ПРОГРАММА ДЛЯ ПОСТРОЕНИЯ ПРАВИЛЬНЫХ МНОГОУГОЛЬНИКОВ ЦИРКУЛЕМ И ЛИНЕЙКОЙ

А. Р. Сафин[1]

Оглавление

[1] safinaskar@mail.ru. 11А класс. Лицей им. Н. И. Лобачевского при Казанском государственном университете. Научные руководители: Скопенков А. Б., МГУ; к. ф.-м. н., доцент КГТУ Бронштейн М. Д.; к. ф.-м. н., доцент КГУ Лернер Э. Ю.

Аннотация

С давних времён людей интересовал вопрос, как строить различные фигуры при помощи циркуля и линейки. Особенно интересны в этом плане были правильные многоугольники.

Я написал компьютерную программу в пакете *Mathematica*, с помощью которой можно построить все построимые правильные многоугольники. По теореме Гаусса-Ванцеля (всюду далее m — натуральное число, p_1, p_2, p_3, ..., p_l — различные простые числа Ферма, то есть числа вида 2^m+1 и q — целое неотрицательное число) число вершин в них имеет вид $2^q p_1 p_2 p_3 \dots p_l$.

Для построения правильного $2^q p_1 p_2 p_3 \dots p_l$-угольника достаточно уметь строить правильные p_1, p_2, p_3, ..., p_l-угольники. Очевидно, что для построения правильного p-угольника достаточно построить отрезок длиной $\cos\frac{2\pi}{p}$. А для этого нужно, чтобы $\cos\frac{2\pi}{p}$ был квадратичной иррациональностью, то есть выражался через натуральные числа при помощи арифметических операций и квадратных корней из неотрицательных чисел. Эти факты доказаны в приложении А. Моя программа выражает $\cos\frac{2\pi}{p}$ в радикалах.

Иоганн Густав Гермес **более 10 лет** разрабатывал способ построения правильного 65 537-угольника и описал его в рукописи размером более 200 страниц, которая хранится в библиотеке Гёттингенского университета ([?]). Моя программа находит соответствующее построение **около минуты**.

Идея о программе и основные принципы её работы взяты из [2]. При написании программы я столкнулся с проблемами (они приводятся в параграфе 2), которые успешно решил вместе с научными руководителями. Программу и настоящий текст я написал сам.

Благодарим М. А. Малахальцева за полезные обсуждения.

1. Теоретическая основа программы

Определение 1. Будем называть действительное число *построимым*, если оно является квадратичной иррациональностью.

Сейчас будет показано, как выразить $\cos\frac{2\pi}{5}$. Общий случай рассмотрим позже. Обозначим $\varepsilon = \cos\frac{2\pi}{5} + i\sin\frac{2\pi}{5}$. Тогда нам нужно найти $\frac{\varepsilon+\varepsilon^4}{2} = \cos\frac{2\pi}{5}$. Мы знаем, что

$$\varepsilon + \varepsilon^2 + \varepsilon^3 + \varepsilon^4 = -1.$$

Обратим внимание, на то что $(\varepsilon+\varepsilon^4)+(\varepsilon^2+\varepsilon^3) = (\varepsilon+\varepsilon^4)(\varepsilon^2+\varepsilon^3) = -1$. Значит, по теореме Виета $\varepsilon+\varepsilon^4$ и $\varepsilon^2+\varepsilon^3$ — корни уравнения x^2+x-1. Решаем его и получаем $\frac{-1+\sqrt{5}}{2}$ и $\frac{-1-\sqrt{5}}{2}$. Так как в данном случае ответ должен быть положительным, $\cos\frac{2\pi}{5} = \frac{-1+\sqrt{5}}{4}$.

Перейдём к общему случаю. Всюду далее $p = 2^m+1$ — простое число и нам нужно выразить $\cos\frac{2\pi}{p}$. Обозначим $\varepsilon = \cos\frac{2\pi}{p} + i\sin\frac{2\pi}{p}$. Разобьём сумму

$$\varepsilon + \varepsilon^2 + \varepsilon^3 + \ldots + \varepsilon^{2^m} = -1$$

на такие 2 части (назовём их $A_{1,0}$ и $A_{1,1}$), чтобы они были построимы и ε и ε^{2^m} попало в $A_{1,0}$. Тогда $A_{1,0}$ и $A_{1,1}$ построимы. Затем аналогично разбиваем $A_{1,0}$ на $A_{2,0}$ и $A_{2,2}$, $A_{1,1}$ — на $A_{2,1}$ и $A_{2,3}$ и так далее. Каждый раз v-ый осколок u-ого уровня мы будем называть $A_{u,v}$ и делить его на $A_{u+1,v}$ и $A_{u+1,v+2^u}$. В конце мы получим $A_{m-1,0} = \varepsilon + \varepsilon^{-1} = 2\cos\frac{2\pi}{p}$.

Пусть g — первообразный корень степени 2^m из 1 по модулю $2^m + 1$, то есть g таково, что остатки от деления на p чисел g^1, g^2, g^3, …, g^{p-1} различны. Будем разбивать так, чтобы для любых целых u и v таких, что $0 \leqslant u \leqslant m-1$ и $0 \leqslant v \leqslant 2^u - 1$, было верно равенство

$$A_{u,v} = \varepsilon^{g^v} + \varepsilon^{g^{2^u+v}} + \varepsilon^{g^{2\cdot 2^u+v}} + \varepsilon^{g^{3\cdot 2^u+v}} + \ldots + \varepsilon^{g^{2^m-2^u+v}}. \tag{1}$$

Это возможно, так как для любых u и v, таких что $v \leqslant 2^{u-1}-1$, верно равенство $A_{u-1,v} = A_{u,v} + A_{u,v+2^{u-1}}$. Также $A_{m-1,0} = \varepsilon + \varepsilon^{-1}$. Попробуем понять, что такое $A_{u,v}A_{u,v+2^{u-1}}$. Безусловно, это линейная комбинация от степеней ε. Обозначим коэффициент при ε^s, где s — целое число, как $\alpha(s)$. Очевидно, что тогда $\alpha(s)$ — это количество решений сравнения

$$g^{2^u k+v} + g^{2^u l+2^{u-1}+v} \equiv s \pmod p \tag{2}$$

относительно k и l. При этом решения, в которых корни имеют одинаковые остатки от деления на 2^{m-u}, считаются за одно.

Если бы оказалось, что $\alpha(0) = 0$ и для любого целого s_0 верно равенство $\alpha(s_0) = \alpha(s_0 g^{2^{u-1}})$, то $A_{u,v}A_{u,v+2^{u-1}}$ было бы линейной комбинацией от $A_{u-1,0}$, $A_{u-1,1}$, $A_{u-1,2}$, $A_{u-1,3}$, …, $A_{u-1,2^{u-1}-1}$, в самом деле,

$$\begin{aligned}
A_{u,v}A_{u,v+2^{u-1}} &= \alpha(1)\varepsilon + \alpha(2)\varepsilon^2 + \alpha(3)\varepsilon^3 + \ldots + \alpha(2^m)\varepsilon^{2^m} = \\
&= \alpha(1)\sum_{i=0}^{2^{m-u+1}-1} \varepsilon^{g^{2^{u-1}i}} + \alpha(g)\sum_{i=0}^{2^{m-u+1}-1} \varepsilon^{g^{2^{u-1}i+1}} + \alpha(g^2)\sum_{i=0}^{2^{m-u+1}-1} \varepsilon^{g^{2^{u-1}i+2}} + \\
&+ \alpha(g^3)\sum_{i=0}^{2^{m-u+1}-1} \varepsilon^{g^{2^{u-1}i+3}} + \ldots + \alpha(g^{2^{u-1}-1})\sum_{i=0}^{2^{m-u+1}-1} \varepsilon^{g^{2^{u-1}i+2^{u-1}-1}} = \\
&= \alpha(1)A_{u-1,0} + \alpha(g)A_{u-1,1} + \alpha(g^2)A_{u-1,2} + \alpha(g^3)A_{u-1,3} + \ldots + \\
&+ \alpha(g^{2^{u-1}-1})A_{u-1,2^{u-1}-1}.
\end{aligned}$$

И это действительно так! Доказательства ниже.

