

Практические занятия по курсу «Разработка ПС»

Данный вариант документа не окончателен и может быть дополнен в течение семестра

Преподаватели:

Иван Григорьевич Корниенко

Алексей Константинович Федин

1 Назначение курса

- 1 Упражнения и практические занятия курса «Разработка ПС» предназначены для закрепления навыков, полученных при прослушивании лекционного материала по соответствующему курсу.

2 Общие требования

2.1 Порядок выполнения практической работы

- 1 Выполнение работы начинается с получения задания.
- 2 Студент должен ознакомиться с заданием на работу. В случае если задание непонятно, он может проконсультироваться у преподавателя.
- 3 Практическая работа выполняется каждым студентом индивидуально. Номер варианта определяется порядковым номером студента в списке группы. Первый в списке получает первый вариант. Если предположить, что вариантов шесть, то седьмой студент по списку получает снова первый вариант. Студент не может выбрать другой вариант.
- 4 После получения и согласования с преподавателем задания на работу студент приступает к выполнению теоретической части работы. В ходе теоретической части работы студент разрабатывает:
 - требования к программному продукту;
 - математические методы и алгоритмы;
 - структуру программы;
 - форматы представления данных.
- 5 После разработки теоретической части студент представляет преподавателю разработанный материал, и, получив разрешение преподавателя, приступает к непосредственному кодированию разработанной программы на компьютере.
- 6 Написав программу, студент должен её тщательно протестировать. После этого он должен сдать программу преподавателю, продемонстрировав её работу и исходный код.
- 7 По завершении работы студент оформляет отчёт, к которому прилагается исходный код программы.

2.2 Требования к программному коду

- 1 Все идентификаторы (названия переменных, функций, классов и модулей) должны отражать назначение именуемых объектов. Все названия должны быть даны на английском языке. Не допускается использование транслитерации и других нестандартных приёмов именования.
- 2 Все константные литералы в коде должны быть объявлены как константы (const).
- 3 Стилль оформления кода (отступы, именование переменных, функций, классов и т.д.) должен последовательно соблюдаться в работе и соответствовать одному из общепринятых стандартов (например, [C# Naming Guidelines](#)).

2.3 Требования к программе

- 1 Программа при запуске должна выводить подробную информацию о назначении программы, авторе, решаемой задаче и предоставляемых результатах. В случае если программа выполнена с использованием графического интерфейса пользователя (приложение Windows) – программа должна выводить соответствующую информацию при запуске и в ответ на выбор соответствующего пункта меню. В этом случае должна быть возможность отключить вывод приветствия при запуске программы.
- 2 Выполнение консольной программы должно быть закольцовано. Завершение работы программы должно производиться только при выборе соответствующего пункта меню.

2.4 Общие требования к работам

- 1 Все ошибки, которые могут возникнуть в программе должны быть обработаны с выдачей соответствующих диагностических сообщений. Метод обработки ошибок должен быть единым для всей программы.
- 2 Интерфейс пользователя должен быть полностью отделён от вычислительных процедур программы.

2.4.1 Работа с файлами

- 1 Во всех работах кроме консольного ввода-вывода (экран, клавиатура) необходимо предоставить возможность:
 - вводить исходные данные из файла;
 - сохранять исходные данные в файле;
 - сохранять результат работы программе в файле.
- 2 Полный путь к файлам в каждом случае задает пользователь.

2.4.2 Тестирование

- 1 Во всех работах должны присутствовать модульные тесты.

2.5 Содержание отчёта

- 1 В отчёт входит описание:
 - постановки задачи;
 - исходных данных;
 - особых ситуаций;
 - математических методов и алгоритмов решения задачи;
 - форматов представления данных в памяти и на внешних носителях;
 - структуры программы;
 - модулей, функций и переменных программы;
 - блок-схем алгоритмов программы;
 - хода выполнения работы;
 - полученных результатов.
- 2 Отчёт оформляется на листах формата А4 с обязательным титульным листом, на котором указываются название работы; ФИО исполнителя; ФИО преподавателей и т.д.

