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 Программная реализация задач

Пояснительная записка

Шифр проекта 020–РГР–02.03.02–№ 3 – ПЗ

Студента Белогривцева Андрея Дмитриевича
фамилия, имя, отчество полностью

Курс 1 Группа МО–241

Направление (специальность) ***02.03.03***
Математическое обеспечение и администрирование информационных систем
код, наименование

Руководитель ***ст. преподаватель***
ученая степень, звание

Федотова И.В.
фамилия, инициалы

Выполнил 30.12.2024
дата, подпись студента

Работа защищена с количеством баллов 12 (двенадцать)

11.01.2025
дата, подпись руководителя

Омск 2024

Содержание

Введение

В настоящее время для решения практических задач широко используются различные языки программирования. Их существует великое множество (Python, C, C++, PHP и д.р). Теоретическое решение практических задач довольно сложное в математическом отношении и общего алгоритма решения, конечно, не существует. Необходимо найти определённые тенденции и закономерности, проанализировать основы механизма работы той или иной задачи.

В тех случаях, когда теоретическое представление задачи доставляет определённую сложность, использование экспериментального и аналитического метода решения становится нерациональным и проблемным. Таким образом возникает необходимость в использовании инструмента, который позволил бы наглядно представить и решить задачу. Конкретно в данной работе будет рассмотрено использование языка программирования C#.

Постановка задачи «Интенсификация производства»

Перед коллективом предприятия “Ни шагу назад” была поставлена задача наращивать каждый день производство продукции на 1.

Требуется определить, какой суммарный объем продукции будет выпущен предприятием за заданный период, если в первый день периода предприятие выпускало P единиц продукции.

Примечания:

Период задается в виде двух календарных дат;

Длительность периода лежит в диапазоне от 1 до 60000;

Високосные годы учитываются по упрощенному правилу: високосным считается год, делящийся нацело на 4;

День начала периода и день его окончания учитываются при подсчете суммарного объема продукции и длительности периода;

Все даты заданы корректно.

Входной файл содержит:

В первой строке – дата начала периода в формате ДД.ММ.ГГГГ;

Во второй строке – дата окончания периода в формате ДД.ММ.ГГГГ;

В третьей строке целое число – начальный выпуск продукции **P** ($0 \le \mathbf{P} \le 5000$).

Выходной файл должен содержать суммарный объем продукции.

Ход решения задачи «Интенсификация производства»

```
using System;
class App
{
   public static void Main()
   {
      Console.WriteLine("Введите дату начала производства:");
      string date1 = Console.ReadLine();
      Console.WriteLine("Введите дату конца производства:");
      string date2 = Console.ReadLine();
      Console.WriteLine("Введите начальную производственную единицу:");
      int workNumber = Convert.ToInt32(Console.ReadLine());
      int date1Day = Convert.ToInt32(date1.Substring(0, 2));
      int date1Month = Convert.ToInt32(date1.Substring(3, 2));
      int date1Year = Convert.ToInt32(date1.Substring(6, 4));
      int date2Day = Convert.ToInt32(date2.Substring(0, 2));
      int date2Month = Convert.ToInt32(date2.Substring(3, 2));
      int date2Year = Convert.ToInt32(date2.Substring(6, 4));
      DateTime trueDate1 = new DateTime(date1Year, date1Month, date1Day);
      DateTime trueDate2 = new DateTime(date2Year, date2Month, date2Day);
      TimeSpan DaysIn = trueDate2 - trueDate1;
      int DaysInWork = DaysIn.Days;
      int result = 0;
      for (int i = 0; i <= DaysInWork; i++)
      {
         result += workNumber;
         workNumber++;
      }
      Console.WriteLine($"Результат: {result}");
   }
}
```

Ввод и вывод данных производится через консоль. Пользователь вводит две переменные (date1, date2), которые представляют из себя две даты, а также третью переменную – начальную производственную единицу. У первых двух переменных (дат) мы выделяем год, месяц и день и записываем их в отдельные переменные (date1Day, date1Month и т.д). Получаем шесть переменных из которых формируются две других переменных типа DateTime, в которых содержатся условные «даты».

