1. **Принципы объектно-ориентированного программирования**

Задание 1. Даны два числа. Найти их сумму, разность, произведение, а также частное от деления первого числа на второе.

Листинг программы:

namespace Task\_1

{

class Task\_1

{

public static void Main(string[] args)

{

Console.Write("Введите первое число: ");

double numOne = double.Parse(Console.ReadLine());

Console.Write("Введите второе число: ");

double numTwo = double.Parse(Console.ReadLine());

double sum = numOne + numTwo;

double sub = numOne - numTwo;

double mul = numOne \* numTwo;

double div = numOne / numTwo; Console.WriteLine($"Сумма = {sum}\nРазность = {sub}\nУмножение = {mul}\nДеление = {div:F2}\n");

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 22 21 | 34 10 264 1,83 |

Анализ результатов:

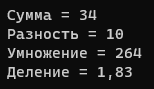


Рисунок 1.2 – Результат работы программы

Источник: собственная разработка

Задание 2. Условные операторы. Проверить истинность высказывания: «Цифры данного трехзначного числа образуют возрастающую или убывающую последовательность»

Листинг программы:

namespace Task\_2

{

class Task\_2

{

public static void Main(string[] args)

{

Console.WriteLine("Цифры данного трехзначного числа образуют возрастающую или убывающую последовательность");

Console.Write("Введите трехзначное число: ");

string num = Console.ReadLine();

if (num[0] < num[1] && num[1] < num[2])

{

Console.WriteLine($"Число {num} образует возрастающую последовательность.");

}

else

{

Console.WriteLine($"Число {num} образует убывающую последовательность.");

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 249 | Число 249 образует возрастающую последовательность. |

Анализ результатов:

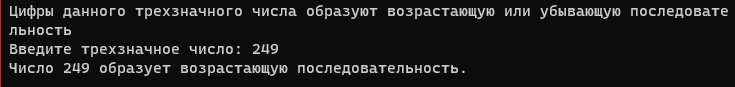


Рисунок 2.1 – Результат работы программы

Источник: собственная разработка

Задание 3. Операторы цикла. Дано целое число N (N>0). Найти значение выражения 1.1 – 1.2 + 1.3 – … (N слагаемых, знаки чередуются). Условный оператор не использовать.

Входные данные: ввести целое число N (1<=n<=20).

Выходные данные: вывести выражения 1.1 – 1.2 + 1.3 – … (N слагаемых, знаки чередуются) с точностью до 4 цифр в дробной части.

Листинг программы:

namespace Task\_3

{

class Task\_3

{

public static void Main(string[] args)

{

Console.Write("Введите целое число N (1<=n<=20): ");

int num = int.Parse(Console.ReadLine());

double number = 1.1;

int minNum = 1;

double result = 0;

for (int i = 0; i < num; i++)

{

result += number \* minNum;

number += 0.1;

minNum \*= -1;

}

Console.WriteLine($"{result:F4}");

}

}

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 | 1,2000 |

Анализ результатов:



Рисунок 3.1 – Результат работы программы

Источник: собственная разработка

Задание 4. Написать программу, которая вычисляет значение функции у:



Рисунок 4.1 – Функция

Источник: собственная разработка

Листинг программы:

namespace Task\_4

{

class Task\_4

{

public static void Main(string[] args)

{

Console.Write("Введите число: ");

double num = double.Parse(Console.ReadLine());

if (num > 6.7)

{

Console.WriteLine($"y= {4 - Math.Pow(Math.E, 4 \* num)}");

}

else

{

Console.WriteLine($"y= {Math.Log(3.5 + num)}");

}

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2,5 | 1,7917... |

Анализ результатов:



Рисунок 4.2 – Результат работы программы

Источник: собственная разработка

Задание 5. Написать программу, которая определяет: является ли треугольник с длинами сторон a, b, c равносторонним.

Листинг программы:

namespace Task\_5

{

class Task\_5

{

public static void Main(string[] args)

{

Console.WriteLine("Введите стороны треугольника:");

Console.Write("a= ");

double sideA = double.Parse(Console.ReadLine());

Console.Write("b= ");

double sideB = double.Parse(Console.ReadLine());

Console.Write("c= ");

double sideC = double.Parse(Console.ReadLine());

if (sideA == sideB && sideB == sideC)

{

Console.WriteLine($"Треугольник равносторонний");

}

else

{

Console.WriteLine($"Треугольник не равносторонний");

}

}

}

}

Таблица 5.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 3 2 | Треугольник не равносторонний |

Анализ результатов:

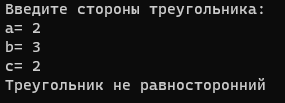


Рисунок 5.1 – Результат работы программы

Источник: собственная разработка

Задание 6. Составить программу (при решении данных задач использовать оператор switch или вложенные операторы if). Дан номер масти m (1<=m<=4) и номер достоинства карты k (6<=k<=14). Определить полное название соответствующей карты в виде «дама пик», «шестерка бубен» и т.д.

