Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Практическое задание №6**

**Тема «Теория чисел»** **(вариант №12)**

Руководитель: Ржеутская Н. В.

Выполнил:

Студент 2 курса 1 группы ФИТ

Немкович Анастасия Вадимовна

Минск 2023

**Цель работы**

Получение основных сведений из курса теории чисел.

**Теоретические сведения**

Ниже рассматриваются: *N* – множество натуральных чисел, *Z* – множество рациональных чисел. Множество целых чисел *Z* – счетное, состоит из элементов 0; ±1; ±2; …; ± *n*,…. На нем определены две алгебраические операции – сложение и умножение. Эти операции обладают следующими свойствами (для любых ):

1. ассоциативность: ; ;

2. коммутативность: ; ;

3. существует нейтральный элемент – 0 и 1 соответственно:



4.  – закон дистрибутивности;

5. для каждого целого  существует единственное противоположное, то есть такое целое *b*, что *a* + *b* = *b* + *a* = 0.

*Теорема 2.1* (*О делении с остатком*). Для любых целых чисел *a* и *b*, , существует единственные целые числа *q* и  , такие, что .

В этом равенстве  называют остатком, а  – частным (неполным частным – при ) от деления *a*  на  При *r* = 0 величины *b* и *q* называют делителями или множителями числа *а*. Читатель со школьной скамьи умеет находить частное и остаток методом деления уголком.

*Следствие.* Пусть  – натуральное число,  Для всякого целого числа *a*  и максимального целого  с условием  существуют единственные целые  такие, что 

Такое равенство записывают сокращённо  или  (если *b* известно по контексту) и называют записью числа *a* в *b* – ичной позиционной системе счисления или системе счисления по основанию *b*. Нам кажется естественной привычная десятичная позиционная система записи целых чисел . В различных ситуациях более удобными оказываются другие основания. К примеру, во всех компьютерах на микроуровне вычисления проводятся в двоичной системе счисления. Для перехода к ней с десятичной применяют промежуточную – 16 - ричную систему счисления.

*Лемма 2.1.* Если в равенстве  все слагаемые – целые числа и все, кроме может быть одного, делятся на целое , то и это исключенное слагаемое делится на .

**Определение 2.1*.***Если целые числа  делятся на целое , то *d*  называют их *общим делителем*.

В дальнейшем речь идет только о положительных целых делителях.

**Определение 2.2.** Максимальный из общих делителей целых чисел  называется их *наибольшим общим делителем* и обозначается через НОД ().

*Теорема 2.2.* Если *,* то НОД *(a, b)*=НОД *(b, c).*

Теорема 2.2 позволила Евклиду (примерно 2300 лет тому назад) обосновать следующий факт.

*Теорема 2.3.* Наибольший общий делитель целых чисел  *a* и *b*   равен последнему отличному от нуля остатку цепочки равенств:

*;*

*;*

*…………………*

**

**

то есть  *=* НОД *.*

Теорема 2.3 формулирует **алгоритм Евклида нахождения наибольшего общего делителя целых чисел**. Его вариантом является следующий – второй способ вычисления наибольшего общего делителя по алгоритму Евклида – вычисляем последовательно разности  до получения последней ненулевой разности, которая и совпадает с НОД *(a, b).*

*Теорема 2.4.* Если *d* = НОД *(a, b)*, то существуют такие целые *u*  и *v*, что выполняется следующее соотношение (Безу): *d = au+ bv.*

**Пример 2.2.** Из примера 2.1 следует, что



Такой способ получения соотношения Безу для конкретных целых чисел называется **расширенным алгоритмом Евклида**. Он состоит из двух этапов: собственно алгоритма Евклида - прогонки вниз и прогонки вверх – последовательного выражения остатков в каждом из шагов предыдущего этапа (с соответствующим приведением подобных на каждом шаге).

**Определение 2.3.** Натуральное число ** называется *простым*, если оно делится только на1 и на себя.

*Теорема 2.5.* Всякое натуральное число ** либо является простым числом, либо имеет простой делитель.

Заметим, что из соотношения  натуральных чисел, больших единицы, следует, что, либо *p,* либо *q* принадлежит отрезку . Легко видеть, что наименьший натуральный делитель ** натурального числа ** является простым числом. Исторически первый метод проверки натурального числа ** на простоту заключается в делении его на простые числа, не превосходящие , носит название “решета Эратосфена”. К настоящему времени разработан достаточно большой цикл алгоритмов проверки числа на простоту.

*Теорема 2.6 (Евклид).* Простых чисел бесконечно много.

Значение простых чисел в том, что они по теореме 2.5 являются составными кирпичиками всех натуральных чисел.

**Определение 2.4.** Целые числа *a*  и  *b* называются***взаимно простыми****,* еслиНОД .

*Теорема 2.7* (*Критерий взаимной простоты целых чисел*). Целые числа  *a* и *b* взаимно просты тогда и только тогда, когда существуют такие целые u и v, что выполняется равенство .

**Следствие.** НОД** тогда и только тогда, когдаНОД иНОД .

Важным в теории чисел и ее приложениях является следующее свойство взаимно простых целых чисел.

