Министерство образования и науки Республики Казахстан

ВОСТОЧНО-КАЗАХСТАНСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Д. СЕРИКБАЕВА

В.Л. Никифоров

Мамандығы: 5В070400 «Есептеу техникасы және программалық қамтамасыз ету»,
5В070500 «Математикалық және компьютерлік моделдеу»

АЛГОРИТМЫ, СТРУКТУРЫ ДАННЫХ И ПРОГРАММИРОВАНИЕ

Методические указания к лабораторным работам, СРС и СРСП для специальностей 5В070400 «Вычислительная техника и программное обеспечение», 5В070500 «Математическое и компьютерное моделирование»

Форма обучения: очная

Усть-Каменогорск
2015

УДК 681.3 (06)

Никифоров В.Л. Алгоритмы, структуры данных и программирование: Методические указания разработаны для более глубокого изучения языка C# студентами специальностей 5В070400 «Вычислительная техника и программное обеспечение» и 5В070500 «Математическое и компьютерное моделирование» / ВКГТУ.- Усть-Каменогорск, 2015.- 61 с.

Методические указания содержат теоретический материал, большое количество примеров решения задач, вариантов индивидуальных заданий для СРС и вопросов, подготовка которых необходима для защиты отчетов лабораторных работ на СРСП. Изложенный материал будет способствовать глубокому усвоению теоретического курса дисциплины «Алгоритмы, структуры данных и программирование», формированию практических навыков по созданию программ на языке C# в консольном приложении средыVisual Studio.NET 2010.

Утверждено методической комиссией факультета информационных технологий и информатики.

Протокол № от 2015г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

При разработке данных методических указаний был использован опыт преподавания дисциплины «Алгоритмизация и языки программирования» и «Программирование на алгоритмических языках» в предыдущие годы.

Методические указания включают теоретический материал, большое количество примеров решения задач, вариантов индивидуальных заданий для СРС и вопросов, подготовка которых необходима для защиты отчетов лабораторных работ на СРСП.

Указания к самостоятельной работе студента предполагают предварительное изучение лекционного материала каждой изучаемой темы дисциплины. Изучение методических указаний к лабораторным работам соответствующей темы дисциплины с его практическим освоением на компьютере. Без этой предварительной подготовки попытки ответить на контрольные вопросы темы или выполнить домашнее задание – обречены на провал.

Работая с данными методическими указаниями, Вы должны ответить на все контрольные вопросы изучаемой темы. Если некоторые вопросы вызывают затруднения, то Вы должны повторить теоретический материал соответствующего раздела темы. Если у Вас возникли затруднения с пониманием теоретического материала, то Вы должны прийти на консультацию к преподавателю в его часы СРСП.

После ответа на контрольные вопросы Вы должны перейти к выполнению домашнего задания изучаемой темы дисциплины. На первом этапе Вы должны осмыслить требования домашнего задания и разработать алгоритм его выполнения. На втором этапе Вы должны разработать программу, выполнить ее отладку на компьютере и провести ее тестирование. Законченное домашнее задание необходимо подготовить к показу преподавателю.

После успешного выполнения домашнего задания попытайтесь самостоятельно выполнить одно из индивидуальных заданий изучаемого модуля дисциплины.

Перечисленный порядок действий необходимо выполнятьпри самостоятельном изучении всех тем изучаемой дисциплины!

Необходимо отметить, что данные методические указания дополняет лекционный материал, но не являются подробным описанием всех аспектов дисциплины. Поэтому студентам предложен список литературы, который они могут использовать для более глубокого изучения материала дисциплины.

ТЕМА 1 ПРОСТЫЕ ОПЕРАТОРЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ C#

1.1Цель первой темы

Знакомство с основами среды Visual Studio.NET и приобретение практических навыков по созданию консольных приложений. Освоение программирования на языке C# с использованием простые операторы.

1.2 Теоретические сведения

1.2.1 Введение

Язык C# является наиболее известной новинкой в области языков программирования. Это первый язык программирования, созданный в 21-м веке. Язык признан международным сообществом. В июне 2006 года Европейской ассоциацией по стандартизации принята уже четвертая версия стандарта этого языка: Standard ECMA-334 C# LanguageSpecifications, 4-th edition.

Руководителем группы, создающей язык C#, является сотрудник Microsoft Андреас Хейлсберг. Он был известен в мире программистов задолго до того, как пришел в Microsoft. Хейлсберг входил в число ведущих разработчиков одной из самых популярных визуальных сред программирования Delphi.

Можно отметить следующие основные достоинства языка C#:

– язык C#создавался и развивается параллельно с каркасом Framework.Net и в полной мере учитывает все его возможности;

– язык C# является полностью объектно-ориентированным языком;

– благодаря каркасу Framework.Net, ставшему надстройкой над операционной системой, программисты языка C# получают преимущества работы с виртуальной машиной;

– Framework.Net поддерживает разнообразие типов приложений на языке C#;

– реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху языка C#.

1.2.2 Система типов языка C#

Стандарт языка C# включает следующий набор фундаментальных типов:

– логический тип (bool);

– символьный тип (char);

– целые типы. Целые типы могут быть одного из трех размеров - short, int, long, сопровождаемые описателем signed или unsigned, который указывает, как интерпретируется значение - со знаком или без оного;

– вещественные типы. Эти типы также могут быть одного из трех размеров - float, double, longdouble;

– тип void, используемый для указания на отсутствие информации.

Язык позволяет конструировать типы:

– указатели (например, int* - типизированный указатель на переменную типа int);

– ссылки (например, double& - типизированная ссылка на переменную типа double);

– массивы (например, char[] - массив элементов типа char).

Язык позволяет конструировать пользовательские типы:

– перечислимые типы (enum) для представления значений из конкретного множества;

– структуры (struct);

– классы (class).

Описание основных типов языка C# приведено в таблице 1.1

Таблица 1.1 Основные типы данных языка C#

Имя типа	Системный тип	Значения	Размер
bool	System.Boolean	true, false	8 бит
sbyte	System.SByte	[-128, 127]	Знаковое, 8-бит
byte	System.Byte	[0, 255]	Беззнаковое, 8-бит
short	System.Short	[-32768, 32767]	Знаковое, 16-бит
ushort	System.UShort	[0, 65535]	Беззнаковое, 16-бит
int	System.Int32	$[-2^{31}, 2^{31}]$	Знаковое, 32-бит
uint	System.UInt32	$[0, 2^{32}]$	Беззнаковое, 32-бит
long	System.Int64	$[-2^{63}, 2^{63}]$	Знаковое, 64-бит
ulong	System.UInt64	$\approx (0, 2^{64})$	Беззнаковое, 64-бит
float	System.Single	$[10^{-45}, 10^{38}]$	7 цифр
double	System.Double	$[10^{-324}, 10^{308}]$	15-16 цифр
decimal	System.Decimal	$[10^{-28}, 10^{28}]$	28-29 значащих цифр
char	System.Char	[U+0000, U+ffff]	16-бит Unicode символ
string	System.String	Строка из символов Unicode	

1.2.3 Выражения

Выражения строятся из операндов – констант, переменных, функций объединенных знаками операций и скобками. При вычислении выражения определяется его значение и тип. Эти характеристики выражения однозначно определяются значениями и типами операндов, входящих в выражение, и правилами вычисления выражения. Правила задаются:

– приоритетом операций (для операций одного приоритета порядок применения – слева направо или справа налево);

– преобразование типов операндов и выбор реализации для перегруженных операций;

– тип и значение результата выполнения операции над заданными значениями операндов определенного типа.

Приоритет и порядок выполнения операций приведен в таблице 1.2

Таблица 1.2 Приоритет и порядок выполнения некоторых операций в C#

Приоритет	Категория	Операции	Порядок
0	Первичные	x.y, f(x), a[x], x++, x--, new	→
1	Унарные	+, -, !, ++x, --x, (T)x	→
2	Мультипликативные	*, /, %	→
3	Аддитивные	+, -	→
4	Сдвиг	<< ,>>	→
5	Отношения, проверка типов	<, >, <=, >=, is, as	→
6	Эквивалентность	==, !=	→
7	Логическое И (AND)	&	→
8	Логическое ИЛИ (OR)	|	→
9	Условное логическое И	&&	→
10	Условное логическое ИЛИ	||	→
11	Условное выражение	? :	←
12	Присваивание	=, *=, /=, %=, +=, -=	←

Любое выражение, взятое в скобки, получает высший приоритет и должно быть вычислено, прежде чем к нему будут применимы какие-либо операции. Скобки позволяют изменить стандартный порядок вычисления выражения и установить порядок, необходимый для вычисления в данном конкретном случае. В сложных выражениях скобки полезно расставлять даже в том случае, если стандартный порядок совпадает с требуемым, поскольку наличие "лишних" скобок зачастую увеличивает наглядность записи выражения.

1.2.4 Преобразование типов

Операция кастинга - приведения операндов к одному типу. В ходе вычисления выражения может возникать необходимость выполнения преобразования типов операндов. По возможности эти преобразования выполняются автоматически, неявно для программиста. Но неявные преобразования ограничены, поскольку могут быть только безопасными. Когда же нужно выполнить опасное преобразование, программист должен задать его явно. Одна из возможностей явного задания преобразования типа состоит в применении операции приведения к типу, называемой также кастингом. Это унарная операция со следующим синтаксисом:

(тип) выражение;

Здесь в скобках указывается тип, к которому следует привести выражение, стоящее за скобками. Нужно понимать, что не всегда существует явное приведение типа источника к типу цели. Операция кастинга применима только для приведения типов внутри арифметического типа. С ее

помощью один арифметический подтип можно привести к другому подтипу, но нельзя, например, целочисленные типы привести к логическому типу bool. Рассмотрим примеры приведения типа:

```
byteb1 = 1, b2 = 2, b3;
//b3 = b1 + b2;
b3 = (byte)(b1 + b2);
```

В этом примере необходимо сложить две переменные типа byte и, казалось бы, никакого приведении типов выполнять не нужно, результат будет также иметь тип byte, согласованный с левой частью оператора присваивания. Однако это не так по той простой причине, что отсутствует операция сложения над короткими числами. Реализация сложения начинается с типа int. Поэтому перед выполнением сложения оба операнда неявно преобразуются к типу int, результат сложения будет иметь тип int, и при попытке присвоить значение выражения переменной типа byte возникнет ошибка периода компиляции. По этой причине оператор во второй строке кода закомментирован. Программист должен явно привести выражение к типу byte, что и демонстрирует третья строка кода, в которой использована операция приведения к типу.

В следующем фрагменте кода демонстрируется еще один пример приведения типа:

```
 int i1,i2;
i1 = -40;
i2 = (int)(1.8 * i1) + 32;
```

Результат умножения имеет тип double по типу первого операнда. Перед тем как выполнять сложение, результат приводится к типу int. После приведениятипа для первого слагаемого (int)(1.8 * i1) сложение будет выполняться над целыми числами, результат будет иметь тип int, и не потребуется никаких преобразований для присвоения полученного значения переменной i2. Если убрать приведение типа в этом операторе, то возникнет ошибка на этапе компиляции.

1.3 Пример выполнения задания на лабораторную работу

Написать программу, которая меняет местами значения переменных a иb. Исходное значение переменной a формируется случайным образом в диапазоне от 0 до 20, а значение переменной bвводится в режиме диалога. Предусмотреть вывод значений переменных до и после обмена их значений. Программу выполнить в консольном приложении, которое разместить на рабочем столе компьютера.

Запускаем среду программирования Visual Studio.NET(см. Рисунок 1.1).

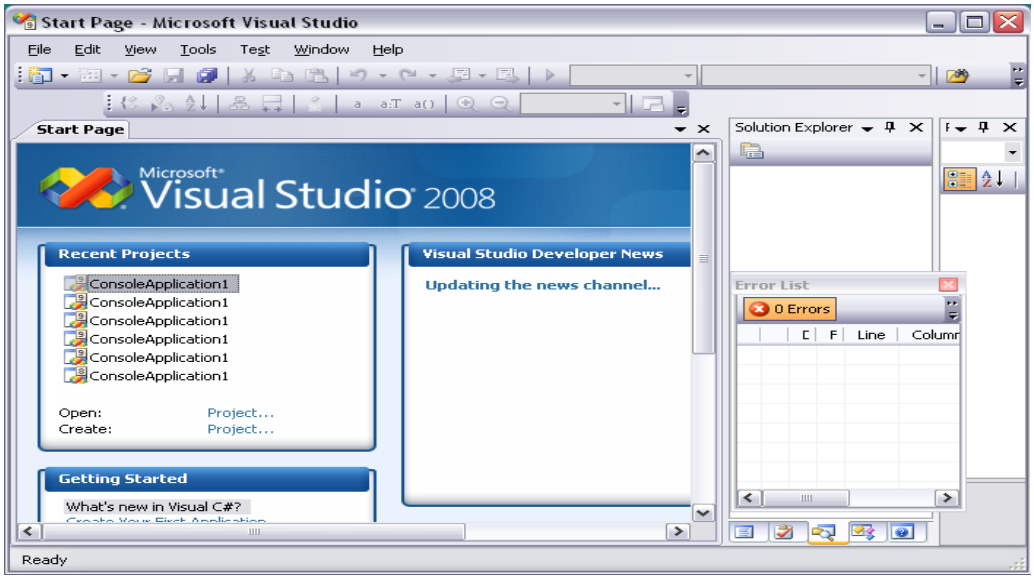

Рисунок 1.1 – Окно 1 – запуска Visual Studio.NET

В окнерисунка 1.1 выбираем команды File/New/Projekt…

Открывается новое окно (NewProject), в котором можно задать тип создаваемого проекта (смотри рисунок 1.2).

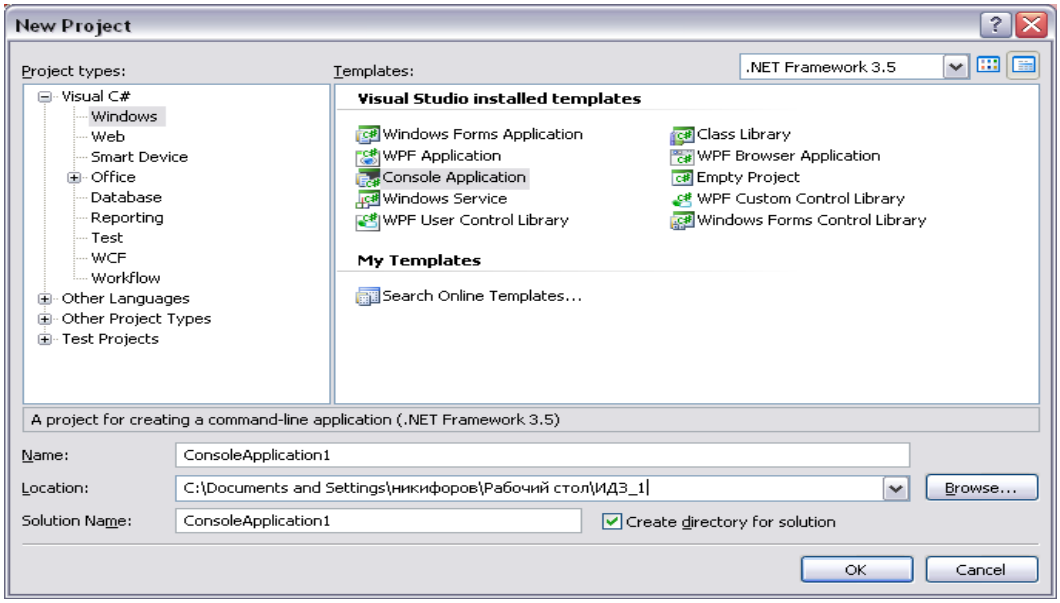

Рисунок 1.2 – Окно 2 – создания нового проекта

В окне 2 выбираем тип проекта ConsoleApplication и задаем место создания папки, в которой будут размещаться файлы создаваемого проекта (Рабочий стол папка ИДЗ_1). Остальные настройки окна можно не менять.

