**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ**

***Институт Принтмедиа и информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 7**

**Дисциплина:** Введение в программирование.

**Выполнил(а):**

**студент(ка) группы 191-726**

Синельникова К.Т.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил:** асс. Кононенко К.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2019**

Оглавление

**[Теория](#_Toc22550572)** [3](#_Toc22550572)

[**Задания** 7](#_Toc22550583)

[**Блок-схемы** 8](#_Toc22550584)

[**Код программы** 12](#_Toc22550585)

[**Результат программы** 16](#_Toc22550586)

# 

# **Теория**

## **Логические операторы**

Следующие операторы выполняют логические операции с использованием логических операндов:

* Унарный ! (логическое отрицание) оператор.
* Бинарные & (логическое И), | (логическое ИЛИ), а также ^ (логическое исключающее ИЛИ) операторы. Эти операторы всегда обрабатывают оба операнда.
* Бинарные && (условное логическое И) и || (условное логическое ИЛИ) операторы. Эти операторы вычисляют правый операнд, только если это необходимо.

Для операндов целочисленных типов операторы &, | и ^ выполняют побитовые логические операции.

## Оператор логического отрицания !

Унарный префиксный оператор ! выполняет логическое отрицание операнда, возвращая true, если операнд имеет значение false, и false, если операнд имеет значение true.

## Оператор логического И &

Оператор & вычисляет логическое И для всех своих операндов. Результат операции x & y принимает значение true, если оба оператора x и y имеют значение true. В противном случае результат будет false.

Оператор & вычисляет оба операнда, даже если левый операнд имеет значение false и результат должен принять значение false, независимо от значения правого операнда.

В следующем примере правый операнд оператора & является вызовом метода, который выполняется независимо от значения левого операнда:

Условный оператор логического И && также вычисляет логическое И для своих операндов, но не вычисляет правый операнд, если левый операнд имеет значение false.

Для операндов целочисленного типа оператор & вычисляет побитовое логическое И своих операндов. Унарный оператор & является оператором AddressOf.

## Оператор логического исключения ИЛИ ^

Оператор ^ вычисляет логическое исключение ИЛИ для всех своих операндов, возвращая true для x ^ y, если x имеет значение true и y имеет значение false или x имеет значение false и y имеет значение true. В противном случае результат будет false. То есть для операндов bool оператор ^ возвращает тот же результат, что и оператор неравенства !=.

Для операндов целочисленного типа оператор ^ вычисляет побитовое логическое исключающее ИЛИ своих операндов.

## Оператор логического ИЛИ |

Оператор | вычисляет логическое ИЛИ для всех своих операндов. Результат операции x | y принимает значение true, если хотя бы один из операторов x или y имеет значение true. В противном случае результат будет false.

Оператор | вычисляет оба операнда, даже если левый операнд имеет значение true и результат должен принять значение true, независимо от значения правого операнда.

В следующем примере правый операнд оператора | является вызовом метода, который выполняется независимо от значения левого операнда:

Условный оператор логического ИЛИ || также вычисляет логическое ИЛИ для своих операндов, но не вычисляет правый операнд, если левый операнд имеет значение true.

Для операндов целочисленного типа оператор | вычисляет побитовое логическое ИЛИ своих операндов.

## Условный оператор логического И &&

Условный оператор логического И && (оператор короткого замыкания) вычисляет логическое И для своих операндов. Результат операции x && y принимает значение true, если оба оператора x и y имеют значение true. В противном случае результат будет false. Если x имеет значение false, y не вычисляется.

В следующем примере правый операнд оператора && является вызовом метода, который не выполняется, если левый операнд имеет значение false:

Оператор логического И & также вычисляет логическое И для своих операндов, но он всегда вычисляет оба операнда.

## Условный оператор логического ИЛИ ||

Условный оператор логического ИЛИ || (оператор короткого замыкания) вычисляет логическое ИЛИ для своих операндов. Результат операции x || y принимает значение true, если хотя бы один из операторов x или y имеет значение true. В противном случае результат будет false. Если x имеет значение true, y не вычисляется.

В следующем примере правый операнд оператора || является вызовом метода, который не выполняется, если левый операнд имеет значение true:

Оператор логического ИЛИ | также вычисляет логическое ИЛИ для своих операндов, но всегда вычисляет оба операнда.