Далее создаём переменную типа TimeSpan, которая является разницей между нашими датами и из неё мы отдельно выделим количество дней. Затем

превращаем её в переменную типа int. Инициализируем переменную result, в которой будет содержаться конечный ответ. Теперь мы можем посчитать количество произведенной продукции: создаём цикл for в котором мы будем суммировать продукцию за каждый день и в каждый последующий день увеличивать производственную единицу на 1. Полученная переменная result будет являться ответом на задачу.

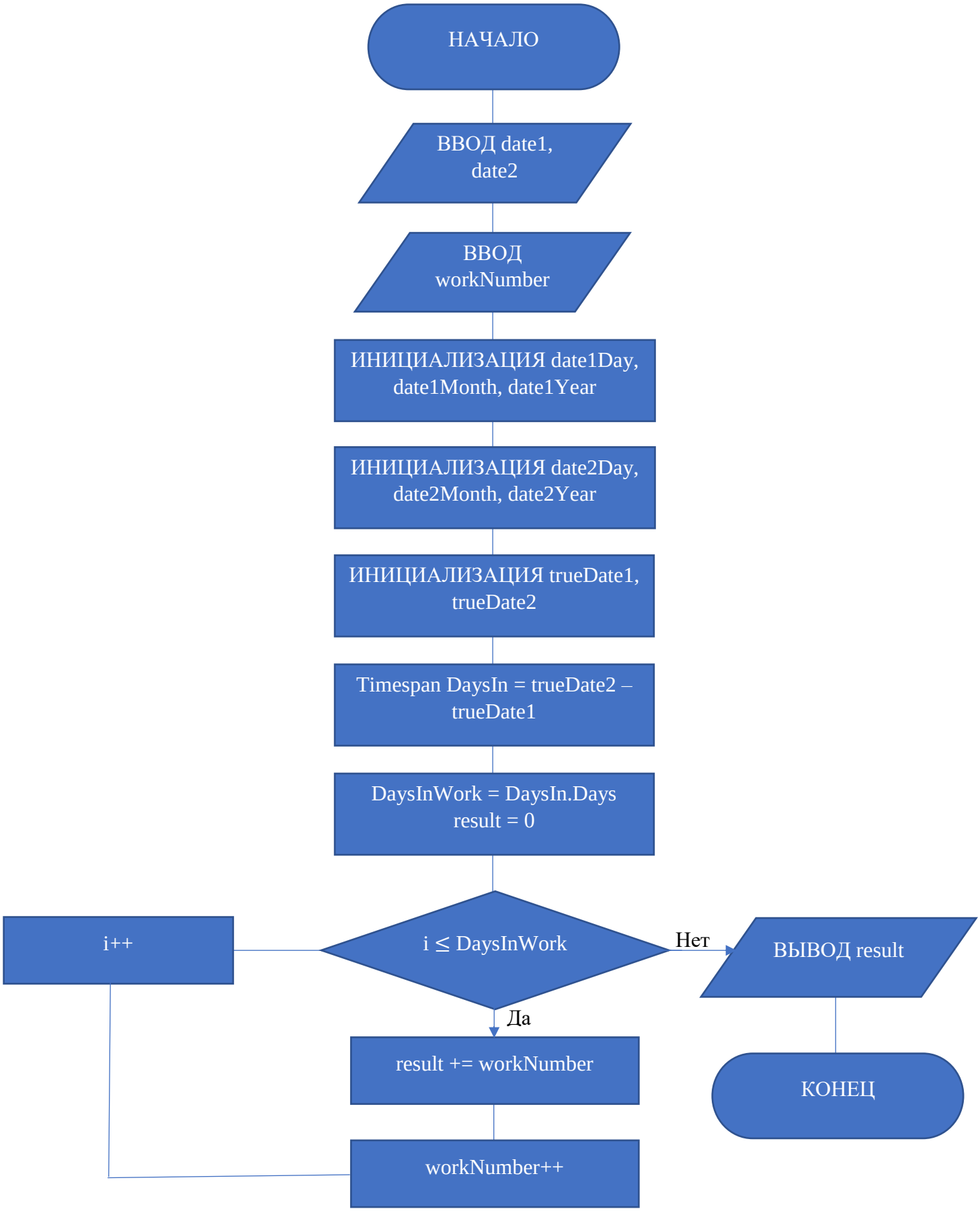

Рисунок 1 – Блок-схема алгоритма задачи «Интенсификация производства»