Лемма 1. $\alpha(0) = 0$.

Идейное резюме доказательства. Предположим противное. Тогда у сравнения (2) есть решение (k_0, l_0) при $s = 0$. Проводим простейшие преобразования и приходим к противоречию в связи с чётностью. □

Доказательство. Предположим противное. Тогда существуют такие целые k_0 и l_0, что

$$g^{2^u k_0+v} + g^{2^u l_0+2^{u-1}+v} \equiv 0 \pmod p.$$

$$g^{2^u k_0+v} \equiv -g^{2^u l_0+2^{u-1}+v} \pmod{p}.$$

$g^{2^{m-1}} = -1$, так как g — первообразный корень, отсюда

$$g^{2^u k_0+v} \equiv g^{2^{m-1}+2^u l_0+2^{u-1}+v} \pmod{p}.$$

$$2^u k_0 + v \equiv 2^{m-1} + 2^u l_0 + 2^{u-1} + v \pmod{2^m}.$$

Разделим сравнение на 2^{u-1}.

$$2k_0 \equiv 2^{m-u} + 2l_0 + 1 \pmod{2^{m-u+1}}.$$

Противоречие в связи с чётностью. □

Лемма 2. Для любого целого s_0 верно равенство $\alpha(s_0) = \alpha(s_0 g^{2^{u-1}})$.

Идейное резюме доказательства. Простейшими преобразованиями преобразуем сравнение (2) при $s = s_0$, $k = k_0$ и $l = l_0$ к сравнению (2) при $s = s_0 g^{2^{u-1}}$, $k = l_0 + 1$ и $l = k_0$. □

Доказательство. Установим соответствие между решениями. Пусть $(k_0; l_0)$ является решением сравнения (2) при $s = s_0$.

$$g^{2^u k_0+v} + g^{2^u l_0+2^{u-1}+v} \equiv s_0 \pmod{p}.$$

Умножим обе части сравнения на $g^{2^{u-1}}$.

$$g^{2^u k_0+2^{u-1}+v} + g^{2^u (l_0+1)+v} \equiv s_0 g^{2^{u-1}} \pmod{p}.$$

Поменяем местами слагаемые в левой части сравнения.

$$g^{2^u (l_0+1)+v} + g^{2^u k_0+2^{u-1}+v} \equiv s_0 g^{2^{u-1}} \pmod{p}.$$

Это значит, что $(k_0; l_0)$ — корень сравнения (2) при $s = s_0$ тогда и только тогда, когда $(l_0 + 1; k_0)$ — корень сравнения (2) при $s = s_0 g^{2^{u-1}}$. □

Замечание 1. Все $A_{u,v}$ являются суммами чисел вида $A_{m-1,v} = \varepsilon^a + \varepsilon^{-a} \in \mathbb{R}$, поэтому в выражении берутся квадратные корни только из неотрицательных чисел.

2. Проблемы, с которыми мы столкнулись и которые успешно решили

2.1. Выбор корня

Когда программа выражает $A_{u,v}$ и $A_{u,v+2^{u-1}}$ как корни квадратного уравнения, непонятно, какой из полученных корней — $A_{u,v}$, а какой — $A_{u,v+2^{u-1}}$. Чтобы определить соответствие, программа считает оба числа непосредственно по формуле (1) и сравнивает. Если $A_{u,v}$ меньше, то меньший из полученных корней — $A_{u,v}$ и наоборот. Существует аналитический метод выбора корня, но он довольно сложен, поэтому время работы программы при аналитическом способе сравнимо со временем при моём способе.

2.2. Точность

При выборе корня (см. предыдущий параграф) нужно, чтобы абсолютная погрешность разности $A_{u,v} - A_{u,v+2^{u-1}}$ была меньше этой разности, иначе будет невозможно наверняка установить соответствие между корнями и $A_{u,v}$ и $A_{u,v+2^{u-1}}$.

Эта проблема легко решается, так как в пакете *Mathematica* реализована возможность считать с любой точностью. Более того, можно хранить вместе с числом его точность и контролировать, не превысила ли погрешность разумные пределы. Об этом можно прочитать на `http://reference.wolfram.com/mathematica/tutorial/NumericalPrecision.html`.

Мы использовали эту возможность. В программе предусмотрен оператор, сравнивающий абсолютную погрешность разности $A_{u,v} - A_{u,v+2^{u-1}}$ с самой разностью. Если погрешность окажется больше, программа выдаёт сообщение об ошибке. Но это сообщение не выдаётся, поэтому можно быть уверенным, что корень выбирается правильно, а значит, программа выдаёт правильное выражение.

2.3. Композиция

Для $p = 65\,537$ итоговое выражение велико: оно содержит $17\,661\,551\,154\,923\,608$ корней. Ясно, что выражение не может уместиться в памяти компьютера, поэтому оно хранится в виде композиции нескольких функций (далее просто композиция). Она очень удобна, так как содержит все части, повторяющиеся в выражении, только по одному разу. Поясню сказанное на примере: для $p = 17$ выражением будет

$$\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\left(-\frac{1}{2}+\frac{\sqrt{17}}{2}\right)+\sqrt{1+\frac{1}{4}\left(-\frac{1}{2}+\frac{\sqrt{17}}{2}\right)^2}\right)+\right.$$
$$\left.+\sqrt{\frac{1}{2}\left(\frac{1}{2}+\frac{\sqrt{17}}{2}\right)-\sqrt{1+\frac{1}{4}\left(-\frac{1}{2}-\frac{\sqrt{17}}{2}\right)^2}+\frac{1}{4}\left(\frac{1}{2}\left(-\frac{1}{2}+\frac{\sqrt{17}}{2}\right)+\sqrt{1+\frac{1}{4}\left(-\frac{1}{2}+\frac{\sqrt{17}}{2}\right)^2}\right)^2}\right).$$

А вот его композиция:

$$A_{1,0} = -\frac{1}{2}+\frac{\sqrt{17}}{2};$$
$$A_{1,1} = -\frac{1}{2}-\frac{\sqrt{17}}{2};$$
$$A_{2,0} = \frac{1}{2}A_{1,0}+\sqrt{-A_{1,0}+\frac{1}{4}A_{1,0}^2-A_{1,1}};$$
$$A_{2,1} = \frac{1}{2}A_{1,1}+\sqrt{-A_{1,0}-A_{1,1}+\frac{1}{4}A_{1,1}^2};$$
$$A_{3,0} = \frac{1}{2}A_{2,0}+\sqrt{\frac{1}{4}A_{2,0}^2-A_{2,1}}.$$

Для $p = 3$, 5, 17 и 257 программа находит композицию и итоговое выражение, а для 65 537 — только итоговое выражение. Для больших p композиция гораздо короче самого выражения, например, для $p = 65\,537$ она содержит всего 2 103 корня.

Обратим внимание: чтобы построить правильный многоугольник, не нужно предварительно превращать композицию в выражение, поэтому исходная задача решена и для $p = 65\,537$.

Программа проверяет правильность своей работы путём приближённого (с учётом погрешности) вычисления $\cos\frac{2\pi}{p}$ и сравнения его с приближением косинуса, полученным стандартными средствами *Mathematica*. Чтобы выполнить такое сравнение, тоже не обязательно знать итоговое выражение, поэтому программа проверила себя для всех p, в том числе для $p = 65\,537$.