Нажимаем кнопку ОК и переходим к окну 3.

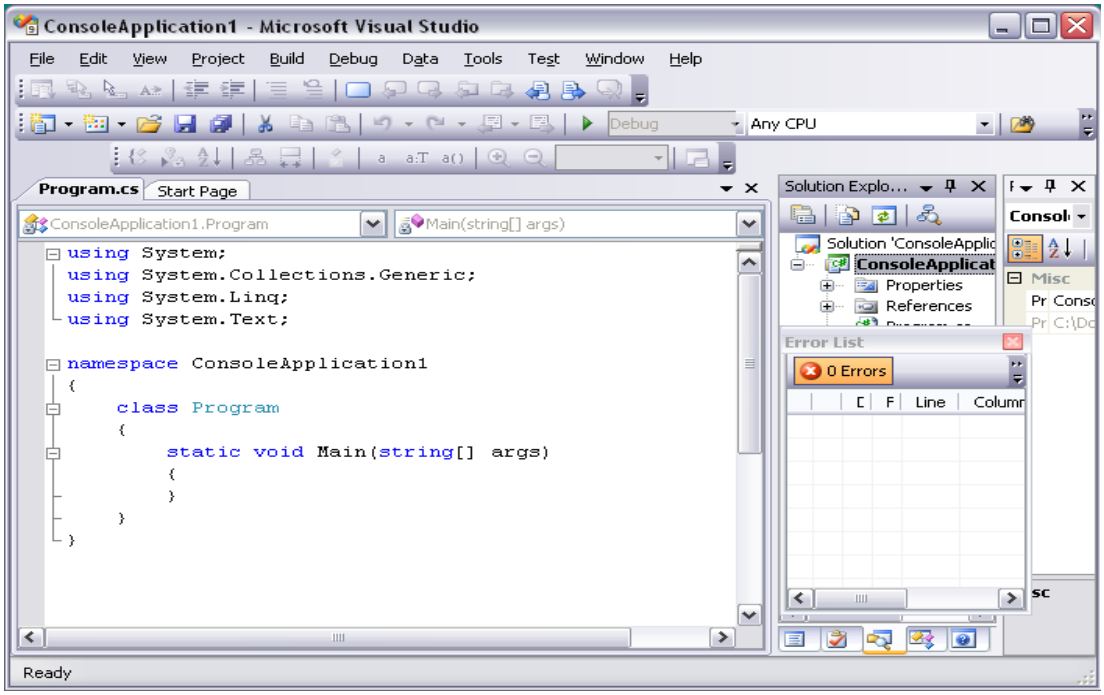

Рисунок 1.3 – Окно 3 среды Visual Studio.NET

В окне 3 на странице редактора Program.cs будет набираться программный код решения задачи.

Алгоритм решения задачи очень простой:

- необходимо организовать ввод в режиме диалога значения переменной a;
- сформировать случайным образом в заданном диапазоне значение переменной b:
- напечатать исходные значения этих переменных:
- выполнить обмен значений этих переменных:
- напечатать новые значения переменных аи b.

Для реализации перечисленных пунктов алгоритма необходимо изучить материал первых двух лекций дисциплины.

Следующим этапом индивидуального задания является построение структурной схемы алгоритма решения задачи.

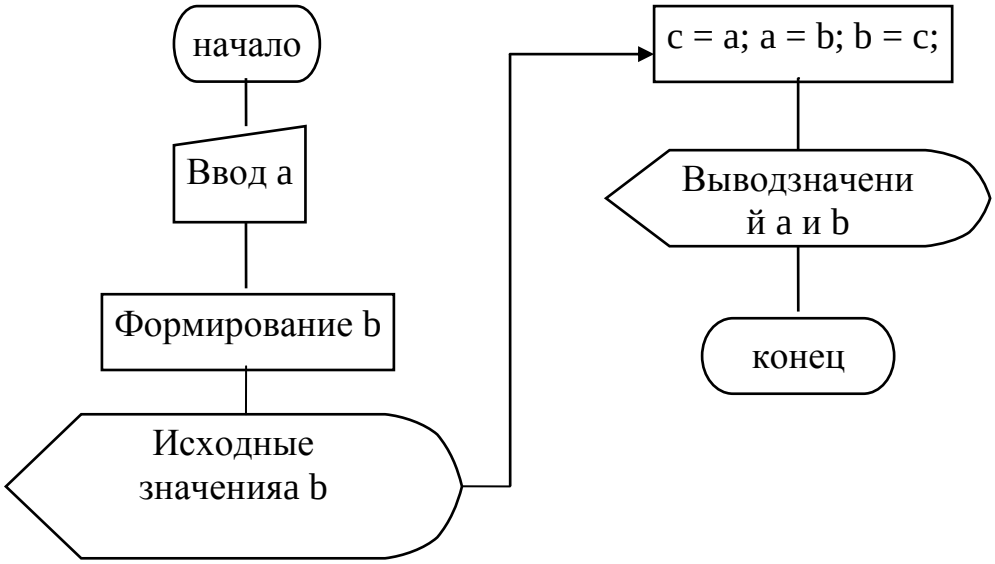

Рисунок 1.4 – Структурная схема алгоритма решения задачи

Используя структурную схему алгоритма решения задачи, разрабатываем код программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
classProgram
  {
staticvoid Main()
    {
int a, b, c;
string buf;
//Ввод значения переменной a в режиме диалога
Console.Write("Введите целое значение a ");
buf = Console.ReadLine();
     a = Convert.ToInt32(buf);
//Формирование случайным образом значения переменной b
Random rnd = newRandom();
     b = rnd.Next() % 21;
//Печать исходных значений
Console.WriteLine("Исходные значения переменных a и b:");
Console.WriteLine("a={0} b={1}", a, b);
//обмен
c = a; a = b; b = c;
//Печать значений переменных a и b после обмена
Console.WriteLine("Новые значения переменных a и b:");
Console.WriteLine("a={0} b={1}", a, b);
```

```
// Задержка рабочего экрана монитора
Console.WriteLine("Для продолжения нажмите клавишу Enter");
Console.ReadLine();
     }
   }
}
```

Работа программы:

Введите целое значение a 35
Исходные значения переменных a и b:
a=35 b=13
Новые значения переменных a и b:
a=13 b=35
Для продолжения нажмите клавишу Enter

При оформлении отчета по лабораторной работе рекомендуется следующая структура и последовательность элементов:

– титульный лист;
– название лабораторной работы;
– цель лабораторной работы;
– индивидуальное задание на лабораторную работу;
– краткие комментарии по выполнению индивидуального задания (при необходимости структурная схема алгоритма решения задачи);
– необходимый программный код индивидуального задания;
– результаты работы программы;
– выводы.

Титульный лист является первой страницей отчета и служит источником информации, необходимой для поиска документа. Поэтому он содержит название министерства, название университета, название кафедры, название дисциплины, название модуля дисциплины, ФИО студента, выполнившего лабораторную работу и ФИО преподавателя, принимающего отчет по лабораторной работе. Внизу титульного листа указывается место и год выполнения лабораторной работы, например, Усть–Каменогорск 2010г.

Титульный лист включают в общую нумерацию страниц отчета, но номер страницы на титульном листе не проставляется.

Название лабораторной работы должно соответствовать названию из методических указаний по выполнению лабораторных работ, часто оно соответствует названию модуля.

Цель лабораторной работы содержит краткое описание цели соответствующего модуля дисциплины.

Индивидуальное задание на лабораторную работу содержит полный текст индивидуального задания, полученного у преподавателя только после выполнения домашнего задания по лабораторной работе.

Краткие комментарии по выполнению индивидуального задания содержат описание алгоритма выполнения индивидуального задания. При

необходимости приводится структурная схема алгоритма или его подробное словесное описание.

Необходимый программный код индивидуального задания содержит либо полный текст кода программы, либо фрагменты кода программы, разработанные студентом и коды добавляемые средой программирования, без которых объяснение выполненной работы невозможно.

Результаты работы программы обычно содержат копии окон работы программы во всех ее режимах.

В выводах обычно отмечается результат выполнения лабораторной работы.

Страницы текста отчета должны соответствовать формату А4.

Печатание отчета выполняется машинописным способом или с применением печатающих и графических устройств вывода ЭВМ на одной стороне листа белой бумаги. Тип шрифта - TimesNewRoman, основной размер шрифта - № 14, допускается № 12. Основной интервал -1, допускается -1,5.

Текст отчета следует печатать, соблюдая следующие разделы полей: правое, верхнее, нижнее и левое – 20 мм.

Абзацный отступ начинается не менее чем с четвертого знака и должен быть одинаков в пределах одного документа.

Вне зависимости от способа выполнения отчета, качество напечатанного текста, иллюстраций, таблиц и приложений должно удовлетворять требованию их чёткого воспроизведения.

Вписывать в отпечатанный текст отчета отдельные слова, формулы и знаки допускается только черными чернилами или черной тушью, при этом плотность вписанного текста должна быть максимально приближена к плотности основного текста.

Опечатки, описки и графические неточности допускается исправлять подчисткой или закрашиванием белой краской и нанесением на том же месте исправленного изображения машинописным способом или от руки черными чернилами лили чёрной тушью.

Повреждения листов отчета, помарки и следы не полностью удалённого текста не допускаются.

Разделы должны иметь порядковые номера в пределах всего отчета, обозначенные арабскими цифрами без точки и записанные с отступом. Подразделы должны иметь нумерацию в пределах каждого раздела. Номер подраздела состоит из номеров раздела и подраздела, разделённых точкой. В конце номера подраздела точки не ставятся. Разделы, как и подразделы, могут состоять из одного или нескольких пунктов.

Разделы, подразделы должны иметь заголовки. Пункты заголовков не имеют.

Заголовки разделов документа следует располагать в середине строки без точки в конце и печатать прописными буквами, не подчёркивая и не выделяя.

Заголовки подразделов документа следует располагать в середине строки и печатать строчными буквами, начиная с первой прописной, выделяя жирным шрифтом.

Если раздел не имеет подразделов, то нумерация пунктов в нем должна быть в пределах этого раздела, и номер пункта должен состоять из номеров раздела и пункта, разделённых точкой. В конце номера пункта точка не ставится.

Если раздел имеет подразделы, то нумерация пунктов должна быть в пределах каждого подраздела и номер пункта должен состоять из номеров раздела, подраздела и пункта, разделённых точками.

Страницы отчета следует нумеровать арабскими цифрами, соблюдая сквозную нумерацию по всему тексту.

Номер страницы отчета проставляют по центру страницы вверху без точки в конце.

Страницы отчета скрепляются скрепкой или размещаются в файле (степлер не использовать).

1.4Домашнее задание на лабораторную работу

Написать программу вычисления длины отрезка, заданного координатами его концов A(x1, y1) и B(x2, y2). Значения координат точки A вводить в режиме диалога, а точки Взадавать случайным образом в диапазоне минус 100 плюс 100. Предусмотреть вывод результатов работы программы на экран монитора.

1.5 Индивидуальные задания для СРС

1.5.1 Индивидуальное задание 1 студента:

1.5.1.1 Написать программу вычисления выражения:

$$Y=3*(X+1)3+4*(X-1)+2.$$

Значение X задавать в режиме диалога. Методом перебора попытайтесь найти такое X при котором Y равен нулю. Предусмотреть вывод результата на экран монитора.

1.5.1.2 Написать программу, которая в вещественном числе, введённом в режиме диалога, выделяет целую и дробную части, и отдельно выводит их на экран монитора.

1.5.1.3 В режиме диалога заданы координаты вершин треугольника. Найти периметр треугольника.

1.5.2 Индивидуальное задание 2 студента:

1.5.2.1 Написать программу вычисления суммы:

$$S=1+1/2+1/3+1/4+1/5.$$

Предусмотреть вывод результата на экран монитора.

1.5.2.2 Написать программу вычисления площади треугольника, стороны которого A, B, C вводятся в режиме диалога. Предусмотреть вывод результата на экран монитора.

1.5.2.3 Написать программу вычисления длины ломаной линии состоящей из трех отрезков, заданных координатами их концов A(x,y), B(x,y), C(x,y) и D(x,y). Значения координат формировать случайным образом в диапазоне от 0 до 100.

1.5.3 Индивидуальное задание 3 студента:

1.5.3.1 Написать программу вычисления выражения:

$$C = A^2 + \sqrt{(A + LnB)/2}\,.$$

Значения переменных A и B, необходимые для решения данного выражения, вводятся в режиме диалога.

1.5.3.2 Написать программу, которая реализует алгоритм обмена значении переменных A и B (без использования промежуточной переменной c). Исходные значения переменных вводить в режиме диалога. Предусмотреть вывод результата до и после обмена.

1.5.3.3 Написать программу вычисления площади круга. Значение радиуса вводится в режиме диалога с ЭВМ и должно быть больше нуля.

1.5.4 Индивидуальное задание 4 студента:

1.5.4.1 Длина отрезка задана в дюймах (1 дюйм = 2,54 см.), вводится в режиме диалога. Перевести значение длины в метрическую систему, то есть выразить её в метрах, сантиметрах и миллиметрах. Так, например, 21 дюйм = 0м 53см 3,4мм.

1.5.4.2 Длина окружности задается в режиме диалога. Найти площадь круга, ограниченного этой окружностью.

1.5.4.3 В режиме диалога заданы координаты двух противоположных вершин прямоугольника. Найти площадь прямоугольника.

1.5.5 Индивидуальное задание 5 студента:

1.5.5.1 Значение X задается в режиме диалога. Использую только арифметические операции вычислить выражение

Y = 1 - 2*X + 3*X2 - 4*X3.

Разрешается использовать не более восьми операций в одном выражении.

1.5.5.2 Угол α задаётся в режиме диалога в градусах, минутах и секундах. Найти его величину в радианах.

1.5.5.3 Длина ребра куба задается в режиме диалога. Найти объем куба и площадь его поверхностей.

1.5.6 Индивидуальное задание 6 студента:

1.5.6.1 Найти площадь кольца, внешний радиус которого равен 100, а внутренний задается в режиме диалога и должен быть меньше 100.

1.5.6.2 Длина стороны равностороннего треугольника задается в режиме диалога. Найти площадь этого треугольника.

1.5.6.3 Смешано v1 литров воды температуры t1 с v2 литрами воды температуры t2. Найти объем и температуру образовавшейся смеси. Значения v1, t1, v2 и t2 задаются в режиме диалога.

1.6 Контрольные вопросы для защиты отчета на СРСП

1.6.1 Что входит в понятие платформы.NET?
1.6.2 Что означает понятие общеязыковая среда выполнения (Common Language Runtime, CLR)?
1.6.3 Что понимается под термином «приложение»?
1.6.4 Что понимается под термином «проект»? Его состав?
1.6.5 Что включает в себя «Пространство имен»?
1.6.6 Что включает в себя среда Visual Studio.NET ?
1.6.7 Понятие алгоритма? Основные свойства алгоритма.
1.6.8 Этапы решения задачи?
1.6.9Какие типы данных языка C# Вы знаете?
1.6.10Какие знаки операций языка C# Вы знаете?
1.6.11Формат записи оператора присваивания языка C#? Пояснить его работа на примере.
1.6.12 Как можно организовать ввод данных «в режиме диалога»? Пример.
1.6.13 Как можно организовать вывод данных на экран монитора? Пример.
1.6.14 Как можно организовать вывод данных на экран монитора в заданном формате? Пример.
1.6.15 Как формируется случайное число в заданном диапазоне? Пример.

Тема 2СЛОЖНЫЕ ОПЕРАТОРЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ C#

2.1 Цель второй темы

Изучение сложных операторов языка C#. Приобретение практических навыков в программировании на языке C# с использованием сложных операторов.

2.2 Теоретические сведения

2.2.1 Оператор условия if

Как в и других языках программирования, в языке C# для выбора одной из нескольких возможностей продолжения программы используются два оператора –if и switch. Первый из них обычно называют оператором условия или оператором условного перехода, второй – оператором переключения программы. Программы, в которых используются оператор if или switch, часто называют программы использующие «ветвящийся» вычислительный процесс. Рассмотрим формат записи и работу каждого из этих операторов. Оператор if имеет следующий формат записи:

```
if(выражение) { операторы_1; }
        else { операторы_2; }
```

Логическое выражение if заключается в круглые скобки и может принимать значение true или false. Если логическое выражение равно true, то программа выполняет только операторы_1. Если логическое выражение равно false, то программа выполняет только операторы_2.

Если операторы_1 или операторы_2 представлены в единственном числе, то фигурные скобки для них можно не использовать.

Часть оператора if, за словом else, может отсутствовать. В этом случае краткая форма записи оператора if задает альтернативный выбор – делать или не делать – выполнять или не выполнять операторы_1. Если внутри фигурных скобок оператора if находятся другие операторы if, то часть else, если она есть, относится к ближайшему перед ней открытому оператору if.

2.2.1 Оператор переключения программы switch

Оператор switch применяется, когда из нескольких вариантов продолжения программы необходимо выбрать вариант, определяемый значением некоторого выражения. Формат записи оператора switch имеет следующий вид:

```
switch (выражение)
{
case константное_выражение_1: [операторы_1 оператор_перехода_1]
        …
case константное_выражение_K: [операторы_Kоператор_перехода_K]
  [default: операторы_Nоператор_перехода_N]
}
```

Ветвь default может отсутствовать. В соответствии с форматом записи допустимо, чтобы после двоеточия следовала пустая последовательность операторов, а не последовательность, заканчивающаяся оператором перехода. Константные выражения в case должны иметь тот же тип, что и switch-выражение.

Работа оператора switch следующая:

– вычисляется значение switch-выражения;

– затем оно поочередно в порядке следования case сравнивается на совпадение с константными выражениями:

– как только достигнуто совпадение, выполняется соответствующая последовательность операторов case-ветви. Поскольку последний оператор этой последовательности является оператором перехода (чаще всего это оператор break), обычно он завершает выполнение оператора switch;

–если case-ветвь была пустой последовательностью операторов. Тогда в случае совпадения константного выражения этой ветви со значением switch-выражения будет выполняться первая непустая последовательность следующей case-ветви;

–если значение switch-выражения не совпадает ни с одним константным выражением, то выполняется последовательность операторов ветви default;

– если нет ветви default, то оператор switch эквивалентен пустому оператору.

В операторе switch каждая case-ветвь должна заканчиваться оператором перехода (забудем на минуту о пустой последовательности), иначе возникнет ошибка периода компиляции.

В формате оператора switch case-выражения могут быть только константным выражением. Не допускается задания диапазона case-выражений или их перечислений.

Если необходимо указать, что нескольким вариантам case-выражений соответствует одна группа операторов, эти case-выражений записываются «пустыми» перед последним case-выражением с этой группой операторов. Например, в случае использования в switch выражения типа string для задания арифметической операции между двумя аргументами, возможны несколько вариантов ответа для выбора операций +, -, *, и /, которые можно представить следующим программным кодом:

```
// Разбор случаев с использованием списков выражений
//arg1 – первый аргумент операции
//arg2 – второй аргумент операции
// result – результат операции
switch (operation)
   {
case "+":
case "Plus":
case "Плюс":
result = arg1 + arg2;
break;
case "-":
case "Minus":
case "Минус":
result = arg1 - arg2;
break;
case "*":
case "Mult":
case "Умножить":
result = arg1 * arg2;
break;
case "/":
case "Divide":
case "Div":
case "разделить":
case "Делить":
result = arg1 / arg2;
```

```
break;
default:
result = 0;
break;
   }
```

Обратите внимание, знак операции над аргументами можно задавать разными способами, что демонстрирует возможность задания списка константных выражений в ветвях оператора switch.

2.2.3 Оператор цикла for

Оператор цикл for, также как оператор if, присутствует во всех языках программирования. Его формат записи имеет следующий вид:

```
for(инициализаторы; условие; список выражений)
{ оператор; }
```

Операторы, стоящие в фигурных скобках, задает тело цикла. Сколько раз будет выполняться тело цикла, зависит от трех управляющих элементов, заданных в круглых скобках. Инициализаторы задают начальное значение одной или нескольких переменных, часто называемых счетчиками или просто переменными цикла. Условие задает условие окончания цикла, которое может принимать значение true или false. Список выражений, записанный через запятую, показывает, как меняются счетчики цикла на каждом шаге выполнения. Если условие цикла истинно, то выполняется тело цикла, затем изменяются значения счетчиков и снова проверяется условие. Как только условие становится ложным, цикл завершает свою работу. В цикле for тело цикла может ни разу не выполняться, если условие цикла ложно после инициализации, а может происходить зацикливание, если условие всегда остается истинным. В нормальной ситуации тело цикла выполняется конечное число раз.

Счетчики цикла зачастую объявляются непосредственно в инициализаторе и соответственно являются переменными, локализованными в цикле, так что после завершения цикла они перестают существовать. В тех случаях, когда предусматривается возможность преждевременного завершения цикла с помощью одного из операторов перехода, счетчики объявляются до цикла, что позволяет анализировать их значения при выходе из цикла.

2.2.4 Циклы с условием

Цикл while (выражение) является универсальным видом цикла, включаемым во все языки программирования. Тело цикла выполняется до тех пор, пока остается истинным выражение while. В языке C# у этого вида цикла две модификации - с проверкой условия в начале и в конце цикла. Первая модификация имеет следующий синтаксис:

```
while(выражение) { оператор; }
```

Эта модификация соответствует стратегии: "сначала проверь, а потом делай". В результате проверки может оказаться, что и делать ничего не нужно. Тело такого цикла может ни разу не выполняться. Конечно же, возможно и зацикливание. В нормальной ситуации каждое выполнение тела цикла - это очередной шаг к завершению цикла.

Цикл, проверяющий условие завершения в конце, соответствует стратегии: "сначала делай, а потом проверь". Тело такого цикла выполняется, по меньшей мере, один раз. Вот синтаксис этой модификации:

```
do
 { оператор; }
while(выражение);
```

2.2.5 Операторы перехода

Операторов перехода, позволяющих прервать естественный порядок выполнения операторов в сложных операторах языкаC#. Они записываются вместе с другими операторами внутри фигурных скобок – теле сложного оператора или его блок. Их несколько и мы рассмотрим их поочередно.

Оператор goto имеет простой формат записи:

goto [метка|caseконстантное_выражение|default];

Все операторы языка C# могут иметь метку - уникальный идентификатор, предшествующий оператору и отделенный от него символом двоеточия. Передача управления помеченному оператору - это классическое применение оператора goto.

Операторы break и continue. В структурном программировании признаются полезными "переходы вперед" (но не назад), позволяющие при выполнении некоторого условия выйти из цикла, из оператора выбора, из блока. Операторы break и continue специально предназначены для этих целей. Оператор break может стоять в теле цикла или завершать case-ветвь в операторе switch. Пример его использования в операторе switch уже демонстрировался. При выполнении оператора break в теле цикла завершается выполнение самого внутреннего цикла. В теле цикла чаще всего оператор break помещается в одну из ветвей оператора if, проверяющего условие преждевременного завершения цикла.

Оператор continue используется только в теле цикла. В отличие от оператора break, завершающего внутренний цикл, continue осуществляет переход к следующей итерации этого цикла.

Еще одним оператором, относящимся к группе операторов перехода, является оператор return, позволяющий завершить выполнение процедуры или функции. Его формат записи имеет следующий вид:

```
return [выражение];
```

Для функций его присутствие и аргумент обязательны, поскольку выражение в операторе return задает значение, возвращаемое функцией.

2.3 Пример выполнения задания на лабораторную работу

В качестве примера выполнения задания на тему сложные операторы языка C# выбрано решение задачи средней сложности из списка индивидуальных заданий прошлых лет.

Задача 2.1 У Вас есть 1000 у.е. Вам необходимо купить акции 5 предприятий. Стоимость акции каждого предприятия формируется случайным образом в диапазоне от 5 до 20 у.е. (остается неизменным на время торгов). Ваша задача купить на всю сумму приблизительно одинаковое количество акций каждого предприятия. Вывести на экран монитора количество купленных акций каждого предприятия и оставшуюся сумму у.е.

Разработаем словесное описание алгоритма решения задачи:

– на первом этапе необходимо сформировать 5 случайных чисел в диапазоне от 5 до 20 – стоимость акций 5 предприятий:

– на втором этапе необходимо организовать цикл покупки акций каждого предприятия поочередно. Необходим цикл с условием – пока наших денег хватает на покупку хотя бы одной акции любого из пяти предприятий.

– в теле цикла необходимы проверки условий, что наших денег достаточно для покупки акции каждого предприятия в отдельности. Если условие выполняется, то мы увеличиваем счетчик купленных акций данного предприятия на 1, а наши деньги уменьшаем на стоимость акции этого предприятия;

– на последнем этапе (по окончании цикла) необходимо вывести на экран монитора количество акций купленных для каждого предприятия и остаток наших денег. Естественно остаток должен быть числом положительным и меньше стоимости акции любого из предприятий.

Исходный код программы:

```
usingSystem;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
classProgram
  {
staticvoid Main(string[] args)
    {
int a1,a2,a3,a4,a5,k1,k2,k3,k4,k5,c;
Random rnd = newRandom();
Console.WriteLine("Программа моделирования покупки акций у 5
предприятий");
// формируем случайным образом стоимость акций 5 предприятий
     a1 = rnd.Next()%16 + 5;
```

