**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и информационных технологий

Кафедра вычислительной математики и программирования

**Отчет по курсовому проекту**

по курсу «Численные Методы»

Студент: Иларионов Д.А.

Группа: М8О-308Б

Преподаватель: Сластушенский Ю.В.

**Курсовая работа**

1. Тема КР:

Решение нелинейных уравнений методом продолжения по параметру.

1. Вариант : **5**
2. Алгоритм:

Когда реализация в разы сложнее самого алгоритма.. Данный алгоритм я сам не смог найти, когда я искал его, мне вылезало много непонятный сайтов с диффурами, ни одного теста или хотя бы шагов алгоритма. Я отчаялся и попросил преподавателя отправить какую-нибудь информацию для данной темы по КП. Он мне помог и кинул пару сайтов, которые я нигде не видел, но там все описано очень хорошо и подробно. На вход принимается функция, начальное приближение и интервал параметра. Решается уравнение методом Ньютона (из приближения вычитаем частное значения функции и значения производной). Оно решается так несколько раз. Каждый раз в виде приближения устанавливается предыдущее решение функции, а в качестве параметра – следующий с шагом. Например приближение у нас = x = 300, а параметр с 1 до 5 с шагом 0.5. Мы решаем методом Ньютона с приближением 300 с параметром 1. Допустим, у нас вышло 324. Тогда мы решаем уже методом Ньютона с приближением 324 с параметром 1.5. Вышло 368 например, решаем с приближением 368 с параметром 2. И так далее, до того, как параметр не станет равным 5. Потом просто строим график по получившимся значениям. Довольно простой алгоритм, но на данную курсовую у меня ушло около недели работы. А дело не в алгоритме. А дело, в его реализации. (Хотя если бы я делал его в питоне, где уже есть библиотеки и функции работы с функциями, вычисления производной, то у меня ушло всего около дня, если не меньше. Но я не уверен, что так можно было бы, и мне казалось это слишком просто).

1. Среда разработки:

Adobe Animate CC – язык – JavaScript (Canvas)

Ссылка на проект на Github: <https://denisolenison.github.io/numeric_methods_KP/>

Ссылка на исх. код на Github: <https://github.com/denisolenison/numeric_methods_KP>

1. Реализация

Это сущий ужас. Вот серьезно. И дело не в реализации алгоритма самого. А сложное – это было создать интерфейс, структуру функций. А самое сложное – вычисление производной по этой функции! Казалось бы, что это просто долго. Но я так сильно уже запутался в собственных структурах, что очень часто возникали баги, например, что я не в ту структуру зашел, и очень много часов уходило на то, чтобы ловить их. Самое сложное – сделать нормальные выходы из функций после генерации из нее производной. Многие значения передаются по ссылке, из-за чего приходилось делать их копии. К счастью, в JavaScript вроде есть автоматический сборщик мусора, так что хоть с проблемами с освобождением памяти не пришлось сталкиваться, иначе, это был бы ад просто. Теперь о самой реализации. Есть структура формулы. И ссылка на всю формулу и на текущее место. В формуле могут быть объекты 6 типов: число, оператор, возведение в степень, корень, параметр и функция. У всех этих структур есть общий параметр ist, по которому можно понять, какая у нас структура. Число имеет тип, это может быть либо константа pi или число эйлера (для более красивого отображения), либо number и произвольное число. Число вводится в отдельном поле, а потом мы его добавляем нажимая на N, либо выбираем функции, связанные с данным числом. Оператор простой тип, в котором просто в его типе указывается оператор символьно. Возведение в степень возводит предыдущий элемент в определенную степень, аргументом является формула. Параметр и корень тоже простые типы. Функция – самый сложный тип. Во-первых, задается то, что внутри функции. Это ссылка на другую формулу. Когда мы выбираем функцию, то мы переходим в нее. Когда мы берем закрывающую скобку, выходим обратно. Скобки тоже представлены в виде функций. Также у некоторых функций есть еще аргумент – дополнительная формула, которая, например у логарифма определяет основание, либо у корня степень, либо у показательной функции число/параметр, которое мы возводим в степень. В то время, как мы возводим что-то в степень параметра или числа – является типом степени, то показательная функция, когда мы возводим само число или параметр в степень с формулой – является функцией. И из каждой формулы есть выход – ссылка на предыдущий массив. У начальной формулы он равен null. Например у нас функция cos(x + p) – sin(p – x) = 0. То в функции косинус у нас аргумента нет, формула примерно formula([root, operator(‘+’), param]), а выход – formula([func(cos), operator('-'), func(sin)]). На массивы указывает атрибут inner, на выход – outer. Более того, в функциях, inner указывает на формулу, и в формуле inner указывает на массив. Именно из-за этого и была основная путаница, я думаю, что данные атрибуты надо было назвать по-разному, потому что при копировании часто неправильно копировалось, например, вместо формулы массив и просто нужные атрибуты были равны null. И я очень долго сидел, чтобы понять проблему. Вот я и реализовал свою собственную программу построения функций. Дальше еще сложнее – производные. Мы разбиваем массив начальной формулы на части, между которыми стоят операторы ‘+’ и ‘-‘. Также в частях мы заменяем все операторы ‘/’ на ‘\*’, а следующий объект возводим в -1 степень. Сокращаем степени в одну. У нас примерно получается массив [объект, степень, оператор(\*), объект, степень, оператор(\*), …. объект, степень]. Рекурсивно вычисляем производную произведения, как [объект, степень] и [объект, степень, оператор(\*), …. объект, степень]. Если правая часть = [], то мы просто вычисляем производную левой части. Таким образом ответом будет (произв л.ч)\*(пр.ч) + (произв пр.ч)\*(л.ч). И так рекурсивно. Затем для следующей части, и так далее, пока не конец формулы. Для функций делаем так. (степень функции)\*(функция^(ст-1))\*(произв. функции с той же внутр. формулой)\*(произв. формулы в функции). Последнее мы так же запускаем рекурсивно. Производная функции определяется в отдельной процедуре и зависит от типа функции. Для скобок она просто равна 1. Эта производная нужна нам для метода Ньютона, однако я сделал кнопку, которая просто преобразует формулу в производную в окошке. Но тут до сих пор некоторые проблемы с выходами при удалении поэлементно, лучше в таком образе формулу удалить целиком. Как это исправить – не знаю, но скорее всего решение окажется простым, потому что я уже сильно запутался в своих же структурах, а дебажить это довольно тяжело и занимает много времени. Также есть процедура сокращения. Например, 2+3 преобразуется в 5. x^0 в 1. Это сделано для меньшей громоздкости формул, но это все еще работает не идеально. Например, производные в итоге получаются очень громоздкими, но зато правильными. Есть еще кое-что, но писать об этом я не буду в отчет)). Функция подставления значений в формулу работает так. Сначала подставляются X и P. Потом рекурсивно вычисляются функции. Затем возведение в степень. Затем умножение/деление. Затем сложение/вычитание. Все как в правилах порядка в математике. Далее уже пользователь выбирает начальное приближение, интервал параметра и количество продолжений. Нажимает на кнопку решить. Идет вычисление для первого приближения. Если значение получилось = NaN (обычно при комплексных числах), +∞. -∞, или метод Ньютона выполняется очень много раз, но разница между значениями постоянно большая, то это значит, что нет решений. И выводится сообщений, что решений для такого интервала параметра, приближения или просто формулы не найдено. Очень редко бывает это ошибочным, но почему-то часто зависит от начального приближение. Думаю, тут проблема уже в самом методе Ньютона. Если все ОК, то уже дальше идет алгоритм продолжения по параметру и выводится график (с осями параметра и значения корня X). График строится поточечно и соединяется линиями, количество точек зависит от количества продолжений. График можно в любое время закрыть и написать другую функцию или выбрать другой интервал параметров/приближение, либо и то и то. Можно еще много чего написать, но я думаю, чтобы лучше понять реализацию, нужно посмотреть на сам код.