Исходные данные 1:

```
12.05.2002
09.04.2004
10
```

Результат работы 1:

```
Введите дату начала производства:
12.05.2002
Введите дату конца производства:
09.04.2004
Введите начальную производственную единицу:
10
Результат: 250941
```

Исходные данные 2:

```
01.01.1901
09.04.2065
5000
```

Результат работы 2:

```
Введите дату начала производства:
01.01.1901
Введите дату конца производства:
09.04.2065
Введите начальную производственную единицу:
5000
Результат: 2099970000
```

Рисунок 2 – Пример работы программы задачи «Интенсификация производства»

Постановка задачи «Зельеварение»

Одним из нелюбимых предметов Невилла Долгопупса, товарища Гарри Поттера с факультета Гриффиндор школы чародейства и волшебства “Хогвартс”, было зельеварение.

Чтобы помочь Невиллу в совершенствовании навыков зельеварения, Гарри придумал зельеварочный комбайн, изготавливающий зелья с помощью определенных заклинаний. Каждое заклинание представляет собой одно слово, формируемое в зависимости от последовательности и методов приготовления зелья.

Последовательность приготовления зелья описывается в виде набора действий, каждое из которых указывает на метод обработки заданного списка ингредиентов:

Смешивание описывается MIX < ингредиент1 ингредиент2, …>;

Растворение в воде описывается WATER < ингредиент1 ингредиент2, …>;

Измельчение описывается DUST < ингредиент1 ингредиент2, …>;

Обжиг описывается FIRE < ингредиент1 ингредиент2, …>.

При этом в качестве любого ингредиента может выступать либо некоторое вещество, задаваемое строковой константой, либо результат выполнения любого из предыдущих действий, задаваемый с помощью номера действия. Название действия и названия ингредиентов разделяются пробелами. В действии участвует как минимум один ингредиент.

Каждое действие переводится в слово по следующему правилу:

Смешивание задается в формируемом заклинании словом MX<список ингредиентов>XM;

Растворение в воде – слово WT<список ингредиентов>TW;

Измельчение – слово DT<список ингредиентов>TD;

Обжиг – слово FR<список ингредиентов>RF.

Где <список ингредиентов> – единое слово, сформированное путем сложения названий ингредиентов или слов, описывающих предыдущие действия.

Последнее действие явно или неявно использует результаты выполнения всех предыдущих действий и является основой для заклинания.

Помогите Невиллу по заданной последовательности действий сформировать заклинание.

Примечание:

При формировании заклинания учитывается регистр названий действий и ингредиентов;

Порядок названий ингредиентов в заклинании должен соответствовать их порядку в действии, т.е. для действия “DUST root tooth” в заклинании правильным считается слово “DTroottoothTD”, а слово “DTtoothrootTD” считается неправильным;

В названии веществ нет цифр, а используются только английские буквы;

каждое действие может быть несколько раз использовано в последующих действиях;

Гарантируется, что длина строки, содержащей сформированное заклинание, не превышает 50000 символов.

Входной файл содержит набор строк, каждая из которых описывает отдельное действие. Строки расположены в порядке выполнения действий. Длина каждой строки не превышает 255 символов. Количество строк не превышает 100.

Выходной файл должен содержать строку, содержащую сформированное заклинание.