```
Console.WriteLine("Стоимость акции 1 предприятия = {0}", a1);
     a2 = rnd.Next() % 16 + 5;
Console.WriteLine("Стоимость акции 2 предприятия = {0}", a2);
     a3 = rnd.Next() % 16 + 5;
Console.WriteLine("Стоимость акции 3 предприятия = {0}", a3);
     a4 = rnd.Next() % 16 + 5;
Console.WriteLine("Стоимость акции 4 предприятия = {0}", a4);
     a5 = rnd.Next() % 16 + 5;
Console.WriteLine("Стоимость акции 5 предприятия = {0}", a5);
// Обнуляем счетчики купленных акций каждого предприятия
     k1=0; k2=0; k3=0; k4=0; k5=0;
// Наши деньги
     c = 1000;
// Запускаем цикл покупки акций
while ((c>=a1)||(c>=a2)||(c>=a3)||(c>=a4)||(c>=a5))
{
if (c>=a1) {k1++; c = c - a1;}
if (c>=a2) {k2++; c = c - a2;}
if (c>=a3) {k3++; c = c - a3;}
if (c>=a4) {k4++; c = c - a4;}
if (c>=a5) {k5++; c = c - a5;}
      }
Console.WriteLine("Купленоакций 1 предприятия = {0} ", k1);
Console.WriteLine("Куплено акций 2 предприятия = {0} ", k2);
Console.WriteLine("Куплено акций 3 предприятия = {0} ", k3);
Console.WriteLine("Куплено акций 4 предприятия = {0} ", k4);
Console.WriteLine("Куплено акций 5 предприятия = {0} ", k5);
Console.WriteLine("Остаток денег  = {0} ", c);
Console.WriteLine("Для продолжения нажмите клавишу Enter");
Console.ReadLine();
    }
  }
}
```

Работа программы:

Программа моделирования покупки акций у 5 предприятий
Стоимость акции 1 предприятия = 15
Стоимость акции 2 предприятия = 16
Стоимость акции 3 предприятия = 11
Стоимость акции 4 предприятия = 8
Стоимость акции 5 предприятия = 15
Куплено акций 1 предприятия = 16
Куплено акций 2 предприятия = 15
Куплено акций 3 предприятия = 15
Куплено акций 4 предприятия = 16
Куплено акций 5 предприятия = 15
Остаток денег = 2
Для продолжения нажмите клавишу Enter

2.4 Домашнее задание на лабораторную работу

Вычислить сумму ряда:

$$s = 1 + \frac{1}{2} + \frac{1}{3} - \frac{1}{4} - \frac{1}{5} + \frac{1}{6} + \frac{1}{7} - \frac{1}{8} - \frac{1}{9} + \ldots + \frac{1}{N}$$

где N – целое число и задается в режиме диалога.

2.5 Индивидуальные задания для СРС

2.5.1 Индивидуальное задание 1 студента:
2.5.1.1 В книге сто страниц. На странице может находиться от 35 до 45 строк (случайное число). В одной строке буква «а» может встречаться от двух до пяти раз (случайное число). Сколько букв «а» напечатано в книге?
2.5.1.2 Написать программу вычисления суммы ряда:

$$S = 1 + \frac{1}{2} - \frac{1}{3} + \frac{1}{4} + \ldots - \frac{1}{n}$$

Значение n вводиться в режиме диалога.
2.5.1.3 Случайным образом формируются координаты X и Y 100 точек. Диапазон значений координат от минус 150 до 150. Подсчитать и напечатать количество точек, расположенных на каждой четверти.
2.5.2 Индивидуальное задание 2 студента:
2.5.2.1 Вы читаете интересную книгу – 200 страниц. Чтение одной страницы занимает от 50 до 80 секунд. Вероятность размещения на странице иллюстраций – 5%. Время просмотра иллюстрации – от 5 до 10 секунд. Сколько чистого времени понадобится для чтения всей книги.
2.5.2.2 Вычислить сумму ряда для заданного в режиме диалога с ЭВМ |x|<1.

$$y = \frac{2!}{x^2 * 3!} + \frac{3!}{x^4 * 4!} + \frac{4!}{x^6 * 5!} + \ldots$$

Вычисления продолжать до тех пор, пока очередной член ряда не становится меньше 0.0001.
2.5.2.3 Случайным образом формируются координаты X и Y центра и R – радиус 50 кругов. Диапазон значений координат от минус 150 до 150, диапазон значения радиуса от 5 до 15. Определить и напечатать, сколько кругов полностью находится в каждой четверти.
2.5.3 Индивидуальное задание 3 студента:
2.5.3.1 Вычислить сумму ряда для заданного в режиме диалога x (x>0 и x<1). Вычисления заканчиваются, когда очередной член ряда становится меньше введенной точности ε=0.0001:

$$\sqrt[3]{1+x} = 1 + \frac{1}{3}x - \frac{2}{2!*3^2}x^2 + \frac{2*5}{3!*3^3}x^3 - \frac{2*5*8}{4!*3^4}x^4 + \ldots$$

2.5.3.2 Автобус за смену делает от 10 до 15 рейсов (случайное число). За рейс перевозит от 130 до 230 пассажиров (случайное число). Стоимость проезда одного пассажира 30 тенге. Вероятность того, что у пассажира удостоверение, проездной или он «заяц» - 30%. Определить доход, полученный за N смен. Число смен вводится в режиме диалога.

2.5.3.3 На книжной полке 20 учебников по истории, физике и химии. Вероятность, что учебник по истории – 40%, по химии – 25%, по физике – 35%. Определить и вывести на экран монитора, сколько учебников по истории, химии и физике?

2.5.4 Индивидуальное задание 4 студента:

2.5.4.1 Вычислить сумму ряда для заданного x < 0. Вычисления заканчиваются, когда очередной член ряда по модулю становится меньше 0.0001:

$$arctg(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

2.5.4.2 Цеху необходимо выполнить объем работ не менее 20 тыс. единиц продукции. Каждый день на работу выходит от 10 до 20 рабочих (случайное число). Производительность каждого рабочего от 5 до 15 единиц в день (случайное число). Сколько дней необходимо затратить на выполнение всего объема работ.

2.5.4.3 Натуральное число X, задано в режиме диалога. Определить сколько цифр в числе. Найти и напечатать первую и последнюю цифры числа.

2.5.5 Индивидуальное задание 5 студента:

2.5.5.1 Вычислить сумму ряда для заданного |x| > 0.

$$\ln\frac{x+1}{x-1} = 2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \ldots\right) \quad |x| > 1$$

Вычисления заканчиваются, когда очередной член ряда по модулю становится меньше 0.0001.

2.5.5.2 Некоторый студент опаздывает на каждое занятие от 3 до 10 минут (случайное число). В неделе 20 занятий. На какой неделе он «наберет» 20 часов опозданий?

2.5.5.3 Случайным образом формируются координаты A(X,Y) и B(X,Y) ста прямоугольников заданных противоположными вершинами. Диапазон значений координат от минус 150 до 150. Подсчитать и напечатать количество прямоугольников расположенных в верхней и нижней четвертях системы координат (если вершины расположены в разных половинах, то этот вариант исключается из рассмотрения).

2.5.6 Индивидуальное задание 6 студента:

2.5.6.1 Исследовать функцию

$$\operatorname{arctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} \ldots \quad x > 1$$

на отрезке от 2 до 10. Вычисления заканчиваются, когда очередной член ряда по модулю становится меньше 0.0001.

2.5.6.2 Случайным образом формируются координаты X и Y 90 точек. Диапазон значений координат от минус 150 до 150. Вывести список точек, расстояние между которыми и началом координат минимально, а сами точки находятся в разных четвертях.

2.5.6.3 В системе координат X, Y «нарисована» мишень на 10 кругов с центром в начале координат и шагом радиуса 10 единиц. Кругу с радиусом 10 единиц соответствует весовое значение в 10 баллов. Для каждого следующего «кольца» мишени баллы уменьшаются от 9 до 1. Случайным образом формируются координаты X и Y 10 точек (десять выстрелов). Диапазон значений координат каждой точки от минус 150 до 150. Определить и напечатать, сколько баллов набирает каждый выстрел и общую сумму баллов всей серии выстрелов.

2.6 Контрольные вопросы для защиты отчета на СРСП

2.6.1 Какие операторы языка программирования C# называют сложными операторами?

2.6.2Как выделяется область действия сложного оператора языка C#?

2.6.3 Какой вычислительный процесс называют «ветвящимся»?

2.6.4 Какой вычислительный процесс называется циклическим?

2.6.5 Формат записи и работа оператора условного перехода if.

2.6.6Формат записи и работа оператора переключения программы switch.

2.6.7 Формат записи и работа оператора цикла for.

2.6.8 Когда целесообразно применять оператор цикла for?

2.6.9 При каких условиях операторы, принадлежащие циклу for, не выполнятся ни одного раза? Пояснить на примере.

2.6.10 Формат записи и работа оператора цикла while

2.6.11 При каких условиях операторы, принадлежащие циклу while, не выполнятся ни одного раза? Пояснить на примере.

2.6.12 Формат записи и работа оператора цикла do – while.

2.6.13 При каких условиях операторы, принадлежащие циклу do – while, не выполнятся ни одного раза?

2.6.14 Можно ли организовать «циклический» вычислительный процесс без операторов цикла? Пояснить на примере.

2.6.15 Какие операторы перехода вы знаете?

ТЕМА 3 ОДНОМЕРНЫЕ МАССИВЫ В ЯЗЫКЕ C#

3.1 Цель третьей темы

Изучение способов организации данных в виде одномерных массивов языка C#. Приобретение практических навыков в программировании на языке C# задач, использующих одномерные массивы.

3.2 Теоретические сведения

3.2.1 Понятие массива

Массив задает способ организации переменных одного типа. Иногда массивом называют упорядоченную совокупность переменных одного типа. Каждая переменная массива имеет индекс – номер переменной в массиве. В языке C#, как и во многих других языках, индексы задаются целочисленным типом. Первая переменная массива имеет 0 индекс, Nпеременная – N-1 индекс.

В некоторых языках программирования при объявлении массива задается число переменных массива. Если число элементов массива известно в момент его объявления и ему может быть выделена память еще на этапе трансляции, то такие массивы называются статическими.

Работа с массивами в языке C# выполняется в два этапа. На первом этапе объявляется тип переменных, объединяемых в массив. На втором этапе – во время работы программы выполняется инициализация массива, т.е. определяется число элементов массива. Поэтому все массивы в языке C# являются динамическими. При инициализации массива ему, как правило, выделяется непрерывная область памяти в куче.

3.2.2 Объявление массива

Объявление одномерных массивов имеет следующий формат:

<тип>[]<объявители>;

где <тип> – тип переменных, объединяемых в массив;

[]– признак одномерного массива;

<объявители> – перечень переменных, объявляемых как переменные массива. Каждый объявитель может быть именем или именем с инициализацией. В первом случае речь идет об отложенной инициализации. Нужно понимать, что при объявлении с отложенной инициализацией сам массив не формируется, а создается только ссылка на массив, имеющая неопределенное значение.Поэтому пока массив не будет реально создан и его элементы инициализированы, использовать его в вычислениях программы нельзя.

Например, объявление трех массивов с отложенной инициализацией имеет следующий вид:

```
int[] a, b, c;
```

Можно каждый массив с отложенной инициализацией объявлять отдельно, например:

```
int[] a;
int[] b;
int[] c;
double[] d;
```

Квадратные скобки приписаны не к имени переменной, а к типу. Они являются неотъемлемой частью определения типа и запись int[]следует понимать как тип, задающий одномерный массив с элементами целого типа.

Что же касается границ изменения индексов, то эта характеристика не является принадлежностью типа, она является характеристикой переменных данного типа – экземпляров, каждый из которых является одномерным массивом со своим числом элементов,задаваемых в объявителе переменной.

3.2.2 Инициализация массива

Если массив объявляется без инициализации, то создается только ссылка на массив со значением void.

Если инициализация выполняется конструктором, то в динамической памяти создается сам массив (выделяется место под массив), элементы которого инициализируются константами соответствующего типа (ноль для арифметики, пустая строка для строковых массивов), и ссылка связывается с этим массивом.

Например:

```
//объявление массивов с отложенной инициализацией
  int[] u;
  u = newint[3];
```

В приведенном примере первоначально объявляется массив целого типа, а затем (во время работы конструктора массива – new) в куче выделяется место под массив на три значения целого типа, в которые записываются нули.

Если массив инициализируется константным массивом, то в памяти создается константный массив, с которым и связывается ссылка.Элементы константного массива необходимо заключать в фигурные скобки, например:

```
int[] x ={5,5,6,6,7,7};//размерность вычисляется, new подразумевается
int[] x = newint[]{5,5,6,6,7,7};   // размерность вычисляется
int[] x = newint[6] {5,5,6,6,7,7}; // считается избыточным описанием без ошибки
```

Чаще в программах массив сначала объявляется, инициализируется (с обязательным указанием количества элементов), а затем заполняется значениями (нулевые значения уже присвоены). Например:

```
int[] b = new int[6]; // элементыравны 0
   Random rnd = new Random();
// формируем случайным образом и выводим его на экран монитора
```

```
for (inti = 0; i< 6; i++)
        {
b[i] = rnd.Next()%101;
Console.Write(" {0}", b[i]);
};                              //Символ ; можно не ставить
       Console.WriteLine();
```

3.3 Пример выполнения задания на лабораторную работу

В качестве примера выполнения задания на тему массивы языка C# выбрано решение простой задачи из списка индивидуальных заданий прошлых лет.

Задача 3.1Сформировать массив 20 целых случайных чисел в диапазоне от минус 40 до 40. Напечатать его. Выполнить сортировку только положительных чисел по убыванию. Напечатать новый массив.

Алгоритм решения задачи определен условием самой задачи – объявляем и инициализируем массив на 20 переменных целого типа; заполняем его случайными целыми числами и сразу выводим на экран монитора; сортируем элементы, значение которых >0; повторно печатаем массив.

Исходный код программы:

```
usingSystem;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
classProgram
  {
staticvoid Main(string[] args)
    {
int i, j, k;
int[] a = newint[20];
Random rnd = newRandom();
// формируем случайным образом массив 20 чисел
// выводим их на экран монитора
Console.WriteLine("Исходный массив: ");
for (i = 0; i <= 19; i++)
      {
a[i] = rnd.Next() % 81 - 40;
Console.Write(" {0}", a[i]);
}
Console.WriteLine();
Console.WriteLine();
// Сортировка положительных чисел массива по убыванию
for (i = 0; i < 19; i++)
for (j = i+1; j <= 19; j++)
if (a[i] >= 0 && a[i] < a[j])
```

