**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Aлгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе Введение (Stepic)

Студенка Кузенкова Елизавета группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

[Задача 1: небольшое число Фибоначчи 3](#_Toc3047770)

[Исходный код к задаче 1 3](#_Toc3047771)

[Задача 2: последняя цифра большого числа Фибоначчи 3](#_Toc3047772)

[Исходный код к задаче 2 4](#_Toc3047773)

[Задача 3: огромное число Фибоначчи по модулю 4](#_Toc3047774)

[Исходный код к задаче 3 4](#_Toc3047775)

[Задача 4: наибольший общий делитель 5](#_Toc3047776)

[Исходный код к задаче 4 6](#_Toc3047777)

[Тест: правила работы с логарифмами 6](#_Toc3047778)

[Тест: правильная скорость роста 6](#_Toc3047779)

[Тест: правильная скорость роста 7](#_Toc3047780)

[Тест повышенной сложности: правильная скорость роста 8](#_Toc3047781)

# Задача 1: небольшое число Фибоначчи

Дано целое число 1≤*n*≤40

, необходимо вычислить *n*-е число Фибоначчи (напомним, что *F*0=0, *F*1=1 и *Fn*=*Fn*−1+*Fn*−2 при *n*≥2

).

**Sample Input:**

3

**Sample Output:**

2

# Исходный код к задаче 1

class Lab1\_1

{

private void DoWork(string[] args)

{

var n = int.Parse(Console.ReadLine());

Console.WriteLine(this.Fib(n));

}

private int Fib(int n)

{

if (n == 1 || n == 2)

{

return 1;

}

else

{

return this.Fib(n - 1) + this.Fib(n - 2);

}

}

public static void Main(string[] args)

{

var app = new Lab1\_1();

app.DoWork(args);

}

}

# Задача 2: последняя цифра большого числа Фибоначчи

Дано число 1≤*n*≤107, необходимо найти последнюю цифру *n*-го числа Фибоначчи.

Как мы помним, числа Фибоначчи растут очень быстро, поэтому при их вычислении нужно быть аккуратным с переполнением. В данной задаче, впрочем, этой проблемы можно избежать, поскольку нас интересует только последняя цифра числа Фибоначчи: если 0≤*a*,*b*≤9

— последние цифры чисел *Fi* и *Fi*+1 соответственно, то (*a*+*b*)mod10 — последняя цифра числа *Fi*+2

**Sample Input:**

317457

**Sample Output:**

2

# Исходный код к задаче 2

class Lab1\_2

{

private void DoWork(string[] args)

{

var n = int.Parse(Console.ReadLine());

Console.WriteLine(this.Fib(n));

}

private int Fib(int n)

{

int[] f = new int[2];

f[0] = 1; f[1] = 1;

for (int i = 2; i <= (n - 1); i++)

f[i % 2] = (f[i % 2] + f[(i + 1) % 2]) % 10;

return f[(n - 1) % 2];

}

public static void Main(string[] args)

{

var app = new Lab1\_2();

app.DoWork(args);

}

}

# Задача 3: огромное число Фибоначчи по модулю

Даны целые числа 1≤*n*≤1018 и 2≤*m*≤105, необходимо найти остаток от деления *n*-го числа Фибоначчи на *m*

**Sample Input:**

10 2

**Sample Output:**

1

# Исходный код к задаче 3

class Lab1\_3

{

/\*\*

\* Метод для определения элемента в последовательности периодов Пизано

\*

\* @param divider делитель

\* @return элемента последовательности периода Пизано

\*/

private static long CalcPisanoPeriods(long divider)

{

long a = 0;

long b = 1;

long c;

for (long i = 0L; i < divider \* divider; i++)

{

c = (a + b) % divider;

a = b;

b = c;

if (a == 0 && b == 1)

{

return i + 1;

}

}

return a;

}

private long GetHugeFibonacci(long elementNumber, long divider)

{

long remainder = elementNumber % CalcPisanoPeriods(divider);

long first = 0L;

long second = 1L;

long result = remainder;

for (int i = 1; i < remainder; i++)

{

result = (first + second) % divider;

first = second;

second = result;

}

return result % divider;

}

private void DoWork(string[] args)

{

var input = Console.ReadLine().Split(' ').Select(long.Parse).ToArray();

Console.WriteLine(this.GetHugeFibonacci(input[0], input[1]));

}

public static void Main(string[] args)

{

var app = new Lab1\_3();

app.DoWork(args);

}

}

# Задача 4: наибольший общий делитель

По данным двум числам 1≤*a*,*b*≤2⋅109 найдите их наибольший общий делитель.

