**Тема:** Разработка и тестирование кода программы на языке С#. Разветвляющиеся алгоритмы.

**Цель работы:**

* Формирование навыков разработки приложения на языке С#.
* Формирование навыков работы в среде Visual Studio.Net
* Изучение операторов языка С# для описания разветвляющихся алгоритмов.

**Оборудование:** Ноутбук MSI Modern, программное обеспечение: ОС, Visual Studio.Net

**Ответы на контрольные вопросы:**

1. Какие операторы С# используются для описания разветвляющихся алгоритмов?

В C# для описания разветвляющихся алгоритмов используются следующие операторы:

* if: Оператор `if` используется для выполнения кода, если определенное условие истинно.
* else: Оператор `else` используется вместе с `if`, чтобы выполнить код в случае, если условие ложно.
* switch: Оператор `switch` может использоваться вместо нескольких операторов `if` для проверки одной переменной на соответствие нескольким значениям.

1. Записать общий вид условного оператора? Как он выполняется на языке c#

Общий вид условного оператора в C# выглядит следующим образом:

if (условие)

{

// Код выполняется, если условие истинно

}

else

{

// Код выполняется, если условие ложно

}

В данном коде условие - это выражение, которое может быть либо истинным (`true`), либо ложным (`false`).

Выполнение условного оператора происходит следующим образом:

1. Сначала оценивается условие.

2. Если условие истинно (`true`), то выполняется код внутри блока `if`. После этого выполнение программы продолжается с первого оператора, расположенного после блока `else` (если он есть).

3. Если условие ложно (`false`), то выполняется код внутри блока `else`. Если блока `else` нет, то выполнение программы продолжается с первого оператора, расположенного после блока `if`.

4. Если в операторе нет блока `else`, и условие ложно, то выполнение просто пропускает блок `if` и продолжается со следующего за ним оператора.

1. Записать общий вид оператора выбора? Как он выполняется?

Оператор выбора в C# представлен оператором `switch`. Общий вид данного оператора выглядит следующим образом:

switch (expression)

{

case constant1:

// код выполняется, если expression равно constant1

break;

case constant2:

// код выполняется, если expression равно constant2

break;

// можно добавить столько case, сколько нужно

default:

// код выполняется, если expression не равно ни одному из предыдущих значений

break;

}

Работа оператора `switch` происходит следующим образом:

1. Сначала вычисляется `expression`.

2. Затем это значение сравнивается с каждым из значений `constant` в операторах `case`, начиная с первого. Если найдено соответствие, выполняется код этого блока `case`, а затем выполнение программы пропускает оставшиеся блоки `case` и продолжается после блока `switch`.

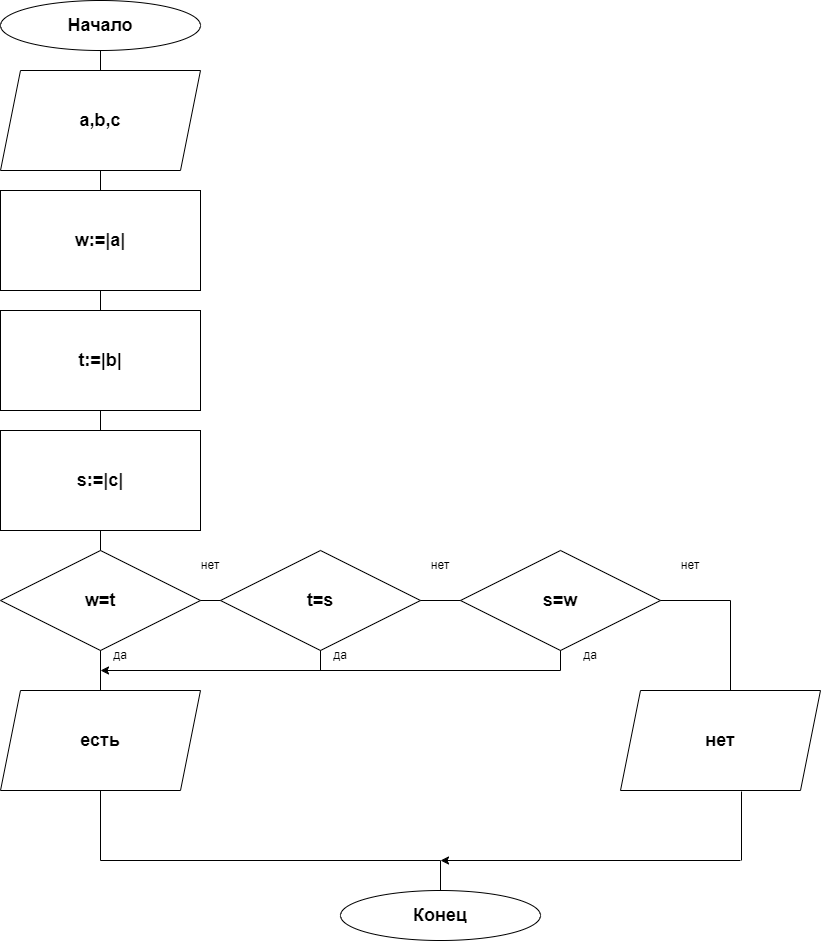
3. Если соответствие не найдено, выполняется код в блоке `default`, если он присутствует. Если блока `default` нет, и ни одно из значений `case` не соответствует `expression`, то выполнение программы просто пропускает блок `switch` и продолжается со следующего за ним оператора.