```
        {
         k = a[i]; a[i] = a[j]; a[j] = k;
}
// выводим их на экран монитора массива после сортировки
Console.WriteLine("Массив после сортировки: ");
for (i = 0; i <= 19; i++)
Console.Write(" {0}", a[i]);
Console.WriteLine();
Console.WriteLine();
Console.WriteLine("Дляпродолженияянажмитеклавишу Enter");
Console.ReadLine();
    }
  }
}
```

Работа программы:
Исходный массив:
-6 21 -7 -15 -36 17 36 1 15 -11 -31 23 31 -10 -24 16 -6 29 -35 24

Массив после сортировки:
-6 36 -7 -15 -36 31 29 24 23 -11 -31 21 17 -10 -24 16 -6 15 -35 1

Для продолжения нажмите клавишу Enter

3.4 Домашнее задание на лабораторную работу

Случайным образом формируются координаты X и Y 100 точек. Диапазон значений координат от -150 до +150. Для каждой четверти напечатать точки, принадлежащие ей, и две точки, расстояние между которыми минимально.

3.5 Индивидуальные задания для СРС

3.5.1 Индивидуальное задание 1 студента:
3.5.1.1 Сформировать массив 25 целых случайных чисел в диапазоне от минус 40 до 60. Напечатать его. Найти 3 maxчисла и поместить их в начало массива (сортировку чисел использовать запрещается). Напечатать новый массив.
3.5.1.2 Сформировать массив 100 целых случайных чисел в диапазоне от 0 до 9. Напечатать его. Найти и напечатать число, чаще других, встречающееся в массиве.
3.5.1.3 Сформировать массив 20 целых случайных чисел в диапазоне от минус 20 до 20. Напечатать его. Выполнить сортировку первых 10 чисел по возрастанию, а вторых 10 чисел – по убыванию значений элементов массива. Напечатать новый массив.

3.5.2 Индивидуальное задание 2 студента:
3.5.2.1 Сформировать массив 30 целых случайных чисел в диапазоне от минус 40 до 40. Напечатать его. Все отрицательные числа поместить в начало массива (сортировку чисел использовать запрещается). Напечатать новый массив.
3.5.2.2 Случайным образом формируются координаты A(X,Y) и B(X,Y) ста прямоугольников заданных противоположными вершинами. Диапазон значений координат от минус 150 до 150. Определить и напечатать, есть ли среди них прямоугольники с одинаковой площадью.
3.5.2.3 Сформировать массив 20 целых случайных чисел в диапазоне от минус 30 до 30. Напечатать его. Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию. Напечатать новый массив.

3.5.3 Индивидуальное задание 3 студента:
3.5.3.1 Сформировать массив 20 целых случайных чисел в диапазоне от минус 30 до 30. Напечатать его. Поменять местами максимальный и минимальный элементы массива. Напечатать новый массив.
3.5.3.2 Сформировать массив 20 целых случайных чисел в диапазоне от минус 50 до 50. Напечатать его. Все четные числа разместить слева, а нечетные – справа. Напечатать новый массив.
3.5.3.3 Сформировать массив 40 целых случайных чисел в диапазоне от минус 40 до 40. Напечатать его. Напечатать произведение элементов, расположенных между максимальным и минимальным элементами массива.

3.5.4 Индивидуальное задание 4 студента:
3.5.4.1 Сформировать массив 30 целых случайных чисел в диапазоне от минус 50 до 50. Напечатать его. Найти числа расположенные на нечётных местах, поместить их в новый массив и вывести его на экран монитора.
3.5.4.2 Случайным образом формируются координаты X и Y центра и R – радиус 50 кругов. Диапазон значений координат от минус 150 до 150, диапазон значения радиуса от 5 до 15. Определить и напечатать, круги, расстояние между окружностями которых, максимальное.
3.5.4.3 Сформировать массив 40 целых случайных чисел в диапазоне от минус 40 до 40. Напечатать его. Сжать массив, удалив из него все элементы, модуль которых не превышает 20 единиц. Освободившиеся в конце массива элементы заполнить нулями. Напечатать новый массив.

3.5.5 Индивидуальное задание 5 студента:
3.5.5.1 Сформировать массив 100 целых случайных чисел в диапазоне от 0 до 100. Напечатать его. Все числа больше 30, но меньше 70 переписать в начало массива. Напечатать новый массив.

3.5.5.2 Сформировать массив 100 целых случайных чисел в диапазоне от 0 до 10. Напечатать его. Напечатать статистику – сколько раз встречается каждое число массива.
3.5.5.3 Случайным образом формируются координаты A(X,Y) и B(X,Y) ста прямоугольников заданных противоположными вершинами. Диапазон значений координат от минус 0 до 150. Определить и напечатать, есть ли среди них прямоугольники с одинаковыми сторонами (отдельно напечатать число прямоугольников, у которых совпали 2 стороны, попарно 4 стороны и все стороны – квадраты).

3.5.6 Индивидуальное задание 6 студента:
3.5.6.1 Сформировать массив 20 целых случайных чисел в диапазоне от минус 50 до 50. Напечатать его. Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах. Напечатать новый массив.
3.5.6.2 Сформировать массив 50 целых случайных чисел в диапазоне от 0 до 60. Напечатать его. Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине – элементы, стоявшие в четных позициях. Напечатать новый массив.
3.5.6.3 Сформировать и напечатать вектора A и B, размерностью 7 элементов каждый. Значения элементов векторов сформировать случайным образом из чисел, принадлежащих диапазону от минус 30 до 30. Найти и напечатать вектор C (размерностью 7), элементы которого определяются произведением соответствующих элементов векторов A и B.

3.6 Контрольные вопросы для защиты отчета на СРСП

3.6.1 Понятие ссылочного типа. Пример.
3.6.2 Понятие массива. Примеры организации данных в виде массива.
3.6.3Как массивы объявляются в программе? Пример.
3.6.4 Как выполняется инициализация массива? Пример.
3.6.5 Как сформировать массив с помощью генератора случайных чисел. Пример.
3.6.6 Ассоциативный алгоритм обращения к элементам массива. Пример.
3.6.7 Алгоритмы сортировки элементов массива методом выбора. Словесное описание алгоритма и фрагмент кода сортировки.
3.6.8 «Пузырьковый» алгоритм сортировки элементов массива. Словесное описание алгоритма и фрагмент кода сортировки.
3.6.9 Алгоритм перестановки (сдвига влево или вправо) данных в массивах. Словесное описание алгоритма и фрагмент кода сортировки.
3.6.10 Понятие динамического массива. Пример.
3.6.11 Понятие поискового массива и ключа поиска. Пример.

3.6.12Алгоритм последовательного поиска элементов массива. Достоинства и недостатки алгоритма последовательного поиска.
3.6.13 Алгоритм блочного поиска элементов массива. Достоинства и недостатки алгоритма блочного поиска.
3.6.14 Алгоритм двоичного поиска элементов массива. Достоинства и недостатки алгоритма двоичного поиска.
3.6.15 Алгоритмы поиска с преобразованием ключа в адрес – хеширование.

ТЕМА 4 ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ – МЕТОДОВ ЯЗЫКАС#

4.1 Цель четвертой темы

Изучение данных и методов классаи приобретение практических навыков по созданию консольных приложений с использованием методов (функций).

4.2 Теоретические сведения

4.2.1 Понятие метода

Язык C# является объектно-ориентированным языком программирования и главную роль в нем играют классы. При этом класс рассматривается как тип данных, который в качестве своих составляющих (наряду с другими) содержит поля класса (его данные) и методы класса (его функции). Методы класса "служат" данным, занимаясь их обработкой.

В языке C# функции существуют только как методы некоторого класса, они не существуют вне класса.

В языке C# нет специальных ключевых слов – method, procedure, function, но сами понятия, конечно же, присутствуют. Синтаксис объявления метода позволяет однозначно определить, чем является метод – процедурой или функцией.

4.2.2Формат записи метода класса

Формат записи метода класса имеет следующий вид:

void или тип_метода имя_метода(список_формальных_параметров)

{ тело метода}

Если вместо типа метода задано значение void, то метод работает как процедура.

Если слово void отсутствует, а задан тип метода, то метод работает как функция.

Обязательным при описании заголовка метода является указание типа метода, имени метода и круглых скобок, наличие которых необходимо и в том случае, если сам список формальных параметров отсутствует.

Имя метода и список формальных параметров составляют сигнатуру метода (необходимые элементы при использовании метода в программе).

Заметьте, в сигнатуру не входят имена формальных параметров, здесь важны типы аргументов. В сигнатуру не входит и тип метода.

Например, метод процедуру можно представить следующим образом:

void poisk() {…};

и ее использование в программе ограничивается следующим указанием:

poisk();

Метод функция при объявлении имеет тип, например:

int kol(){…};

и при использовании в программе должна присваиваться переменной целого типа (в соответствии с типом метода), например:

d= kol();

4.2.3Формальные параметры метода класса

Список формальных параметров метода может быть пустым и это довольно типичная ситуация для методов класса. Список может содержать фиксированное число параметров, разделяемых символом запятой.

Формат записи одного формального параметра:

[ref или out или params] тип_параметра имя_параметра

Значения, указанные в квадратных скобках, являются необязательными. Обязательным является указание типа и имени параметра.

Несмотря на фиксированное число формальных параметров, есть возможность при вызове метода передавать ему произвольное число фактических параметров. Для реализации этой возможности в списке формальных параметров необходимо задать ключевое слово params. Оно может появляться в объявлении лишь для последнего параметра списка, объявляемого как массив произвольного типа. При вызове метода этому формальному параметру соответствует произвольное число фактических параметров.

Содержательно все параметры метода разделяются на три группы: входные, выходные и обновляемые.

Параметры первой группы передают информацию методу, их значения в теле метода только читаются.

Параметры второй группы представляют собой результаты метода, они получают значения в ходе работы метода.

Параметры третьей группы выполняют обе функции. Их значения используются в ходе вычислений и обновляются в результате работы метода. Выходные параметры всегда должны сопровождаться ключевым словом out, обновляемые – ref. Что же касается входных параметров, то они задаются без ключевого слова.

Если параметр объявлен как выходной с ключевым словом out, то в теле метода обязательно должен присутствовать оператор присваивания, задающий значение этому параметру. В противном случае возникает ошибка еще на этапе компиляции.

Тело метода представляет собой последовательность операторов и описаний переменных, заключенную в фигурные скобки. Если речь идет о теле функции, то в блоке должен быть хотя бы один оператор, возвращающий значение функции в форме return<выражение> (тип выражения должен совпадать с типом функции).

Переменные, описанные в теле метода, считаются локальными в этом методе.

В записи операторов тела метода участвуют имена локальных переменных метода, имена полей класса (для методов они рассматриваются как глобальные переменные класса) и имена параметров метода.

4.3 Пример выполнения задания на лабораторную работу

Рассмотрим чисто учебную программу для работы с массивом с помощью методов.

Задача 4.1Сформировать массив из 20 случайных целых чисел в диапазоне от минус 50 до 50. Напечатать его. Выполнить сдвиг значений массива на 1 разряд влево. Программу оформить в виде методов, реализующих каждое задание задачи. В методах использовать входные, выходные и обновляемые параметры. Предусмотреть меню.

В соответствии с условиями задачи необходимо разработать 3 метода:

– создание массива (используем выходной параметр для массива);

– печать массива (используем входной параметр для массива);

– сдвиг массива (используем обновляемый параметр для массива).

Для организации меню программы используем цикл whileи оператор switch ();

Исходный код программы:

```
usingSystem;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Text;
namespace ConsoleApplication1
{
classProgram
  {
publicstaticvoid sozd(outint[] ma)
    {
     ma = newint[20];
Random rnd = newRandom();
for (int i = 0; i < 20; i++)
       ma[i] = rnd.Next() % 101 - 50;
Console.WriteLine("Массивсоздан !!");
    }

publicstaticvoid zadvig(refint[] ma)
    {
int k;
for (int i = 0; i < 19; i++)
```

```
      {
       k = ma[i]; ma[i] = ma[i + 1]; ma[i + 1] = k;
}
Console.WriteLine("Сдвиг массива на 1 разряд выполнен !");
}

publicstaticvoid prinmas(int[] ma)
    {
for (int i = 0; i < 20; i++)
Console.Write(" {0}", ma[i]);
Console.WriteLine();
    }

staticvoid Main()
    {
int[] a = newint[20];
int k = 0;
string buf;
while (k < 4)
      {
Console.WriteLine("1 - Создатьмассив 20 чисел");
Console.WriteLine("2 - Переместить массив на 1 разряд влево");
Console.WriteLine("3 - Печать массива");
Console.WriteLine("4 - Выход из программы");
Console.WriteLine("Введите пункт меню программы");
buf = Console.ReadLine();
       k = Convert.ToInt32(buf);
switch (k)
        {
case 1: sozd(out a);  break;
case 2: zadvig(ref a); break;
case 3: prinmas(a); break;
default: break;
}
       }
}
  }
}
```

Работа программы:
1 - Создать массив 20 чисел
2 - Переместить массив на 1 разряд влево
3 - Печать массива
4 - Выход из программы
Введите пункт меню программы
1
Массив создан !!
1 - Создать массив 20 чисел
2 - Переместить массив на 1 разряд влево
3 - Печать массива
4 - Выход из программы

Введите пункт меню программы
3
-24 -20 40 46 -26 -16 -45 -46 -39 32 38 -38 -18 -2 3 -26 -40 -17 -34 -39
1 - Создать массив 20 чисел
2 - Переместить массив на 1 разряд влево
3 - Печать массива
4 - Выход из программы
Введите пункт меню программы
2
Сдвиг массива на 1 разряд выполнен !
1 - Создать массив 20 чисел
2 - Переместить массив на 1 разряд влево
3 - Печать массива
4 - Выход из программы
Введите пункт меню программы
3
-20 40 46 -26 -16 -45 -46 -39 32 38 -38 -18 -2 3 -26 -40 -17 -34 -39 -24
1 - Создать массив 20 чисел
2 - Переместить массив на 1 разряд влево
3 - Печать массива
4 - Выход из программы
Введите пункт меню программы

4.4 Домашнее задание на лабораторную работу

Используя метод-функцию вычислить сумму ряда для заданного в режиме диалога x (x>0 и x<1). Вычисления заканчиваются, когда очередной член ряда по модулю становится меньше введенной точности ε=0.0001:

$$\sqrt[3]{1+x} = 1 + \frac{1}{3}x - \frac{2}{2!*3^2}x^2 + \frac{2*5}{3!*3^3}x^3 - \frac{2*5*8}{4!*3^4}x^4 + \ldots$$

4.5 Индивидуальные задания для СРС

Внимание! В учебных целях каждый метод-функция или метод-процедура выполняемой задачи должен иметь выходные и обновляемые формальные параметры. Без использования этих параметров индивидуальное задание не принимается к защите.

4.5.1 Индивидуальное задание 1 студента:
4.5.1.1 Случайным образом формируются координаты 20 отрезков (Y1,Y2) и (X1,X2). Значения координат – целые числа от 0 до 100. Определить номер отрезка, имеющего максимальную длину. Использовать метод-процедуру.
4.5.1.2 Написать программу вычисления ряда:

$$Y=\sum_{N=0}^{100}(x^{2*N+1}/(2*N+1)*\sin(2*N+1)*x/10),$$

где x – задаётся в режиме диалога в интервале от 0 до 1; N – изменяется от 0 до 100 с шагом 1. Использовать метод-функцию.

4.5.1.3 Написать программу вычисления корней квадратного уравнения вида ax2+bx+c = 0. Значения a, b и c вводить в режиме диалога. Предусмотреть проверку существования корней уравнения и выдать соответствующие сообщения. Для решения задачи использовать метод-процедуру.

4.5.2 Индивидуальное задание 2 студента:

4.5.2.1 Имеется круг, заданный координатами центра (50,50) и радиусом = 30. Случайным образом формируются 10 точек (x,y) – координаты в диапазоне от 0 до 100. Определить и сколько точек попало в круг. Использовать метод-процедуру.

4.5.2.2 Вычислить сумму Y = 1*2*3+2*3*4+3*4*5+…+(n-1)*n*(n+1),
n задаётся в режиме диалога. Использовать метод-функцию.

4.5.2.3 В цикле 20 раз формируются случайные целые числа в диапазоне от 0 до 100. Напечатать все значения этих чисел. Использовать метод-функцию для определения max и min чисел. Напечатать их.

4.5.3 Индивидуальное задание 3 студента:

4.5.3.1 Случайным образом формируются координаты X и Y 100 точек. Диапазон значений координат от -150 до +150. Подсчитать и напечатать количество точек, расположенных на каждой четверти. Отдельно учитывать точки, расположенные на осях координат. Использовать метод-процедуру.

4.5.3.2 Вычислить сумму ряда для заданного в режиме диалога и |x|>1.

$$y=\frac{2!}{x^2*3!}+\frac{3!}{x^4*4!}+\frac{4!}{x^6*5!}+\ldots$$

Вычисления продолжать до тех пор, пока очередной член ряда не становится меньше 0.0001. Использовать метод-функцию.

4.5.3.3 Сформировать массив 20 целых случайных чисел в диапазоне от минус 30 до 30. Напечатать его. Поменять местами максимальный и минимальный элементы массива. Напечатать новый массив. Каждое действие оформить в виде методов-процедур.

4.5.4 Индивидуальное задание 4 студента:

4.5.4.1 Случайным образом формируются координаты A(X,Y) и B(X,Y) ста прямоугольников заданных противоположными вершинами. Диапазон значений координат от -150 до +150. Подсчитать и напечатать количество прямоугольников расположенных в верхней и нижней половинах системы координат(если вершины расположены в разных половинах, то этот вариант исключается из рассмотрения). Использовать метод-процедуру.

4.5.4.2 Используя метод-функцию вычислить сумму ряда для заданного |x| > 1.

$$\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots\right)\quad |x|>1$$

Вычисления заканчиваются, когда очередной член ряда по модулю становится меньше 0.0001.

4.5.4.3 Сформировать массив 25 целых случайных чисел в диапазоне от минус 40 до 60. Напечатать его. Найти 3 max числа и поместить их в начало массива (сортировку чисел использовать запрещается). Напечатать новый массив. Каждое действие оформить в виде методов-процедур.

4.5.5 Индивидуальное задание 5 студента:

4.5.5.1 Случайным образом формируются координаты X и Y 60 точек. Диапазон значений координат от -150 до +150. Вывести список точек, расстояние между которыми максимально, а сами точки находятся в разных четвертях. Использовать метод-процедуру.

4.5.5.2 Используя метод-функцию вычислить сумму ряда для x = 5.

$$\operatorname{arctg} x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}\ldots\quad x>1$$

Вычисления заканчиваются, когда очередной член ряда по модулю становится меньше 0.0001.

4.5.5.3 Сформировать массив 100 целых случайных чисел в диапазоне от 0 до 9. Напечатать его. Найти и напечатать число, чаще других, встречающееся в массиве. Каждое действие оформить в виде методов-процедур.

4.5.6 Индивидуальное задание 6 студента:

4.5.6.1 Случайным образом формируются координаты X и Y 60 точек. Диапазон значений координат от -150 до +150. Вывести список точек, расстояние между которыми минимально, а сами точки находятся в одной четверти. Использовать метод-процедуру.

4.5.6.2 Вычислить сумму ряда для заданного x<1. Вычисления заканчиваются, когда очередной член ряда становится меньше 0.001:

$$arctg(x)=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots$$

Использовать метод-процедуру.

4.5.6.3 Случайным образом формируются координаты A(X,Y) и B(X,Y) ста прямоугольников заданных противоположными вершинами. Диапазон значений координат от -150 до +150. Подсчитать и напечатать количество прямоугольников расположенных на каждой четверти (если вершины расположены в разных четвертях, то этот вариант исключается из рассмотрения). Использовать метод-процедуру.

4.6 Контрольные вопросы для защиты отчета на СРСП

4.6.1 Понятие метода классаязыкаC#. Пример.
4.6.2 Формат записи метода класса языка C#.
4.6.3Какие спецификаторы доступа методов класса вы знаете? Пример.
4.6.4 Механизмы обмена данными методов класса с `staticvoidMain()` в языке C#?
4.6.5Какие входные формальные параметры методов класса Вы знаете? Пример.
4.6.6Какие выходные формальные параметры методов класса Вы знаете? Пример.
4.6.7Какие обновляемые формальные параметры методов класса Вы знаете? Пример.
4.6.8 Понятие функциив языке C#. Пример.
4.6.9 Как и где задается тип функциив языке C#? Пример.
4.6.10 Понятие процедуры в языке C#. Пример.
4.6.11 Понятие локальных и глобальных переменных метода класса языка C#. Пример.
4.6.12 Какие «имена» могут участвовать в записи операторов тела методакласса языка C#?
4.6.13 Можно ли внутри одного метода (например, `void Main())` объявлять другой метод? Пример.
4.6.14 Понятие рекурсии. Пример.
4.6.15 Достоинства и недостатки рекурсии.

5 МНОГОМЕРНЫЕ МАССИВЫ В ЯЗЫКЕ C#

5.1 Цель пятой темы

Изучение правил объявления и инициализации многомерных массивов иприобретение практических навыков по созданию консольных приложений с использованием многомерных массивов – матриц.

5.2 Теоретические сведения

5.2.1 Понятие многомерных массивов

Разделение массивов на одномерные массивы и многомерные носит исторический характер. Никакой принципиальной разницы между ними нет. Одномерные массивы - это частный случай многомерных массивов.

Одномерные массивы позволяют задавать такие математические структуры, как векторы, двумерные - матрицы, трехмерные - кубы данных, массивы большей размерности - многомерные кубы данных.

Размерность массива это характеристика типа. Формат записи при объявлении многомерного массива имеет следующий вид:

<тип>[, … ,] <объявители>;

Например: int[,] matri; int[,,] kubi;

Размерность массива задается запятыми, которые устанавливаются в квадратных скобках.

Число запятых, увеличенное на единицу, и задает размерность массива. Что касается объявителей, то все, что сказано для одномерных массивов, справедливо и для многомерных. Можно лишь отметить, что хотя явная инициализация с использованием многомерных константных массивов возможна, но применяется редко из-за громоздкости такой структуры. Мы будем ее применять в разделе графы при задании матрицы смежности.

Обычно инициализацию многомерных массивов выполняют программно при создании массива.

5.2.2 Массивы массивов

Еще одним видом массивов C# являются массивы массивов, называемые также «ломаными» массивами (jaggedarrays).

Такой массив массивов можно рассматривать как одномерный массив, его элементы являются массивами, элементы которых, в свою очередь снова могут быть массивами, и так может продолжаться до некоторого уровня вложенности.

В каких ситуациях может возникать необходимость в таких структурахданных? Эти массивы могут применяться для представления деревьев, у которых узлы могут иметь произвольное число потомков. Таковым может быть, например, генеалогическое дерево. Вершины первого уровня - Fathers, представляющие отцов, могут задаваться одномерным массивом, так что Fathers[i] - это i-й отец. Вершины второго уровня представляются массивом массивов - Children, так что Children[i] - это массив детей i-го отца, а Children[i][j] - это j-й ребенок i-го отца. Для представления внуков понадобится третий уровень, так что GrandChildren [i][j][k] будет представлять к-го внука j-го ребенка i-го отца.

Есть некоторые особенности в объявлении и инициализации таких массивов. Если при объявлении типа многомерных массивов для указания размерности использовались запятые, то для «ломанных» массивов применяется более ясная символика - совокупности пар квадратных скобок; например, int[][] задает массив, элементы которого - одномерные массивы элементов типа int.

Сложнее с созданием самих массивов и их инициализацией. Здесь нельзя вызвать конструктор newint[3][5], поскольку он не задает «ломанный» массив. Фактически нужно вызывать конструктор для каждого массива на самом нижнем уровне. В этом и состоит сложность объявления таких массивов. Например: для абстрактного массива массивов masmasiцелого типа выполним объявление и инициализацию следующим образом:

```
int[][]masmasi = new int[3][]
  {
```

```
        newint[] {5,7,9,11},
        newint[] {2,8},
        newint[] {6,12,4}
};
```

Массивmasmasiимеетвсегодвауровня. Можно считать, что у него три элемента, каждый из которых является массивом. Для каждого такого массива необходимо вызвать конструктор new, чтобы создать внутренний массив. В данном примере элементы внутренних массивов получают значение, будучи явно инициализированы константными массивами. Этот же массив можно объявить и инициализировать не явно следующим образом:

```
int[][]masmasi  = new int[3][]
{
        newint[4],
        newint[2],
        newint[3]
};
```

В этом случае элементы массива получат при инициализации нулевые значения. Реальную инициализацию нужно будет выполнять программным путем. Стоит заметить, что в конструкторе верхнего уровня константу 3 можно опустить и писать просто newint[][]. Самое забавное, что вызов этого конструктора можно вообще опустить, он будет подразумеваться:

```
int[][]masmasi  =
{
        newint[4],
        newint[2],
        newint[3]
};
```

Но вот конструкторы нижнего уровня необходимы.

5.2.3 Алгоритмы линейной алгебры

Матрицей называется набор чисел, состоящий из m строк и n столбцов. Для программиста матрица - это двумерный массив. Матрица называется квадратной, если m = n, и прямоугольной - в противном случае. Числа m и n определяют размерность матрицы. Над прямоугольными матрицами определены операции транспонирования, сложения, умножения.

Пусть A - матрица размерности m*n (из m строк и n столбцов) с элементами $a_{i,j}$. Транспонированной матрицей B = AT называют матрицу размерности n*m, элементы которой $b_{i,j} = a_{j,i}$. В транспонированной матрице строки исходной матрицы становятся столбцами.

$$A = \left\| \begin{array}{l} a_{1,1}, a_{1,2} \dots a_{1,n} \\ a_{2,1}, a_{2,2} \dots a_{2,n} \\ \dots \\ a_{m,1}, a_{m,2} \dots a_{m,n} \end{array} \right\| \quad B = A^T = \left\| \begin{array}{l} a_{1,1}, a_{2,1} \dots a_{m,1} \\ a_{1,2}, a_{2,2} \dots a_{m,2} \\ \dots \\ a_{1,n}, a_{2,n} \dots a_{m,n} \end{array} \right\|$$

Операция сложения определена над прямоугольными матрицами одинаковой размерности. Пусть A, B, C - прямоугольные матрицы размерности m*n. Тогда сумма матриц определяется естественным образом:

$$A = \left\| \begin{array}{l} a_{1,1}, a_{1,2} \dots a_{1,n} \\ a_{2,1}, a_{2,2} \dots a_{2,n} \\ \dots \\ a_{m,1}, a_{m,2} \dots a_{m,n} \end{array} \right\| \quad B = \left\| \begin{array}{l} b_{1,1}, b_{1,2} \dots b_{1,n} \\ b_{2,1}, b_{2,2} \dots b_{2,n} \\ \dots \\ b_{m,1}, b_{m,2} \dots b_{m,n} \end{array} \right\|$$

$$C = A + B = \left\| \begin{array}{l} a_{1,1} + b_{1,1}, a_{1,2} + b_{1,2} \dots a_{1,n} + b_{1,n} \\ a_{2,1} + b_{2,1}, a_{2,2} + b_{2,2} \dots a_{2,n} + b_{2,n} \\ \dots \\ a_{m,1} + b_{m,1}, a_{m,2} + b_{m,2} \dots a_{m,n} + b_{m,n} \end{array} \right\|$$

Операция умножения определена над прямоугольными матрицами, у которых число столбцов первого сомножителя равно числу строк второго сомножителя.

Матрица произведения имеет число строк, равное числу строк первого сомножителя, и число столбцов, равное числу столбцов второго сомножителя.

Пусть A - матрица размерности m*p, B - размерности p*n, тогда матрица C= A*B имеет размерность m*n. Элементы матрицы произведения определяются как сумма попарных произведений элементов строки первого сомножителя на элементы столбца второго сомножителя.

$$A = ||a_{i,j}|| \quad i = 1, \dots m; \; j = 1, \dots p; \quad B = ||b_{j,k}|| \quad j = 1, \dots p; \; k = 1, \dots n$$

$$C = A * B = ||c_{i,k}|| \quad i = 1, \dots m; \; k = 1, \dots n;$$

$$c_{i,k} = \sum_{j=1}^{p} a_{i,j} * b_{j,k};$$

Умножение всегда определено для прямой и транспонированной матрицы. Если A - прямоугольная матрица размерности m*n, то всегда определена квадратная матрица B размерности m*m:

$$B = A * A^T = B^T = (A * A^T)^T = (A^T)^T * A^T = A * A^T$$

Результатом такого произведения является симметричная матрица. Квадратная матрица называется симметричной, если $a_{i,j} = a_{j,i}$ для всех i и j, или, что то же, если $A = A^T$. Операции транспонирования, сложения и умножения обладают следующими свойствами:

$$(A^T)^T = A; \quad (A + B)^T = A^T + B^T; \quad (A * B)^T = B^T * A^T$$

5.2.4 Квадратные матрицы

Квадратная матрица называется диагональной, если все элементы, кроме диагональных элементов,равны нулю, то есть $a_{i,j} = 0$ при $i/ = j$.

Квадратная матрица называется единичной, если все элементы, кроме диагональных, равны нулю, а диагональные элементы равны единице. То есть $a_{i,j} = 0$ при $i/ = j$ и $a_{i,j} = 1$ при $i = j$. Единичная матрица обозначается обычно буквой E, и она играет роль единицы при умножении матриц,

поскольку для любой квадратной матрицы A и единичной матрицы E той же размерности имеют место соотношения:

$$A * E = E * A = A$$

Для квадратных матриц определена функция над ее элементами, называемая определителем. Обозначается определитель обычно с помощью одинарных линий вокруг набора чисел, задающих матрицу:

$$D(A) = \begin{vmatrix} a_{1,1}, a_{1,2} \dots a_{1,n} \\ a_{2,1}, a_{2,2} \dots a_{2,n} \\ \dots \\ a_{n,1}, a_{n,2} \dots a_{n,n} \end{vmatrix}$$

Функция, задающая определитель, обладает рядом важных свойств.
Определитель диагональной матрицы равен произведению диагональных элементов. Отсюда следует, что определитель матрицы E равен 1.

Определитель матрицы не меняется при выполнении над матрицей элементарных преобразований. Под элементарной операцией (преобразованием) понимается прибавление к любой строке матрицы линейной комбинации других ее строк. В частности, если к строке матрицы с номером j прибавить строку с номером $k(k/=j)$, умноженную на некоторое число, то определитель матрицы не изменится.

Если все элементы одной строки матрицы умножить на некоторое число q, то определитель матрицы изменится в q раз (умножается на q).

Если переставить местами строки j и k, то модуль определителя не изменится, но изменится знак, если разность $|k-j|$ является нечетным числом.
Определитель произведения матриц равен произведению определителей:

$$D(A * B) = D(A) * D(B)$$

Не приводя общего формального определения, рассмотрим ниже алгоритм вычисления определителя матрицы, основанный на его свойствах.

Если определитель квадратной матрицы A не равен нулю, то существует обратная матрица, обозначаемая как A^{-1}. Прямая и обратная матрицы связаны соотношением:

$$A * A^{-1} = A^{-1} * A = E$$

Операции транспонирования, умножения и обращения матриц связаны соотношениями:

$$(A^T)^{-1} = (A^{-1})^T; \quad (A * B)^{-1} = B^{-1} * A^{-1}$$

Множество квадратных матриц одной размерности с определителем, отличным от нуля образуют группу по умножению. В группе есть единичный элемент, для каждого элемента существует обратный к нему, и произведение элементов принадлежит группе.

Определители широко используются при нахождении корней системы из n линейных уравнений с n неизвестными.

5.3 Пример выполнения задания на лабораторную работу

Рассмотрим чисто учебную задачу, позволяющую сформировать матрицы A – 5*3 и B – 3*4 случайных целых чисел в диапазоне от 0 до 5.

Необходимо напечатать эти матрицы, а затем найти и напечатать матрицу C, представляющую собой результат произведения матрицы A*B. Программа должна содержать меню.

Алгоритм произведения матриц подробно рассмотрен в теоретическом разделе 5 модуля данных методических указаний. Диапазон значений матриц A и B выбран маленьким для простоты проверки значений матрицы C.

Исходный код программы:

```
usingSystem;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
classProgram
 {
publicstaticvoid sozd(int[,] ma, int[,] mb)
  {
Random rnd = newRandom();
Console.WriteLine("Матрицысозданы!!");
Console.WriteLine("МатрицаA 5*3:");
for (int i = 0; i < 5; i++)
   {
for (int j = 0; j < 3; j++)
    {
     ma[i, j] = rnd.Next() % 6;
Console.Write(ma[i, j] + "\t");
   }
Console.WriteLine();
   }
Console.WriteLine();
Console.WriteLine("МатрицаB 3*4:");
for (int i = 0; i < 3; i++)
  {
for (int j = 0; j < 4; j++)
    {
     mb[i, j] = rnd.Next() % 6;
Console.Write(mb[i, j] + "\t");
}
Console.WriteLine();
   }
Console.WriteLine();
  }

publicstaticvoid umnmatr(int[,] mc, int[,] ma, int[,] mb)
  {
int S;
```

```
for (int i = 0; i < 5; i++)
for (int j = 0; j < 4; j++)
     {
      S = 0;
for (int k = 0; k < 3; k++)
       S = S + ma[i, k] * mb[k, j];
      mc[i,j] = S;
}
Console.WriteLine("Матрица C 5*4:");
for (int i = 0; i < 5; i++)
   {
for (int j = 0; j < 4; j++)
    {
Console.Write(mc[i, j] + "\t");
    }
Console.WriteLine();
   }
Console.WriteLine();
  }

staticvoid Main()
  {
int[,] a = newint[5, 3];
int[,] b = newint[3, 4];
int[,] c = newint[5, 4];
int k = 0;
string buf;
while (k < 3)
   {
    Console.WriteLine("1 - Создать и напечатать матрицы A 5*3 и
B 3*4");
Console.WriteLine("2 - Найти и напечатать матрицу C = A * B");
Console.WriteLine("3 - Выход из программы");
Console.WriteLine("Введите пункт меню программы");
buf = Console.ReadLine();
    k = Convert.ToInt32(buf);
switch (k)
    {
case 1: sozd(a,b);  break;
case 2: umnmatr(c,a,b); break;
default: break;
    }
   }
  }
}
}
```

Работа программы:
1 - Создать и напечатать матрицы A 5*3 и B 3*4
2 - Найти и напечатать матрицу C = A * B
3 - Выход из программы
Введите пункт меню программы