**Sample Input 1:**

18 35

**Sample Output 1:**

1

**Sample Input 2:**

14159572 63967072

**Sample Output 2:**

4

# Исходный код к задаче 4

class Lab1\_4

{

private int GetNOD(int firstValue, int secondValue)

{

if (firstValue == 0)

{

return secondValue;

}

else if (secondValue == 0)

{

return firstValue;

}

else if (firstValue >= secondValue)

{

return GetNOD(firstValue % secondValue, secondValue);

}

else

{

return GetNOD(firstValue, secondValue % firstValue);

}

}

private void DoWork(string[] args)

{

var input = Console.ReadLine().Split(' ').Select(int.Parse).ToArray();

Console.WriteLine(this.GetNOD(input[0], input[1]));

}

public static void Main(string[] args)

{

var app = new Lab1\_4();

app.DoWork(args);

}

}

# Тест: правила работы с логарифмами

Отметьте **все** верные утверждения.

+

# Тест: правильная скорость роста

Отметьте **все** верные утверждения.

+

+

+

+

+

+

+

Чтобы проверить равенства вида *f*(*n*)=*O*(*g*(*n*)) нужно найти lim*n*→∞*f*(*n*)/*g*(*n*) . Если он равен константе, равенства верны и для *O*, и для Ω, и для Θ. Если бесконечности, то только для Ω. Если нулю, то только для *O*.

