МИНОБРНАУКИ РОССИИ

ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ЭКОНОМИКИ И СЕРВИСА

КАФЕДРА МАТЕМАТИКИ И МОДЕЛИРОВАНИЯ

Методические рекомендации по выполнению лабораторных работ по дисциплине (модулю)

**ОСНОВЫ АЛГОРИТМИЗАЦИИ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ**

Направление и профиль подготовки:

38.03.05 Бизнес-информатика. Бизнес-аналитика

Форма обучения

очная

Составитель:

*Штука Виктор Игоревич, кандидат физико-математических наук, доцент, Кафедра математики и моделирования*

Утверждены на заседании кафедры математики и моделирования от 20.03.2020, протокол № 8.

Владивосток, 2020 г.

**Лабораторная работа №1. Тема «Основные принципы алгоритмизации и программирования»**

Нарисовать блок-схему и cоставить программу вычисления площади треугольника по двум сторонам и углу между ними (угол в градусах).

**Разбор лабораторной работы, пример**

*Задание ориентировано на восприятие, понимание и закрепление базовых навыков организации блок-схем алгоритмов, понятий передачи управления и структурного подхода как результатов работы, цель которой состоит в их достижении. Отчёт по лабораторной работе должен отражать основные моменты и положения, которые необходимы для её практической реализации, такие как определение проблематики работы, разбор задания и его формализации в виде блок-схемы.*

Чтобы выполнить задание, необходимо обратиться к формуле для вычисления площади треугольника по двум сторонам (a и b) и углу γ (радианы) между ними вида S=a·b·cosγ/2

Связь радиальной и градусной мер определяется соотношением 180/π=c/γ, где c – угол в градусах. Учитывая обязательность указания трёх параметров a, b, и c и основываясь на базовых принципах построения блок-схем, за точкой входа должен последовать блок ввода параметров, далее должен последовать следующий блок перевода градусной меры в радианную и соответствующий расчёт площади по приведённой выше формуле. Результат вычислений должен быть определён также дополнительным блоком, после которого желательно добавить блок ожидания и точку выхода для того, чтобы последующий псевдокод, основанный на составленной блок-схеме, мог быть легко интерпретирован на язык программирования и реализован в конкретной среде разработки.

Приветствие

Ввод a, b, c

Вычисление γ

Вычисление S

Вывод на экран S

Ожидание нажатия

точка входа

точка выхода

**Лабораторная работа №2. Тема «Основные элементы языка C#»**

Дано натуральное *n*. Составить программу, вычисляющую

**Разбор лабораторной работы, пример**

*Используя среду разработки Microsoft Visual Studio, необходимо организовать консольное приложение (Console Application) для реализации расчёта соответствующего выражения как результата работы, цель которой состоит в приобретении навыков алгоритмизации. Отчёт по лабораторной работе должен отражать основные моменты и положения, которые необходимы для её практической реализации, такие как определение проблематики работы, разбор задания и его формализацию в виде программного кода.*

Изначально, программа должна запросить у пользователя натуральное число n, чтобы использовать его впоследствии, для чего желательно при запуске приложения вывести уведомление пользователю о том, что эта программа делает и какие действия ожидаются от пользователя. Соответствующие операции относятся к методам класса Console, а обращение к ним происходит по стандартной для объектно-ориентированного подхода к программированию схеме.

Метод ***Console.ReadLine()*** необходим для считывания введённой с клавиатуры информации, результатом которого является строка (тип данных string). Чтобы получить из строки целое число, воспользуйтесь специальной функцией класса ***Convert***, результат которой уже будет целым числом. В данном случае

***int n=Convert.ToInt32(Console.ReadLine());//ввод целого числа с клавиатуры***.

Метод ***Console.WriteLine(string s)*** выводит на экран строку s и имеет ряд перегрузок (вариантов реализации) для форматированного вывода с дополнительными аргументами.

Метод ***Console.ReadKey()*** даёт возможность пользователю ознакомиться с результатами вычислений до тех пор, пока пользователь не нажмёт любой клавиш на клавиатуре, после чего происходит передача управления следующей инструкции кода.

Реализация этапа вычислений может быть основана на рекурсии или с использованием циклических структур типа do, while или for. Поскольку определён явный критерий останова (прекращения выполнения цикла), то из предложенных структур подходящей и наиболее простой окажется цикл со счётчиком for, который в «восходящем» варианте синтаксически записывается так ***for(int i=1; i<=n; i++){}***, если в теле цикла (блок определён фигурными скобками “{}”) присутствует лишь одна инструкция, то фигурные скобки могут быть опущены.