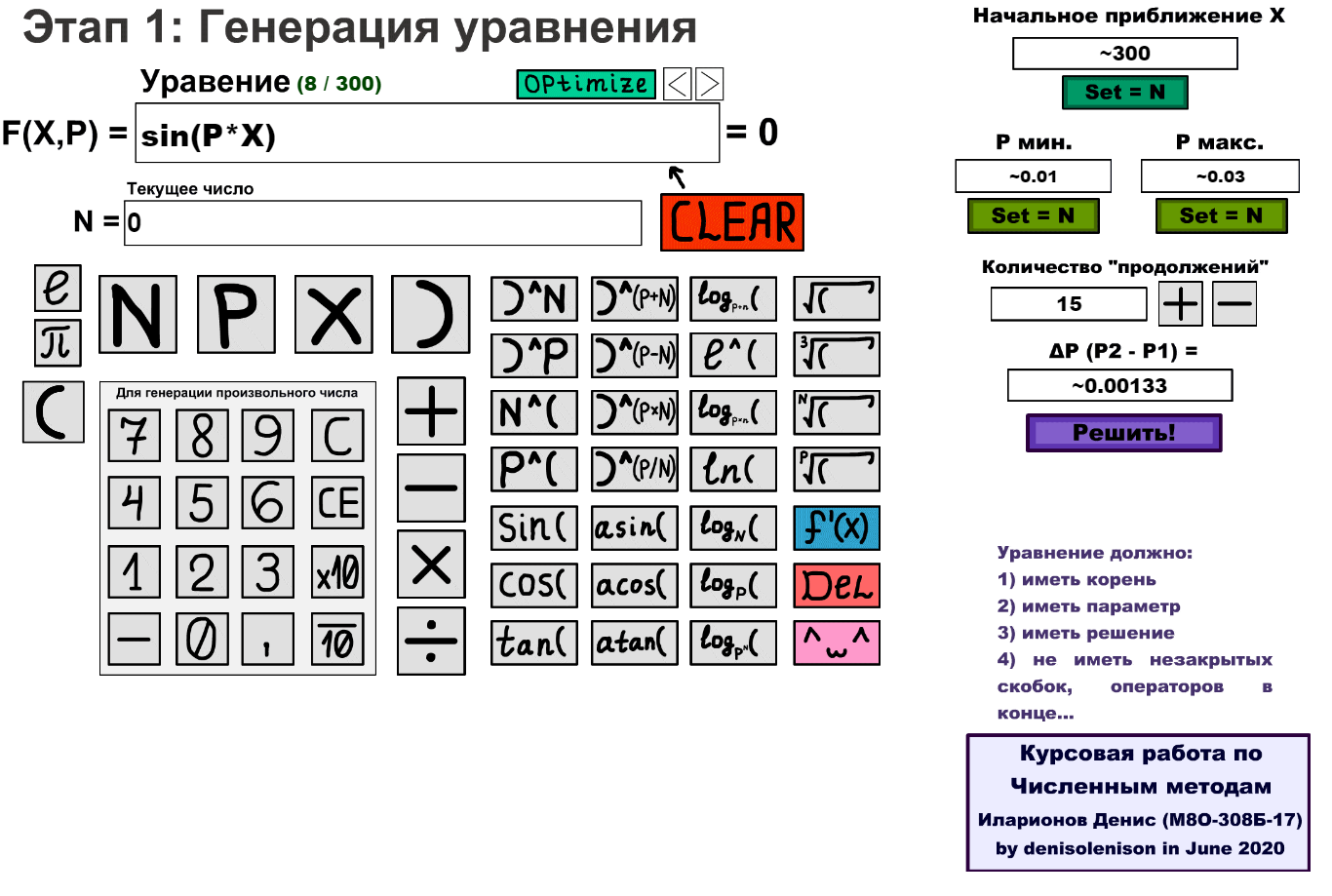
1. Руководство для пользователя

Число вводится в сером окошке слева. Оно не вводится сразу в формулу, а вводится в отдельные переменные. При нажатии на кнопку N оно преобразуется и добавляется в формулу. Также при выборе функции или степени, где нарисована буква N, функция добавляется с этим числом. Можно число умножить или разделить на 10. Кнопка C стирает 1 цифру или точку числа. Кнопка CE стирает число полностью. Чуть правее – операторы +, -, \* , /. Их можно ставить только после числа, корня, параметра, степени или функции. Два оператора подряд ставить нельзя. Сверху кнопки N – добавляет число в формулу. P – добавляет параметр. X – добавляет корень. Открывающая и закрывающая скобки – для нужного приоритета операций. Слева еще кнопки e, pi. Они меняют число на эту константу и число больше не вводится. Чтобы снова можно было самому вводить число, нужно нажать на эту же кнопку еще раз. Справа – функции. Некоторые – с аргументом, который чаще всего представляет параметр, число, или формулу из них (типо p\*x). Кнопка clear очищает полностью формулу. Кнопка Del удаляет один объект из формулы. Кнопка f’(x) преобразует формулу в ее производную и меняет ее. Что делает розовая кнопка – писать я сюда не буду)). Кнопка optimize сверху сокращает некоторые части формулы. Справа задаем начальное приближение. Все значение справа устанавливаются равными N, тому числу, которое мы вводили. Сначала задаем Pmax, а потом Pmin. Либо наоборот, если числа отрицательные. Но мы не можем задать Pmin > Pmax, либо Pmax < Pmin. Устанавливаем количество продолжений кнопками. Нажимаем решить – и опа, получаем ошибку. Ну или если все норм, то появляется новое окошко с графиком. Можем закрыть его и еще поэкспериментировать. Думаю, это все.