## Составное присваивание

Для бинарного оператора op выражение составного присваивания в форме за исключением того, что x вычисляется только один раз.

Операторы &, | и ^ поддерживают составное присваивание. Условные логические операторы && и || не поддерживают составное присваивание.

## Приоритет операторов

В следующем списке перечислены логические операторы в порядке убывания приоритета:

* Оператор логического отрицания !
* Оператор логического И &
* Оператор логического исключающего ИЛИ ^
* Оператор логического ИЛИ |
* Условный оператор логического И &&
* Условный оператор логического ИЛИ ||

Порядок вычисления, определяемый приоритетом операторов, можно изменить с помощью скобок (()).

# **Задания**

1. Даны два целых числа: A, B. Проверить истинность высказывания: «Справедливы неравенства A > 2 и B ≤ 3»

2. Даны три целых числа: A, B, C. Проверить истинность высказывания: «Справедливо двойное неравенство A < B < C».

3. Дано целое положительное число. Проверить истинность высказывания: «Данное число является четным двузначным».

4. Дано трехзначное число. Проверить истинность высказывания: «Цифры данного числа образуют возрастающую или убывающую последовательность».

5. Дано четырехзначное число. Проверить истинность высказывания: «Данное число читается одинаково слева направо и справа налево».

6. Даны целые числа a, b, c, являющиеся сторонами некоторого треугольника. Проверить истинность высказывания: «Треугольник со сторонами a, b, c является прямоугольным».

7. Даны целые числа a, b, c. Проверить истинность высказывания: «Существует треугольник со сторонами a, b, c».

**Блок-схемы** 

Рисунок 1 — Блок-схема к заданию 1



Рисунок 2 — Блок-схема к заданию 2



Рисунок 3 — Блок-схема к заданию 3



Рисунок 4 — Блок-схема к заданию 4



Рисунок 5 — Блок-схема к заданию 5



Рисунок 6 — Блок-схема к заданию 6



Рисунок 7 — Блок-схема к заданию 7

# **Код программы**

|  |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. namespace Код\_Лабораторной\_7\_\_1\_7\_\_шарп 7. { 8. class Program 9. { 10. static void Main(string[] args) 11. { 12. int A, B; 13. Console.Write("Введите A: "); 14. A = int.Parse(Console.ReadLine()); 15. Console.Write("Введите B: "); 16. B = int.Parse(Console.ReadLine()); 17. Console.Write("Справедливы неравенства A > 2 и B <= 3: " + ((A > 2) && (B <= 3))); 18. Console.ReadKey(); 19. } 20. } 21. } |

Листинг 1 — Задание 1 (Справедливы неравенства A > 2 и B ≤ 3)

|  |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. namespace Код\_Лабораторной\_7\_\_1\_7\_\_шарп 7. { 8. class Program 9. { 10. static void Main(string[] args) 11. { 12. int A, B, C; 13. Console.Write("Введите A: "); 14. A = int.Parse(Console.ReadLine()); 15. Console.Write("Введите B: "); 16. B = int.Parse(Console.ReadLine()); 17. Console.Write("Введите C: "); 18. C = int.Parse(Console.ReadLine()); 19. Console.Write("Справедливо двойное неравенство A < B < C: " + ((A < B) || (B < C))); 20. Console.ReadKey(); 21. } 22. } 23. } |

Листинг 2 — Задание 2 (Справедливо двойное неравенство A < B < C)

|  |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. namespace Код\_Лабораторной\_7\_\_1\_7\_\_шарп 7. { 8. class Program 9. { 10. static void Main(string[] args) 11. { 12. int A; 13. Console.Write("Введите число: "); 14. A = int.Parse(Console.ReadLine()); 15. Console.Write("Данное число является четным двузначным: " + (((A < 100) && (A >= 10)) && (A % 2 == 0))); 16. Console.ReadKey(); 17. } 18. } 19. } |

Листинг 3 — Задание 3 (Данное число является четным двузначным)

|  |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. namespace Код\_Лабораторной\_7\_\_1\_7\_\_шарп 7. { 8. class Program 9. { 10. static void Main(string[] args) 11. { 12. int Q, A, B, C; 13. Console.Write("Введите трехзначное число: "); 14. Q = int.Parse(Console.ReadLine()); 15. A = Q / 100; 16. B = (Q % 100) / 10; 17. C = Q % 10; 18. Console.Write("Цифры данного числа образуют возрастающую или убывающую последовательность: " + ((A < B) && (B < C) || (A > B) && (B > C))); 19. Console.ReadKey(); 20. } 21. } 22. } |