Элементарные математические операции сложения, вычитания, умножения и деления производятся компилятором без обращения к дополнительным библиотекам, классам и модулям, поэтому для записи результата достаточно будет завести переменную p нецелочисленного типа (например, float или double), в которую будет записываться результат произведения. Инициализировать эту переменную можно уже на этапе объявления единицей, поэтому использовать придётся лишь одно выражение для этой инструкции вида ***double p=1;//init p*** – за двойным «слэшем» (символ «/») стоят однострочные комментарии, которые необходимы для того, чтобы разработчик оставлял соответствующие пометки о том, за что отвечает конкретная инструкция, а также для отладки приложения.

Блок цикла для расчёта произведения будет представлен одним выражением ***p=p\*(1+1/(i\*i))***, в котором переменная p перезаписывается новым значением, умноженным на очередной сомножитель, тем самым используя результат предыдущих умножений и сохраняя его. Очерёдность выполнения алгебраических действий определена также, как и в обычной алгебре и если имеется необходимость указать порядок их выполнения, используются круглые скобки “()”. Окончание инструкции обязательно снабжается символом ‘;’.

**Лабораторная работа №3. Тема «Операторы управления и функции»**

1. Реализовать функцию Delete(s, i, n), возвращающую строку, полученную удалением из строки s n символов, начиная с i-го. Не использовать стандартную процедуру Delete.
2. Реализовать функцию UnSeparator(s, sep), возвращающую строку, полученную удалением из исходной всех лишних разделителей (т.е. последовательность из N идущих подряд символов-разделителей заменяется на 1 разделитель). Символы-разделители заданы множеством sep.

**Разбор лабораторной работы, пример**

*Используя среду разработки Microsoft Visual Studio, необходимо организовать консольное приложение (Console Application) для реализации возможностей функционального программирования на примере строкового типа данных как результата работы, цель которой состоит в приобретении навыков и умений обращения функциями, а также со строками для формирования компонент будущих модулей и библиотек обработки информации. Отчёт по лабораторной работе должен отражать основные моменты и положения, которые необходимы для её практической реализации, такие как определение проблематики работы, разбор задания и его формализацию в виде программного кода.*

Функция Delete в том варианте, в котором её необходимо реализовать может быть представлена традиционным образом и записана как ***static string Delete(string s, int i, int n){…}***, а её тело с учётом того, что пользоваться стандартной процедурой нельзя, желательно организовать в виде обработки массива символов, заведя для выходной строки массив из m-n элементов, где m – длина строки s или ***int m=s.Length***.

Результатом функции Delete с параметрами i=4, n=2 для строки ‘сорока’ должна стать строка ‘соро’. Хорошо, чтобы параметры i и n не позволяли бы выходить за пределы строки, вызывая, тем самым, ошибку обращения к несуществующим элементам массива.

Функцию UnSeparator(string s, string sep) оформить возможно через организацию одного цикла для прохода по элементам строки s. Чтобы удалять следующие подряд разделители, необходимо будет использовать символ пустой строки ‘’, а также дополнительную переменную для хранения индекса начала последовательности разделителей. Так как строка есть массив символов, выход за пределы массива (если не использовать символ пустой строки при замене) чреват фатальными последствиями для программы. Для того, чтобы функция вернула строку в качестве результата, необходимо употребление специального ключевого слова ***return***.

Результатом функции UnSeparator для строки ‘Лес --- множество деревьев...;;’ и разделителей ‘.-;’ должна стать строка ‘Лес – множество деревьев.’.

**Лабораторная работа №4. Тема «Указатели. Массивы»**

В файле 1.txt дана матрица nxm, причем её размерность заранее неизвестна. Записать в файл 2.txt столбец, составленный из максимумов строк матрицы.

**Разбор лабораторной работы, пример**

*Используя среду разработки Microsoft Visual Studio, необходимо организовать консольное приложение (Console Application) для анализа и обработки данных, хранящихся в файловой системе как результата работы, цель которой состоит в приобретении умений и навыков обращения с массивами и анализа данных. Отчёт по лабораторной работе должен отражать основные моменты и положения, необходимые для её практической реализации, такие как определение проблематики работы, разбор задания и его формализацию в виде программного кода.*

Чтение и запись происходит аналогично описанной для лабораторной работы №3 методике, поэтому здесь важно обратить внимание именно на устройство массива, его организацию и систему вложенных циклов, ответственных за выявление максимумов значений элементов в строках матрицы. Важно не забывать о том, что индексация в массивах начинается с нуля, поэтому индекс последнего элемента в строке из n символов будет n-1. Положим, что матрица определена целыми числами, тогда её считывание из файла будет записано в следующей форме