```
1
Матрицы созданы!!
Матрица A 5*3:
2      1      3
5      5      3
2      0      2
4      4      0
4      2      3

Матрица B 3*4:
0      5      0      3
0      3      5      3
3      0      5      2

1 - Создать и напечатать матрицы A 5*3 и B 3*4
2 - Найти и напечатать матрицу C = A * B
3 - Выход из программы
Введите пункт меню программы
2
Матрица C 5*4:
9      13     20     15
9      40     40     36
6      10     10     10
0      32     20     24
9      26     25     24

1 - Создать и напечатать матрицы A 5*3 и B 3*4
2 - Найти и напечатать матрицу C = A * B
3 - Выход из программы
Введите пункт меню программы
```

5.4 Домашнее задание на лабораторную работу

Сформировать матрицу 5*5 случайных целых чисел в диапазоне от -20 до 60. Напечатать матрицу. Найти максимальный элемент и удалить строку и столбец матрицы, на пересечении которых находится максимальный элемент. Оставшиеся значения переписать в новую матрицу 4*4. Напечатать новую матрицу. В программе использовать меню.

5.5 Индивидуальные задания для СРС

5.5.1 Индивидуальное задание 1 студента:
5.5.1.1 Сформировать матрицу A5×5 случайных целых чисел в диапазоне от минус 50 до 50. Напечатать матрицу. Найти минимальный элемент, и, если он не в первой строке, то поменять местами первую строку и строку, в которой находится минимальный элемент. Напечатать новую матрицу.
5.5.1.2 Текст некоторой строки вводится в режиме диалога и содержит простое предложение, в котором слова отделяются символом «пробел». Добавить лишние «пробелы» между словами так, чтобы длина строки стала кратна 60, а «пробелы» распределились по строке равномерно.
5.5.1.3 Сформировать матрицу A7×7 случайных чисел в диапазоне от минус 20 до 60. Напечатать ее. Вычислить и напечатать сумму значений элементов расположенных «ниже» главной и побочной диагоналей матрицы.

5.5.2 Индивидуальное задание 2 студента:
5.5.2.1 Сформировать матрицу A6×6 случайных целых чисел в диапазоне от 0 до 100. Напечатать матрицу. Все числа в каждой строке матрицы расположить в убывающем порядке. Напечатать новую матрицу.
5.5.2.2 Строка содержит простое предложение, в котором слова отделяются символом «пробел». Распечатать слова, имеющие максимальную и минимальную длину.
5.5.2.3 Сформировать матрицу A10×10 случайных целых чисел в диапазоне от 0 до 9. Напечатать матрицу. Подсчитать сколько в исходной матрице 0, 1, 2, …, 9.

5.5.3 Индивидуальное задание 3 студента:
5.5.3.1 Сформировать матрицу A5×5 случайных целых чисел в диапазоне от минус 20 до 70. Напечатать матрицу. Найти максимальный элемент и удалить в исходной матрице столбец, в котором находится максимальный элемент. Переписать оставшиеся значения в новую матрицу B5×4. Напечатать новую матрицу.
5.5.3.2 Строка, введенное в режиме диалога, содержит условие оператора if языка C#, включающее не более 2 скобок. Определить, правильно ли записано введенное в режиме диалога условие.
5.5.3.3 Сформировать матрицу A6×6 случайных целых чисел в диапазоне от минус 30 до 50. Напечатать матрицу. Числа главной и побочной диагоналей матрицы, расположить в возрастающем порядке. Напечатать новую матрицу.

5.5.4 Индивидуальное задание 4 студента:
5.5.4.1 Сформировать матрицу A6×6 случайных целых чисел в диапазоне от минус 50 до 50. Напечатать матрицу. Подсчитать сумму положительных и отрицательных элементов. Если сумма отрицательных чисел по модулю больше суммы положительных, то начиная с первого отрицательного числа

менять знаки чисел на плюс до тех пор, пока сумма положительных по модулю не станет больше суммы отрицательных. Напечатать новую матрицу.
5.5.4.2 Строка, введенное в режиме диалога, содержит простое предложение, в котором слова отделяются символом «пробел». Сформировать новую строку, в которой поменять местами слова исходной строки, имеющие максимальную и минимальную длину.
5.5.4.3 Сформировать матрицу A5×5 случайных чисел в диапазоне от 0 до 20. Напечатать ее. Найти и напечатать только повторяющиеся числа матрицы и их частоту повторения.

5.5.5 Индивидуальное задание 5 студента:
5.5.5.1 Сформировать и напечатать матрицу A6x6 целых случайных чисел в диапазоне от 0 до 20. Расположить столбцы матрицы в порядке возрастания их сумм элементов. Напечатать новую матрицу.
5.5.5.2 Текст некоторой строки вводится в режиме диалога и содержит простое предложение, в котором слова отделяются символом «пробел». Сформировать новую строку, в которой слова строки расположены в порядке возрастания в них числа букв.
5.5.5.3 Сформировать матрицу A10×10 случайных целых чисел в диапазоне от 0 до 20. Напечатать матрицу. Найти и напечатать номер строки, в которой больше всего одинаковых чисел.

5.5.6 Индивидуальное задание 6 студента:
5.5.6.1 Сформировать матрицу A6x6 случайных чисел в диапазоне от 0 до 20. Напечатать ее. Построить и напечатать вектор из повторяющихся чисел матрицы.
5.5.6.2 Строка, введенное в режиме диалога, содержит простое предложение, в котором слова отделяются символом «пробел». Определить есть ли в тексте одинаковые слова. Напечатать их с указанием частоты встречаемости в предложении.
5.5.6.3 Сформировать и напечатать матрицу A8×8 целых случайных чисел в диапазоне от минус 50 до 50. Напечатать все диагонали матрицы параллельные главной.

5.6 Контрольные вопросы для защиты отчета на СРСП

5.6.1 Понятие многомерного массива. Примеры.
5.6.2 Как объявляется и инициализируется многомерный массив? Пример.
5.6.3 Приведите пример и объясните как формировать матрицу A5×5 вещественных чисел в диапазоне от 0 до 10.
5.6.4 Приведите пример и объясните как организация вывода матрицы A5×5 на экран монитора.
5.6.5 Объясните алгоритм поиска максимального числа в матрице A5×5.

5.6.6 Понятие массива массивов. Объявление и инициализация таких массивов. Пример.
5.6.7 Объясните алгоритм сложения двух матриц. Пример.
5.6.8 Объясните алгоритм умножения двух матриц. Пример.
5.6.9 Понятие строковой переменной в языке C#.Пример.
5.6.10 Для чего предназначены escape-последовательностей в языке **C#**? Примеры.
5.6.11 Как сравниваются строковые переменные в языке C#? Пример.
5.6.12 Как можно удалить из текста некоторый известный фрагмент? Пример.
5.6.13 Как можно вставить в текст некоторый известный фрагмент? Пример.
5.6.14 Как можно определить количество символов в некотором тексте? Пример.
5.6.15 Назначение методов Split и Join. Примеры.

6 АЛГОРИТМЫ ОБХОДА ГРАФОВ

6.1 Цель шестой темы

Изучение основных понятия теории графов и некоторых алгоритмов обработки графов иприобретение практических навыков по созданию консольных приложений для решения «транспортных» задач с использованием графов.

6.2 Теоретические сведения

6.2.1 Стягивающие деревья. Основные понятия

В теории графов существует два простых определения понятия дерева. Не разбирая теоремы, доказывающие справедливость этих определений приведем их текст. Первое определение утверждает, что деревом является связанный граф, в котором число дуг на единицу меньше числа вершин.

Второе определение более очевидное – деревом называется связанный граф без циклов.

Дерево, полученное из графа путем удаления некоторых ребер, называется стягивающим деревом этого графа или каркасом.

Например, на рисунке 6.1 представлен граф типа «звезда», который может быть представлен несколькими стягивающими деревьями. Один из вариантов стягивающего дерева также изображен на рисунке 6.1.

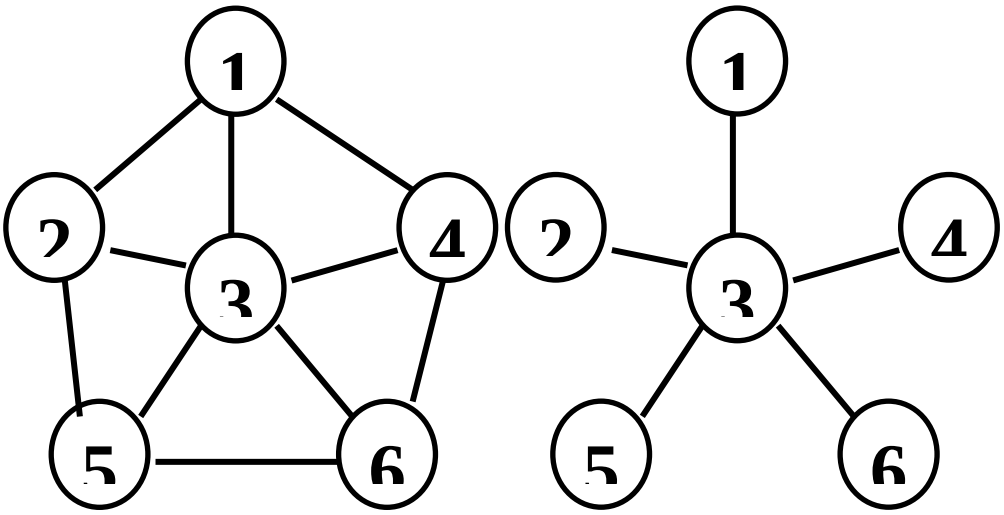

Рисунок 6.1 – Связанный граф и его стягивающее дерево.

В теории графов неиспользованные ребра (удаленные ребра графа из которого получено стягивающее дерево) называются хордами.

Если к стягивающему дереву добавить произвольную хорду, то полученный граф содержит в точности один цикл.

На этой идее строятся алгоритмы получения всего множества циклов графа, что часто имеет прикладной характер, например, при анализе электрических цепей по закону Кирхгофа.

6.2.2 Алгоритм построения стягивающего дерева

Существует множество различных алгоритмов нахождения стягивающих деревьев.

Рассмотрим алгоритм нахождения стягивающего дерева, в котором использован известный Вам алгоритм Дейкстры – нахождения минимальных маршрутов от заданной вершины до остальных вершин графа.

Это возможно, так как существует теорема, доказывающая, что «набор» минимальных маршрутов от заданной вершины до остальных вершин графа есть стягивающее дерево.

Напомню идею алгоритма нахождения минимального маршрута между вершинами графа. Например, существуют три смежные вершины 1,2 и 3. Расстояние между вершинами 1 - 2 равно 5, вершинами 2 – 3 равно 3, а вершинами 3 – 1 равно 1. При выборе маршрута между вершинами 1 – 2 графа используется маршрут 1 – 3 – 2. Ребро 1 – 2 удаляется из графа.

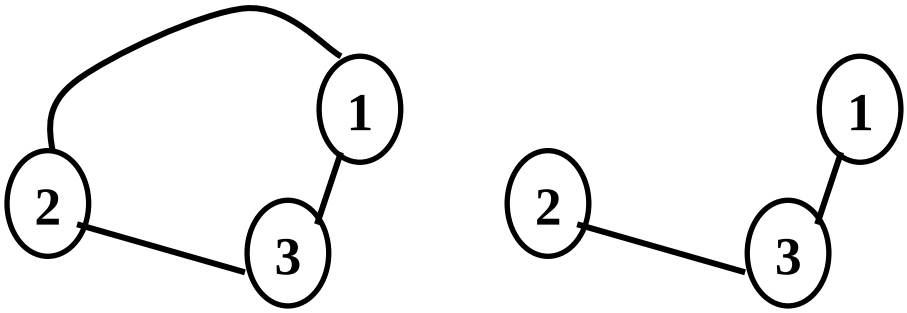

Рисунок 6.2 – Идея алгоритма нахождения минимального маршрута

Использование приведенного алгоритма позволяет строить стягивающие деревья. Так как «перебор» вершин графа в алгоритме осуществляется в порядке возрастания вершин (использован цикл for), то при «равенстве» двух маршрутов в стягивающем дереве остается первый вариант.

В качестве исходного графа для программной реализации задачи построения стягивающего дерева связанного графа, выбран граф, для которого уже существует программная реализация в визуальной среде программирования Delphi (граф изображен на рисунке 6.3).

Алгоритм построения стягивающего дерева использует алгоритм нахождения минимальных маршрутов Дейкстры. Отличие от ранее рассмотренной программой реализации этого алгоритма заключается в том, что можно задавать значение начальной вершины.

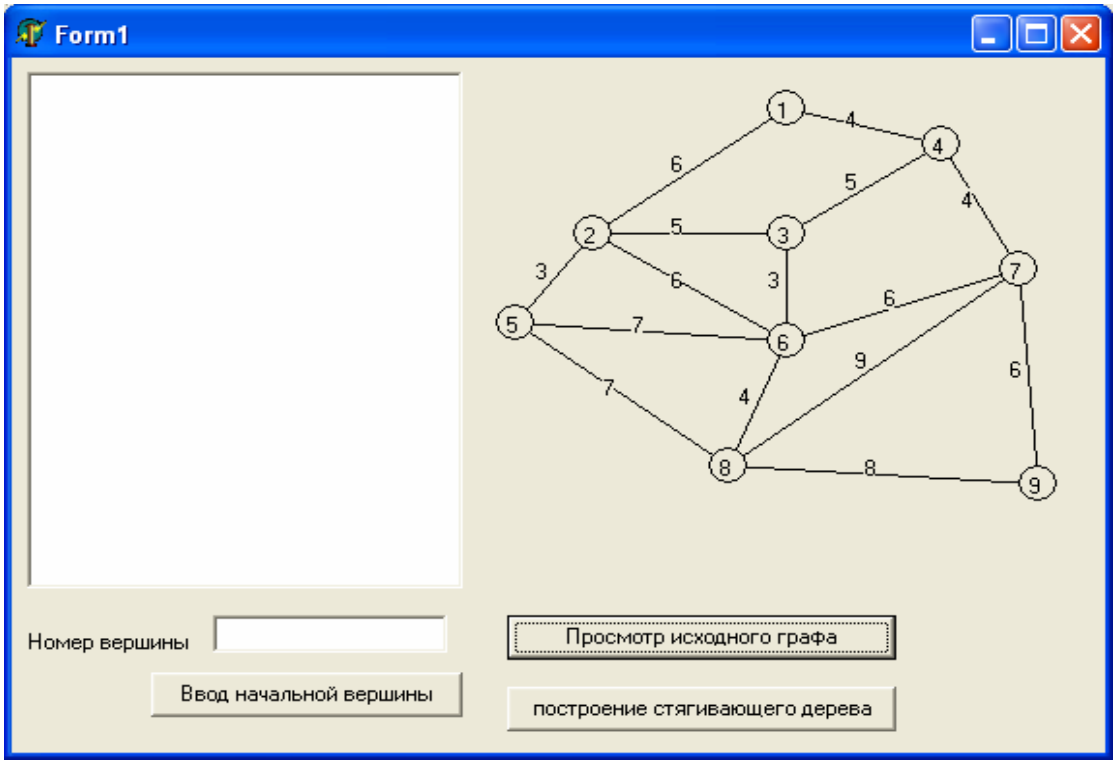

Рисунок 6.3 – Исходный связанный граф

6.2.3 Алгоритм Дейкстры

Алгоритм Дейкстры требует использования трех массивов, размерность которых соответствует количеству вершин графа.

Первый массив, массив «постоянных» вершин, будет хранить минимальные маршруты от заданной вершины графа до всех остальных его вершин. Первоначально в этот массив записывается номер начальной вершины (вершины, от которой будем находить минимальные расстояния для всех других вершин графа). Название массива соответствует названию выбранных вершин в алгоритме Дейкстры, согласно которому сначала все вершины графа объявляются «временными», а выбранные вершины называются «постоянными». Начальная вершина объявляется «постоянной».

Второй массив предназначен для хранения минимальных расстояний от заданной вершины графа до всех остальных его вершин. Первоначально в него переписывается строка матрицы смежности, соответствующая номеру выбранной вершины.

В третьем массиве логического типа отмечаются выбранные (постоянные) вершины графа. Первоначально «постоянной» вершиной графа является только начальная вершина.

Далее выбирается «временная» вершина, до которой расстояние от начальной (постоянной) вершины минимально. Эта вершина объявляется вершиной k. Проверяются все маршруты от «постоянной» вершины до всех «временных» вершин графа как непосредственно, так и через вершину k.Минимальные расстояния заносятся вво второй массив, а в первый массив, в соответствующую позицию, записывается k, если маршрут минимального расстояния проходил через вершину k.

Далее вершина k объявляется «постоянной» (это фиксируется в третьем массиве) и алгоритм поиска следующей вершины графа повторяется относительно новой «постоянной» вершины.

6.3 Пример выполнения задания на лабораторную работу

Задача 6.1 Для связанного графа, изображенного на рисунке 6.3 построить стягивающее дерево. Номер начальной вершины графа задавать в режиме диалога. Стягивающее дерево представить списком маршрутов от заданной вершины до остальных вершин графа.

Исходный код программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
classProgram
 {
```

```
publicstaticint[,] a = newint[9, 9]
   {{   0,   6,1000,   4,1000,1000,1000,1000,1000},
    {   6,   0,   5,1000,   3,   6,1000,1000,1000},
    {1000,   5,   0,   5,1000,   3,1000,1000,1000},
    {   4,1000,   5,   0,1000,1000,   4,1000,1000},
    {1000,   3,1000,1000,   0,   7,1000,   7,1000},
    {1000,   6,   3,1000,   7,   0,   6,   4,1000},
    {1000,1000,1000,   4,1000,   6,   0,   9,   6},
    {1000,1000,1000,1000,   7,   4,   9,   0,   8},
    {1000,1000,1000,1000,1000,1000,   6,   8,   0}};

publicstaticint[] d = newint[10];
publicstaticint[] post = newint[10];
publicstaticbool[] t = newbool[10];
publicstaticint i, j, p, k, minras, begver;

publicstaticvoid poisk()
{
string buf;
//Ввод значения переменной a в режиме диалога
Console.WriteLine("Введите номер вершины графа целое значение 0
- 8");
buf = Console.ReadLine();
   begver = Convert.ToInt32(buf);
//начальные установк
for (i = 0; i < 9; i++)
{
     post[i] = begver;
     t[i] = true;
d[i] = a[begver,i];
    }
   t[begver] = false;
   post[begver] = 0;
   minras = 0;
for (i = 0; i < 8; i++)
    {
// поисквершины k
     minras = 1000;
for (j = 0; j < 9; j++)
if ((t[j] == true) && (minras > d[j]))
{
        minras = d[j]; k = j;
}
// поиск маршрутов и минимальных расстояний через вершину k
t[k] = false;
for (j = 0; j < 9; j++)
if ((t[j] == true) && (d[j] > d[k] + a[k,j]))
      {
       d[j] = d[k] + a[k,j];
       post[j] = k;
      }
    }
```

```
  }


publicstaticvoid print()
{
Console.WriteLine("Печатаемматрицусмежности : ");
for (i = 0; i < 9; i++)
    {
for (j = 0; j < 9; j++)
Console.Write("\t" + a[i, j]);
Console.WriteLine();
    }
Console.WriteLine();
// печатьномероввершинграфа
for (i = 0; i < 9; i++) Console.Write("\t" + i);
Console.WriteLine();
// печать массива минимальных расстояний
for (i = 0; i < 9; i++) Console.Write("\t" + d[i]);
Console.WriteLine();
// печатьмассивамаршрутов
for (i = 0; i < 9; i++) Console.Write("\t" + post[i]);
Console.WriteLine();
Console.WriteLine();
int dlin;
// печать маршрутов стягивающего дерева
for (i = 0; i < 9; i++)
    {
     dlin = d[i];
Console.Write("Путь № {0}-{1} длина пути = {2}\t\t{3}", i,
begver, dlin,i);
p = i;
if (i != begver)
      {
do
        {
         p = post[p];
if (p!=0) Console.Write("\t" + p);
}
while (p != 0);
Console.WriteLine();
      }
elseConsole.WriteLine();
    }
  }

staticvoid Main()
  {
   poisk();
   print();
Console.ReadLine();
  }
 }
```

```
}
```