Программа также считает количество сложений и вычитаний, умножений, делений и корней в композиции и итоговом выражении. При этом количество этих операций в итоговом выражении для $p = 65\,537$ было получено нами косвенным путём, а именно анализом композиции.

3. Сокращение времени работы программы

3.1. Нахождение $\alpha(s)$

Это самое существенное сокращение времени работы программы. Без него запуск программы для $p = 65\,537$ оказался бы невозможным.

Как найти $\alpha(s)$? Можно смоделировать умножение $A_{u,v}$ и $A_{u,v+2^{u-1}}$ как многочленов с одной переменной ε. Тогда первые 2^{u-1} коэффициента результата будут равны $\alpha(s)$. Но при $p = 65\,537$ $A_{u,v}$ и $A_{u,v+2^{u-1}}$ будут содержать до 32 768 слагаемых, значит, это моделирование потребует до 1 073 741 824 операций, и всё это только для нахождения одних $A_{u,v}$ и $A_{u,v+2^{u-1}}$.

Понятно, что такой путь совершенно неприемлим. Поэтому мы моделируем умножение $A_{u+1,v}$ на $\varepsilon^{g^{2^{u-1}+v}}$. Обозначим коэффициент при ε^s в результате этого как $\beta(s)$. Все $\alpha(s)$ находятся исходя из следующей леммы.

Лемма 3. Для любого s_0 верно равенство

$$\alpha(s_0) = \beta(s_0) + \beta(s_0 g^{2^{u-1}}) + \beta(s_0 g^{2\cdot 2^{u-1}}) + \beta(s_0 g^{3\cdot 2^{u-1}}) + \ldots + \beta(s_0 g^{2^m - 2^{u-1}}). \quad (3)$$

Идейное резюме доказательства. Составляем сравнение, количеством корней которого является $\beta(s)$. Тогда правая часть (3) является количеством корней некоторой совокупности сравнений. Объединяя их и вводя новые переменные, приходим к сравнению (2) при $s = s_0$. □

Доказательство. Очевидно, что $\beta(s)$ — это количество решений сравнения

$$g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s \pmod p$$

относительно x. При этом решения с одинаковыми остатками по модулю 2^{m-u-1} считаются за одно. Тогда правая часть (3) равна количеству решений совокупности сравнений

$$\left[\begin{array}{l} g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 \pmod p \\ g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 g^{2^{u-1}} \pmod p \\ g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 g^{2\cdot 2^{u-1}} \pmod p \\ g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 g^{3\cdot 2^{u-1}} \pmod p \\ \vdots \\ g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 g^{2^m - 2^{u-1}} \pmod p \end{array}\right.$$

относительно x. Решения считаются за одно по тому же принципу.

Все сравнения, номера которых имеют остаток 1 по модулю 4, объединим в одно, 2 — в другое, 3 — в третье, 0 — в четвёртое. Приходим к совокупности

$$\left[\begin{array}{l} g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 g^{2^{u+1}y} \pmod p \\ g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 g^{2^{u+1}y+2^{u-1}} \pmod p \\ g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 g^{2^{u+1}y+2\cdot 2^{u-1}} \pmod p \\ g^{2^{u+1}x+v} + g^{2^{u-1}+v} \equiv s_0 g^{2^{u+1}y+3\cdot 2^{u-1}} \pmod p \end{array}\right.$$

относительно x и y. Решения, в которых корни имеют одинаковые остатки по модулю 2^{m-u-1}, считаются за одно. Разделим обе части первого сравнения на $g^{2^{u+1}y}$, второго — на $g^{2^{u+1}y+2^{u-1}}$, третьего — на $g^{2^{u+1}y+2\cdot 2^{u-1}}$ и четвёртого — на $g^{2^{u+1}y+3\cdot 2^{u-1}}$.

$$\left[\begin{array}{l} g^{2^{u+1}(x-y)+v} + g^{-2^{u+1}y+2^{u-1}+v} \equiv s_0 \pmod p \\ g^{2^{u+1}(x-y)-2^{u-1}+v} + g^{-2^{u+1}y+v} \equiv s_0 \pmod p \\ g^{2^{u+1}(x-y)-2\cdot 2^{u-1}+v} + g^{-2^{u+1}y-2^{u-1}+v} \equiv s_0 \pmod p \\ g^{2^{u+1}(x-y)-3\cdot 2^{u-1}+v} + g^{-2^{u+1}y-2\cdot 2^{u-1}+v} \equiv s_0 \pmod p. \end{array}\right.$$

Введём новые переменные.

- В первом сравнении: $k = x - y$ и $l = -y$
- Во втором: $k = x - y - \frac{3}{4}$ и $l = -y - \frac{1}{4}$ (здесь и далее $\frac{a}{b}$ — это такое c, что $bc \equiv a \pmod p$)
- В третьем: $k = x - y - \frac{1}{2}$ и $l = -y - 1$
- В четвёртом: $k = x - y - 1\frac{1}{4}$ и $l = -y - 1\frac{1}{4}$

Получим:

$$\left[\begin{array}{l} g^{2^{u+1}k+v} + g^{2^{u+1}l+2^{u-1}+v} \equiv s_0 \pmod p \\ g^{2^{u+1}k+2^u+v} + g^{2^{u+1}l+2^{u-1}+v} \equiv s_0 \pmod p \\ g^{2^{u+1}k+v} + g^{2^{u+1}l+2^u+2^{u-1}+v} \equiv s_0 \pmod p \\ g^{2^{u+1}k+2^u+v} + g^{2^{u+1}l+2^u+2^{u-1}+v} \equiv s_0 \pmod p. \end{array}\right.$$

Объединим все сравнения в одно.

$$g^{2^u k+v} + g^{2^u l+2^{u-1}+v} \equiv s_0 \pmod{p}.$$

Решения, в которых корни имеют одинаковые остатки по модулю 2^{m-u}, считаются одинаковыми. Сравниваем полученное сравнение с (2). □

3.2. Выражение не всех $A_{u,v}$

Большинство $\alpha(s)$ равны нулю, поэтому чтобы выразить $\cos\frac{2\pi}{p}$ через квадратные радикалы, необязательно выражать все $A_{u,v}$. Поэтому находятся только те $A_{u,v}$, которые используются в итоговом выражении. Для этого в программе применяется рекурсия. В программе есть процедура, к которой в качестве аргументов передаются u и v и которая выражает $A_{u,v}$. Она находит все $\alpha(s)$ и вызывает себя рекурсивно для тех осколков более низкого уровня, для которых $\alpha(s) \neq 0$. Также она вызывает $A_{u-1,v}$, так как $A_{u,v} + A_{u,v+2^{u-1}} = A_{u-1,v}$. В самой программе эта процедура вызывается для $A_{m-1,0}$ и результат делится на 2. В табл. 1 на с. 9 в последних двух строках отчётливо видно, насколько сокращаются вычисления, особенно для больших p.

3.3. Повторное выражение $A_{u,v}$

В описанном в предыдущем параграфе алгоритме может случиться так, что указанная процедура вызовется несколько раз для одного и того же $A_{u,v}$. Поэтому программа в специальном массиве отмечает каждый вызов процедуры.

3.4. Дохождение до уровня $m-1$

Мы находим $\cos\frac{2\pi}{p}$ как $\frac{A_{m-1,0}}{2}$, хотя можно было как $\operatorname{Re} A_{m,0}$. Благодаря тому, что мы выбрали первый способ, программа доходит лишь до $m-1$-го уровня и это немного сокращает время своей работы.

Заключение

В этом параграфе приведены наши основные результаты.

Результат 1. Я написал программу для выражения $\cos\frac{2\pi}{p}$ в квадратных радикалах в пакете *Mathematica*.