***StreamReader s=new StreamReader(@’1.txt’);//инициализация считывания***

***int[, ] a= new int[n, m];//объявление массива из n столбцов и m строк***

***for(int i=0; i<n; i++) for(int j=0; j<m; j++) a[i, j]= Convert.ToInt32(s.Read());***

***s.Close();//очень важная операция, предотвращающая многие ошибки***

Определение максимального элемента в строке происходит путём прохода по элементам с одновременным сравнением их с текущим максимальным значением и последующей записью результата в файл. Осуществляется это для каждой строки всего в один проход, а максимальное значение хранится при этом в буферной переменной b (вначале её нужно сопоставить с первым элементом строки), и инструкция получится также однострочной с использованием лишь одного условного оператора

***StreamWriter s=new StreamWriter(@’2.txt’);//инициализация записи***

***…//блок цикла для прохода по строкам***

***int b=a[0, j]; for(i=1; i<n; i++) If[a[i, j]>b] b=a[i, j]; s.WriteLine(b);***

***…//завершения блока цикла***

***s.Close();//очень важная операция, предотвращающая многие ошибки***

**Лабораторная работа №5. Тема «Работа с файлами. Структуры и классы»**

В файле 1.txt дана строка символов. Записать в файл 2.txt строку, полученную из исходной в соответствии с правилом: заменить все вхождения подстроки ‘zxc’ подстрокой ‘asdf’, удалить все символы ‘.’.

**Разбор лабораторной работы, пример**

*Используя среду разработки Microsoft Visual Studio, необходимо организовать консольное приложение (Console Application) для реализации текстового алгоритма, исполняющего несколько элементарных действий как результата работы, цель которой и состоит в приобретении навыков элементарной обработки информации и умению обращения к файловой системе. Отчёт по лабораторной работе должен отражать основные моменты и положения, которые необходимы для практической реализации работы, такие как определение её проблематики, разбор задания и его формализацию в виде программного кода.*

Для чтения данных из файла и записи данных в файл используется модуль System.IO, а его методы StreamReader() и StreamWriter() для инициализации этапа чтения и записи соответственно, задавая в качестве аргумента имя файла из папки (директории) проекта или же указывая его абсолютный адрес в файловой системе. Существует возможность построчного обращения с файлом при помощи методов ReadLine() и WriteLine(). Ознаменованием конца файла является «нулевая» строка (null), встреча с которой и будет являться критерием останова цикла с предусловием while. Чтение из файла может быть выполнено построчно стандартной процедурой, после которой доступ к файлу прекращается и оперативная память, выделенная для его временного хранения, освобождается посредством метода Close()

***StreamReader sr= new StreamReader(@'1.txt’);//инициализация считывания***

***string s;//переменная для считываемой строки***

***while(s=sr.ReadLine()!=null) Console.WriteLine(s);//чтение файла***

***sr.Close();//завершение работы с файлом***

Аналогичным образом обстоят дела с записью данных в файл 2.txt.

Замена подстроки другой подстрокой происходит посредством обращения к классу String (строки), к которому относятся переменные все строкового типа, и его методу Replace(). В данном случае инструкция с будет иметь вид s=s.Replace(‘zxc’, ’asdf’) для замены подстроки и s=s.Replace(‘.’, ’’), т.е. аналогичной заменой символа точки на пустую строку – такой приём встречается в прикладных задачах форматирования строк довольно часто.

Также существует возможность поэлементного прохода по строке, поскольку строка представляет собой массив символов (char[]) и содержит в себе полный функционал операций со списками и массивами.

**Лабораторная работа №6. Тема «Сортировка и поиск»**

Сгенерировать случайным образом массив А размера 10х10 (заполнить его целых чисел в пределах от -100 до 100). Затем необходимо

1. Вывести его в файл;

2. Выписать все четные элементы;

3. Определить сколько элементов массива делится на 3;

4. Сформировать новый массив B такой, что

5. Сформировать новый массив С так, чтобы в нем элементы массива А были упорядочены по возрастанию по строкам (A00 – наименьший элемент ВСЕГО массива, A99 – наибольший элемент ВСЕГО массива).

**Разбор лабораторной работы, пример**

*Используя среду разработки Microsoft Visual Studio, необходимо организовать консольное приложение (Console Application) для закрепления навыков анализа и обработки структурированных данных как результата работы, цель которой и состоит в его достижении. Отчёт по лабораторной работе должен отражать основные моменты и положения, которые необходимы для её практической реализации, такие как определение проблематики работы, разбор задания и его формализацию в виде программного кода.*

Чтобы заполнить массив случайными целыми числами из диапазона, необходимо воспользоваться классом Random, создав его экземпляр, а затем обратившись к его методу Next(). На эту операцию потребуется всего несколько инструкций

***Random rnd= new Random();//конструктор для случайного числа***

***foreach(a in A) a=rnd.Next(-100, 100);//заполнение массива A случайными числами***

Признаком делимости одного числа на другое будет являться нулевой остаток от деления, за которым закреплена простая операция %, т.е. условие a%b==0 означает что число a делится на число b без остатка или a кратно b. Данное свойство необходимо использовать в этой работе при организации проверки делимости элемента на 2, 3, а также формировании из исходного массива a нового массива b, соответствующего определённым выше критериям. Иногда, удобнее использовать обратные условия, например, такое a%b!=0, чтобы обеспечить непосредственное исполнения инструкции, а для остальных случаев оставить условие по умолчанию в контексте употребления операторов if-else.

