других языках программирования. Однако C#-массивы имеют существенные отличия: они относятся к ссылочным типам данных, более того - реализованы как объекты. Фактически имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации массива. А за освобождением памяти следит система сборки мусора - неиспользуемые массивы автоматически утилизируются данной системой.

Рассмотрим различные типы массивов.

1. Одномерные массивы

Одномерный массив - это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в C# начинается с нуля, то есть, если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерный массив в C# реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив,

затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяется размер массива.

В общем случае процесс объявления переменной типа массив, и выделение необходимого объема памяти может быть разделено. Кроме того на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

Форма записи	Пояснения
базовый_тип [] имя_массива; *Например*: int [] a;	Описана ссылка на одномерный массив, которая в дальнейшемможет быть использована: 1. для адресации на уже существующий массив; 2. передачи массива в метод в качестве параметра 3. отсроченного выделения памяти под элементы массива.
базовый_тип [] имя_массива = new базовый_тип [размер]; *Например*: int []a=new int [10];	Объявлен одномерный массив заданного типа и выделена память под одномерный массив указанной размерности. Адрес данной области памяти записан в ссылочную переменную. Элементы массива равны нулю. **Замечание**. Надо отметить, что в C# элементам массива присваиваются начальные значения по умолчанию в зависимости от базового типа. Для арифметических типов - нули, для ссылочных типов - null, для символов - пробел.
базовый_тип [] имя_массива= {список инициализации}; *Например*: int []a={0, 1, 2, 3};	Выделена память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации.

Обращения к элементам массива происходи с помощью индекса, для этого нужно указать имя массива и в квадратных скобках его номер. Например, a[0], b[10], c[i].

Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Рассмотрим несколько простых примеров работы с одномерными массивами.

Пример 1.

```
static void Main()
{
 int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
 int i;
 for (i = 0; i < 10; ++i)
  Console.WriteLine(myArray[i]
  );
}
```

Задание. Измените программу так, чтобы числа выводились в строчку.

Пример 2.

```
static void Main()
{
```

```
 int[] myArray = new
 int[10];int i;
 for (i = 0; i <
  10; i++)
  myArray[i] = i *
  i;
 for (i = 0; i < 10; i++)
  Console.WriteLine(myArray[i]
  );
}
```

Задание. Измените программу так, чтобы обрабатывался массив из n чисел.

Хотя при инициализации массива нет необходимости использовать операцию new, все же массив можно инициализировать следующим образом:

```
int [] myArray = new int [] {99, 10, 100, 18, 78, 23, 163, 9, 87,
49};
```

Несмотря на избыточность, данная форма инициализации массива может оказаться полезной в том случае, когда уже существующей ссылке на одномерный массив присваивается ссылка на новый массив. Например:

```
static void Main()
{
 int[] myArray = { 0, 1, 2, 3,
 4, 5};int i;
 for (i = 0; i < 10; i++)
  Console.Write("
  "+myArray[i]);
 Console.WriteLine("\nНовый массив: ");
 myArray = new int[] { 99, 10, 100, 18, 78, 23, 163, 9, 87, 49 };
 // 1
 for (i = 0; i < 10; i++)
  Console.Write(" " +
  myArray[i]);
}
```

Следует отметить, что первоначально переменная myArray ссылалась на 6-ти элементный массив. В строке 1 переменной myArray была присвоена ссылка на новый 10-элементный массив, в результате чего исходный массив оказался неиспользуемым, т.к. на него теперь не ссылается ни одинобъект. Поэтому он автоматически будет удален сборщиком мусора.

2. Массивы и исключения

Выход за границы массива в C# расценивается как ошибка, в ответ на которую генерируетсяисключение - IndexOutOfRangeException.

Рассмотрим следующий пример:

```
static void Main()
{
 int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
 i
 n
 t
```

```
 i
 ;

 t
 r
 y
 {
  for (i = 0; i <= 10; i++) Console.WriteLine(myArray[i]);
 }

 catch (IndexOutOfRangeException)
 {
  Console.WriteLine("Exception: Выход за границу диапазона");
 }
}
```

Задание. Добавьте в программу обработчики исключений FormatException и OutOfMemoryException. Вспомните, что они контролируют.

3. Массив как параметр

Так как имя массива фактически является ссылкой, то он передается в метод по ссылке и, следовательно, все изменения элементов массива, являющегося формальным параметром, отразятся на элементах соответствующего массива, являющимся фактическим параметром.