Листинг программы:

namespace Task\_6

{

class Task\_6

{

public static void Main(string[] args)

{

Console.Write("Введите номер карты: ");

int numCard = int.Parse(Console.ReadLine());

Console.Write("Введите масть карты: ");

int suitCard = int.Parse(Console.ReadLine());

switch (numCard)

{

case 6:

Console.Write("шестерка ");

break;

case 7:

Console.Write("семерка ");

break;

case 8:

Console.Write("восьмерка ");

break;

case 9:

Console.Write("девятка ");

break;

case 10:

Console.Write("десятка ");

break;

case 11:

Console.Write("валет ");

break;

case 12:

Console.Write("дама ");

break;

case 13:

Console.Write("король ");

break;

case 14:

Console.Write("туз ");

break;

default:

Console.WriteLine("нет такого номера карты");

break;

}

switch (suitCard)

{

case 1:

Console.Write("крести");

break;

case 2:

Console.Write("пик");

break;

case 3:

Console.Write("бубен");

break;

case 4:

Console.Write("чирва");

break;

default:

Console.WriteLine("нет такой масти карты");

break;

}

}

}

}

Таблица 6.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 7 4 | Семерка чирва |

Анализ результатов:

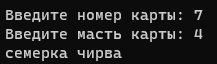


Рисунок 6.1 – Результат работы программы

Источник: собственная разработка

Задание 7. Вывести на экран (задачу решите тремя способами - используя операторы цикла while, do while и for): все целые четные числа из диапазона от А до В (A<=B), оканчивающиеся на цифру Х или У в обратном порядке;

Листинг программы:

namespace Task\_7

{

class Task\_7

{

public static void Main(string[] args)

{

Console.Write("Введите начальное значение: ");

int start = int.Parse(Console.ReadLine());

Console.Write("Введите конечное значение: ");

int end = int.Parse(Console.ReadLine());

Console.Write("Введите целое число (x) : ");

int numX = int.Parse(Console.ReadLine());

Console.Write("Введите целое число (y): ");

int numY = int.Parse(Console.ReadLine());

Console.WriteLine("Цикл for");

for (int i = end; i >= start; i--)

{

if (i % 2 == 0)

{

if (i % 10 == numX || i % 10 == numY)

{

Console.Write($"Числo: {i} ");

}

}

}

Console.WriteLine("\nЦикл while");

int j = end;

while ( j >= start)

{

if (j % 2 == 0)

{

if (j % 10 == numX || j % 10 == numY)

{

Console.Write($"Числo: {j} ");

}

}

j--;

}

Console.WriteLine("\nЦикл do while");

j = end;

do

{

if (j % 2 == 0)

{

if (j % 10 == numX || j % 10 == numY)

{

Console.Write($"Числo: {j} ");

}

}

j--;

}

while (j >= start);

}

}

}

Таблица 7.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 46 2 3 | 42 32 22 12 2 |

Анализ результатов:

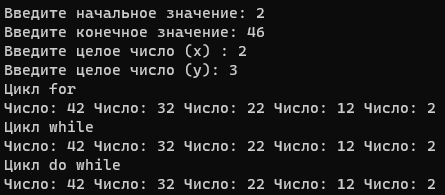


Рисунок 7.1 – Результат работы программы

Источник: собственная разработка

Задание 8. Даны два целых числа A и B (A<B). Найти произведение всех целых чисел от A до B включительно. Входные данные: ввести два целых числа A, B (1<=A, B<=10). Выходные данные: вывести произведение всех целых чисел от A до B включительно.

Листинг программы:

namespace Task\_8

{

class Task\_8

{

public static void Main(string[] args)

{

Console.Write("Введите начальное значение: ");

int start = int.Parse(Console.ReadLine());

Console.Write("Введите конечное значение: ");

int end = int.Parse(Console.ReadLine());

int result = 1;

for (int i = start; i <= end; i++)

{

result \*= i;

}

Console.WriteLine($"Произведение: {result}");

}

}

}

Таблица 8.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 6 | 720 |

Анализ результатов:

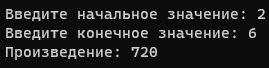


Рисунок 8.1 – Результат работы программы

Источник: собственная разработка

Задание 9. Табулирование функций. Составить программу вычисления значений функции F(x) на отрезке [A, B] в точках xi =x+H, где H=(B-A)/M, M – заданное целое число.





Рисунок 9.1 – Функция

Источник: собственная разработка

Листинг программы:

namespace Task\_9

{

class Task\_9

{

public static void Main(string[] args)

{

const int numM = 20;

const double start = Math.PI / 4;

const double end = 4 / Math.PI;

double numH = (end - start) / numM;

for (double i = start; i <= end; i += numH)

{

Console.WriteLine($"F({i})= {Math.Cos(1 / i)}");

}

}

}

}

Таблица 9.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 20 π/4 4/π | F(0,7853…) = 0,2931… |

Анализ результатов:

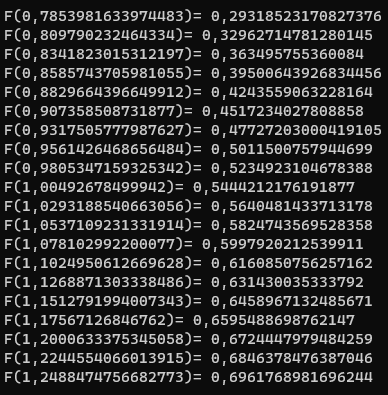


Рисунок 9.2 – Результат работы программы

Источник: собственная разработка

Задание 10. Написать программу, находящую все двухзначные числа, сумма квадратов цифр которых кратна 13.

Листинг программы:

namespace Task\_10

{

class Task\_10

{

public static void Main(string[] args)

{

Console.WriteLine("Числа сумма квадратов цифр которых кратна 13:");

for (int i = 10; i < 100; i++)

{

if ((Math.Pow(i / 10, 2) + Math.Pow(i % 10, 2)) % 13 == 0)

Console.Write($"{i} ");

}

}

}

}

Таблица 10.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 15 18 23 32 46 47 51 64 69 74 79 81 96 97 |

Анализ результатов:



Рисунок 10.1 – Результат работы программы

Источник: собственная разработка