Упорядочивание элементов массива происходит с применением методов обычной (пузырьковой, выбора и т.д.) или улучшенной (логарифмической и т.п.) сортировки. В данном случае желательно завести дополнительный массив-строку D из 100 элементов для промежуточных действий, сегменты которого и будут составлять уже столбцы массива c. Распаковка и упаковка должна происходить по следующей схеме

***int[] D = new int[100];//выделение памяти под буферный массив***

***for(i=0; i<10; i++) for(j=0; j<10; j++) D[i+j\*10]=A[i, j];//распаковка матрицы***

***…//сортировка элементов d по возрастанию***

***for(int k=0; k<100; k++) C[k/10, k%10]=D[k];//упаковка строки в матрицу***

**Лабораторная работа №7. Тема «Графика в Windows»**

Цель работы – изобразить функцию. Для этого выбрать три характерных варианта графика. Вывод графика осуществлять в созданном на экране окне. У пользователя программа запрашивает параметры графика (A, B и т.д.) и цвет в который будет окрашен график.

Улитка Паскаля: {x=Acos2(t)+Bcos(t); y=Acos(t)sin(t)+Bsin(t), A>B, B>0, 0<=t<2Pi}

Рассмотреть случаи, когда B>=2A; A<B<2A; A>B.

**Разбор лабораторной работы, пример**

*Используя среду разработки Microsoft Visual Studio, необходимо организовать форму оконного приложения (Windows Forms Application) для реализации соответствующего функционала с использованием графического и интерактивного интерфейсов как результата работы, цель которой состоит в приобретении умений и навыков создания и компоновки графических приложений с соответствующим функционалом, а также возможностью интерактивного взаимодействия с пользователем. Отчёт по лабораторной работе должен отражать основные моменты и положения, которые необходимы для её практической реализации, такие как определение проблематики работы, разбор задания и его формализацию в виде программного кода.*

Чтобы реализовать заложенный в задании функционал, на поле формы для ввода параметров графика и выбора его цвета желательно поместить два элемента-бегунка NumericUpDown и один элемент ColorDialog (обращение к нему желательно сделать через кнопку Button), сам же график удобно будет изобразить внутри элемента PictureBox.

Для правильного отображения важно учитывать, что начало координат в приложениях располагается традиционно в левом верхнем углу, поэтому соотносить такое представление с принятым в математике необходимо по формулам

x’·a=k·(x-xc)·w, y’·b=k·(y-yc)·h,

где (x’, y’) – экранные координаты, (x, y) – реальные координаты, (xc, yc) – реальные координаты сдвига графика, (a, b) – область отрисовки в реальных координатах, (w, h) – ширина и высота экранной области отрисовки, k – масштабный коэффициент.

Важно отметить, что тригонометрические функции класса Math работают с радианной мерой аргумента, поэтому градусы необходимо переводить в радианы, используя встроенную константу Math.PI для числа π в ответственных задачах (здесь же значение 2π≈6.28). Параметрическую зависимость желательно завести в функции fx и fy, уместив всё в одну процедуру ReDraw() вида

***Graphics g= pictureBox1.CreateGraphics();//создание «холста» для рисования***

***int w= pictureBox1.width; int h= pictureBox1.height;//определение параметров***

***Point[] pt= new Point[360];//выделение памяти под массив точек ломанной***

***g.Clear(pictureBox1.BackColor);//предварительная очистка «холста»***

***for(int t=0; t<360; t++){//инициализация цикла для градусов t-параметра***

***pt[i].X=(int)((k\*w\*fx(A, B, t/6.28)-xc)/a);***

***pt[i].Y=(int)((k\*h\*fy(A, B, t/6.28)-yc)/b);}***

***g.DrawLines(Pen.Red, pt);//изображение ломанной красного цвета на «холсте»***

***g.Dispose();//размещение примитивов на «холсте» и его освобождение***

Чтобы добавить приложению интерактивности, воспользуйтесь обработчиком событий Click() для кнопки и ValueChanged() для бегунков, тогда при изменении параметров A или B будет происходить перерисовка графика! Для этого достаточно связать эти события (находятся во вкладке свойств элементов) с инструкцией, вызывающей метод ReDraw(), описанный выше.