2.5.1 Постановка задачи

- 1 Постановка задачи указывает, какая цель должна быть достигнута при разработке программы. Какую задачу должна решать программа, и в каких условиях будет функционировать.

2.5.2 Исходные данные

- 1 Исходными данными являются любые данные, которые программа получает для обработки.
- 2 Описание исходных данных должно содержать:
 - семантику (назначение) данных;
 - единицы изменения;
 - представление в программе.

2.5.3 Особые ситуации

- 1 Под особыми ситуациями понимаются ситуации, в которых поведение программы может не соответствовать поведению, ожидаемому пользователем.

- 2 Все особые ситуации должны быть описаны и соответствующим образом обработаны в программе.
- 3 Примерами особых ситуаций являются:
 - некорректный ввод;
 - отсутствие ожидаемых программой файлов;
 - возможное деление на ноль в ходе вычислений;
 - нехватка оперативной памяти.

2.5.4 Математические методы и алгоритмы решения задач

- 1 Все используемые программой алгоритмы и математические методы решения задач должны быть описаны в специальном разделе в форме и полноте, достаточной для восприятия другими разработчиками.

2.5.5 Форматы представления данных

- 1 Для всех пользовательских типов данных (не являющихся частью языка) должны быть документированы назначение и мотивация выбора конкретного типа данных.
- 2 Должны быть документированы форматы всех внешних ресурсов. Структура данных, сохраняемых в файлах и т.д.

2.5.6 Структура программы

- 1 Разработанная структура программы (разделение на модули, интерфейсы, шаблоны проектирования) должна быть документирована.
- 2 Должна быть описана основная последовательность работы программы (вызова функций, методов и т.д.).
- 3 Все модули, функции, методы и пользовательские типы данных должны быть соответствующим образом документированы в отчёте.

2.5.7 Блок-схемы алгоритмов программы

- 1 Построение блок-схем алгоритмов регламентируется ГОСТ 19.701-90 (ИСО 5807-85) «Единая система программной документации. Схемы алгоритмов программ, данных и систем. Условные обозначения и правила выполнения».

2.5.8 Описание хода выполнения работы

- 1 В отчёте должно содержаться подробное описание хода выполнения работы.
- 2 Основное внимание должно быть уделено выполнению работы за компьютером.
- 3 Должны быть описаны все новые методы и приёмы, использованные в ходе выполнения работы. Таковыми могут быть:
 - создание проекта и запуск программы;
 - работа с отладчиком (точки останова и протоколирования, просмотр значений переменных);
 - поиск ошибок в программе;
 - работа с устройствами ввода-вывода.
- 4 Из отчёта должно быть ясно:
 - как выполнялась работа;
 - какие были проблемы и как они были решены;
 - какие проблемы остались нерешёнными;
 - какие моменты были или остались непонятными.

2.5.9 Результаты работы программы

- 1 Необходимо указать, какие результаты производит программа.
- 2 Необходимо указать в каком формате пользователь получает результат.

- 3 Должно быть приведено три варианта результатов выполнения программы с различными исходными данными.

2.5.10 Исходный текст программы

- 1 Исходный текст программы распечатывается и прилагается к отчёту.

2.5.11 Документирование и комментирование исходного текста

- 1 Все пользовательские типы данных должны быть прокомментированы.
- 2 Все функции, классы и модули должны быть прокомментированы.
- 3 Каждый модуль должен начинаться с комментария, указывающего его назначение, автора, используемые алгоритмы.
- 4 Каждая нетривиальная функция должна предваряться комментарием, описывающим:
 - назначение;
 - входные данные;
 - результаты.
- 5 В функциях, где соответствующее описание будет полезным, также следует описать:
 - предусловия;
 - постусловия;
 - инварианты.