Рассмотрим пример передачи массива как параметра:

```
class Program
{
 static void Print(int n, int[] a) //n - размерность массива, a - ссылка на
 массив
 {
  for (int i = 0; i < n; i++) Console.Write("{0} ",
    a[i]);Console.WriteLine();
 }


 static void Change(int n, int[] a)
 {
  for (int i = 0; i < n; i++)
    if (a[i] > 0) a[i] = 0; // изменяются элементы массива
 }

 static void Main()
 {
  int[] myArray = { 0, -1, -2, 3, 4, 5, -6, -7, 8, -9 };
  Print(10,
  myArray);
  Change(10,
  myArray);
```

```
  Print(10,
  myArray);
 }
}
```

4. Массив как объект

Мы уже говорили о том, что массивы в C# реализованы как объекты. Если говорить более точно, то они реализованы на основе базового класса Array, определенного в пространстве имен System. Данный класс содержит различные свойства и методы. Например, свойство Length позволяет определять количество элементов в массиве. Преобразуем предыдущий пример:

```
class Program
{
 static void Print(int[] a) // передаем только ссылку на массив
 {
  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] = 0;
 }
 static void Main()

 {
  int[] myArray = { 0, -1, -2, 3, 4, 5, -6, -7, 8, -9 };
  Print(myA
  rray);
  Change(my
  Array);
  Print(myA
  rray);
 }
}
```

Другие свойства и методы класса Array приведены в следующей таблице:

Элемент	Вид	Описание
Length	Свойство	Количество элементов массива (по всем размерностям)
BinarySearch	статический метод	Двоичный поиск в отсортированном массиве
Clear	статический метод	Присваивание элементам массива значений по умолчанию
Copy	статический метод	Копирование заданного диапазона элементов одного массива в другой
CopyTo	экземплярный метод	Копирование всех элементов текущего одномерного массива в другой массив
GetValue	экземплярный метод	Получение значения элемента массива

IndexOf	статический метод	Поиск первого вхождения элемента в одномерный массив
LastIndexOf	статический метод	Поиск последнего вхождения элемента в одномерный массив
Reverse	статический метод	Изменение порядка следования элементов на обратный
SetValue	экземплярны йметод	Установка значения элемента массива
Sort	статический метод	Упорядочивание элементов одномерного массива

Вызов статических методов происходит через обращение к имени класса, например, Array.Sort(myArray). В данном случае мы обращаемся к статическому методу Sort класса Array и передаем данному методу в качестве параметра объект myArray - экземпляр класса Array.

Обращение к свойству или вызов экземплярного метода производится через обращение к экземпляру класса, например, myArray.Length или myArray.GetValue(i).

Пример:

```
class Program
{
 static void Main()
 {
  try
  {
   int[] MyArray;
   Console.Write("Введите размерность
   массива: ");int n =
   int.Parse(Console.ReadLine());
   MyArray = new int[n];
   for (int i = 0; i < MyArray.Length; ++i)
   {
    Console.Write("a[{0}]=",i);
    MyArray[i] = int.Parse(Console.ReadLine());
   }
   PrintArray("исходный массив:",
   MyArray);Array.Sort(MyArray);
   PrintArray("массив отсортирован по возрастанию",
   MyArray);Array.Reverse(MyArray);
   PrintArray("массив отсортирован по убыванию", MyArray);

  }
  catch (FormatException)
  {
   Console.WriteLine("неверный формат ввода данных");
  }
  catch (OverflowException)
  {
   Console.WriteLine("переполнение");
  }
  catch (OutOfMemoryException)
  {
```

```
    Console.WriteLine("недостаточно памяти для создания нового
   }
  }
  static void PrintArray(string a, int[] mas)
  {
   Console.WriteLine(a);
   for (int i = 0; i < mas.Length; i++) Console.Write("{0} ",
    mas[i]);Console.WriteLine();
  }
 }
```

Задание. Добавьте в программу метод InputArray, предназначенный для ввода с клавиатуры элементов массива. Продемонстрируйте работу данного метода.

5. Многомерные массивы

Многомерные массивы имеют более одного измерения. Чаще всего используются двумерные массивы, которые представляют собой таблицы. Каждый элемент массива имеет два индекса, первый определяет номер строки, второй - номер столбца, на пересечении которых находится элемент.Нумерация строк и столбцов начинается с нуля.

