Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

|  |  |
| --- | --- |
| Факультет (институт) | *Информационных технологий и компьютерных систем* |
|  |  |
| Кафедра | *Прикладная математика и фундаментальная информатика* |
|  |  |

**Расчетно-графическая работа**

|  |  |
| --- | --- |
| по дисциплине | ***Алгоритмизация и программирование*** |
|  |  |
| на тему | Разработка программы |

Пояснительная записка

|  |  |
| --- | --- |
| **Шифр проекта** | 020-РГР-02.03.02-№ 9-ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Есакова Кирилла Николаевича | | | | | |
|  |  |  |  | | фамилия, имя, отчество полностью | | | | | |
|  |  |  | Курс | *1* |  | Группа | | ФИТ-232 | | |
|  |  |  |  |  |  | |  |  | |  |
|  | | | **Направление (специальность)** | | | | | ***02.03.02*** | | |
|  | | | *Фундаментальная информатика и информационные технологии* | | | | | | | |
|  |  |  | код, наименование | | | | | | | |
|  |  |  | Руководитель | | ***ст. преподаватель*** | | | | | |
|  |  |  | ученая степень, звание | | | | | |
|  |  |  | ***Федотова И.В.*** | | | | | | | |
|  |  |  | фамилия, инициалы | | | | | | | |
|  |  |  | Выполнил | | 17.05.2024 | | | | | |
|  |  |  | дата, подпись студента | | | | | |
|  |  |  | **Работа защищена с количеством баллов** | | | | | | | |
|  |  |  |  | | | | | |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | дата, подпись руководителя |  |  |  |

Омск 2024

Содержание

[Введение 3](#_Toc166920750)

[Теоретическая часть 4](#_Toc166920751)

[Разработка кода 5](#_Toc166920752)

[Результаты программы 8](#_Toc166920753)

[Заключение 10](#_Toc166920754)

[Список используемой литературы 11](#_Toc166920755)

# **Введение**

В ходе программы нам необходимо написать программу, с помощью которой можно получить результат выражения, записанного в польской нотации, а также проверить, правильно ли расставлены скобки в строке. Для этого мы будем использовать такую коллекцию, как с Stack.

# **Теоретическая часть**

Stack – коллекция переменного размера экземпляров одинакового заданного типа, обслуживаемую по принципу "последним пришел – первым вышел" (LIFO).

Для получения количества элементов стека применяется свойство Count.

В классе Stack можно выделить следующие методы:

* Clear: очищает стек
* Contains: проверяет наличие в стеке элемента и возвращает true при его наличии
* Push: добавляет элемент в стек в верхушку стека
* Pop: извлекает и возвращает первый элемент из стека
* Peek: просто возвращает первый элемент из стека без его удаления

Постфиксная (или обратная польская) запись арифметического выражения – это способ записи выражений, в котором знак операции записывается не между операндами, а после операндов.

Числа добавляются в стек, а результат операций применяется к двум последним числам в стеке, которые из стека удаляются. Затем результат кладется в стек.

То есть выражение «2 3 + 4 \*» в постфиксной записи может быть вычислено так:

* Число 2 кладется в стек.
* Число 3 кладется в стек.
* Из стека извлекаются числа 3 и 2, к ним применяется операция сложения, результат (число 5) кладется в стек.
* Число 4 кладется в стек.
* Из стека удаляются числа 5 и 4, результат их умножения 20 кладется в стек.

В итоге в стеке оказывается одно число – 20, которое и есть результат вычисления выражения.

# **Разработка кода**

class Program

{

static void Main()

{

Menu menu = new Menu();

menu.Run();

}

}

public class Menu

{

public void Run()

{

bool exit = false;

while (!exit)

{

Console.WriteLine(

"Выберите действие: \n" +

"1) Данные о студенте\n" +

"2) Польская запись\n" +

"3) Правильная последовательность скобок\n" +

"4) Выход из программы\n"

);

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

Console.ReadLine();

switch (choice)

{

case 1:

Console.Clear();

Console.WriteLine("Есаков Кирилл Николаевич, ФИТиКС, ФИТ-232/2, 1 курс");

Console.WriteLine("Нажмите Enter, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

break;

case 2:

Console.Clear();

Console.WriteLine("Введите выражение в обратной польской нотации:");

string str1 = Console.ReadLine();

try

{

double result = PolishEntry(str1);

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

}

catch (DivideByZeroException ex)

{

Console.WriteLine(ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("Запись введена неверно!");

}

Console.WriteLine();

Console.WriteLine("Нажмите Enter, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

break;

case 3:

Console.Clear();

Console.WriteLine("Введите строку со скобками:");

string str = Console.ReadLine();

bool isCorrect = Scaples(str);

Console.WriteLine();

if (isCorrect)

{

Console.WriteLine("Скобки расставлены верно!");