2.6 Защита и сдача работы

- 1 Работа защищается преподавателям, ведущим практические работы.
- 2 Для защиты необходимо иметь отчёт о проделанной работе и продемонстрировать работоспособную программу.
- 3 Подпись за работу выставляет преподаватель, которому работа была защищена.
- 4 Окончательная сдача работ является допуском к экзамену (или зачёту) по предмету.

2.6.1 Окончательная сдача работ

- 1 Для сдачи работ, необходимо представить:
 - комплект отчётов по работам;
 - папку с исходными кодами программ и выполняемыми модулями.
- 2 В папке должен содержаться файл readme.txt, в котором должно быть указано:
 - Ф.И.О. выполнившего работы
 - Год, название предмета, названия работ.

2.7 Пример выполнения задания

При выполнении работы будет использоваться компилятор Microsoft Visual Studio 2017.

2.7.1 Постановка задачи

Методом Монте-Карло вычислить число «пи».

2.7.2 Исходные данные

В качестве исходных данных программа использует вводимое пользователем число испытаний при проведении эксперимента.

2.7.3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации.

- Если пользователь ввёл число испытаний меньше одного, то эксперимент провести невозможно.

- Если пользователь ввёл число испытаний меньше разумного предела для проведения статистического эксперимента, результаты могут быть недостоверными.
- Если пользователь ввёл очень большое число испытаний, то эксперимент может занять значительное время, о чём пользователь должен быть предупреждён.

2.7.4 Математические методы и алгоритмы решения задач

Согласно постановке задачи для составления программы будет использован метод Монте-Карло, который заключается в случайном выборе координат точек внутри квадрата заданного размера.

Для каждой точки будет проверяться попадание во вписанную в квадрат окружность. Таким образом, по окончании эксперимента мы будем располагать двумя числами:

- N – число случайно выбранных точек;
- M – число точек, попавших внутрь вписанной окружности.

Поскольку нам известна площадь квадрата и площадь вписанной окружности, то мы можем вычислить отношение площадей этих фигур. Если принять сторону квадрата за единицу, то его площадь будет равна одной квадратной единице.

Площадь круга, вписанного в этот квадрат, может быть определена по формуле:

$$S = \pi r^2 = \frac{\pi d^2}{4} \quad (1)$$

Таким образом, число π можно выразить через площадь и диаметр вписанного круга следующим образом:

$$\pi = 4 \frac{S}{d^2} \quad (2)$$

Площадь вписанного круга можно найти из отношения M к N. Они относятся друг к другу так, как относится площадь вписанной окружности к площади квадрата. А площадь квадрата нам известна – она единична. Исходя из этого, получаем полную формулу для вычисления числа π по известным нам числам M и N.

$$\pi = \frac{4}{d^2} \cdot \frac{M}{N} \quad (3)$$

Все величины в данной формуле нам известны. Однако, как было указано выше, M является статистической величиной, которая будет рассчитана в результате проверки попадания случайных точек в окружность. Для генерации набора точек и проверки их попадания в круг необходимо написать программу.

2.7.5 Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемые в программе

Имя	Тип	Описание
tries	int	Число попыток в эксперименте
inCircle	int	Число точек, попавших в окружность

Для задания минимального и максимального пределов числа экспериментов используются следующие константы

Таблица 2 – Константы, используемые в программе

Имя	Тип	Значение	Описание
LowerLimit	const int	100	Минимальное число экспериментов, обеспечивающих достоверность
UpperLimit	const int	10000000	Число экспериментов, при превышении которого вычисления могут быть долгими

Для задания координат точки на плоскости используется структура *Point2D*, в которой задаются *x* и *y* координаты точки.

2.7.6 Структура программы

В силу своей простоты программа помещена в одном исполняемом модуле. Программа разделена на несколько функций:

Таблица 3 – Функции, составляющие программу

Имя	Описание
GeneratePoint	Создание точки в заданных координатах
IsInsideCircle	Проверка на нахождение точки внутри окружности
ProcessInput	Обработка ввода пользователя

2.7.7 Описание хода выполнения работы

1. В ходе работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio 2015. В нём был создан проект.