Объявить двумерный массив можно одним из предложенных способов:тип [,] имя_массива;

тип [,] имя__массива = new тип [размер1, размер2];

тип [,] имя__массива={{элементы 1-ой строки}, … , {элементы n-ой строки}};

тип [,] имя_массива= new тип [,]{{элементы 1-ой строки}, … ,{элементы n-ой строки}};строки}};

Например:

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

Замечания.

1. Как и в случае с одномерными массивами, последние два описания являются избыточными.
2. При работе с многомерными массивами можно использовать приемы, которые мы рассмотрели для одномерных массивов.
3. При обращении к свойству Length для двумерного массива мы получим общее количество элементов в массиве. Чтобы получить количество строк нужно обратиться к методу GetLengthс параметром 0. Чтобы получить количество столбцов - к методу GetLength с параметром 1.

Пример:

```
class Program
{
 static void PrintArray(string a, int[,] mas)
 {
  Console.WriteLine(a);

  for (int i = 0; i < mas.GetLength(0); i++)
```

```
 {
  for (int j = 0; j <
   mas.GetLength(1); j++)
   Console.Write("{0} ", mas[i, j]);
   Console.WriteLine();
 }
}
static void Change(int[,] mas)
{
 for (int i = 0; i <
  mas.GetLength(0); i++) for (int j =
  0; j < mas.GetLength(1); j++)
   if (mas[i, j] % 2 == 0) mas[i, j] = 0;
}
static void Main()
{
 try
 {
  int[,] MyArray ={ { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
  PrintArray("исходный массив:",
  MyArray);Change(MyArray);
  PrintArray("итоговый массив", MyArray);
 }
 catch (FormatException)
 {
  Console.WriteLine("неверный формат ввода данных");
 }
 catch (OverflowException)
 {
  Console.WriteLine("переполнение");
 }
 catch (OutOfMemoryException)
 {
  Console.WriteLine("недостаточно   памяти   для   создания   нового
 }
}
```

Задания.

1. Добавьте в программу метод InputArray, предназначенный для ввода с клавиатуры элементовмассива. Продемонстрируйте работу данного метода.
2. Измените метод Change так, чтобы он вычислял сумму четных элементов двумерного массива.

6. Ступенчатые массивы

В ступенчатых массивах количество элементов в разных строках может быть различным. Впамяти ступенчатый массив хранится в виде массива массивов. Структура ступенчатого массива:

Массив	a	a[0]		a[0][0]	a[0][1]	…
		a[1]				

	…	a[1][0]	a[1][1]	…	
	a[n]				
		a[n][0]	a[n][1]	…	

Объявление ступенчатого массива:

тип [][] имя_массива;

Например:

```
int [][]a;
```

Фактически мы объявили одномерный массив ссылок на целочисленные одномерные массивы. При таком описании потребуется не только выделять память под одномерный массив ссылок, но и под каждый из целочисленных одномерных массивов. Такое распределение памяти позволяет определять произвольную длину каждой строки массива (отсюда и произошло название массива - ступенчатый). Например:

int [][] a= new int [3][]; // Создаем три строки **a[0]=new int [2];** // 0-ая строка ссылается на 2-х элементый одномерный массив **a[1]=new int [3];** // 1-ая строка ссылается на 3-х элементый одномерный массив **a[2]=new int [10];** // 2-ая строка ссылается на 10-х элементый одномерный массивДругой способ выделения памяти:

int [][] a= {new int [2], new int [3], new int [10]};

Так как каждая строка ступенчатого массива фактически является одномерным массивом, то с каждой строкой можно работать как с экземпляром класса Array. Это является преимуществомступенчатых массивов перед двумерными массивами.