Ход решения задачи «Зельеварение»

```
using System;
class App
{
   public static void Main()
   {
      Console.WriteLine("Введите количество действий");
      var n = Convert.ToInt32(Console.ReadLine());
      Console.WriteLine("Ввод производится в формате: (MIX или WATER или
DUST или FIRE)пробел(ИНГРИДИЕНТ)");
      var result = "";
      for (int i = 0; i < n; i++)
      {
         var x = Convert.ToString(Console.ReadLine());
         if (x.Substring(0,3) == "MIX")
         {
            var ingridient = x.Substring(4);
            ingridient = ingridient.Replace(" ", "");
            ingridient = ingridient.Replace("1", "");
            ingridient = ingridient.Replace("2", "");
            ingridient = ingridient.Replace("3", "");
            ingridient = ingridient.Replace("4", "");
            ingridient = ingridient.Replace("5", "");
            ingridient = ingridient.Replace("6", "");
            ingridient = ingridient.Replace("7", "");
            ingridient = ingridient.Replace("8", "");
            ingridient = ingridient.Replace("9", "");
            ingridient = ingridient.Replace("0", "");
            ingridient = ingridient.Replace("-", "");
            //запись строки в результат
            result = result + ingridient;
            result = "MX" + result + "XM";
         }
         else if (x.Substring(0, 5) == "WATER")
         {
            var ingridient = x.Substring(6);
            ingridient = ingridient.Replace(" ", "");
            ingridient = ingridient.Replace("1", "");
            ingridient = ingridient.Replace("2", "");
            ingridient = ingridient.Replace("3", "");
            ingridient = ingridient.Replace("4", "");
            ingridient = ingridient.Replace("5", "");
            ingridient = ingridient.Replace("6", "");
            ingridient = ingridient.Replace("7", "");
            ingridient = ingridient.Replace("8", "");
```

```
            ingridient = ingridient.Replace("9", "");
            ingridient = ingridient.Replace("0", "");
            ingridient = ingridient.Replace("-", "");
            result = result + ingridient;
            result = "WT" + result + "TW";
        }
        else if (x.Substring(0, 4) == "DUST")
        {
            var ingridient = x.Substring(5);
            ingridient = ingridient.Replace(" ", "");
            ingridient = ingridient.Replace("1", "");
            ingridient = ingridient.Replace("2", "");
            ingridient = ingridient.Replace("3", "");
            ingridient = ingridient.Replace("4", "");
            ingridient = ingridient.Replace("5", "");
            ingridient = ingridient.Replace("6", "");
            ingridient = ingridient.Replace("7", "");
            ingridient = ingridient.Replace("8", "");
            ingridient = ingridient.Replace("9", "");
            ingridient = ingridient.Replace("0", "");
            ingridient = ingridient.Replace("-", "");
            result = result + ingridient;
            result = "DT" + result + "TD";
        }
        else if (x.Substring(0, 4) == "FIRE")
        {
            var ingridient = x.Substring(5);
            ingridient = ingridient.Replace(" ", "");
            ingridient = ingridient.Replace("1", "");
            ingridient = ingridient.Replace("2", "");
            ingridient = ingridient.Replace("3", "");
            ingridient = ingridient.Replace("4", "");
            ingridient = ingridient.Replace("5", "");
            ingridient = ingridient.Replace("6", "");
            ingridient = ingridient.Replace("7", "");
            ingridient = ingridient.Replace("8", "");
            ingridient = ingridient.Replace("9", "");
            ingridient = ingridient.Replace("0", "");
            ingridient = ingridient.Replace("-", "");
            result = result + ingridient;
            result = "FR" + result + "RF";
        }
        else
        {
            Console.WriteLine("Ввод сделан не по формату, строка не
засчитывается");
```

```
            }
        }
        Console.WriteLine($"Получили заклинание {result}");
    }
}
```

Ввод и вывод данных производится через консоль. В самом начале вводится переменная, которая определяет количество, введенных пользователем, ингредиентов. Также заранее создаётся пустая строковая переменная result, которая будет являться нашим результатом работы. После введения названия заклинания и ингредиента программа считывает название заклинания, а затем удаляет все лишние символы из введенного ингредиента. В переменную result записывается обработанный ингредиент, а между ним ставятся буквы, в зависимости от введенного заклинания. Программа выполняет это действие пока пользователь не введет заданное им количество ингредиентов.