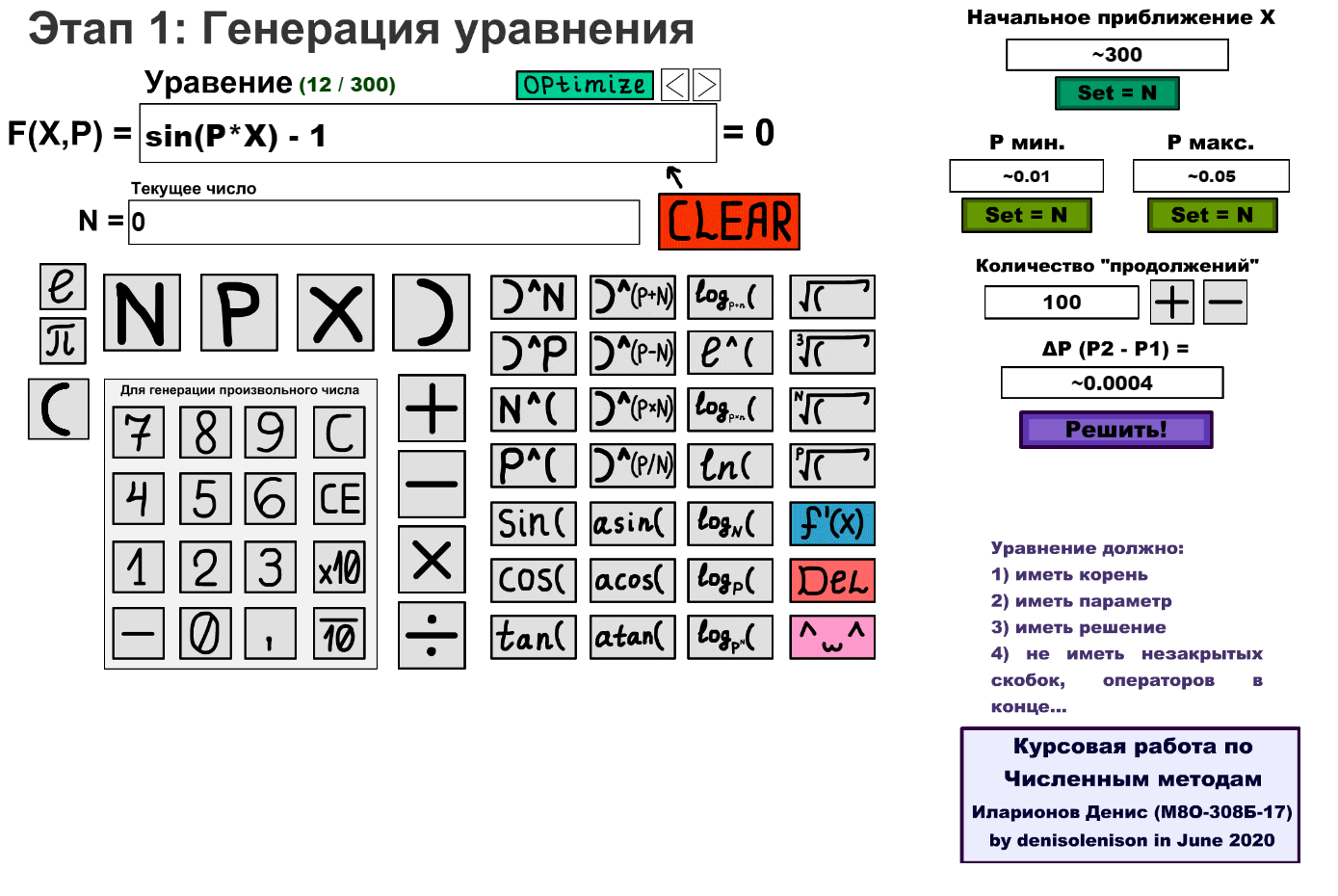
1. Тесты (скриншоты из программы с разными функциями):

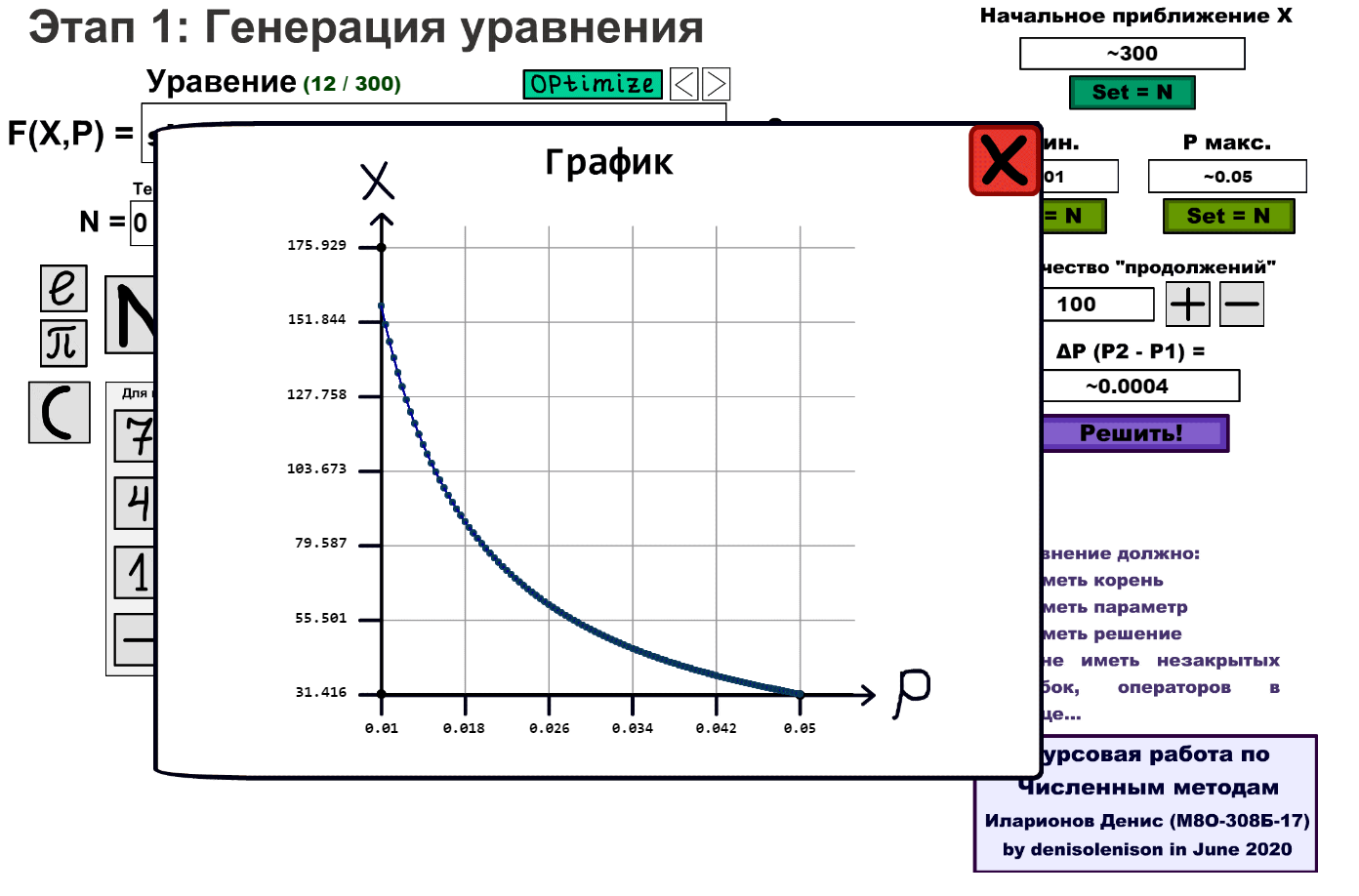
Также, некоторые тесты содержатся в видео, по данной ссылке:

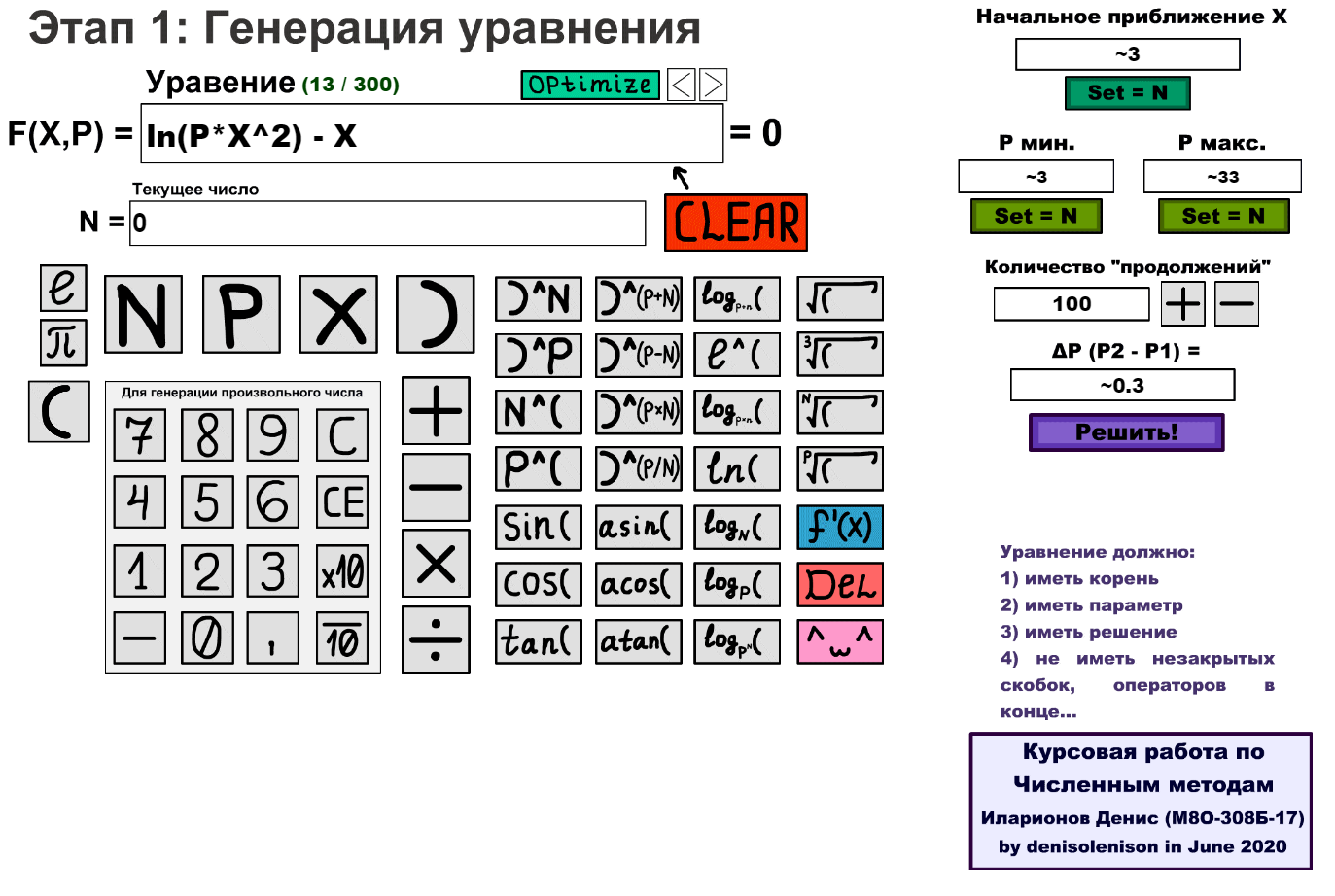
<https://www.youtube.com/watch?v=I6CsHvRKrmU>