Работапрограммы:

Введите номер вершины графа целое значение 0 - 8

4

Печатаем матрицу смежности :

0	6	1000	4	1000	1000	1000	1000	1000
6	0	5	1000	3	6	1000	1000	1000
1000	5	0	5	1000	3	1000	1000	1000
4	1000	5	0	1000	1000	4	1000	1000
1000	3	1000	1000	0	7	1000	7	1000
1000	6	3	1000	7	0	6	4	1000
1000	1000	1000	4	1000	6	0	9	6
1000	1000	1000	1000	7	4	9	0	8
1000	1000	1000	1000	1000	1000	6	8	0

0	1	2	3	4	5	6	7	8
9	3	8	13	0	7	13	7	15
1	4	1	2	0	4	5	4	7

Путь № 0-4 длина пути = 9 0 1 4

Путь № 1-4 длина пути = 3 1 4

Путь № 2-4 длина пути = 8 2 1 4

Путь № 3-4 длина пути = 13 3 2 1 4

Путь № 4-4 длина пути = 0 4

Путь № 5-4 длина пути = 7 5 4

Путь № 6-4 длина пути = 13 6 5 4

Путь № 7-4 длина пути = 7 7 4

Путь № 8-4 длина пути = 15 8 7 4

6.4 Домашнее задание на лабораторную работу

Разработать программу для реализации алгоритма обхода графа в «ширину».Использоватьсвойграф.Количество вершин не менее 10.

6.5 Индивидуальные задания для СРС

6.5.1 Создать граф – авиационных перевозок, узлы которого дополнительно включают символьное поле – название областных центров республики. Предусмотреть поиск минимального маршрута перемещения от заданного областного центра до всех остальных центров республики.

6.5.2 Создать свой граф не менее 10 вершин, узлы которого соответствуют некоторой электронной схеме без активных элементов. Найти сопротивление каждой цепей между узлами схемы, заданными в режиме диалога.

6.5.3 Схему автобусных маршрутов города представить структурой типа граф. Узлы структуры соответствуют остановкам автобусных маршрутов и дополнительно включают название остановок. Предусмотреть просмотр номеров маршрутов по названию остановки.
6.5.4 Создать граф имен студентов группы (допускается использование одинаковых имен). Предусмотреть поиск студентов по имени заданному в режиме диалога.
6.5.5 Генеалогическое дерево некоторого рода представлено графом не более 12 вершин. Узел каждой вершины графа дополнительно включает пол представителя рода. Организовать поиск и печать всех особ женского пола с помощью обхода графа в «глубину».
6.5.6 Четыре трамвайных маршрута города представлены структурой типа граф. Узлы структуры соответствуют остановкам трамвайных маршрутов и дополнительно включают название остановок. Для двух названий остановок, введенных в режиме диалога, найти минимальный маршрут перемещения от первой остановки до второй (по минимальной сумме расстояний пройденных остановок).
6.5.7 Генеалогическое дерево некоторого рода представлено графом не более 14 вершин. Узел каждой вершины графа дополнительно включает имя представителя рода. Организовать поиск наиболее часто встречающегося мужского и женского имени.
6.5.8 Четыре трамвайных маршрута города представлены структурой типа граф. Узлы структуры соответствуют остановкам трамвайных маршрутов и дополнительно включают название остановок. Для двух названий остановок, введенных в режиме диалога, найти минимальный маршрут перемещения от первой остановки до второй (по количеству пройденных остановок).
6.5.9 Генеалогическое дерево некоторого рода представлено графом не более 15 вершин. Узел каждой вершины графа дополнительно включает основной вид деятельности представителя рода и время его работы по этой профессии. Организовать поиск профессии с максимальным временем работы.
6.5.10 Четыре трамвайных маршрута города представлены структурой типа граф. Узлы структуры соответствуют остановкам трамвайных маршрутов и дополнительно включают название остановок. Напечатать названия остановок города в порядке убывания числа маршрутов, проходящих через эти остановки.
6.5.11 Иерархическая структура каталогов диска С компьютера представлена структурой типа граф, узлы которого соответствуют папкам каталога и дополнительно включают название папок. Определить, есть ли на диске одинаковые папки. Напечатать их и отобразить путь к ним от корневого каталога.
6.5.12 Создать свой граф не менее 10 вершин, узлы которого дополнительно включают символьное поле. Разработать алгоритм обхода графа в «глубину» только по «гласным» вершинам графа(дополнительное символьное поле содержит «гласный» символ).

6.5.13. Иерархическая структура каталогов диска С компьютера представлена структурой типа граф, узлы которого соответствуют папкам каталога и дополнительно включают название папок. Определить, сколько раз на диске встречается папка с названием Games.
6.5.14 Создать свой граф не менее 10 вершин, узлы которого дополнительно включают символьное поле. Разработать алгоритм обхода графа в «глубину» только по «согласным» вершинам графа (дополнительное символьное поле содержит «согласный» символ).
6.5.15 Схему автобусных маршрутов района области представить структурой типа граф (не менее 10 вершин). Узлы структуры соответствуют названиям поселков района. Предусмотреть просмотр номеров маршрутов по названию остановки. Для названия поселка, введенного в режиме диалога, найти минимальный маршрут перемещения от районного центра до этого поселка (по минимальной сумме расстояний пройденных остановок).
6.5.16 Водопроводная сеть микрорайона города представлена ориентированным графом не менее 15 вершин, а дуги – пропускной способности участка сети (количество воды, подаваемое в секунду). Определить какое максимальное количество воды можно подавать в узлы A, B и C из некоторого узла X. Значение всех узлов задаются в режиме диалога. Использовать равномерное распределение подаваемой воды между принимающими узлами (если это могут обеспечить подводящие дуги).
6.5.17 Создать свой граф не менее 10 вершин, узлы которого соответствуют некоторой электронной схеме без активных элементов. Найти сопротивление цепей между узлами схемы, заданными в режиме диалога (учитывать параллельное и последовательное соединение схемы).
6.5.18 Лабиринт представлен графом не менее 16 вершин, где вершины соответствуют перекресткам или тупикам. Известны узлы входа и выхода лабиринта. Найти минимальный маршрут прохождения лабиринта.
6.5.19 Четыре трамвайных маршрута города представлены структурой типа граф. Узлы структуры соответствуют остановкам трамвайных маршрутов и дополнительно включают название остановок. Для двух названий остановок, введенных в режиме диалога, найти минимальный маршрут перемещения от первой остановки до второй (по количеству совершаемых пересадок, но не количеству остановок). Напечатать путь перемещения с указанием названий остановок и номеров маршрутов трамваев, на которые необходимо садится или пересаживаться.
6.5.20 Четыре трамвайных маршрута города представлены структурой типа граф. Узлы структуры соответствуют остановкам трамвайных маршрутов и дополнительно включают название остановок. Для двух названий остановок, введенных в режиме диалога, найти минимальный маршрут перемещения от первой остановки до второй (по количеству остановок). Напечатать путь перемещения с указанием названий остановок и номеров маршрутов трамваев, на которые необходимо садится или пересаживаться.

6.6 Контрольные вопросы для защиты отчета на СРСП

6.6.1 Понятие графа. Примеры.
6.6.2 Понятие инцидентности. Пример.
6.6.3 Понятие смежности. Пример.
6.6.4 Понятие пути графа. Какой путь графа называется простым? Пример.
6.6.5 Какой граф называется связанным? Пример.
6.6.6 Как в памяти ЭВМ можно разместить информацию о графе?
6.6.7 Объясните представление графа в памяти ЭВМ в виде списков смежных вершин.
6.6.8 Алгоритм поиска минимальных расстояний между вершинами графа.
6.6.9 Алгоритм нахождения минимальных маршрутов между вершинами графа. 6.6.10 Алгоритм обхода графа в «глубину».
6.6.11 Алгоритм обхода графа в «ширину».
6.6.12 Алгоритм построения «стягивающих» деревьев графа.
6.6.13 Алгоритм нахождения всех маршрутов между двумя заданными вершинами графа.
6.6.14 Зачем в алгоритме поиска всех циклов графа просмотренным вершинам графа «возвращается» свойство новой вершины?
6.6.15 Объясните структуру стека, которая используется в алгоритме поиска всех маршрутов графа между двумя заданными вершинам.

7 Список рекомендуемой литературы

7.1. Основная литература

7.1.1 Презентации лекций по дисциплине «Алгоритмизация и основы программирования» для студентов специальности 5В070400 – смотри портал кафедры ИС http : \\ www.do.ektu.kz
7.1.2 В.В. ФароновСоздание приложений с помощьюC# Руководство программиста. - М.: “Эксмо”, 2008г.
7.1.3Т.А. ПавловскаяC#, Программирование на языке высокого уровня. Учебник для вузов, СПб.: Питер, 2009г.
7.1.4Д. Кнут. Искусство программирования для ЭВМ. Т.3./ Сортировка и поиск / - М.:Мир,1976.

7.2 Дополнительная литература

7.2.1 Э. Йодан Структурное программирование и конструирование программ. М.: ”Мир”, 1989г.
7.2.2 Н. ВиртАлгоритмы и структуры данных. М. Изд-во «МИР», 1989г.
7.2.3 Э.ТроелсенC# и платформа .NET Библиотека программиста, СПб,: Питер, 2007г.