2. В начальном варианте программы было перепутано условие выхода из цикла *while*. Было указано:

```
while (InputSuccess) {
```

в результате чего тело цикла ни разу не выполнялось. Для выявления проблемы было использовано пошаговое выполнение программы, в ходе которого выяснилась причина ошибки, и программа была исправлена.

2.7.8 Результаты работы программы

В результате вычислений программа выводит примерное значение числа «пи». Точность вычислений определяется числом проведённых экспериментов и качеством используемого генератора случайных чисел.

2.7.9 Исходный текст программы

```
[ Начало программы ---]
// Lab1.cpp
// Практическая работа №0.
// Использование языка C# для математических расчётов
// Расчёт числа "пи" методом Монте-Карло
// Студент группы NNN, Фамилия Имя Отчество. 2018 год
using System;
```

```
namespace Method_Monte_Carlo
{
```

```

class Program
{
    static void Main(string[] args)
    {
        int inCircle = 0;
        int tries = 0;
        tries = ProcessInput();

        var rand = new Random();

        // Генерация и проверка точек
        for (int i = 1; i <= tries; ++i)
        {
            Point2D pt = GeneratePoint(rand);
            if (IsInsideCircle(pt))
                ++inCircle;
        }

        // Вычисление числа "пи"
        Console.WriteLine("Число пи = ");
        Console.WriteLine(4 * (inCircle / (double)tries));

        Console.ReadKey();
    }

    /// <summary>
    /// Генерация случайной точки с координатами x и y в
    /// интервале от нуля до единицы
    /// </summary>
    static Point2D GeneratePoint(Random rand)
    {
        Point2D pt = new Point2D
        {
            X = rand.NextDouble(),
            Y = rand.NextDouble(),
        };

        return pt;
    }

    /// <summary>
    /// Проверка нахождения точки внутри окружности
    /// </summary>
    static bool IsInsideCircle(Point2D pt)
    {
        const double circleRadius = 0.5;
        Point2D circle = new Point2D
        {
            X = 0.5,
            Y = 0.5,
        }
    }
}

```



```

};

double distFromCenter = Math.Sqrt(
    Math.Pow(circle.X - pt.X, 2) +
    Math.Pow(circle.Y - pt.Y, 2)
);

return distFromCenter < circleRadius;
}

/// <summary>
/// Обработка ввода пользователя
/// </summary>
static int ProcessInput()
{
    const int LowerLimit = 100;
    const int UpperLimit = 10000000;
    bool inputSuccess = false;
    int tries = 0;

    while (!inputSuccess)
    {
        Console.Write("Введите число экспериментов:");
        string consoleInput = Console.ReadLine();

        if (!int.TryParse(consoleInput, out tries))
        {
            Console.WriteLine(
                "Число экспериментов должно быть числом.");
            continue;
        }
        if (tries <= 0)
        {
            Console.WriteLine(
                "Число экспериментов должно быть положительным.");
            continue;
        }
        inputSuccess = true;
    }
    if (tries < LowerLimit)
    {
        Console.WriteLine("Результат может быть неточным.");
    }
    if (tries > UpperLimit)
    {
        Console.WriteLine("Вычисления могут быть длительными.");
    }
    return tries;
}

/// <summary>
/// Структура, описывающая точку на плоскости.
/// </summary>

```

```

    private struct Point2D
    {
        public double X, Y;
    };
}
}
[--- Конец программы.]

```

3 Работа №1. Алгоритмы и структуры данных

Первая лабораторная работа предназначена для приобретения практического опыта в создании простейшего приложения с использованием языка программирования C#.

