Программирование на C#

Семинар №6

Модуль №1

Тема:

Вложенные циклы

Циклы повторения решений

Контроль начальных значений

Задания преподавателя к семинару

1. Повторяем теорию, изучаем демонстрационные задачи.
2. Выполняем задания, представленные на слайдах.
3. Любой из выполненных проектов дополните блоком повтора решений.
4. Используйте в программе контроль ввода данных через Parse и TryParse. Сформулируйте отличия в механизме действия обоих методов.

Полезные материалы к семинару

1. Создание вложенных циклов https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/tutorials/branches-and-loops-local#use-loops-to-repeat-operations
2. TryParse https://learn.microsoft.com/ru-ru/dotnet/api/system.int32.tryparse

Demo 01. Вложенные циклы

NB! *Основной принцип работы вложенных циклов: внутренний цикл выполняется быстрее обрамляющего цикла.*

Вывод:

```
a = 1  b = 1 c = 1  sum = 3
a = 1  b = 1 c = 2  sum = 7
a = 1  b = 2 c = 1  sum = 11
a = 1  b = 2 c = 2  sum = 16
a = 2  b = 1 c = 1  sum = 20
a = 2  b = 1 c = 2  sum = 25
a = 2  b = 2 c = 1  sum = 30
a = 2  b = 2 c = 2  sum = 36
```

```
int a;
    int b;
    int c;
    int sum = 0;
    for (a = 1; a < 3; a++)
    {
        for (b = 1; b < 3; b++)
        {
            for (c = 1; c < 3; c++) //Самый быстрый цикл.
            {
                sum += a + b + c;
                Console.Write($"a = {a} b = {b} c = {c}
sum = {sum}\n");
            }
        }
    }
    Console.ReadLine();
```

ToDo 01: Вложенные Циклы

Измените код Demo1 таким образом, чтобы вывести сумму для каждой возможной комбинации переменных без нарастающего итога

Корректный вывод:

a = 1 b = 1 c = 1 sum = 3
a = 1 b = 1 c = 2 sum = 4
a = 1 b = 2 c = 1 sum = 4 …..

Self 01: Вложенные Циклы

Используя вложенные циклы do… while() реализуйте вывод на экран таблицу значений логической функции для всех комбинаций значений параметров `F = !(p & q) & !(p|!q):`

Demo 02. Вложенные циклы

```
double s,     //Значение суммы.
       sl;    //Очередное слагаемое (оно же результат произведения).
int k,     //Номер очередного слагаемого.
    n,     //Предельный номер слагаемого.
    i;     //Номер сомножителя.
string strin, strout;
Console.Write("Введите предел суммирования: ");
strin = Console.ReadLine();
n = int.Parse(strin);
if (n < 2) {
    Console.Write("Введено ошибочное значение");
    Console.ReadLine();
    return;
}
for (s = 0, k = 2; k <= n; k++) //Вычисляем сумму
{ for (sl = 1, i = 1; i < k; i++) //Вычисляем произведение
        sl *= Math.Sin(Math.PI * i / k);
    s += sl;
}
strout = string.Format($"При n={n} сумма={s:f3}\n");
Console.Write(strout);
Console.WriteLine();
```

Для введенного с клавиатуры натурального n (n>2) вычислить

$$s = \sum_{k=2}^{n} \prod_{i=1}^{k-1} sin\frac{i\pi}{k}$$

Контроль n с выходом из программы при ошибке

Вывод:

```
Введите предел суммирования: 20

При n=20 сумма=3,000
```

Demo 03. Циклы повтора решений

Организация цикла повтора позволяет тестировать программу с разными наборами входных параметров не выходя из текущей рабочей сессии. Используем цикл с постусловием.

```
char rep;
do
 {
 Console.Clear();
  . . . . . .  // Здесь  наша программа.
 Console.WriteLine("Для повтора вычислений нажмите клавишу Y: ");
 rep = char.Parse(Console.ReadLine());
 } while (rep == 'Y' || rep == 'y');
```

Очистка экрана

Условие продолжения цикла. Блок повтора

ToDo 02. Организуйте блок повтора в любой выполненной программе семинара.