**Ход работы:**

1. Разработать алгоритм задачи и представить его в виде схемы программы
2. В среде Visual Studio.Net создать новый проект и набрать файл с программой
3. Выполнить тестирование и отладку программу
4. Результаты представить в виде отчета
5. Сделать вывод о проделанной работе

**Задание на лабораторную работу:** Определить, имеется ли среди чисел A,B,C хотя бы одна пара взаимно противоположных чисел

**Схема программы**



**Листинг с исходным кодом**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace противоположные\_числа

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите число а");

int a = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите число b");

int b = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите число c");

int c = Convert.ToInt32(Console.ReadLine());

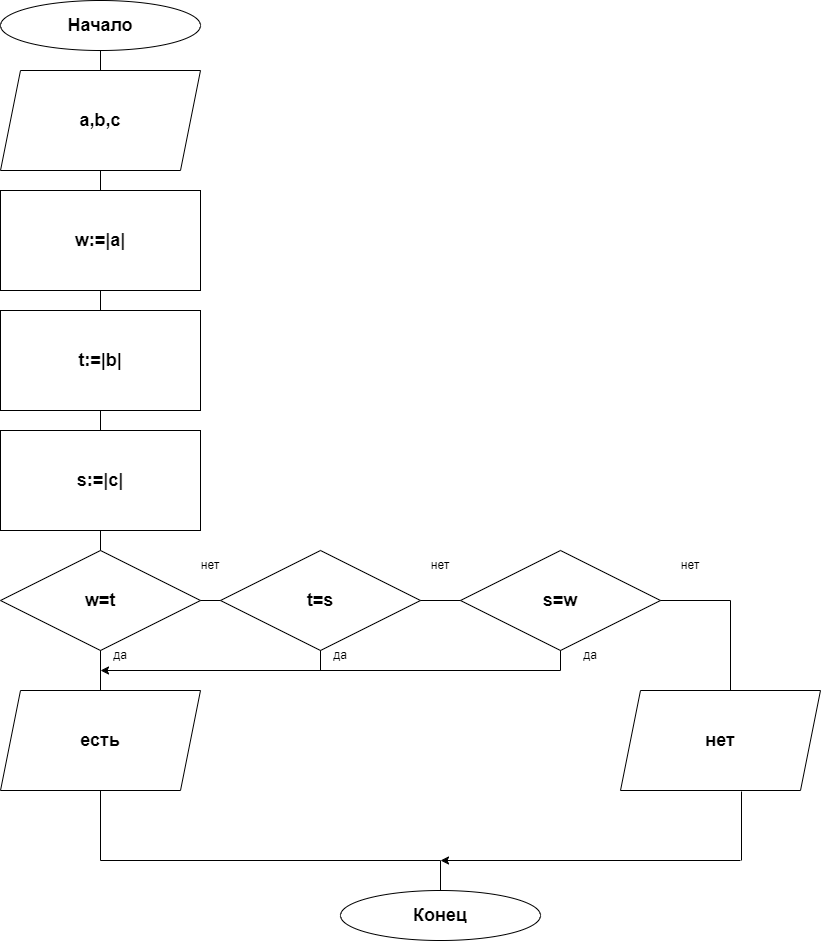
int w = Math.Abs(a);

int t = Math.Abs(b);

int s = Math.Abs(c);

if (w == t || w == s || s == t)

{



Console.WriteLine("Имеются взаимно противоположные числа");