Исходные данные 1:

```
DUST aa
FIRE 1
WATER 2
MIX 3
DUST 4
```

Результат работы 1:

```
Введите количество действий
5
Ввод производится в формате: (MIX или WATER или DUST или FIRE)пробел(ИНГРИДИЕНТ)
DUST aa
FIRE 1
WATER 2
MIX 3
DUST 4
Получили заклинание DTMXWTFRDTaaTDRFTWXMTD
```

Исходные данные 2:

```
DUST XqK DWYMfDBZ ZZdnIqo
FIRE 1 Odg
WATER 1
MIX 3 2 xJZYkKXL
DUST 4 ceeJr
```

Результат работы 2:

```
Введите количество действий
5
Ввод производится в формате: (MIX или WATER или DUST или FIRE)пробел(ИНГРИДИЕНТ)
DUST XqK DWYMfDBZ ZZdnIqo
FIRE 1 Odg
WATER 1
MIX 3 2 xJZYkKXL
DUST 4 ceeJr
Получили заклинание DTMXWTFRDTXqKDWYMfDBZZZdnIqoTDOdgRFTWxJZYkKXLXMceeJrTD
```

Рисунок 3 – Пример работы программы задачи «Зельеваренье»

Постановка задачи «Отгадай число»

Известен следующий фокус. Фокусник предлагает выполнить действия следующего характера: задумайте число, прибавьте 2, умножьте на 3, отнимите 5, отнимите задуманное число и т.д. После этого по названному полученному результату фокусник определяет задуманное число.

Пусть задумано некоторое целое число X. Требуется после выполнения ряда действий по известному результату R определить это число.

Примечание:

Гарантируется, что имеется только один ответ;

Гарантируется, что во время выполнения действий какие-либо промежуточные результаты не превышают по модулю 2 000 000 000.

Входной файл

Первая строка содержит количество действий N ($0 \le N \le 100$).

Следующие N строк содержат описания действий в последовательности их выполнения, причем в каждой строке указывается одно действие в формате S V, где:

S – Тип действия, состоящий из одного символа: "*" – умножить; "-" – отнять; "+" - прибавить;

V – Аргумент действия. Может быть целым числом ($|V| \le 100$) либо символом "x". Символ "x" может применяться только в действиях "-" и "+" и обозначает, что нужно отнять или прибавить задуманное число, соответственно.

Последняя строка содержит результат R ($|R| \le 2\,000\,000\,000$).

Выходной файл должен содержать одно целое число – задуманное число X.

Ход решения задачи «Отгадай число»

```
using System;
using System.Diagnostics.CodeAnalysis;
using System.Diagnostics.Metrics;
class App
{
   public static void Main()
   {
      Console.WriteLine("Введите количество действий:");
      var n = Convert.ToInt32(Console.ReadLine());
      string[] moves = new string[n];
      int[] numbers = new int[n];
      int[] exes = new int[n];
      for (int i = 0; i < n; i++)
      {
         Console.WriteLine("Введите арифметическую операцию, а затем
число (если вводите X, то операцией может быть только + или -):");
         var move = Convert.ToString(Console.ReadLine());
         var x = Convert.ToString(Console.ReadLine());
         if (x == "x")
         {
            exes[i] = 1;
            moves[i] = move;
         }
         else
         {
            moves[i] = move;
            numbers[i] = Int32.Parse(x);
         }

      }
      double resultX = 1;
      double integerCoeficient = 0;

      for (int i = 0; i < moves.Length; i++)
      {
         var moveInt = moves[i];
         var NumInt = numbers[i];
         var NumX = exes[i];
         if (NumX == 0)
         {
            if (moveInt == "+")
            {
               integerCoeficient += NumInt;
            }
```

```
            if (moveInt == "-")
            {
                integerCoeficient -= NumInt;
            }
            if (moveInt == "*")
            {
                integerCoeficient *= NumInt;
                resultX *= NumInt
            }
        }
        else
        {
            if (moveInt == "+")
            {
                resultX += NumX;
            }
            if (moveInt == "-")
            {
                resultX -= NumX;
            }
        }
    }
    Console.WriteLine("Введите число, которое должно быть получено в качестве результата (убедитесь, что загаданное число может существовать)");
    double R = Convert.ToDouble(Console.ReadLine());
    if (resultX == 0 && integerCoeficient != R)
    {
        Console.WriteLine("Ошибка: загаданное число не может быть получено набранным набором действий");
    }
    else if (resultX == 0 && integerCoeficient == R)
    {
        Console.WriteLine($"Загаданное число равно {R}");
    }
    else
    {
        var result = (R -  integerCoeficient) / resultX;
        Console.WriteLine($"Загаданное число равно {result}");
    }
```

Ввод и вывод данных производится через консоль. В начале пользователь вводит переменную, которая задаёт количество арифметических операций, а затем последовательно вводит саму операцию и число с ней. Составляется

список условных «действий», который ввел пользователь, а также 2 других списка, которые по сути являются последовательностью введенных чисел. По данным спискам программа понимает когда был введен x, а когда конкретная константа. Составляется уравнение вида: x = <число введенное пользователем в конце>, с которым осуществляются все заданные пользователем операции. В конце программа решает данное уравнение, деля правую часть уравнения на коэффициент при x. Также отдельно предусмотрены случай, когда пользователь может ввести такое уравнение, где не существует решений (например: x - x + 10 = 2000).