Пример:

```
class Program
{
 static void Main()
 {
  try
  {
   int[][] MyArray;
   Console.Write("Ввведите количесвто
   строк: ");int n =
   int.Parse(Console.ReadLine()); MyArray
   = new int[n][];
   for (int i = 0; i < MyArray.Length; i++)
   {
    Console.Write("введите количество элементов в {0} строке:
    ", i);int j = int.Parse(Console.ReadLine());
    MyArray[i] = new int[j];
    for (j = 0; j < MyArray[i].Length; j++)
    {
     Console.Write("a[{0}][{1}]= ", i, j);
     MyArray[i][j] =
     int.Parse(Console.ReadLine());
    }
```

```
     }
     PrintArray("исходный массив:", MyArray);
     for (int i = 0; i < MyArray.Length; i++)
      Array.Sort(MyArray[i]);PrintArray("измененный массив",
      MyArray);
    }
    catch (FormatException)
    {
     Console.WriteLine("неверный формат ввода данных");
    }
    catch (OverflowException)
    {
     Console.WriteLine("переполнение");
    }
    catch (OutOfMemoryException)
    {
     Console.WriteLine("недостаточно памяти для создания нового
    }

   }
   static void PrintArray(string a, int[][] mas)
   {
    Console.WriteLine(a);
    for (int i = 0; i < mas.Length; i++)
    {
     for (int j = 0; j < mas[i].Length; j++)
Console.Write("{0} ",mas[i][j]);
     Console.WriteLine();
    }
   }
  }
```

Задание. Добавьте в программу метод MakeArray, предназначенный для создания ступенчатого массива, в котором количество элементов в каждой строке больше номера строки в два раза. А сам элемент равен сумме номеров строки и столбца, в котором он находится. Продемонстрируйте работу данного метода.

7. Оператор foreach и его использование при работе с массивами

Оператор foreach применяется для перебора элементов в специальным образом организованной группе данных, в том числе и в массиве. Удобство этого вида цикла заключается в том, что нам не требуется определять количество элементов в группе и выполнять перебор по индексу - мы просто указываем на необходимость перебрать все элементы группы. Синтаксис оператора:

```
foreach (<тип> <имя> in <группа>) <тело цикла>
```

где *имя* определяет локальную по отношению к циклу переменную, которая будет по очередипринимать все значения из указанной *группы*, а *тип* соответствует базовому типу элементов *группы*.

Ограничением оператора foreach является то, что с его помощью можно только просматриватьзначения элементов в группе данных, но нельзя их изменять.

Рассмотрим несколько примеров использования оператора foreach:для работы с одномерными массивами:

```
static void PrintArray(string a, int [] mas)
{
 Console.WriteLine(a);
 foreach (int x in mas)Console.Write("{0}
 ", x);Console.WriteLine();
}
```

для работы с двумерными массивами:

```
static int Sum (int [,] mas)
{
 int s=0;
 foreach (int x in mas)
 s += x;return s;
}
```

для работы со ступенчатыми массивами:

```
static void PrintArray3(string a, int[][] mas)
{
 Console.WriteLine(a);
 for (int i = 0; i < mas.Length; i++)
 {
  foreach (int x in mas[i]) Console.Write("{0}
  ", x);Console.WriteLine();
 }
}
```

8. Вставка и удаление элементов в массивах

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 состояние*
a	0	1	4	16	0	0	0	0	0	

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 случаеразмерность 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.Lengt
    h;
    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 идобавьте 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 ступенчатыймассив.

```
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
   необходимоint [,]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 идобавьте 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 ступенчатыймассив.