}

else { Console.WriteLine("Взаимно противоположных чисел нет"); }

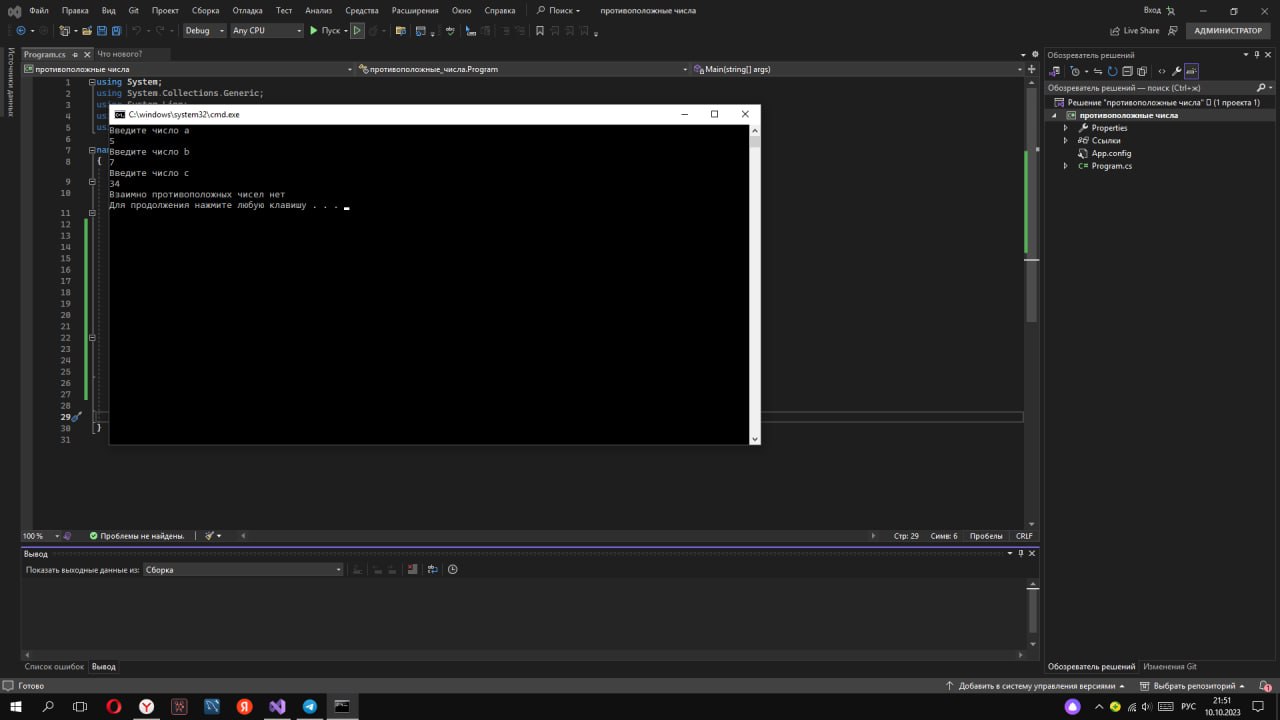
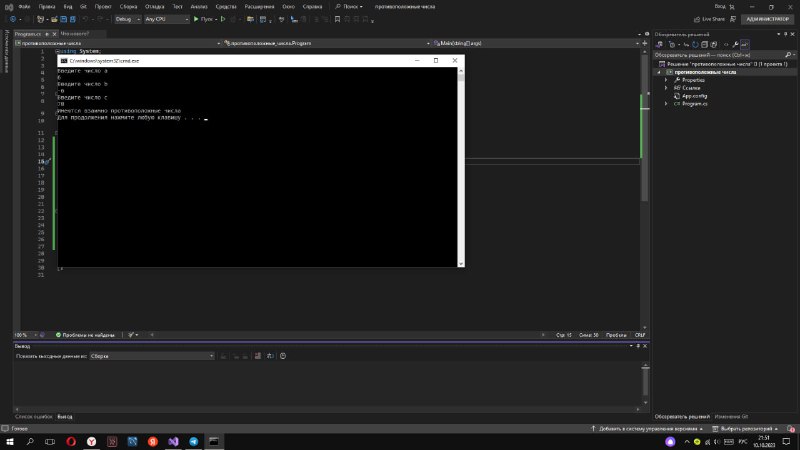
Console.ReadKey();

}

}

}

**Результаты тестирования**



**Вывод по работе:** В ходе работы были изучены и применены разветвляющиеся алгоритмы в контексте программирования на языке C#. Эти алгоритмы, использующие условные операторы и циклы, являются фундаментальными элементами большинства программ. Разработка и тестирование кода на языке C# с использованием разветвляющихся алгоритмов помогает создавать эффективные и надежные приложения.