Я использовал пакет *Mathematica* по следующим причинам:

- Он позволяет работать со сложными выражениями, содержащими корни
- С его помощью можно производить расчёты с любой заранее заданной точностью
- Сложные программы на этом пакете имеют маленький код, так как:

– *Mathematica* использует функциональное программирование

– В *Mathematica* есть большое количество встроенных функций

Таблица 1. Результаты работы

p		**3**	**5**	**17**	**257**	**65 537**
Время работы в секундах		0.2187500	0.2656250	0.2812500	3.9062500	67.7500000
Количество операций в итоговом выражении	Сложений и вычитаний	0	1	23	24 191	27 253 971 763 199 999
	Делений	1	3	33	31 425	35 323 102 309 847 217
	Возведений в квадрат	0	0	4	3 616	4 034 565 273 323 608
	Квадратных корней	0	1	16	15 712	17 661 551 154 923 608
Количество операций в композиции	Сложений и вычитаний	0	1	10	188	86 548
	Делений	0	2	10	78	4 206
	Возведений в квадрат	0	0	3	37	2 101
	Квадратных корней	0	1	5	39	2 103
Количество $A_{u,v}$		0	1	7	127	32 767
Количество вычисленных $A_{u,v}$		0	1	5	39	2 103

Результат 2. Я опробовал свою программу для $p = 3, 5, 17, 257$ и $65\,537$, то есть для всех известных чисел Ферма, и всюду убедился в правильности ответов. В табл. 1 на с. 9 собраны результаты работы программы. Сами итоговые выражения и композиции находятся в приложении Г. Время работы указано для компьютера Intel Pentium 4 с частотой процессора 2.40 ГГц и оперативной памятью размером 512 МБ. А простых чисел Ферма, больших, чем $65\,537$ не найдено, и даже есть гипотеза, что их нет ([1]).

Программа обладает тем существенным преимуществом, что она находит композицию, которая короче самого выражения и важнее его. Под композицией подразумевается цепочка формул, в каждой из которых используются результаты предыдущих формул. В нашем случае формулы содержат только арифметические операции и квадратные корни. Отметим, что сборка выражений из часто встречающихся составных частей — один из распространённых приёмов при аналитических вычислениях. Она эффективна за счёт многократного использования ранее вычисленных выражений. Например, для $p = 65\,537$ композиция содержит 2103 квадратных корня, а итоговое выражение — $17\,661\,551\,154\,923\,608$.

Результат 3. С помощью своей программы я получил итоговое выражение для $p = 3, 5, 17$ и 257 и композицию для $p = 3, 5, 17, 257$ и $65\,537$ (см. приложение Г).

Результат 4. Я оценил времы работы программы ($O(2^m m^2 \ln m)$) и объём используемой ей памяти ($O(p^{\frac{\log_2 p - 3}{2}} \ln p)$, если не находить итоговое выражение, то $O(p)$).

Результат 5. Скорость моей программы была значительно увеличена за счёт того, что:

- Она выражает через радикалы только те числа вида $A_{u,v}$, которые обязательно присутствуют в итоговой формуле
- Она выражает их только по одному разу

- Она находит $\alpha(s)$ через $\beta(s)$

Без них программа работала бы не для всех p. Об этих сокращениях времени работы программы можно прочитать в параграфе 3.

Результат 6. В процессе работы я столкнулся со следующими сложностями, об успешном решении которых написано в параграфе 2:

- Было непонятно, какой из двух корней уравнения выбрать
- Для подсчётов не хватало машинной точности
- Итоговое выражение для $p = 65\,537$ не помещалось в память компьютера

Направление дальнейшей деятельности:

- Выразить $\cos\frac{2\pi}{n}$, если n не имеет вид $2^q p_1 p_2 p_3 \dots p_l$, через радикалы степеней, отличных от второй

A. Связь между построимостью правильных многоугольников и квадратичными иррациональностями

Прошу прощения, за то что привожу такие очевидные факты.

Лемма 4. Любую квадратичную иррациональность можно построить циркулем и линейкой.

Идейное резюме доказательства. Сложение, вычитание, умножение, деление и извлечение квадратного корня можно реализовать циркулем и линейкой. □

Доказательство. Достаточно доказать, что если a и b можно построить циркулем и линейкой, то $a + b$, $a - b$, ab, $\frac{a}{b}$, $\sqrt{a}$ тоже.

Для $a + b$ и $a - b$ это очевидно. Для умножения нужно построить фигуру на рис. 1 так, чтобы $OA = 1$, $OC = a$ и $AB = b$. Очевидно, что тогда $CD = ab$. Деление проводится с помощью этой же фигуры. Нужно лишь построить её так, чтобы $OA = b$, $OC = a$ и $AB = 1$. Очевидно, что тогда $CD = \frac{a}{b}$.

Чтобы извлечь корень, нужно построить фигуру на рис. 2 так, чтобы $AC = 1$ и $CB = a$. Очевидно, что тогда $CD = \sqrt{a}$. □

Лемма 5. Если можно построить правильные p_1, p_2, p_3, …, p_l-угольники, то можно построить и правильный $2^q p_1 p_2 p_3 \dots p_l$-угольник.

Идейное резюме доказательства. Применим $l - 1$ раз алгоритм Евклида для центральных углов многоугольников, потом q раз разделим полученный центральный угол пополам. □

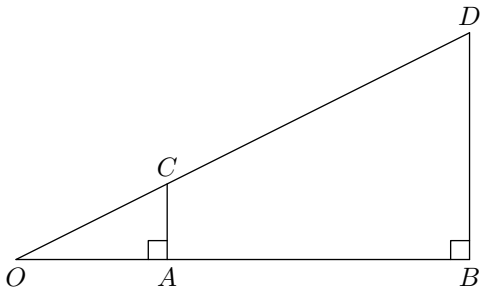

Рис. 1. Умножение и деление циркулем и линейкой

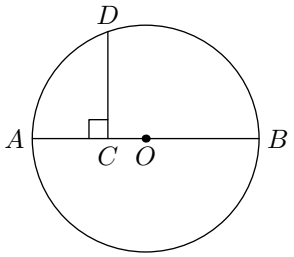

Рис. 2. Извлечение квадратного корня циркулем и линейкой

Доказательство. Если можно построить правильный p-угольник, то можно построить и его центральный угол, равный $\frac{2\pi}{p}$. По двум углам можно построить их разность, поэтому проведём алгоритм Евклида для углов $\frac{2\pi}{p_1}$ и $\frac{2\pi}{p_2}$. Так как p_1 и p_2 взаимно просты, то можно подобрать такие целые a и b, что $ap_1 + bp_2 = 1$, но нельзя подобрать такие целые a и b, чтобы $0 < ap_1 + bp_2 < 1$. Отсюда есть такие целые a и b, что

$$\frac{a}{p_2} + \frac{b}{p_1} = \frac{1}{p_1 p_2},$$

но нет таких, чтобы

$$0 < \frac{a}{p_2} + \frac{b}{p_1} < \frac{1}{p_1 p_2}.$$

Таким образом, мы построили угол, равный $\frac{2\pi}{p_1 p_2}$. Затем применим алгоритм Евклида к углам $\frac{2\pi}{p_1 p_2}$ и $\frac{2\pi}{p_3}$ и так далее. Наконец, разделим результат q раз пополам. □

Лемма 6. Если $2^m + 1$ — простое число, то существует такое целое неотрицательное r, что $m = 2^r$

Доказательство. Пусть m — не степень двойки. Тогда существуют такие натуральное a и нечётное $b \neq 1$, что $m = ab$. Но тогда $2^m + 1 = (2^a + 1)(2^{a(b-1)} - \ldots - 2^{3a} + 2^{2a} - 2^a + 1)$. Противоречие. □