```
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.ReadLin
   e());Console.Write("m = ");
   m=int.Parse(Console.ReadLin
   e());
//выделяем памяти больше чем
   неообходимоint
   [][]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);
   }
  }
}
```

Варианты заданий.

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

1. Дана последовательность из n действительных чисел. Подсчитать количество

максимальныхэлементов.

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. Подсчитать сумму элементов, расположенных между первым максимальным и последним минимальными элементами. Если максимальный элемент встречается позже минимального, то выдать сообщение об этом.
3. Найти минимум из положительных элементов.

II. Задачи из данного пункта решить, используя двумерный массив.

1. Дан массив размером n×n, элементы которого целые числа. Подсчитать среднее 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.ReadLin
   e());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;
   doub
   le
   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 элементов, расположенных под побочной диагональю.
3. 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.

III. Задачи из данного пункта решить, используя ступенчатый массив.

1. Дан массив размером n×n, элементы которого целые числа. Найти максимальный элемент вкаждой 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. Для каждой строки найти последний четный элемент и записать данные в новый массив.
3. Для каждой строки подсчитать сумму элементов, не попадающих в заданный интервал, изаписать данные в новый массив.

IV. Разобрать примеры из *8. Вставка и удаление элементов в массивах.*

1. В одномерном массиве, элементы которого - целые числа, удалить из массива повторяющиесяэлементы, оставив только их первые вхождения.
2. В двумерном массиве, элементы которого - целые числа, вставить новую строку после строки,в которой находится первый встреченный минимальный элемент.

Индивидуальные задания.

1. Ввести два одномерных целочисленных массива A и B из N и M элементов соответственно. Сформировать массив C, добавив в него сначала элементы массива A, затем элементы массива B, кратные числу K.
2. Ввести одномерный целочисленный массив A. Сформировать массив B, записав в него все элементы массива A, стоящие после минимального. Массив B отсортировать по убыванию.
3. Ввести два одномерных целочисленных массива A и B из N и M элементов соответственно. Сформировать массив C, который должен содержать все элементы массивов A и B, отсортированные по убыванию.
4. Ввести два одномерных целочисленных массива A и B из N и M элементов

соответственно. Сформировать массив С, который должен содержать элементы массивов А и В, стоящие на четных местах, отсортированные по возрастанию.

5. Ввести одномерный целочисленный массив А. Сформировать массив В, содержащий все повторяющиеся элементы массива А, следующие за максимальным.

6. Ввести одномерный целочисленный массив А. Сформировать массив В, содержащий все неповторяющиеся элементы массива А, следующие за максимальным.

7. Ввести два одномерных целочисленных массива А и В из N и M элементов соответственно. Сформировать массив С, который должен содержать элементы массива А, находящиеся в массиве В.

8. Ввести два одномерных целочисленных массива А и В из N и M элементов соответственно. Сформировать массив С, который должен содержать элементы массива А, отсутствующие в массиве В.

9. Ввести одномерный массив А. Сформировать массив С, который должен содержать все повторяющиеся элементы массива А, предшествующие последнему отрицательному.

10. Ввести одномерный динамический массив А. Сформировать массив С, который должен содержать переставленные в обратном порядке элементы массива А между максимальным из отрицательных и минимальным элементами.

11. Ввести одномерный массив А. Сформировать массив С, который должен содержать переставленные в обратном порядке элементы массива А между максимальным и предпоследним отрицательным элементом.

12. Ввести два одномерных целочисленных массива А и В из N и M элементов соответственно. Сформировать массив С, добавив в него сначала все элементы массива А, затем элементы массива В, стоящие на нечетных местах.

13. Ввести два одномерных целочисленных массива А и В из N и M элементов соответственно. Сформировать массив С, добавив в него сначала четные элементы массива А, затем нечетные элементы массива В.

14. Ввести одномерный целочисленный массив А. Сформировать массив С, который должен содержать элементы массива А, являющиеся простыми числами.

15. Ввести два одномерных целочисленных массива А и В из N и M элементов соответственно. Сформировать массив С, который должен содержать чередующиеся элементы массивов А и В. Если размеры массивов не совпадают, оставшиеся элементы массива большего размера дописать в массив С в обратном порядке.

16. Ввести одномерный целочисленный массив А. Сформировать массив В, записав в него все четные элементы массива А, стоящие после максимального. Массив В отсортировать по убыванию.

17. Ввести одномерный целочисленный массив А. Сформировать массив В, записав в него все нечетные элементы массива А, стоящие после минимального. Массив В отсортировать повозрастанию.

18. Ввести одномерный динамический массив А. Сформировать массив С, который должен содержать переставленные в обратном порядке элементы массива А между первым максимальным и последним минимальным элементами.

19. Ввести два одномерных целочисленных массива А и В из N и M элементов соответственно. Сформировать массив С, добавив в него первый элемент массива А, затем первый элемент массива В, далее второй элемент массива А, затем второй элемент массива В и т.д. Если N≠M для реализации алгоритма заполнения добавлять нулевые значения, когда закончатся элементы массива меньшего размера.

20. Ввести два одномерных целочисленных массива А и В из N и M элементов соответственно. Сформировать массив С, добавив в него первый элемент массива А, затем последний элемент массива В, далее второй элемент массива А, затем предпоследний

элемент массива B и т.д. Если N≠M, прекратить заполнение массива C когда закончатся элементы массива меньшего размера .

Контрольные вопросы

1. Что представляет собой массив как структура данных?
2. Какие способы задания массивов Вы знаете?
3. Какие данные могут выступать в качестве индексов и элементов массивов?
4. Какие операции применимы к массивам?
5. В чем особенность работы с массивами идентичных типов?
6. Как осуществить сравнение массивов?
7. Как осуществить удаление и вставку элементов в массив?
8. Как осуществить передачу элементов произвольного массива в процедуры и функции?