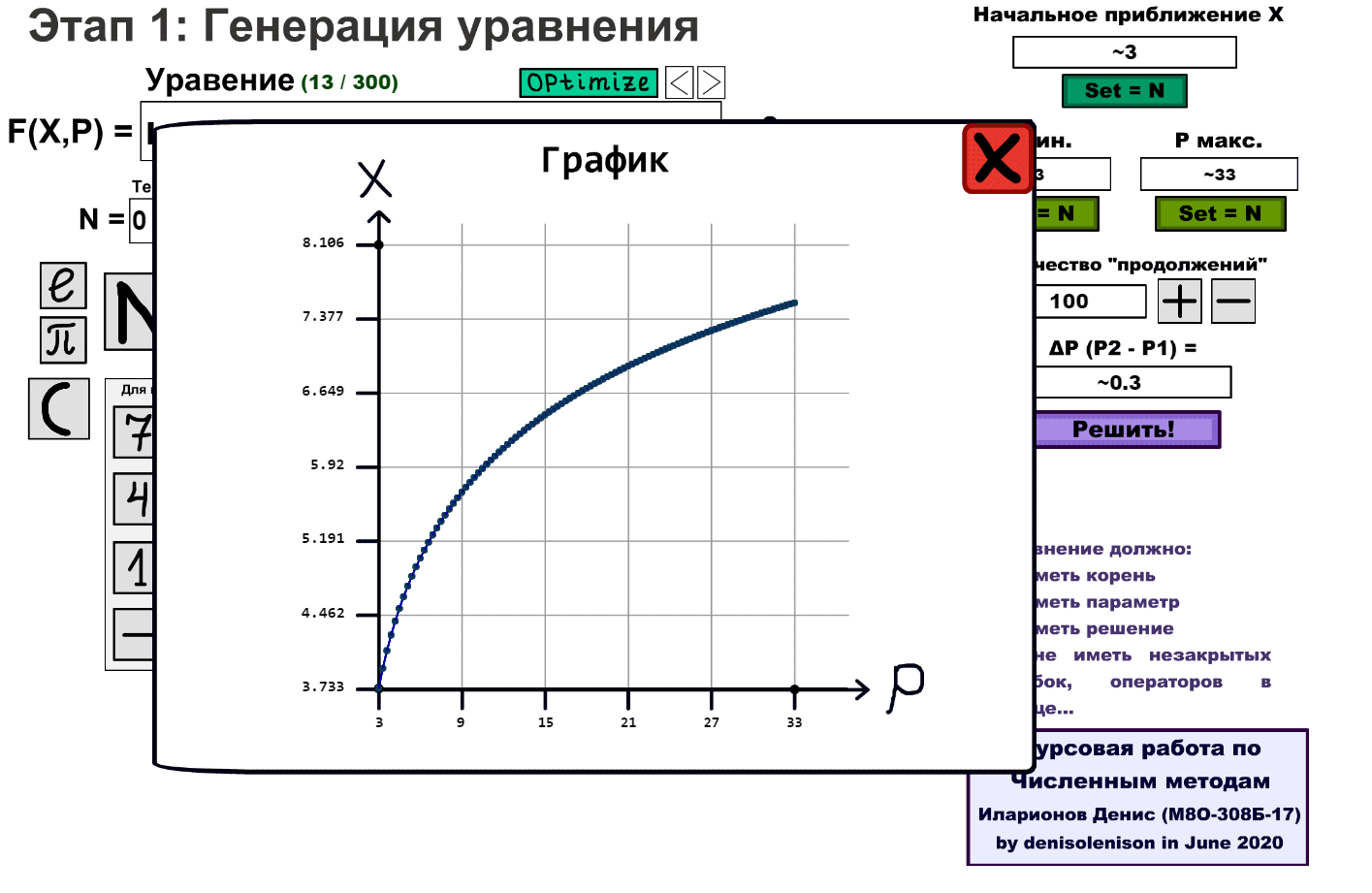


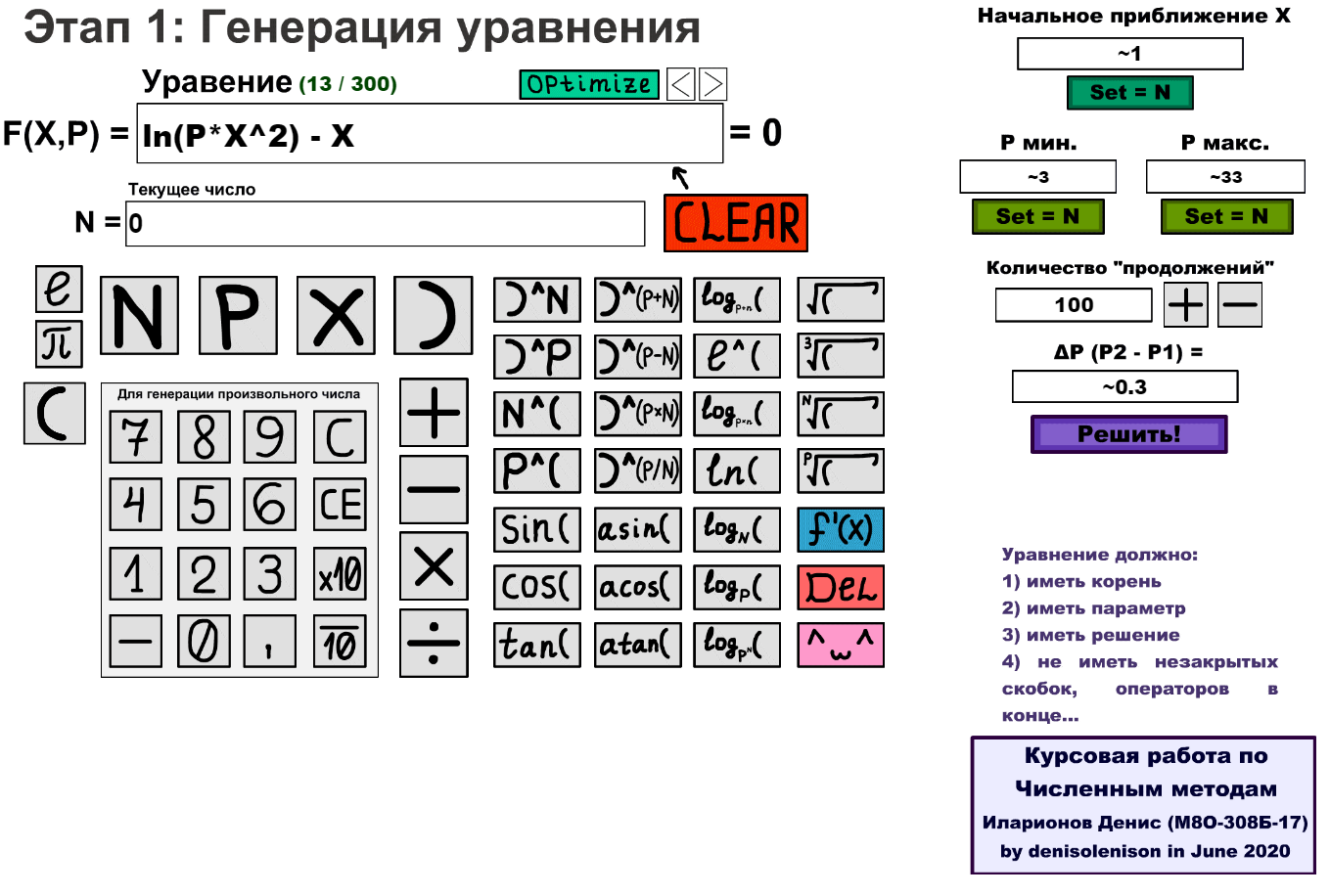














1. Данный курсовой проект выполнен: **19 июня 2020.**
2. Вывод (он будет коротким) – убил немало нервов, когда дебажил баги, связанные со структурами и выходами из формул. Зато, я укрепил некоторые навыки ООП, работу со структурами, ссылками, вложенными структурами, определенными видами деревьев, и рекурсивными функциями. Реализовал собственную программу представления функций, сделал сам вычисление производных любой функции. А также выполнил саму курсовую работу, реализовав алгоритм “продолжения по параметру” уже используя собственную программу и структуры. Я бы мог еще иногда обновлять данную программу, улучшая алгоритм сокращения функций, но пока не уверен, понадобится ли она еще где-нибудь, кроме данного курсового проекта. Зато еще укрепил некоторые навыки работы в среде разработки, стал более внимательным, и получил некоторый опыт, осознал, что лучше изначально представить структуры и классы простыми, реализовать несколько простых функций работы с ними и потом использовать уже их, чем постоянно переписывать одно и то же и путаться. Так правда удобнее, понадобится немного больше времени в начале, но зато потом уйдет намного меньше времени и будет меньше проблем. Полученный опыт мне еще пригодится, так как я и сам нередко работаю в данной среде разработки.
3. Код (JavaScript) – его очень много, около 3к строк. Тут содержится только тот код, который я писал сам. Код из файла index.js, который сгенерировался сам программой для создания интерфейса программы тут не учитывается.
4. *//Число N*
5. **var** custN = 0;
6. **var** custNafter = 0;
7. **var** hasPoint = **false**;
8. **var** vedZeros = 0;
9. **var** hasMinus = **false**;
11. **var** isE = **false**;
12. **var** isPI = **false**;

15. *//Формула*
16. **var** formula = **new** theFormula([], **null**);
18. *//Текущее место*
19. **var** curPlace = formula;

22. **var** graphOpen = **false**;

25. **this**.err2.visible = **false**;


29. **var** xStar = 0;
30. **var** pMin = 0;
31. **var** pMax = 1;
33. **var** conts = 10;
35. **var** dP = 0.1;

38. **var** MAX\_SIGNS = 32;
39. **var** fShift = 0;
40. **var** signsNow = 0;

43. window.setInterval(checkFps, 1000);
45. **function** sound(src) {
46. **this**.sound = document.createElement("audio");
47. **this**.sound.src = src;
48. **this**.sound.setAttribute("preload", "auto");
49. **this**.sound.setAttribute("controls", "none");
50. **this**.sound.style.display = "none";
51. document.body.appendChild(**this**.sound);
52. **this**.play = **function** () {
53. **this**.sound.play();
54. }
55. **this**.pause = **function** () {
56. **this**.sound.pause();
57. }
58. **this**.stop = **function** () {
59. **this**.sound.pause();
60. **this**.sound.currentTime = 0;
61. }
62. }
64. **function** checkFps() {
65. fps = Math.max(1, fps2);
66. fps2 = 0;
67. }

70. **function** theFormula(inner, outer) {
71. **this**.inner = inner;
72. **this**.outer = outer; *//чтобы выйти из формулы*
73. }

76. *//+ - \* /*
77. **function** operator(type) {
78. **this**.ist = "operator";
79. **this**.type = type;
80. }
82. *//cos, sin, tg, ctg...*
83. **function** funct(type, arg, outer) {
84. **this**.ist = "function";
85. **this**.type = type;
86. **this**.inner = **new** theFormula([], outer);
87. **this**.arg = arg;
88. **this**.exit = **false**;
89. }
91. **function** makeBracket(inner, outer) {
92. **var** newBr = **new** funct("bracket", 0, outer);
93. newBr.inner.inner = inner;
94. newBr.exit = **true**;
95. **return** newBr;
96. }
98. **function** root() {
99. **this**.ist = "root";
100. }
102. **function** par() {
103. **this**.ist = "param";
104. }
106. *//pi / e ...*
107. **function** numb(typ, value) {
108. **this**.ist = "numb";
109. **this**.type = typ;
110. **this**.value = value;
111. }
113. *//X^n - тип power, но n^(...) - тип function*
114. **function** power(b, outer) {
115. **this**.ist = "power";
116. **this**.pow\_to = **new** theFormula(b, outer);
117. }