3.1 Варианты заданий

- 1 Для заданной функции на заданном интервале найти требуемое значение методом половинного деления.
- 2 Для заданной функции на заданном интервале найти требуемое значение методом Ньютона (касательных).
- 3 Для заданной функции на заданном интервале найти требуемое значение методом хорд.
- 4 Для заданной окружности и луча в плоскости определить, пересекает ли луч окружность. Найти координаты точек пересечения.
- 5 Для заданных отрезков на плоскости определить, пересекаются ли они. Найти координаты точки пересечения.
- 6 Для заданной точки и треугольника на плоскости определить, принадлежит ли точка треугольнику.
- 7 В заданном интервале указать все числа, удовлетворяющие одновременно двум условиям:
 - это простые числа;
 - эти числа ряда Фибоначчи.
- 8 Вычислить интеграл заданной функции методом трапеций и методом парабол.
- 9 Для заданного луча и треугольника в трёхмерном пространстве определить, пересекает ли луч треугольник. Указать точку пересечения, если луч пересекает треугольник.
- 10 Для заданного прямоугольника и отрезка на плоскости определить, пересекаются ли они. Найти координаты точек пересечения.
- 11 Для двух окружностей (в плоскости) определить, имеют ли они общую область. Найти площадь данной области.

4 Работа №2. Процесс проектирования

Студент выполняет задание обычной или повышенной сложности

Лабораторная работа предназначена для приобретения практического опыта в работе с наборами объектов в языке программирования C#.

4.1 Варианты заданий обычной сложности

- 1 Напишите программу, находящую медиану массива. То есть индекс ячейки массива, сумма элементов слева от которой минимально отличается от суммы элементов справа.
- 2 В трехмерном пространстве заданы точки (тройками значений x, y, z) и сфера (центр и радиус). Напишите программу, выводящую точки (их координаты), которые попадают в заданную пользователем сферу.
- 3 Напишите программу сортировки элементов массива (любым методом) так, чтобы все чётные элементы оказались в левой части массива, а все нечётные – в правой.
- 4 Напишите программу сортировки элементов массива (любым методом) так, чтобы отрицательные элементы чередовались с положительными.

- 5 Напишите программу двоичного поиска заданного значения в упорядоченном массиве. Программа должна возвращать индекс искомого элемента в массиве или каким-либо образом информировать об отсутствии искомого элемента в массиве.
- 6 Напишите программу, находящую в массиве вещественных чисел последовательность, имеющую максимальную сумму. Программа должна выводить начальный и конечный элемент.
- 7 Напишите программу, удаляющую из массива все дублирующиеся элементы.
- 8 Напишите программу, находящую в массиве две неубывающие последовательности максимальной длины.
- 9 Напишите программу, находящую в массиве значение, встречающееся чаще всего.
- 10 Напишите программу, находящую значение из заданного интервала, отсутствующее в массиве.

4.2 Варианты заданий повышенной сложности

- 1 Напишите программу, находящую в двумерном массиве вещественных чисел двумерный подмассив, сумма элементов которого максимальна для заданного массива.
- 2 Напишите программу, осуществляющую умножение «больших» чисел. Под большим числом понимается целое число, которое может содержать до 100 знаков.
- 3 Напишите программу, находящую в массиве числа, сумма которых наиболее близка к заданному числу.
- 4 Используя знаки математических операций +, -, *, / из заданных чисел постройте выражение, значение которого равно заданному.
- 5 Написать программу, удаляющую из заданного массива минимальное число элементов для того, чтобы массив стал упорядоченным.
- 6 Напишите программу, выполняющую улучшенную сортировку методом прямого обмена («шейкерную сортировку») и сортировку с уменьшающимися расстояниями (сортировку Шелла). Сравните количество операций сравнения и обмена в этих алгоритмах для разных случаев (разных исходных массивов). Выведите результаты в виде таблицы.
- 7 Напишите программу, выполняющую сортировку методами прямого обмена («пузырьковую сортировку»), прямого выбора и прямой вставки. Сравните количество операций сравнения и обмена в этих алгоритмах для разных случаев (разных исходных массивов). Выведите результаты в виде таблицы.