Demo 04. Анализ нажатых клавиш

С помощью структуры ConsoleKeyInfo мы можем получать данные о клавишах, нажатых на клавиатуре.
Усовершенствуем цикл повторения решения

```
do {
// Решение задачи
    Console.WriteLine("Для выхода из программы нажмите ESC...");
}
while (Console.ReadKey().Key != ConsoleKey.Escape);
```

https://learn.microsoft.com/ru-ru/dotnet/api/system.console.readkey?view=net-6.0

https://learn.microsoft.com/ru-ru/dotnet/api/system.consolekey?view=net-6.0

https://learn.microsoft.com/ru-ru/dotnet/api/system.consolekeyinfo.key?view=net-6.0

Методы контроля ввода значений

Метод **тип.Parse(string s)**

пробует получить объект заданного типа из его строкового представления. При успешной попытке возвращает число, при неудаче – исключение FormatException.

Пример: int k = int.Parse(string s) – попытка преобразовать строку в целое число.

Метод **тип.TryParse(string s, out тип result)**

проверяет, можно ли получить число из строки. Если это возможно - возвращает **true** и полученное число out-параметром, иначе возвращает false и default(тип) через **out**-параметр. Исключение не вызывает.

```
string s = Console.ReadLine();
if (int.TryParse(s, out int i))
    Console.WriteLine($"Это целое число: {i}");
        else
            Console.WriteLine("Введенная строка вовсе не число.");
```

bool result = int.TryParse(input, out number);

Присваивание результата вызова метода **TryParse()** только к объекту типа bool!

Self. Выполнить самостоятельно

Self 02. Напечатать таблицу истинности логической функции

$$\left(\overline{A \vee B \& C}\right) \vee A ,$$

где &, $\vee$, – - знаки логических операций И, ИЛИ, НЕ.

Self 03. Распечатать все четырехзначные натуральные десятичные числа из диапазона [2000..3000], в записи которых нет двух одинаковых цифр. Подсчитать количество таких чисел.

Self 04. В выражении ((((1 ? 2) ? 3) ? 4) ? 5) ? 6 вместо каждого знака ? поставить знак одной из операций +, -, *, / так, чтобы результат вычислений был равен 35.

Self 05. Для заданного натурального числа **N** и вещественного числа **A** вычислить и вывести на экран:

$$S = \sum_{i=1}^{N} \frac{1}{A^i} \quad , \qquad P = \prod_{i=1}^{N} (A - i)$$

Предел суммирования (произведения) **N** и значение **A** ввести с клавиатуры. Стандартную функцию **pow** *не использовать*.

Self. Выполнить самостоятельно

Self 06. Вычислить и вывести на экран таблицу значений функции одной переменной

$$S(n) = \sum_{k=1}^{n} k!$$

Изменение аргумента задано в виде: **начальное значение (шаг) конечное значение** и составляет: n = 5(1)12. На каждой строке экрана вывести текущее значение аргумента и значение функции.

Self 07. Напишите программу, которая позволяет пользователю ввести с клавиатуры значение вещественной переменной x и целочисленной $1 \le n \le 20$. Определить и вывести на экран значение, вычисленное по формуле:

$$(\ln x - \frac{2}{9}) \sum_{k=1}^{n} \left(|x - k| cos \frac{\sqrt[3]{kx}}{2} \right)$$

При некорректных данных выводить строку “Wrong input” и запрашивать повторно ввод корректных значений. После вывода результатов расчёта пользователь может выйти из программы или запросить повторение работы для ввода новых данных. Вывод вещественных значений в консольное окно осуществляется с точностью до трёх знаков после десятичного разделителя.

Self. Выполнить самостоятельно

Self 08.

- Составить программу для вычисления таблицы значений функции для N значений X, изменяющихся от X_0 с шагом dX и M значений Z, меняющихся от Z_0 с шагом dZ. $N > 0, M > 0$ - целые, $X_0, Z_0, dX > 0, dZ > 0$ - вещественные
- Реализовать и отладить программу для следующих функций:

$$Xarctg\frac{X}{\sqrt{Z}} - ln\sqrt[3]{X^2 + Z} + 1$$

$$e^{\sqrt{Z}} + \sqrt[3]{X^4}\left(1 + \frac{X - {Z}/{X}}{X + {Z}/{X}}\right)|sinX|$$

- Некорректные данные и аварийные ситуации обработать. Предусмотреть цикл повторения решения
- *Дополнительно. Реализовать экранное текстовое меню, позволяющее выбрать функцию, для которой будет построена таблица значений