121. *//копирование формулы*
122. **function** copyF(form) {
123. **var** newForm = **new** theFormula([], form.outer);
124. **var** newInner = copyArr(form.inner, newForm);
125. newForm.inner = newInner;
126. **return** newForm;
127. }

130. **function** copyArr(inner, nOuter) {
131. **var** newInner = [];
132. **for** (**var** i = 0 ; i < inner.length ; ++i) {
133. **if** (inner[i].ist == "operator") {
134. newInner.push(**new** operator(inner[i].type));
135. }
136. **else** **if** (inner[i].ist == "root") {
137. newInner.push(**new** root());
138. }
139. **else** **if** (inner[i].ist == "param") {
140. newInner.push(**new** par());
141. }
142. **else** **if** (inner[i].ist == "numb") {
143. newInner.push(**new** numb(inner[i].type, inner[i].value));
144. }
145. **else** **if** (inner[i].ist == "power") {
146. newInner.push(**new** power(copyArr(inner[i].pow\_to.inner), nOuter));
147. }
148. **else** **if** (inner[i].ist == "function") {
149. newInner.push(copyFunc(inner[i], nOuter));
150. }
151. }
152. **return** newInner;
153. }

156. **function** copyThis(thiss, nOuter) {
157. **var** newThis;
158. **if** (thiss.ist == "operator") {
159. **return** **new** operator(thiss.type);
160. }
161. **else** **if** (thiss.ist == "root") {
162. **return** **new** root();
163. }
164. **else** **if** (thiss.ist == "param") {
165. **return** **new** par();
166. }
167. **else** **if** (thiss.ist == "numb") {
168. **return** **new** numb(thiss.type, thiss.value);
169. }
170. **else** **if** (thiss.ist == "power") {
171. **return** **new** power(copyArr(thiss.pow\_to.inner), thiss.pow\_to.outer);
172. }
173. **else** **if** (thiss.ist == "function") {
174. **return** copyFunc(thiss, nOuter);
175. }
176. **else** {
177. **return** thiss;
178. }
179. }

182. **function** copyFunc(func1, newOuter) {
183. **var** newFunc = **new** funct(func1.type, copyThis(func1.arg, newOuter), newOuter);
184. newFunc.inner.inner = copyArr(func1.inner.inner, newFunc.inner);
185. newFunc.exit = func1.exit
186. **return** newFunc;
187. }

190. *//Добавляем цифру к числу*
191. **function** addTo(num1, num2) {
192. num1 \*= 10;
193. num1 += num2;
194. num1 = Math.round(num1);
195. **return** num1;
196. }
198. **function** addX(X) {
199. **if** ((custN.toString().length + custNafter.toString().length + vedZeros) <= 15) {
200. **if** (hasPoint) {
201. **if** (custNafter == 0 && X == 0) {
202. vedZeros += 1;
203. }
204. **else** {
205. custNafter = addTo(custNafter, X);
206. }
207. }
208. **else** {
209. custN = addTo(custN, X);
210. }
211. }
212. }
214. **function** setVZ(num) {
215. **var** curSt = "";
216. **for** (**var** i = 0 ; i < num ; ++i) {
217. curSt += "0";
218. }
219. **return** curSt;
220. }
222. *//Количество ведущих нулей после 1 знака части после запятой*
223. **function** za1(num) {
224. **var** nStr = num.toString();
225. **var** zeros = 0;
226. **for** (**var** i = 1 ; i < nStr.length ; ++i) {
227. **if** (nStr[i] == "0") {
228. zeros++;
229. }
230. **else** {
231. **break**;
232. }
233. }
234. **return** zeros;
235. }
237. **function** wrOP(str) {
238. **if** (str == "+" || str == "-") {
239. **return** " " + str + " ";
240. }
241. **else** **return** str;
242. }

245. **function** toTheNum(first, second, zeros, sign) {
246. **var** num = first;
247. **if** (second != 0) {
248. **var** logNum = Math.floor(Math.log(second) / Math.log(10)) + 1 + zeros;
249. **var** num2 = second / Math.pow(10, logNum);
250. **return** (num + num2) \* sign;
251. }
252. **else** **return** num \* sign;
253. }



258. **function** writeEq(formula) {
259. **var** str = "";
260. **var** elements = formula.inner.length;
261. **for** (**var** i = 0 ; i < elements ; ++i) {
262. str += writeThe(formula.inner[i]);
263. }
264. **return** str;
265. }

268. **function** writeThe(object) {
269. **if** (object.ist == "operator") {
270. **return** wrOP(object.type);
271. }
272. **else** **if** (object.ist == "root") {
273. **return** "X";
274. }
275. **else** **if** (object.ist == "param") {
276. **return** "P";
277. }
278. **else** **if** (object.ist == "numb") {
279. **if** (object.type == "e") {
280. **return** "e";
281. }
282. **else** **if** (object.type == "pi") {
283. **return** "π";
284. }
285. **else** {
286. **return** object.value;
287. }
288. }
289. **else** **if** (object.ist == "power") {
290. **if** (object.pow\_to.inner.length == 1) {
291. **return** "^" + writeThe(object.pow\_to.inner[0]);
292. }
293. **else** {
294. **return** "^(" + writeEq(object.pow\_to) + ")";
295. }
296. }
297. **else** **if** (object.ist == "function") {
298. **var** theTx = "";
299. **if** (object.type == "power") {
300. theTx += writeThe(object.arg) + "^(" + writeEq(object.inner);
301. }
302. **else** **if** (object.type == "log") {
303. theTx += "log" + "[" + writeEq(object.arg.inner) + "](" + writeEq(object.inner);
304. }
305. **else** **if** (object.type == "root") {
306. theTx += "root" + "[" + writeEq(object.arg.inner) + "](" + writeEq(object.inner);
307. }
308. **else** **if** (object.type == "bracket") {
309. theTx += "(" + writeEq(object.inner);
310. }
311. **else** {
312. theTx += object.type + "(" + writeEq(object.inner);
313. }
315. **if** (object.exit) {
316. theTx += ")";
317. }
318. **return** theTx;
319. }
320. }

323. **function** getThePart(text, begin, end) {
324. **var** textAns = "";
325. **for** (**var** i = begin ; i < end ; ++i) {
326. textAns += text[i];
327. }
328. **return** textAns;
329. }



334. **this**.addEventListener("tick", main.bind(**this**));
335. **function** main() {
336. **if** (isE) {
337. **this**.custNum.text = "e";
338. }
339. **else** **if** (isPI) {
340. **this**.custNum.text = "π";
341. }
342. **else** {
343. **if** (hasPoint == **false**) {
344. **this**.custNum.text = (hasMinus ? "-" : "") + Math.round(custN);
345. }
346. **else** {
347. **this**.custNum.text = (hasMinus ? "-" : "") + Math.round(custN) + "." + setVZ(vedZeros) + (custNafter ? Math.round(custNafter) :"");
348. }
349. }
351. signsNow = writeEq(formula).length;
352. **this**.curEq.text = getThePart(writeEq(formula), fShift, Math.min(fShift+32, signsNow));
353. **if** (signsNow > fShift+32) {
354. **this**.curEq.text += "..";
355. }

358. **this**.sizeTT.text = "(" + Math.round(signsNow) + " / 300)";
360. **if** (**this**.err2.visible) {
361. **this**.err2.alpha -= 1/(5\*fps);
362. }
363. **if** (**this**.err2.alpha <= 0) {
364. **this**.err2.visible = **false**;
365. **this**.err2.alpha = 1;
366. }