Б. Текст программы без комментариев

```
createbutton[p_]:=Button[p,
time=AbsoluteTime[];
SelectionMove[ButtonNotebook[],All,ButtonCell];
SelectionMove[ButtonNotebook[],Next,CellGroup];
NotebookDelete[ButtonNotebook[]];
program::aT="aT[u,v0]-aT[u,v0+2^(u-1)]==0";
post[a_,b___]:=a[b];
program[m_]:=(
precision=100;
Clear[aA,aT,aR,aE];
g=PrimitiveRoot[2^m+1];
Do[
gin[i]=PowerMod[g,i,2^m+1];
logg[gin[i]]=i,
{i,0,2^m-1}
];
a[0,0]=aE[0,0]=-1;
aR[0,0]=N[-1,precision];
aA[0,0]=Null;
aC=Table[Null,{2^m}];
aEPMPrivate[0,0]=0;
aEDivisionPrivate[0,0]=0;
aESquarePrivate[0,0]=0;
aERootPrivate[0,0]=0;
aCPM=0;
aCDivision=0;
aCSquare=0;
aCRoot=0;
aT[u_,v_]:=(aT[u,v]=Sum[Cos@(N[2,10]Pi gin[2^ui+v])/(2^m+1),{i,0,2^(m-u)-1}]);
aA[u_,v_]:=(
aA[u-1,Mod[v,2^(u-1)]];
v0=Mod[v,2^(u-1)];
aEPMPrivate[u,v]=2aEPMPrivate[u-1,v0]+If[u!=1,1,0];
aEDivisionPrivate[u,v]=2aEDivisionPrivate[u-1,v0]+2;
aESquarePrivate[u,v]=2aESquarePrivate[u-1,v0]+If[u!=1,1,0];
aERootPrivate[u,v]=2aERootPrivate[u-1,v0]+1;
a1=gin[v0+2^(u-1)];
If[aT[u,v0]-aT[u,v0+2^(u-1)]==0,Message[program::aT]];
tab=(
aA[u-1,#[[1]]];
aEPMPrivate[u,v]=aEPMPrivate[u,v]+aEPMPrivate[u-1,#[[1]]]+1;
aEDivisionPrivate[u,v]=aEDivisionPrivate[u,v]+aEDivisionPrivate[u-1,#[[1]]];
aESquarePrivate[u,v]=aESquarePrivate[u,v]+aESquarePrivate[u-1,#[[1]]];
aERootPrivate[u,v]=aERootPrivate[u,v]+aERootPrivate[u-1,#[[1]]];
a[u-1,#[[1]]]Length@#
)&/@Split@Sort@Table[Mod[logg@Mod[gin[2^ui+v0]+a1,2^m+1],2^(u-1)],{i,0,2^(m-u)-1,2}];
v0=Mod[v,2^(u-1)];
b=a[u-1,v0]/2;
root=Sqrt[b^2-Plus@@tab];
aA[u,v]=Null;
aCPrivate=If[(aT[u,v0]<aT[u,v0+2^(u-1)])==(v==v0),b-root,b+root];
aCPM=aCPM+Plus@@(Length@#-1&/@Extract[aCPrivate,Position[aCPrivate,_Plus]]);
aCDivision=aCDivision+Count[aCPrivate,1/2,Infinity]+Count[aCPrivate,-1/2,Infinity]+
Count[aCPrivate,1/4,Infinity]-Count[aCPrivate,Power[_,1/2],Infinity];
aCSquare=aCSquare+Count[aCPrivate,Power[_,2],Infinity];
aCRoot=aCRoot+Count[aCPrivate,Power[_,1/2],Infinity];
aC[[aCRoot]]=Row[{Subscript[A, u,v],aCPrivate/.a[u1_,v1_]->Subscript[A, u1,v1]},"="];
aR[u,v]=N[1,precision]aCPrivate/.a->aR;
If[m<16,aE[u,v]=post[HoldForm,aCPrivate/.a->aE]]
);
aA[m-1,0];
aEPM=aEPMPrivate[m-1,0];
aEDivision=aEDivisionPrivate[m-1,0]+1;
aESquare=aESquarePrivate[m-1,0];
aERoot=aERootPrivate[m-1,0];
aC=Sort@Take[aC,aCRoot]
```

```
);
m=Log[2,p-1];
Print["p = "<>ToString[p]<>". Пожалуйста, подождите..."];
program[m];
Print["Время счёта:                "<>ToString[AbsoluteTime[]-time]<>" секунд"];
Print["Приближённое значение, полученное программой:"];
If[Precision@aR[m-1,0]>=1,Print[aR[m-1,0]/2],Print["ОШИБКА. В ответе нет ни одной значащей цифры"]];
Print["Косинус, подсчитанный другим образом:"];
Print[N[Cos@(2Pi)/p,precision]];
Print["Количество Subscript[A, u,v]:             "<>ToString[2^(m-1)-1]];
Print["Количество вычисленных Subscript[A, u,v]: "<>ToString@aCRoot];
Print["---- ИТОГОВОЕ ВЫРАЖЕНИЕ ----"];
If[m<16,Print@OpenerView@{"Итоговое выражение",aE[m-1,0]/2}];
Print["Сложений:                   "<>ToString@aEPM];
Print["Делений:                    "<>ToString@aEDivision];
Print["Возведений в квадрат:       "<>ToString@aESquare];
Print["Взятий корня:               "<>ToString@aERoot];
Print["---- КОМПОЗИЦИЯ ----"];
Print@OpenerView@{"Композиция",Column@aC};
Print["Сложений:                   "<>ToString@aCPM];
Print["Делений:                    "<>ToString@aCDivision];
Print["Возведений в квадрат:       "<>ToString@aCSquare];
Print["Взятий корня:               "<>ToString@aCRoot];
Print["Полное время:               "<>ToString[AbsoluteTime[]-time]<>" секунд"],
Method->"Queued"];
CellPrint@ExpressionCell[Row[Join[{Выберите,p},createbutton/@{3,5,17,257,65537}],"  "],"Section"]
```