}

else

{

Console.WriteLine("Неверно расставленны скобки!");

}

Console.WriteLine("Нажмите Enter, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

break;

case 4:

Console.Clear();

exit = true;

break;

default:

Console.WriteLine("Некорректный ввод!");

Console.ReadKey();

Console.Clear();

break;

}

}

}

static double PolishEntry(string str)

{

var notation = str.Split();

string[] operations = new string[] { "+", "-", "\*", "/" };

Stack<double> stack = new Stack<double>();

foreach (string elem in notation)

{

bool isDouble = double.TryParse(elem, out double number);

if (isDouble)

{

stack.Push(number);

}

else if (Array.IndexOf(operations, elem) != -1)

{

if (stack.Count < 2)

{

throw new Exception("Запись введена неверно!");

}

double n1 = stack.Pop();

double n2 = stack.Pop();

if (elem == "+")

{

stack.Push(n2 + n1);

}

else if (elem == "-")

{

stack.Push(n2 - n1);

}

else if (elem == "\*")

{

stack.Push(n2 \* n1);

}

else if (elem == "/")

{

if (n1 == 0)

{

throw new DivideByZeroException("Нельзя делить на ноль!");

}

else

{

stack.Push(n2 / n1);

}

}

}

}

if (stack.Count != 1)

{

throw new Exception("Запись введена неверно!");

}

return stack.Peek();

}

static bool Scaples(string str)

{

char[] parens = new char[] { '(', ')', '[', ']', '{', '}' };

Stack<char> stack = new Stack<char>();

foreach (char c in str)

{

int index = Array.IndexOf(parens, c);

if (c == '(' || c == '[' || c == '{')

{

stack.Push(c);

}

else if (index != -1)

{

if (stack.Count > 0 && stack.Peek() == parens[index - 1])

{

stack.Pop();

}

else

{

stack.Push(c);

}

}

}

return stack.Count == 0;

}

}

# **Результаты программы**

Рассмотрим, как работает программа.

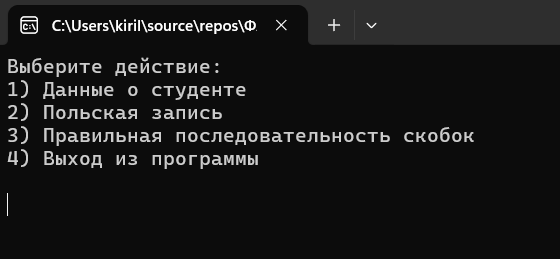


Рисунок 1 – Меню программы

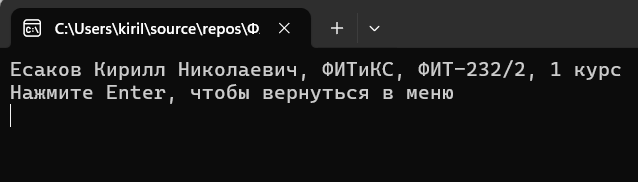


Рисунок 2 – Пункт "Данные о студенте"

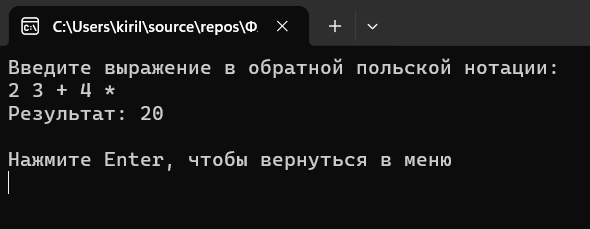


Рисунок 3 ­– Пункт "Польская запись"

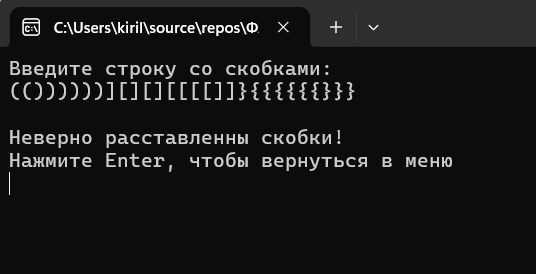


Рисунок 4 – Пункт "Правильная последовательность скобок"

# **Заключение**

В конечном итоге была разработана программа с использованием коллекции Stack, с помощью которой мы вычислили значение выражения, записанного в польской записи, а также проверили корректность последовательности скобок. Эта программа помогла закрепить знания, полученные во время изучения языка программирования C#.

# **Список используемой литературы**

1. Metanit, <https://metanit.com/sharp/tutorial/4.8.php>, (дата обращения: 17.05.2024)
2. Microsoft Learn, <https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.stack-1?view=net-8.0>, (дата обращения: 17.05.2024)
3. Фоксфорд, <https://foxford.ru/wiki/informatika/obratnaya-polskaya-notatsiya?utm_referrer=https%3A%2F%2Fyandex.ru%2F>, (дата обращения 17.05.2024)