369. **if** (signsNow >= 300) {
370. **this**.sizeTT.color = '#AA0000';
371. **this**.warning.visible = **true**;
372. }
373. **else** {
374. **this**.sizeTT.color = '#004400';
375. **this**.warning.visible = **false**;
376. }

379. dP = (pMax - pMin) / (conts);
381. **this**.xStarT.text = "~" + Math.round(xStar\*10000)/10000;
383. **this**.pMinT.text = "~" + Math.round(pMin\*10000)/10000;
384. **this**.pMaxT.text = "~" + Math.round(pMax\*10000)/10000;

387. **this**.continos.text = Math.round(conts);
388. **this**.deltaP.text = "~" + Math.round(dP\*100000)/100000;
390. }

393. **this**.add1.addEventListener("click", add1f.bind(**this**));
394. **function** add1f() {
395. addX(1);
396. }
398. **this**.add2.addEventListener("click", add2f.bind(**this**));
399. **function** add2f() {
400. addX(2);
401. }
403. **this**.add3.addEventListener("click", add3f.bind(**this**));
404. **function** add3f() {
405. addX(3);
406. }
408. **this**.add4.addEventListener("click", add4f.bind(**this**));
409. **function** add4f() {
410. addX(4);
411. }

414. **this**.add5.addEventListener("click", add5f.bind(**this**));
415. **function** add5f() {
416. addX(5);
417. }
419. **this**.add6.addEventListener("click", add6f.bind(**this**));
420. **function** add6f() {
421. addX(6);
422. }
424. **this**.add7.addEventListener("click", add7f.bind(**this**));
425. **function** add7f() {
426. addX(7);
427. }
429. **this**.add8.addEventListener("click", add8f.bind(**this**));
430. **function** add8f() {
431. addX(8);
432. }
434. **this**.add9.addEventListener("click", add9f.bind(**this**));
435. **function** add9f() {
436. addX(9);
437. }
439. **this**.add0.addEventListener("click", add0f.bind(**this**));
440. **function** add0f() {
441. addX(0);
442. }
444. **var** trig = 0;
445. **var** trigTim = 0;
447. **this**.formuLeft.addEventListener("mouseover", setTrig.bind(**this**));
448. **function** setTrig() {
449. trig = -1;
450. }
452. **this**.formuLeft.addEventListener("mouseout", setTrig2.bind(**this**));
453. **function** setTrig2() {
454. trig = 0;
455. trigTim = 0;
456. }
458. **this**.formuLeft.addEventListener("click", AsetTrig2.bind(**this**));
459. **function** AsetTrig2() {
461. **if** (fShift > 0) {
462. fShift -= 1;
463. }
464. trig = 0;
465. trigTim = 0;
466. }
468. **this**.formuRight.addEventListener("mouseover", setTrig12.bind(**this**));
469. **function** setTrig12() {
470. trig = 1;
471. }
473. **this**.formuRight.addEventListener("mouseout", setTrig22.bind(**this**));
474. **function** setTrig22() {
475. trig = 0;
476. trigTim = 0;
477. }
479. **this**.formuRight.addEventListener("click", AsetTrig22.bind(**this**));
480. **function** AsetTrig22() {
481. **if** (signsNow > fShift+32) {
482. fShift += 1;
483. }
484. trig = 0;
485. trigTim = 0;
486. }
488. **var** fpSSS = 0;
490. **this**.addEventListener("tick", settingTrigger.bind(**this**));
491. **function** settingTrigger() {
492. fpSSS += 1;
493. **if** (trig == 1) {
494. trigTim += 1/fps;
495. }
496. **else** **if** (trig == -1) {
497. trigTim -= 1/fps;
498. }
499. **if** (trigTim >= 3) {
500. **if** (signsNow > fShift+32 && fpSSS%3==0) {
501. fShift += 1;
502. }
503. }
504. **else** **if** (trigTim <= -3 && fpSSS%3==0) {
505. **if** (fShift > 0) {
506. fShift -= 1;
507. }
508. }
509. }

512. **this**.addPt.addEventListener("click", addPtf.bind(**this**));
513. **function** addPtf() {
514. hasPoint = **true**;
515. }
517. **this**.removeOne.addEventListener("click", removeOnef.bind(**this**));
518. **function** removeOnef() {
519. **if** (vedZeros > 0 && custNafter==0) {
520. vedZeros -= 1;
521. }
522. **else** **if** (!hasPoint) {
523. custN = Math.floor(custN/10);
524. }
525. **else** **if** (custNafter != 0) {
526. custNafter = Math.floor(custNafter/10);
527. }
528. **else** {
529. hasPoint = **false**;
530. }
531. }
533. **this**.removeAll.addEventListener("click", removeAllf.bind(**this**));
534. **function** removeAllf() {
535. custN = 0;
536. custNafter = 0;
537. hasPoint = **false**;
538. vedZeros = 0;
539. hasMinus = **false**;
540. }
542. **this**.mult10.addEventListener("click", mult10f.bind(**this**));
543. **function** mult10f() {
544. **if** ((custN.toString().length + custNafter.toString().length + vedZeros) <= 15) {
545. **if** (!hasPoint || custNafter == 0 && vedZeros == 0) {
546. custN = Math.round(custN\*10);
547. }
548. **else** **if** (vedZeros > 0) {
549. custN = Math.round(custN\*10);
550. vedZeros -= 1;
551. }
552. **else** **if** (custNafter != 0) {
553. **var** maxPtPt = Math.log(custNafter) / Math.log(10);
554. **var** maxPtPt = Math.floor(maxPtPt);
555. vedZeros += za1(custNafter);
556. **var** t10power = Math.pow(10, maxPtPt);
557. **var** valX = Math.floor(custNafter/t10power);
558. custNafter = Math.round(custNafter - t10power\*valX);
559. custN = addTo(custN, valX);
560. }
561. **if** (custNafter == 0 && vedZeros == 0) {
562. hasPoint = **false**;
563. }
564. }
565. }
567. **this**.div10.addEventListener("click", div10f.bind(**this**));
568. **function** div10f() {
569. **if** ((custN.toString().length + custNafter.toString().length + vedZeros) <= 15) {
570. **var** lastDigit = custN%10;
571. **if** (lastDigit == 0) {
572. **if** (hasPoint) {
573. vedZeros += 1;
574. }
575. custN = Math.floor(custN/10);
576. }
577. **else** **if** (custN !=0) {
578. hasPoint = **true**;
579. custN = Math.floor(custN/10);
580. **var** newNum = lastDigit \* Math.pow(10, vedZeros);
581. vedZeros = 0;
582. **var** newLog;
583. **if** (custNafter != 0) {
584. newLog = Math.floor(Math.log(custNafter) / Math.log(10))+1;
585. }
586. **else** newLog = 0;
587. custNafter += newNum \* Math.pow(10, newLog);
588. }
589. }
590. }
592. **this**.setE.addEventListener("click", setEf.bind(**this**));
593. **function** setEf() {
594. **if** (!isE) {
595. isE = **true**;
596. isPI = **false**;
597. }
598. **else** {
599. isE = **false**;
600. isPI = **false**;
601. }
602. }
604. **this**.setPI.addEventListener("click", setPIf.bind(**this**));
605. **function** setPIf() {
606. **if** (!isPI) {
607. isPI = **true**;
608. isE = **false**;
609. }
610. **else** {
611. isPI = **false**;
612. isE = **false**;
613. }
614. }