B. Текст программы с комментариями

```
createbutton[p_]:=Button[p,
time=AbsoluteTime[];
SelectionMove[ButtonNotebook[],All,ButtonCell];
SelectionMove[ButtonNotebook[],Next,CellGroup];
NotebookDelete[ButtonNotebook[]];(*Этот код предназначен для генерации кнопки, к выражению косинуса
в радикалах отношения не имеет*)

program::aT="aT[u,v0]-aT[u,v0+2^(u-1)]==0";(*Этот код нужен для вывода сообщения об ошибке в случае
недостаточной точности*)

post[a_,b___]:=a[b]; (*Нужен для применения оператора к аргументам после их вычисления*)

program[m_]:=((*program --- это процедура, содержащая как бы всю программу в целом. К ней в качестве
параметра передаётся m (см. работу). Процедура создаёт функцию aE[u_, v_] (E --- сокращение от
Expression), возвращающую представление Subscript[A, u,v] через квадратные радикалы (см. работу),
aR[u_, v_] (Real), возвращающую вещественное приближение для Subscript[A, u,v]. Результаты
выполнения этих функций для всех возможных u и v записываются в память во время выполнения program и
при обращении к функциям они ничего не считают, а сразу выдают уже посчитанное значение, как будто
они не функции, а массивы. Также программа заносит в переменную aC (Constructor) список строк в
композиции, в aCPM (Plus, Minus) --- количество сложений и вычитаний в композиции, aCDivision ---
делений, aCSquare --- возведений в квадрат, aCRoot --- корней, aEPM, aEDivision, aESquare, aERoot
--- то же, но в итоговом выражении. При m >= 16 aR не работает, так как иначе память компьютера не
вместила бы ответ. Но aC, aR, aCPM, ..., aEPM, ... работают при m >= 16. Более того, при p == 65537
было проверено равенство aR[m - 1, 0]/2 == Cos[2\[Pi]/65537] (кстати, аналогичное было проверено для
всех возможных p), поэтому мы можем быть уверены, что и при p == 65537 программа работает верно.
aEPM и тому подобные считались во время выполнения программы, поэтому и были получены для p ==
65537*)

precision=100;(*Количество знаков после запятой в приближении для итогового выражения. Это
приближение нужно, чтобы сравнить его с косинусом, полученным другим образом*)

Clear[aA,aT,aR,aE];(*Этот оператор нужен, чтобы при повторном запуске program с другим m старые
значения переменных не вошли в противоречие с новыми. aA (All) --- это основная рекурентная
процедура. aT (Test) вычисляет Subscript[A, u,v] непосредственно по определению (см. работу). О том
для чего это нужно, см. работу. aA[u_, v_] делает следующее:
1. Выражает Subscript[A, u,v] через Subscript[A, u-1,0], Subscript[A, u-1,1], Subscript[A, u-1,2],
```

```
Subscript[A, u-1,3], ..., Subscript[A, u-1,2^(u-1)-1] и записывает результат в aCPrivate
2. Запускает те из aA[u - 1, 0], aA[u - 1, 1], aA[u - 1, 2], aA[u - 1, 3], ..., aA[u - 1, 2^(u-1) -
1] реккурентно, которые присутствуют в выражении
3. Присваивает aA[u, v] = Null, чтобы исключить повторный вызов
4. Подставляет полученные в реккурентно вызванных процедурах aE в выражение и записывает результат в
aE[u, v]
5. Подставляет полученные в реккурентно вызванных процедурах aR в выражение и записывает результат в
aR[u, v]
6. Вносит вклад в вычисление aC, aCPM, ..., aEPM, ...*)

g=PrimitiveRoot[2^m+1]; (*Что такое g, объяснено в работе*)

Do[
gin[i]=PowerMod[g,i,2^m+1];
logg[gin[i]]=i,
{i,0,2^m-1}
];(*logg --- это как бы логарифм по основанию g. В качестве аргумента к нему передаётся остаток по
модулю 2^m, в качестве ответа он выдаёт остаток по модулю 2^m+1. Строго говоря, для данного a он
выдаёт такое b, что g^b == a. В этом цикле считаются значения функции при всех возможных параметрах,
чтобы далее пользоваться этой функцией как массивом. gin, наоборот, предназначено для возведения g в
степень по модулю 2^m+1*)

a[0,0]=aE[0,0]=-1;(*a --- это просто заголовок, он не считается. Внутри программы aCPrivate выражено
через a, а затем вместо него подставляются aE или aR. a[0, 0] приравнивается к -1, чтобы, например,
для p == 17 aC[1, 0] возвращало -1/2+Sqrt[17]/2, а не 1/2a[0,0]+Sqrt[-4a[0,0]+1/4a[0,0]^2]*)

aR[0,0]=N[-1,precision]; (*Это нужно для правильной работы случая p == 3*)

aA[0,0]=Null;(*Это нужно, чтобы рекурсия имела конечную глубину*)

aC=Table[Null,{2^m}];(*Потом мы заполним этот массив*)

aEPMPrivate[0,0]=0;
aEDivisionPrivate[0,0]=0;
aESquarePrivate[0,0]=0;
aERootPrivate[0,0]=0;

aCPM=0;
aCDivision=0;
aCSquare=0;
aCRoot=0;

aT[u_,v_]:=(aT[u,v]=Sum[Cos@(N[2,10]Pi gin[2^ui+v])/(2^m+1),{i,0,2^(m-u)-1}]); (*Конструкция
aT[u_,v_]:=(aT[u,v]=...) представляет собой реализацию динамического программирования: после первого
вызова aT для некоторых u и v результат запоминается и используется при повторных вызовах*)

aA[u_,v_]:=(
aA[u-1,Mod[v,2^(u-1)]];(*Subscript[A, u-1,v0] == Subscript[A, u,v0] + Subscript[A, u,v0+2^(u-1)],
поэтому знание Subscript[A, u-1,v0] необходимо для вычисления Subscript[A, u,v]*)

v0=Mod[v,2^(u-1)];

aEPMPrivate[u,v]=2aEPMPrivate[u-1,v0]+If[u!=1,1,0];(*Subscript[A, u-1,v0] дважды встречается в
выражении + знак между -Subscript[A, u-1,v0]/2 и корнем. Если u == 1, то под корнем на один плюс
или минус меньше, поэтому в этом случае единицу не прибавляем*)
aEDivisionPrivate[u,v]=2aEDivisionPrivate[u-1,v0]+2;(*2 деления: оба отмечены здесь красным цветом:
-Subscript[A, u-1,v0]/2\[PlusMinus]Sqrt[(Subscript[A, u-1,v0]/2)^2-...]*)
aESquarePrivate[u,v]=2aESquarePrivate[u-1,v0]+If[u!=1,1,0];(*Если u == 1, то квадрата нет: он сразу
же вычисляется*)
aERootPrivate[u,v]=2aERootPrivate[u-1,v0]+1;

a1=gin[v0+2^(u-1)];(*Основой алгоритма является моделирование умножения
\[Epsilon]^g^v0+\[Epsilon]^g^(v0+2*2^u)+\[Epsilon]^g^(v0+4*2^u)+\[Epsilon]^g^(v0+6*2^u)+...+
\[Epsilon]^g^(v0+2^m-2*2^u) на g^(v0+2^(u-1)). Почему нужно моделировать именно это, написано в
работе. Так как все вычисления, какие можно, надо выносить из цикла, почитаем g^(v0+2^(u-1)) сразу и
запомним результат в a1. Сам цикл ниже*)

If[aT[u,v0]-aT[u,v0+2^(u-1)]==0,Message[program::aT]];(*Здесь происходит контроль точности*)
```

```
(*Это самая сложная и важная часть программы. Здесь мы нахадим \[Alpha](1), \[Alpha](g), \[Alpha](
g^2), \[Alpha](g^3), ..., \[Alpha](g^(2^(u-1)-1)). tab --- это {\[Alpha](1)a[u - 1, 0], \[Alpha](g)
a[u - 1, 1], \[Alpha](g^2)a[u - 1, 2], \[Alpha](g^3)a[u - 1, 3], ..., \[Alpha](g^(2^(u-1)-1))a[u -
1, 2^(u-1) - 1]}. Разбор этой части см. в конце*)
tab=(
aA[u-1,#[[1]]];
aEPMPrivate[u,v]=aEPMPrivate[u,v]+aEPMPrivate[u-1,#[[1]]]+1;
aEDivisionPrivate[u,v]=aEDivisionPrivate[u,v]+aEDivisionPrivate[u-1,#[[1]]];
aESquarePrivate[u,v]=aESquarePrivate[u,v]+aESquarePrivate[u-1,#[[1]]];
aERootPrivate[u,v]=aERootPrivate[u,v]+aERootPrivate[u-1,#[[1]]];
a[u-1,#[[1]]]Length@#
)&/@Split@Sort@Table[Mod[logg@Mod[gin[2^ui+v0]+a1,2^m+1],2^(u-1)],{i,0,2^(m-u)-1,2}];

v0=Mod[v,2^(u-1)];

b=a[u-1,v0]/2;(*b --- это b из формулы корней квадратного уравнения x^2+2bx+c=0: Subscript[x, 1,2]==
-b\[PlusMinus]Sqrt[b^2-c]*)

root=Sqrt[b^2-Plus@@tab];(*Это Sqrt[b^2-c] из приведённой выше формулы корней*)

aA[u,v]=Null;(*О необходимости этого оператора сказано выше*)

aCPrivate=If[(aT[u,v0]<aT[u,v0+2^(u-1)])==(v==v0),b-root,b+root];(*О выборе корня см. работу*)

aCPM=aCPM+Plus@@(Length@#-1&/@Extract[aCPrivate,Position[aCPrivate,_Plus]]);
aCDivision=aCDivision+Count[aCPrivate,1/2,Infinity]+Count[aCPrivate,-1/2,Infinity]+
Count[aCPrivate,1/4,Infinity]-Count[aCPrivate,Power[_,1/2],Infinity];
aCSquare=aCSquare+Count[aCPrivate,Power[_,2],Infinity];
aCRoot=aCRoot+Count[aCPrivate,Power[_,1/2],Infinity]; (*Количество операций в композиции*)

aC[[aCRoot]]=Row[{Subscript[A, u,v],aCPrivate/.a[u1_,v1_]->Subscript[A, u1,v1]},"="];(*Находим новую
строчку в композиции*)

aR[u,v]=N[1,precision]aCPrivate/.a->aR;(*Находим aR[u,v]*)

If[m<16,aE[u,v]=post[HoldForm,aCPrivate/.a->aE]](*Находим aE[u,v], если, конечно p меньше 65537.
Потом "замораживаем" его при помощи HoldForm, чтобы избежать лишних сокращений и сохранить
первозданный вид выражения*)
);

aA[m-1,0];

aEPM=aEPMPrivate[m-1,0];

aEDivision=aEDivisionPrivate[m-1,0]+1;(*Ответ равен Subscript[A, m-1,0]/2, поэтому добавляется одно
деление*)

aESquare=aESquarePrivate[m-1,0];

aERoot=aERootPrivate[m-1,0];

aC=Sort@Take[aC,aCRoot]
);
m=Log[2,p-1];
Print["p = "<>ToString[p]<>". Пожалуйста, подождите..."];
program[m];
Print["Время счёта:                "<>ToString[AbsoluteTime[]-time]<>" секунд"];
Print["Приближённое значение, полученное программой:"];
If[Precision@aR[m-1,0]>=1,Print[aR[m-1,0]/2],Print["ОШИБКА. В ответе нет ни одной значащей цифры"]];
Print["Косинус, подсчитанный другим образом:"];
Print[N[Cos@(2Pi)/p,precision]];
Print["Количество Subscript[A, u,v]:             "<>ToString[2^(m-1)-1]];
Print["Количество вычисленных Subscript[A, u,v]: "<>ToString@aCRoot];(*Оно совпадает с количеством
строк в алгоритме сборки, то есть с количеством корней в ней*)
Print["---- ИТОГОВОЕ ВЫРАЖЕНИЕ ----"];
If[m<16,Print@OpenerView@{"Итоговое выражение",aE[m-1,0]/2}];(*OpenerView нужен для того, чтобы
скрыть итоговое выражение, так как если оно велико, попытка вывести его на экран может привести к
```

```
зависанию компьютера*)
Print["Сложений:                    "<>ToString@aEPM];
Print["Делений:                     "<>ToString@aEDivision];
Print["Возведений в квадрат:        "<>ToString@aESquare];
Print["Взятий корня:                "<>ToString@aERoot];
Print["---- КОМПОЗИЦИЯ ----"];
Print@OpenerView@{"Композиция",Column@aC};
Print["Сложений:                    "<>ToString@aCPM];
Print["Делений:                     "<>ToString@aCDivision];
Print["Возведений в квадрат:        "<>ToString@aCSquare];
Print["Взятий корня:                "<>ToString@aCRoot];
Print["Полное время:                "<>ToString[AbsoluteTime[]-time]<>" секунд"],

Method->"Queued"];
CellPrint@ExpressionCell[Row[Join[{Выберите,p},createbutton/@{3,5,17,257,65537}],"  "],"Section"]
(*Эти строчки касаются генерации кнопки*)
```