Исходные данные 1:

```
4
* 3
+ 7
- x
- -5
2
```

Результат работы 1:

```
Введите количество действий:
4
Введите арифметическую операцию, а затем число (если вводите X, то операцией может быть только + или -):
*
3
Введите арифметическую операцию, а затем число (если вводите X, то операцией может быть только + или -):
+
7
Введите арифметическую операцию, а затем число (если вводите X, то операцией может быть только + или -):
-
x
Введите арифметическую операцию, а затем число (если вводите X, то операцией может быть только + или -):
-
-5
Введите число, которое должно быть получено в качестве результата (убедитесь, что загаданное число может существовать)
2
Загаданное число равно -5
```

Исходные данные 2

```
4
+ 2
* 3
- 5
- x
7
```

Результат работы 2:

```
Введите количество действий:
4
Введите арифметическую операцию, а затем число (если вводите X, то операцией может быть только + или -):
+
2
Введите арифметическую операцию, а затем число (если вводите X, то операцией может быть только + или -):
*
3
Введите арифметическую операцию, а затем число (если вводите X, то операцией может быть только + или -):
-
5
Введите арифметическую операцию, а затем число (если вводите X, то операцией может быть только + или -):
-
x
Введите число, которое должно быть получено в качестве результата (убедитесь, что загаданное число может существовать)
7
Загаданное число равно 3
```

Рисунок 4 – Пример работы программы «Отгадай число»

Постановка задачи «Крестьянин и чёрт»

Идет крестьянин и плачется: "Эхма! Жизнь моя горькая! Заела нужда совсем! Вот в кармане только несколько монет, да и те сейчас нужно отдать. И как это у других бывает, что на всякие свои деньги они еще деньги получают? Хоть бы кто помочь мне захотел".

Только успел это сказать, как глядь, а перед ним черт стоит и говорит: "Вот видишь этот мост через реку. Стоит тебе перейти через мост, и у тебя будет вдвое больше денег, чем есть. Перейдешь опять, и снова станет вдвое больше. Но за то, что я у тебя деньги удваиваю, после каждого перехода ты мне должен отдавать по K монет".

"Ой ли," – сказал крестьянин "ну-ка, попробуем". Перешел мост, и деньги у него удвоились. Отдал он черту K монет, перешел мост еще раз, и опять деньги удвоились. Снова отдал крестьянин черту K монет.

Однако после Z переходов и отдач черту по K монет оказалось, что у крестьянина не осталось ни одной монеты.

Требуется определить, сколько комбинаций условий перехода через мост может быть, если известно, что у крестьянина изначально было не более MaxN монет. Комбинацией условий перехода является тройка чисел N, K, Z, где N – начальное количество монет у крестьянина, K – количество монет, отдаваемых черту после каждого перехода, Z – количество переходов. Естественно, что для этой тройки должно выполняться условие, что после Z циклов у крестьянина не должно остаться монет.

Входной файл содержит целое число MaxN - максимальное количество, которое может быть изначально у крестьянина ($1 \le MaxN \le 2000000000$).

Выходной файл должен содержать одно целое число - количество комбинаций условий перехода через мост.