*Лемма 2.2.* Пусть произведение целых чисел *ab* делится на целое число *с* и НОД . Тогда *b*  делится на  *с*.

*Теорема 2.8**(Основная теорема арифметики)*. Всякое целое число ** однозначно раскладывается в произведение простых множителей

*.*

Если в этом равенстве собрать одинаковые множители, то получим каноническое разложение целого числа: .

*Теорема 2.9.* Пусть *-* натуральное число*,* . Для любых целых чисел *a* и *b* следующие условия равносильны:

*1) a и b имеют одинаковые остатки от деления на *

*2) a – b делится на m, то есть a – b = mq для подходящего целого q;*

*3) a = b + mq для некоторого целого q.*

**Определение 2.5.**Целые числа *а* и *b* называются сравнимыми по модулю *m*, если они удовлетворяют одному из условий теоремы 2.9.Этот факт обозначают формулой ** илии называют данную формулу сравнением.

Основные свойства сравнений:

**1.** Пусть *.* Тогда  для всякого целого *c*, то есть к обеим частям сравнения можно добавить (или вычесть из обеих частей) одно и то же число.

**2.** Сравнения можно почленно складывать и вычитать: если **, *,* то  

**3.** Сравнения можно почленно перемножать: если ** *,* то **.

**4.** Сравнения можно почленно возводить в любую натуральную степень: если *,* то **.

**5.** Если в сравнении ** числа *a*, *b*, *m* имеют общий множитель *d*, то на него сравнение можно сократить: **.

**6.** Сравнение можно сократить на общий множитель, взаимно простой с модулем: если **, НОД (*d*, *m*) = 1, то из сравнения  следует сравнимость  и  по модулю .

**7.** Сравнение можно умножить на любой целый множитель: если **, то  для всякого целого *t*.

**8.** Рефлексивность: ** для любого целого *а* и всякого натурального *m* >1.

**9.** Симметричность: если **, то **.

**10.** Транзитивность: если **, **, то .

*Теорема 2.10*(***Малая теорема Ферма***). Пусть *p –* простое число и целое число *a* не делится на . Тогд*а .*

Теория сравнений и малая теорема Ферма позволяют быстро находить остаток от деления большого числа на простое число.

**Ход работы**

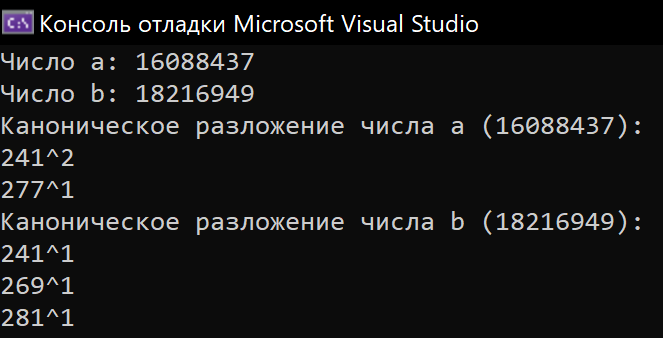
При a = 16088437, b = 18216949:

1. Найти канонические разложения чисел а, b.
2. Найти НОД (a, b) пользуясь a) алгоритмом Евклида; б) разложением чисел на простые множители.
3. С помощью расширенного алгоритма Евклида найти целые u, v, удовлетворяющие соотношению Безу: au + bv = НОД (a, b).
4. Найти остаток от деления 19952004 на 16.

**Задание 1.**

В разложении числа простые множители могут повторяться. Повторяющиеся простые множители можно записать более компактно, используя [степень числа](http://www.cleverstudents.ru/powers/powers.html). Пусть в разложении числа a простой множитель p1 встречается s1 раз, простой множитель p2 – s2 раз, и так далее, pn – sn раз. Тогда разложение на простые множители числа a можно записать как a=p1s1·p2s2·…·pnsn. Такая форма записи представляет собой так называемое **каноническое разложение числа на простые множители**.

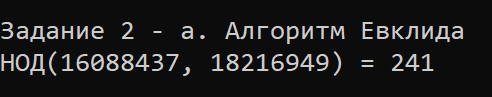
|  |
| --- |
| public static class PrimeFactorization  {  public static void PrintPrimeFactorization(int num)  {  if (num <= 0)  {  Console.WriteLine("Число должно быть положительным.");  return;  }  Dictionary<int, int> primeFactors = new Dictionary<int, int>();  for (int factor = 2; factor <= num; factor++)  {  while (num % factor == 0)  {  if (primeFactors.ContainsKey(factor))  primeFactors[factor]++;  else  primeFactors[factor] = 1;  num /= factor;  }  }  foreach (var kvp in primeFactors)  {  Console.WriteLine($"{kvp.Key}^{kvp.Value}");  }  }  } |



**Задание 2.**

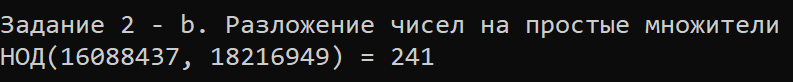
а) Алгоритм Евклида

|  |
| --- |
| public class EvclidForNOD  {  public static int FindNOD(int a, int b)  {  while (b != 0)  {  int temp = b;  b = a % b;  a = temp;  }  return a;  }  } |