Г. Выражения и композиции

$$p = 3$$

Композиция: пуста.
Итоговое выражение: $-\frac{1}{2}$.

$$p = 5$$

Композиция:

$$A_{1,0} = -\frac{1}{2} + \frac{\sqrt{5}}{2}.$$

Итоговое выражение: $\frac{1}{2}\left(-\frac{1}{2} + \frac{\sqrt{5}}{2}\right)$.

$$p = 17$$

Композиция:

$$A_{1,0} = -\frac{1}{2} + \frac{\sqrt{17}}{2};$$
$$A_{1,1} = -\frac{1}{2} - \frac{\sqrt{17}}{2};$$
$$A_{2,0} = \frac{1}{2}A_{1,0} + \sqrt{-A_{1,0} + \frac{1}{4}A_{1,0}^2 - A_{1,1}};$$
$$A_{2,1} = \frac{1}{2}A_{1,1} + \sqrt{-A_{1,0} - A_{1,1} + \frac{1}{4}A_{1,1}^2};$$
$$A_{3,0} = \frac{1}{2}A_{2,0} + \sqrt{\frac{1}{4}A_{2,0}^2 - A_{2,1}}.$$

Итоговое выражение:

$$\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\left(-\frac{1}{2}+\frac{\sqrt{17}}{2}\right)+\sqrt{1+\frac{1}{4}\left(-\frac{1}{2}+\frac{\sqrt{17}}{2}\right)^2}\right)+\right.$$

$$\left.+\sqrt{\frac{1}{2}\left(\frac{1}{2}+\frac{\sqrt{17}}{2}\right)-\sqrt{1+\frac{1}{4}\left(-\frac{1}{2}-\frac{\sqrt{17}}{2}\right)^2}+\frac{1}{4}\left(\frac{1}{2}\left(-\frac{1}{2}+\frac{\sqrt{17}}{2}\right)+\sqrt{1+\frac{1}{4}\left(-\frac{1}{2}+\frac{\sqrt{17}}{2}\right)^2}\right)^2}\right).$$

$$p = 257$$

Композиция:

$$A_{1,0} = -\frac{1}{2}+\frac{\sqrt{257}}{2};$$

$$A_{1,1} = -\frac{1}{2}-\frac{\sqrt{257}}{2};$$

$$A_{2,0} = \frac{1}{2}A_{1,0}+\sqrt{-16A_{1,0}+\frac{1}{4}A_{1,0}^2-16A_{1,1}};$$

$$A_{2,1} = \frac{1}{2}A_{1,1}+\sqrt{-16A_{1,0}-16A_{1,1}+\frac{1}{4}A_{1,1}^2};$$

$$A_{2,2} = \frac{1}{2}A_{1,0}-\sqrt{-16A_{1,0}+\frac{1}{4}A_{1,0}^2-16A_{1,1}};$$

$$A_{2,3} = \frac{1}{2}A_{1,1}-\sqrt{-16A_{1,0}-16A_{1,1}+\frac{1}{4}A_{1,1}^2};$$

$$A_{3,0} = \frac{1}{2}A_{2,0}+\sqrt{-2A_{2,0}+\frac{1}{4}A_{2,0}^2-5A_{2,1}-4A_{2,2}-5A_{2,3}};$$

$$A_{3,1} = \frac{1}{2}A_{2,1}-\sqrt{-5A_{2,0}-2A_{2,1}+\frac{1}{4}A_{2,1}^2-5A_{2,2}-4A_{2,3}};$$

$$A_{3,2} = \frac{1}{2}A_{2,2}+\sqrt{-4A_{2,0}-5A_{2,1}-2A_{2,2}+\frac{1}{4}A_{2,2}^2-5A_{2,3}};$$

$$A_{3,3} = \frac{1}{2}A_{2,3}-\sqrt{-5A_{2,0}-4A_{2,1}-5A_{2,2}-2A_{2,3}+\frac{1}{4}A_{2,3}^2};$$

$$A_{3,4} = \frac{1}{2}A_{2,0}-\sqrt{-2A_{2,0}+\frac{1}{4}A_{2,0}^2-5A_{2,1}-4A_{2,2}-5A_{2,3}};$$