Листинг 4 — Задание 4 (Цифры данного числа образуют возрастающую или убывающую последовательность)

Листинг 5 — Задание 5 (Данное число читается одинаково слева направо и справа налево)

|  |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. namespace Код\_Лабораторной\_7\_\_1\_7\_\_шарп 7. { 8. class Program 9. { 10. static void Main(string[] args) 11. { 12. int Q, A, B, C, D; 13. Console.Write("Введите четырехзначное число: "); 14. Q = int.Parse(Console.ReadLine()); 15. A = Q / 1000; 16. B = (Q % 1000) / 100; 17. C = (Q % 100) / 10; 18. D = Q % 10; 19. Console.Write("«Данное число читается одинаково слева направо и справа налево: " + ((A == D) && (B == C))); 20. Console.ReadKey(); 21. } 22. } 23. } |

# 

Листинг 6 — Задание 6 (Треугольник со сторонами a, b, c является прямоугольным)

|  |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. namespace Код\_Лабораторной\_7\_\_1\_7\_\_шарп 7. { 8. class Program 9. { 10. static void Main(string[] args) 11. { 12. int a, b, c; 13. Console.Write("Введите a: "); 14. a = int.Parse(Console.ReadLine()); 15. Console.Write("Введите b: "); 16. b = int.Parse(Console.ReadLine()); 17. Console.Write("Введите c: "); 18. c = int.Parse(Console.ReadLine()); 19. Console.Write("Треугольник со сторонами a, b, c является прямоугольным: " + ((a \* a + b \* b) == (c \* c) || (a \* a + c \* c) == (b \* b) || (c \* c + b \* b) == (a \* a))); 20. Console.ReadKey(); 21. } 22. } 23. } |

Листинг 7 — Задание 7 (Существует треугольник со сторонами a, b, c)

|  |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. namespace Код\_Лабораторной\_7\_\_1\_7\_\_шарп 7. { 8. class Program 9. { 10. static void Main(string[] args) 11. { 12. int a, b, c; 13. Console.Write("Введите a: "); 14. a = int.Parse(Console.ReadLine()); 15. Console.Write("Введите b: "); 16. b = int.Parse(Console.ReadLine()); 17. Console.Write("Введите c: "); 18. c = int.Parse(Console.ReadLine()); 19. Console.Write("Существует треугольник со сторонами a, b, c: " + ((a + b > c) && (a + c > b) && (c + b > a))); 20. Console.ReadKey();} 21. } 22. } 23. } |

# **Результат программы**

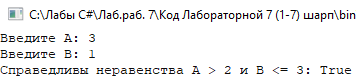


Рисунок 8 — Результат выполнения программы 1

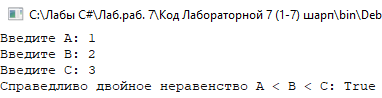


Рисунок 9 — Результат выполнения программы 2

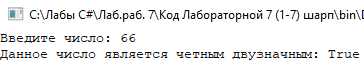


Рисунок 10 — Результат выполнения программы 3

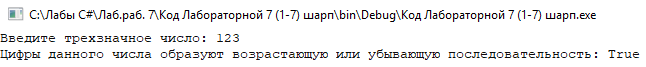


Рисунок 11 — Результат выполнения программы 4

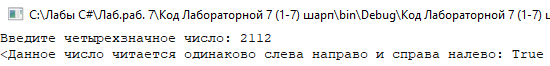


Рисунок 12 — Результат выполнения программы 5

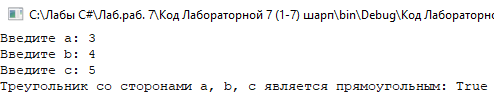


Рисунок 13 — Результат выполнения программы 6

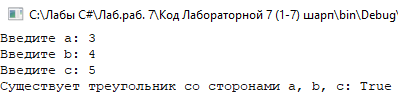


Рисунок 14 — Результат выполнения программы 7