5 Работа №3. Методология программирования

Студент выполняет задание обычной или повышенной сложности.

Необходимо написать приложение с использованием технологии WinForms для построения графика функции и вывода таблицы значений функции. Пользователь задает правую и левую границу, шаг, коэффициенты (при их наличии). При невозможности построить график функции в заданном интервале пользователю выдается предупреждение об этом с предложением сменить границы построения. Если график функции из-за коэффициентов вырождается в точку или не может быть построен пользователь также видит предупреждение.

5.1 Варианты заданий обычной сложности

1. Верзьера Аньези.

$$y = \frac{a^3}{a^2 + x^2}$$

2. Декартов лист

$$y = \pm x \sqrt{\frac{l+x}{l-3x}}, \text{ где } l = \frac{3a}{\sqrt{2}}$$

3. Овал Кассини

$$y = \pm \sqrt{\sqrt{a^4 + 4c^2 x^2} - x^2 - c^2}$$

4. Лемниската Бернулли

$$y = \pm \sqrt{\sqrt{c^4 + 4x^2 c^2} - x^2 - c^2}$$

5. Синусоида

$$y = a + b \sin(cx + d).$$

6. Циклоида

$$x = r \arccos \frac{r-y}{r} - \sqrt{2ry - y^2}$$

7. Трактриса

$$x = \pm \left(a \ln \frac{a + \sqrt{a^2 - y^2}}{y} - \sqrt{a^2 - y^2} \right), \text{ при } y \in (0, a)$$

8. Цепная линия

$$y = \frac{a}{2}(e^{x/a} + e^{-x/a})$$

9. Кубика Чирнгауза

$$27ay^2 = (a-x)(8a+x)^2$$

10. Прямая строфоида

$$y = \pm x \sqrt{\frac{a+x}{a-x}}.$$

11. Циссоида Диокла

$$y^2 = \frac{x^3}{2a-x}.$$

12. Лемниската Жероно

$$y = \pm \sqrt{\frac{a^2 x^2 - x^4}{a^2}}.$$

13. Астроида

$$x^{2/3} + y^{2/3} = R^{2/3}$$

14. Квадратриса

$$x = y \operatorname{ctg} \frac{\pi y}{2R}$$

5.2 Варианты заданий повышенной сложности

Необходимо выполнить задание из вариантов обычной сложности добавив в программу возможность экспорта исходных данных и результатов расчета в MS Excell

6 Работа №4. Многопроцессорное и многопоточное программирование

Студент выполняет задание обычной или повышенной сложности.

Необходимо написать приложение с использованием технологии WinForms реализующие вариант задания. Программа должна позволять добавлять новые сущности с использованием интерфейса и редактировать существующие. Сущности, добавленные в программу должны сохраняться между запусками приложения. Для хранения данных необходимо использовать СУБД SQLite. Необходимо предусмотреть возможность сохранения списка существующих сущностей в файл.

6.1 Варианты заданий обычной сложности

Необходимо написать программу

1. Хранения списка банковских счетов.
2. Хранения списка студентов.
3. Хранения расписания поездов.
4. Хранения списка товаров в магазине.
5. Хранения списка должников.
6. Хранения естественных астрономических объектов (звезды, планеты, галактики).
7. Хранения искусственных астрономических объектов (искусственные спутники земли, межпланетные станции).
8. Хранения списка книг в библиотеки.
9. Хранения фильмотеки.
10. Хранения результатов спортивных соревнований.

6.2 Варианты заданий повышенной сложности

Необходимо выполнить задание из вариантов обычной сложности используя паттерн MVP (или MVVM если используется WPF) и библиотеку Autofac для внедрения зависимостей. Необходимо использовать асинхронное подключение к СУБД (SQLiteAsyncConnection)