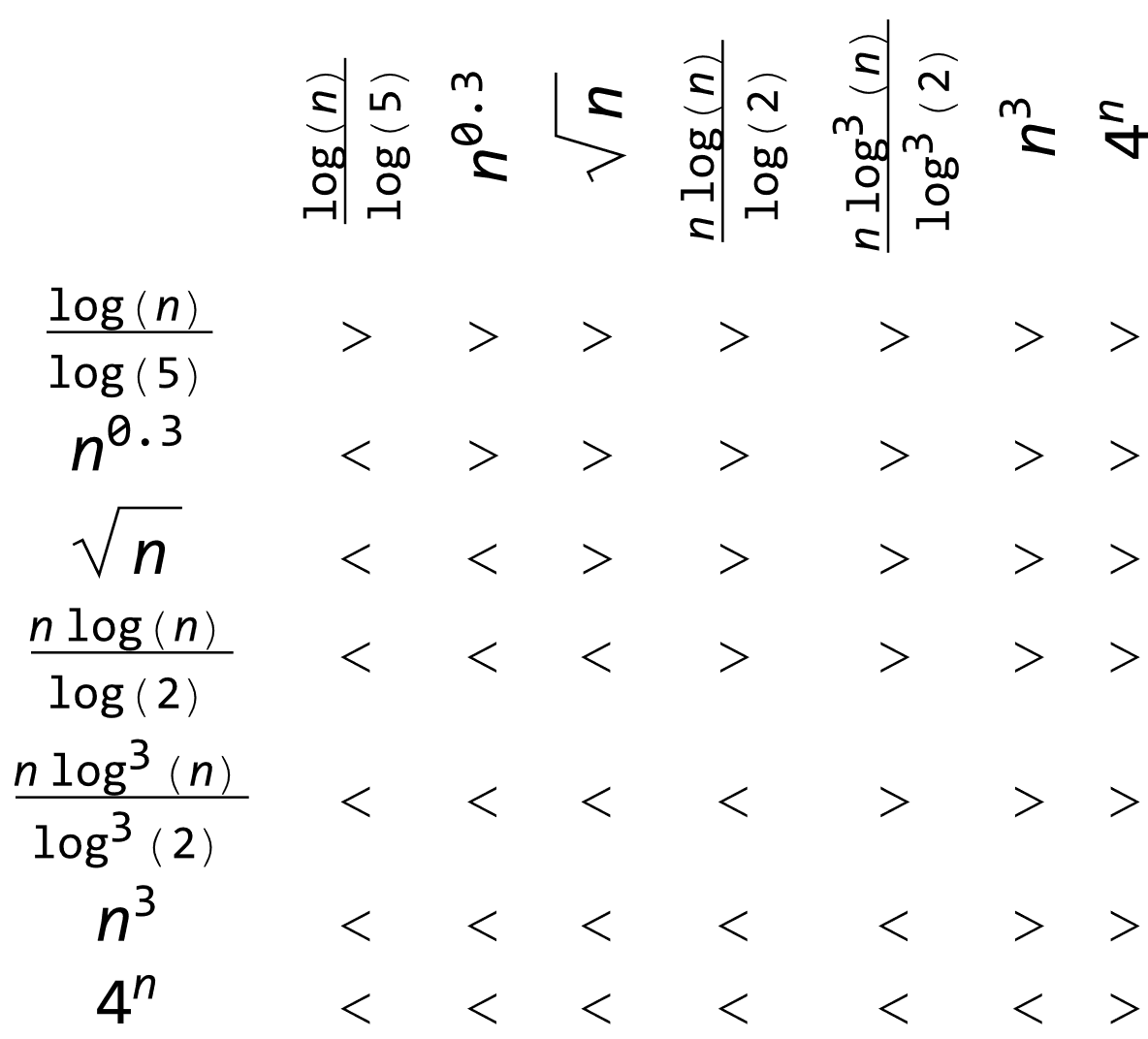
# Тест: правильная скорость роста

Упорядочите данные семь функций по возрастанию скорости роста (сверху — медленнее всего растущая функция, снизу — быстрее всего растущая).

ff = Inactivate[{ }];  
sf = Sort[ff, {f1, f2} => Limit[Activate[f1/f2], n -> Infinity] != Infinity];

{}

Grid[tableData = Transpose[Prepend[Transpose[Prepend[  
 (f1 => ((f2 => If[Limit[Activate[f1/f2], n -> Infinity] != Infinity, ">", "<"]) /@ sf)) /@ sf,   
 Rotate[TraditionalForm[#], Pi/2] & /@ sf]], Prepend[TraditionalForm /@ sf, ""]]]]



# Тест повышенной сложности: правильная скорость роста

Упорядочите данные функции по возрастанию скорости роста (сверху — медленнее всего растущая функция, снизу — быстрее всего растущая):

ff = Inactivate[{  
 Sqrt[Log[4, n]], Log[3, n], Log[2, n!],   
 n/Log[5, n], Sqrt[n], Log[2, n]^2,   
 Log[2, Log[2, n]], 3^Log[2, n], 2^(3 n), n^2,   
 n^Log[2, n], n^Sqrt[n], 4^n,   
 2^n, 7^Log[2, n], n!,   
 Log[2, n]^Log[2, n], 2^2^n  
}];  
sf = Sort[ff, {f1, f2} => Limit[Activate[f1/f2], n -> Infinity] != Infinity];

{  
 Log[2,Log[2,n]],Sqrt[Log[4,n]],Log[3,n],  
 Log[2,n]^2,Sqrt[n],n\*Log[5,n]^(1),  
 Log[2,n!],3^Log[2,n],n^2,  
 7^Log[2,n],Log[2,n]^Log[2,n],n^Log[2,n],  
 n^Sqrt[n],2^n,4^n,  
 2^(3\*n),n!,2^2^n  
}

Grid[tableData = Transpose[Prepend[Transpose[Prepend[  
 (f1 => ((f2 => If[Limit[Activate[f1/f2], n -> Infinity] != Infinity, ">", "<"]) /@ sf)) /@ sf,   
 Rotate[TraditionalForm[#], Pi/2] & /@ sf]], Prepend[TraditionalForm /@ sf, ""]]]]