б) Разложение чисел на простые множители

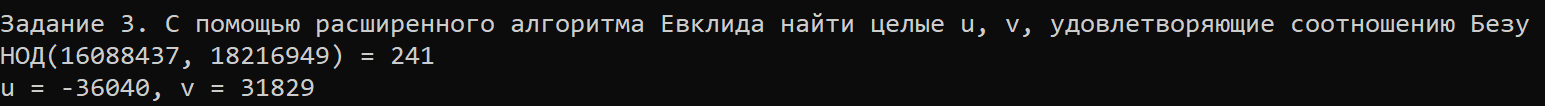
|  |
| --- |
| public class PrimeFactorizationUtility  {  // Функция для разложения числа на простые множители и возврата словаря простых множителей и их степеней  public static Dictionary<int, int> PrimeFactorization(int num)  {  Dictionary<int, int> primeFactors = new Dictionary<int, int>();  for (int factor = 2; factor <= num; factor++)  {  while (num % factor == 0)  {  if (primeFactors.ContainsKey(factor))  primeFactors[factor]++;  else  primeFactors[factor] = 1;  num /= factor;  }  }  return primeFactors;  }  // Функция для нахождения НОД двух чисел на основе разложения на простые множители  public static int FindGCDUsingPrimeFactorization(int a, int b)  {  Dictionary<int, int> primeFactorsA = PrimeFactorization(a);  Dictionary<int, int> primeFactorsB = PrimeFactorization(b);  int gcd = 1;  foreach (var kvpA in primeFactorsA)  {  if (primeFactorsB.ContainsKey(kvpA.Key))  {  int commonPower = Math.Min(kvpA.Value, primeFactorsB[kvpA.Key]);  gcd \*= (int)Math.Pow(kvpA.Key, commonPower);  }  }  return gcd;  }  } |



**Задание 3.**

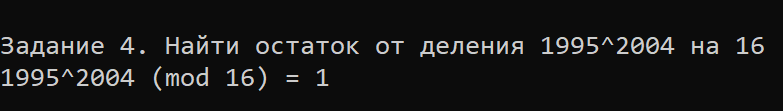
С помощью расширенного алгоритма Евклида найти целые числа u, v, удовлетворяющие соотношению Безу: au + bv = НОД (a, b).

|  |
| --- |
| public class ExtendedEuclideanAlgorithm  {  public static void ExtendedGCD(int a, int b, out int nodplus, out int u, out int v)  {  if (b == 0)  {  nodplus = a;  u = 1;  v = 0;  }  else  {  ExtendedGCD(b, a % b, out nodplus, out u, out v);  int temp = u;  u = v;  v = temp - (a / b) \* v;  }  }  } |



**Задание 4.** Найти остаток от деления 19952004 на 16.

|  |
| --- |
| public class ModularPowerCalculator  {  public static long CalculateModularPower(long baseNumber, long exponent, long modulus)  {  if (modulus == 1)  {  return 0;  }  long result = 1;  baseNumber = baseNumber % modulus;  while (exponent > 0)  {  if (exponent % 2 == 1)  {  result = (result \* baseNumber) % modulus;  }  exponent = exponent >> 1;  baseNumber = (baseNumber \* baseNumber) % modulus;  }  return result;  }  } |

****

**Вывод**

Многие криптографические алгоритмы базируются на результатах классической теории чисел. Теория чисел очень древняя наука, которая сейчас переросла в направление «Арифметическая геометрия». Но даже самые давние фундаментальные результаты этой науки только в наше время находят применения в очень востребованной ныне прикладной науке – криптографии.

Для понимания выполняемых операций в криптографических преобразованиях одним из наиболее часто используемых инструментов является расширенный алгоритм Евклида для нахождения мульипликативно обратного по модулю некоторого целого числа. Надо отметить, что в общем случае применение этого алгоритма значительно шире и затрагивает не только криптографические преобразования, но и, например, теорию алгебраического кодирования.

При компьютерной обработке информация представляется в виде наборов из 0 и 1 (двоичных наборов). Каждому такому набору ставится в соответствие натуральное число, запись которого в двоичной системе счисления (двоичная запись) совпадает с этим набором. Таким образом, компьютерная обработка информации сводится к обработке натуральных чисел. В современной криптографии сообщения представляются символами некоторого конечного алфавита (или последовательностями таких символов). Этим символам ставят в соответствие числа от 0 до N – 1, где N – число элементов (мощность) алфавита. Поэтому шифрование и расшифровывание сообщений представляют собой преобразование натуральных чисел, меньших N.

Разделом математики, предметная область которого – натуральные числа, является теория чисел. Таким образом, современная криптография связана с использованием результатов теории чисел, имеющей долгую историю развития. Ввиду конечности алфавитов сообщений важную роль играет раздел теории чисел – сравнения, в котором числа, имеющие одинаковые остатки от деления на фиксированное число (модуль), считаются одинаковыми.