Ход решения задачи «Крестьянин и чёрт»

```
using System;
using System.Diagnostics.Metrics;
Console.WriteLine("Крестьянини и чёрт");
Console.WriteLine("Введите число, равное максимальному стартовому количеству денег у крестьянина:");
int nGlobal = Convert.ToInt32(Console.ReadLine());
int kGlobal = 0;
int z = 0;
if (nGlobal == 0)
{
   Console.WriteLine("Нисколько монет со страта... на что вы рассчитываете? :))");
}
else
{
   Console.WriteLine("Начинаем перебор...");
   Console.WriteLine("                N K Z");
   int countThrees = 0;
   for (int n = 1; n <= nGlobal; n++)
   {
      var nFor = n;
      for (int k = 2; (k != ((n * 2) + 1)); k++)
      {
         while (nFor != 0)
         {
            nFor = nFor * 2;
            nFor = nFor - k;
            if (nFor == 0)
            {
               z++;
               Console.WriteLine($"Подходящая тройка {n} {k} {z}");
               countThrees++;
               break;
            }
            if (nFor < 0)
            {
               break;
            }
            if (z > 7)
            {
               break;
            }
            z++;
         }
```

```
            nFor = n;
            z = 0;
        }
    }
    Console.WriteLine($"Количество подходящих троек равно {countThrees}");
}
```

Ввод и вывод данных производится через консоль. Пользователь вводит переменную n, равную максимальному начальному количеству денег у крестьянина. Программа начинает выполнять алгоритм, описанный в задаче, то есть начинает перебирать сумму денег от 1 до n и производить с ней следующие действия: умножает сумму денег на 2, а затем отнимает от неё количество денег, которое будет забирать чёрт при проходе через мост (очевидно, что подбирать количество денег, забираемое чертом для каждого n стоит пока оно не станет равно $\frac{n+1}{2}$, где n – кол-во денег у крестьянина). Как только программа видит, что в результате получился 0, то она записывает нужную нам тройку. Если она увидит, что в результате получилось отрицательное число, или наши действия зациклились, то программа переходит к следующему n. Так происходит пока программа не обработает все n, заданные пользователем.

Исходные данные 1:

3
Входные данные
1000
Выходные данные
1599

Результат работы 1:

```
Крестьянини и чёрт
Введите число, равное максимальному стартовому количеству денег у крестьянина:
1000
Начинаем перебор...
                N K Z
Подходящая тройка 1 2 1
Подходящая тройка 2 4 1
Подходящая тройка 3 4 2
Подходящая тройка 3 6 1
Подходящая тройка 4 8 1
Подходящая тройка 5 10 1
Подходящая тройка 6 8 2
Подходящая тройка 6 12 1
```

```
Подходящая тройка 993 1986 1
Подходящая тройка 994 1136 3
Подходящая тройка 994 1988 1
Подходящая тройка 995 1990 1
Подходящая тройка 996 1328 2
Подходящая тройка 996 1992 1
Подходящая тройка 997 1994 1
Подходящая тройка 998 1996 1
Подходящая тройка 999 1332 2
Подходящая тройка 999 1998 1
Подходящая тройка 1000 2000 1
Количество подходящих троек равно 1599
```

Исходные данные 2:

2
Входные данные
32
Выходные данные
49

Результат работы 2:

```
Подходящая тройка 27 54 1
Подходящая тройка 28 32 3
Подходящая тройка 28 56 1
Подходящая тройка 29 58 1
Подходящая тройка 30 32 4
Подходящая тройка 30 40 2
Подходящая тройка 30 60 1
Подходящая тройка 31 32 5
Подходящая тройка 31 62 1
Подходящая тройка 32 64 1
Количество подходящих троек равно 49
```

Рисунок 5 – Пример работы программы «Крестьянин и чёрт»

Заключение

В данной работе было рассмотрено решение четырёх задач с использованием языка программирования C#. Каждое задание требовало применение различных алгоритмических подходов и методов.

Таким образом, теоретические и практические знания о вышеописанном языке программирования были значительно укреплены. Также сюда следует отнести логику и знания об алгоритмизации и программировании в целом, что положительно скажется на дальнейшей работе с данным языком. Опыт, стремление и решение всё более сложных задач - залог успешного карьерного пути в IT индустрии.

Список литературы

1) Microsoft Learn документация по языку C# https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 05.11.2024).
2) Stack Overflow https://stackoverflow.com/ (дата обращения: 11.11.2024).
3) Википедия C# https://ru.wikipedia.org/wiki/C_Sharp (дата обращения: 15.11.2024).
4) Metanit C# https://metanit.com/sharp/ (дата обращения: 06.12.2024).