$$A_{3,5} = \frac{1}{2}A_{2,1}+\sqrt{-5A_{2,0}-2A_{2,1}+\frac{1}{4}A_{2,1}^2-5A_{2,2}-4A_{2,3}};$$

$$A_{3,6} = \frac{1}{2}A_{2,2}-\sqrt{-4A_{2,0}-5A_{2,1}-2A_{2,2}+\frac{1}{4}A_{2,2}^2-5A_{2,3}};$$

$$A_{3,7} = \frac{1}{2}A_{2,3}+\sqrt{-5A_{2,0}-4A_{2,1}-5A_{2,2}-2A_{2,3}+\frac{1}{4}A_{2,3}^2};$$

$$A_{4,0} = \frac{1}{2}A_{3,0}+\sqrt{-2A_{3,0}+\frac{1}{4}A_{3,0}^2-2A_{3,2}-A_{3,4}-2A_{3,5}-A_{3,6}};$$

$$A_{4,1} = \frac{1}{2}A_{3,1} + \sqrt{-2A_{3,1} + \frac{1}{4}A_{3,1}^2 - 2A_{3,3} - A_{3,5} - 2A_{3,6} - A_{3,7}};$$

$$A_{4,2} = \frac{1}{2}A_{3,2} + \sqrt{-A_{3,0} - 2A_{3,2} + \frac{1}{4}A_{3,2}^2 - 2A_{3,4} - A_{3,6} - 2A_{3,7}};$$

$$A_{4,3} = \frac{1}{2}A_{3,3} + \sqrt{-2A_{3,0} - A_{3,1} - 2A_{3,3} + \frac{1}{4}A_{3,3}^2 - 2A_{3,5} - A_{3,7}};$$

$$A_{4,4} = \frac{1}{2}A_{3,4} + \sqrt{-A_{3,0} - 2A_{3,1} - A_{3,2} - 2A_{3,4} + \frac{1}{4}A_{3,4}^2 - 2A_{3,6}};$$

$$A_{4,5} = \frac{1}{2}A_{3,5} + \sqrt{-A_{3,1} - 2A_{3,2} - A_{3,3} - 2A_{3,5} + \frac{1}{4}A_{3,5}^2 - 2A_{3,7}};$$

$$A_{4,6} = \frac{1}{2}A_{3,6} - \sqrt{-2A_{3,0} - A_{3,2} - 2A_{3,3} - A_{3,4} - 2A_{3,6} + \frac{1}{4}A_{3,6}^2};$$

$$A_{4,7} = \frac{1}{2}A_{3,7} + \sqrt{-2A_{3,1} - A_{3,3} - 2A_{3,4} - A_{3,5} - 2A_{3,7} + \frac{1}{4}A_{3,7}^2};$$

$$A_{4,8} = \frac{1}{2}A_{3,0} - \sqrt{-2A_{3,0} + \frac{1}{4}A_{3,0}^2 - 2A_{3,2} - A_{3,4} - 2A_{3,5} - A_{3,6}};$$

$$A_{4,9} = \frac{1}{2}A_{3,1} - \sqrt{-2A_{3,1} + \frac{1}{4}A_{3,1}^2 - 2A_{3,3} - A_{3,5} - 2A_{3,6} - A_{3,7}};$$

$$A_{4,10} = \frac{1}{2}A_{3,2} - \sqrt{-A_{3,0} - 2A_{3,2} + \frac{1}{4}A_{3,2}^2 - 2A_{3,4} - A_{3,6} - 2A_{3,7}};$$

$$A_{4,11} = \frac{1}{2}A_{3,3} - \sqrt{-2A_{3,0} - A_{3,1} - 2A_{3,3} + \frac{1}{4}A_{3,3}^2 - 2A_{3,5} - A_{3,7}};$$

$$A_{4,12} = \frac{1}{2}A_{3,4} - \sqrt{-A_{3,0} - 2A_{3,1} - A_{3,2} - 2A_{3,4} + \frac{1}{4}A_{3,4}^2 - 2A_{3,6}};$$

$$A_{4,13} = \frac{1}{2}A_{3,5} - \sqrt{-A_{3,1} - 2A_{3,2} - A_{3,3} - 2A_{3,5} + \frac{1}{4}A_{3,5}^2 - 2A_{3,7}};$$

$$A_{4,14} = \frac{1}{2}A_{3,6} + \sqrt{-2A_{3,0} - A_{3,2} - 2A_{3,3} - A_{3,4} - 2A_{3,6} + \frac{1}{4}A_{3,6}^2};$$

$$A_{4,15} = \frac{1}{2}A_{3,7} - \sqrt{-2A_{3,1} - A_{3,3} - 2A_{3,4} - A_{3,5} - 2A_{3,7} + \frac{1}{4}A_{3,7}^2};$$

$$A_{5,0} = \frac{1}{2}A_{4,0} + \sqrt{-A_{4,0} + \frac{1}{4}A_{4,0}^2 - A_{4,1} - A_{4,2} - A_{4,5}};$$

$$A_{5,1} = \frac{1}{2}A_{4,1} + \sqrt{-A_{4,1} + \frac{1}{4}A_{4,1}^2 - A_{4,2} - A_{4,3} - A_{4,6}};$$

$$A_{5,15} = \frac{1}{2}A_{4,15} + \sqrt{-A_{4,0} - A_{4,1} - A_{4,4} - A_{4,15} + \frac{1}{4}A_{4,15}^2};$$

$$A_{5,23} = \frac{1}{2}A_{4,7} + \sqrt{-A_{4,7} + \frac{1}{4}A_{4,7}^2 - A_{4,8} - A_{4,9} - A_{4,12}};$$

$$A_{5,24} = \frac{1}{2}A_{4,8} + \sqrt{-A_{4,8} + \frac{1}{4}A_{4,8}^2 - A_{4,9} - A_{4,10} - A_{4,13}};$$

$$A_{5,25} = \frac{1}{2}A_{4,9} + \sqrt{-A_{4,9} + \frac{1}{4}A_{4,9}^2 - A_{4,10} - A_{4,11} - A_{4,14}};$$

$$A_{6,0} = \frac{1}{2}A_{5,0} + \sqrt{\frac{1}{4}A_{5,0}^2 - A_{5,1} - A_{5,23}};$$

$$A_{6,56} = \frac{1}{2}A_{5,24} + \sqrt{-A_{5,15} + \frac{1}{4}A_{5,24}^2 - A_{5,25}};$$

$$A_{7,0} = \frac{1}{2}A_{6,0} + \sqrt{\frac{1}{4}A_{6,0}^2 - A_{6,56}}.$$

Итоговое выражение: слишком велико, чтобы быть помещённым сюда.

$$p = 65\,537$$

Композиция: слишком велика, чтобы быть помещённой сюда.

Итоговое выражение: не было получено, так как не помещается в памяти компьютера. Но поставленная задача всё равно решена для $p = 65\,537$ (см. параграф 2.3).

Литература

[1] Brent R. P. *Integer Factorization Algorithms Illustrated by the Factorization of Fermat Numbers*. Computer Sciences Laboratory, Australian National University, presented at CANT'97, Sydney, 5 December 1997.

[2] Козлов П. Ю., Скопенков А. Б. *В поисках утраченной алгебры: в направлении Гаусса (подборка задач).*
`http://arxiv.org/abs/0804.4357v1`

[3] Гаусс К. Ф. *Арифметические исследования* // Труды по теории чисел. М.: Изд-во АН СССР, 1959.

[4] Гиндикин С. *Дебют Гаусса* // Квант, №1, 1972.

[5] Кириллов А. А. *О правильных многоугольниках, функции Эйлера и числах Ферма* // Квант, №7, 1977. Квант, №6, 1994.