617. **this**.senNxStar.addEventListener("click", senNxStarf.bind(**this**));
618. **function** senNxStarf() {
619. **if** (isE) {
620. xStar = 2.718281828459;
621. removeAllf();
622. }
623. **else** **if** (isPI) {
624. xStar = 3.14159265359;
625. removeAllf();
626. }
627. **else** {
628. xStar = toTheNum(custN, custNafter, vedZeros, bll(hasMinus));
629. removeAllf();
630. }
631. }

634. **this**.senPMin.addEventListener("click", senPMinf.bind(**this**));
635. **function** senPMinf() {
636. **if** (isE) {
637. **if** (2.718281828459 < pMax) {
638. pMin = 2.718281828459;
639. removeAllf();
640. }
641. }
642. **else** **if** (isPI) {
643. **if** (3.14159265359 < pMax) {
644. pMin = 3.14159265359;
645. removeAllf();
646. }
647. }
648. **else** {
649. **if** (toTheNum(custN, custNafter, vedZeros, bll(hasMinus)) < pMax) {
650. pMin = toTheNum(custN, custNafter, vedZeros, bll(hasMinus));
651. removeAllf();
652. }
653. }
654. }

657. **this**.senPMax.addEventListener("click", senPMaxf.bind(**this**));
658. **function** senPMaxf() {
659. **if** (isE) {
660. **if** (2.718281828459 > pMin) {
661. pMax = 2.718281828459;
662. removeAllf();
663. }
664. }
665. **else** **if** (isPI) {
666. **if** (3.14159265359 > pMin) {
667. pMax = 3.14159265359;
668. removeAllf();
669. }
670. }
671. **else** {
672. **if** (toTheNum(custN, custNafter, vedZeros, bll(hasMinus)) > pMin) {
673. pMax = toTheNum(custN, custNafter, vedZeros, bll(hasMinus));
674. removeAllf();
675. }
676. }
677. }

680. **var** trig2 = 0;
681. **var** trigTim2 = 0;
682. **var** fpSSS2 = 0;
684. **this**.addEventListener("tick", settingTrigger2.bind(**this**));
685. **function** settingTrigger2() {
686. fpSSS2 += 1;
687. **if** (trig2 == 1) {
688. trigTim2 += 1/fps;
689. }
690. **else** **if** (trig2 == -1) {
691. trigTim2 -= 1/fps;
692. }
693. **if** (trigTim2 >= 3) {
694. **if** (conts < 300 && fpSSS2%3<=1) {
695. conts += 1;
696. }
697. }
698. **else** **if** (trigTim2 <= -3 && fpSSS2%3<=1) {
699. **if** (conts > 1) {
700. conts -= 1;
701. }
702. }
704. **if** (fShift > signsNow-32) {
705. fShift = Math.max(0, signsNow-32);
706. }

709. }
711. **this**.plusCon.addEventListener("click", plusConf.bind(**this**));
712. **function** plusConf() {
713. **if** (conts < 300) {
714. conts++;
715. }
716. trigTim2 = 0;
717. trig2 = 0;
718. }
720. **this**.plusCon.addEventListener("mouseover", plusConf2.bind(**this**));
721. **function** plusConf2() {
722. **if** (conts < 300) {
723. trig2 = 1;
724. }
725. }
727. **this**.plusCon.addEventListener("mouseout", plusConf3.bind(**this**));
728. **function** plusConf3() {
729. trigTim2 = 0;
730. trig2 = 0;
731. }

734. **this**.minusCon.addEventListener("click", minusConf.bind(**this**));
735. **function** minusConf() {
736. **if** (conts > 1) {
737. conts--;
738. }
739. trigTim2 = 0;
740. trig2 = 0;
741. }
743. **this**.minusCon.addEventListener("mouseover", minusConf2.bind(**this**));
744. **function** minusConf2() {
745. **if** (conts > 1) {
746. trig2 = -1;
747. }
748. }
750. **this**.minusCon.addEventListener("mouseout", minusConf3.bind(**this**));
751. **function** minusConf3() {
752. trigTim2 = 0;
753. trig2 = 0;
754. }

757. **this**.graphWindow.closGraph.addEventListener("click", closGR.bind(**this**));
758. **function** closGR() {
759. **this**.graphWindow.visible = **false**;
760. graphOpen = **false**;
761. }

764. **function** badNumber(num) {
765. **if** (isNaN(num) || num == **Infinity** || num == -**Infinity**) {
766. **return** **true**;
767. }
768. **else** {
769. **return** **false**;
770. }
771. }

774. **var** pArray = [];
775. **var** xArray = [];

778. **function** newTone\_Method(x, p) {
779. **var** xPrev = x;
780. **var** xNext = x;
781. **var** iters = 0;
783. while ((Math.abs((xNext/xPrev) - 1) > 0.0000001 && iters < 1000) || iters < 5) {
785. **var** forDer = findDerivative(copyF(formula), formula.outer);
786. **for** (**var** i = 0 ; i < 10 ; ++i) {
787. forDer.inner = optimize(forDer.inner, 0);
788. }
790. **var** fxn = podstava(xNext, p, copyF(formula));
791. **var** dfxn = podstava(xNext, p, copyF(forDer));
792. xPrev = xNext;
793. xNext = xPrev - (fxn/dfxn);
794. ++iters;
795. }
796. **if** (iters >= 999) {
797. **return** **Infinity**;
798. }
799. **else** {
800. **return** xNext;
801. }
802. }


806. **this**.solveBt.addEventListener("click", solving.bind(**this**));
807. **function** solving() {
808. pArray = [];
809. xArray = [];
811. **var** xPrev = xStar;
812. **var** xNext = xStar;
813. **var** iters = 0;
815. while ((Math.abs((xNext/xPrev) - 1) > 1e-15 && iters < 1000) || iters < 5) {
817. **for** (**var** i = 0 ; i < 10 ; ++i) {
818. formula.inner = optimize(formula.inner, 0);
819. }
820. **var** forDer = findDerivative(copyF(formula), formula.outer);
821. **for** (**var** i = 0 ; i < 10 ; ++i) {
822. forDer.inner = optimize(forDer.inner, 0);
823. }
825. **var** fxn = podstava(xNext, pMin, copyF(formula));
826. *//console.log(fxn);*
827. **var** dfxn = podstava(xNext, pMin, copyF(forDer));
828. xPrev = xNext;
829. xNext = xPrev - (fxn/dfxn);
830. ++iters;
831. }
833. **if** (iters > 999 || badNumber(xNext)) {
834. **this**.err2.visible = **true**;
835. **this**.err2.alpha = 1;
836. }
837. **else** {
838. pArray.push(pMin);
839. xArray.push(xNext);
840. **var** step = 1;
841. **var** curP = pMin + dP;
842. **var** xThis = xNext;
843. while (Math.round(step) <= Math.round(conts)) {
844. xThis = newTone\_Method(xThis, curP);
845. **if** (badNumber(xThis)) {
846. **break**;
847. }
848. **else** {
849. pArray.push(curP);
850. xArray.push(xThis);
851. ++step;
852. curP = curP + dP;
853. }
854. }
856. stage.addChild(**this**.graphWindow);
857. **this**.graphWindow.visible = **true**;
858. **this**.graphWindow.x = 500;
859. **this**.graphWindow.y = 370;

862. graphOpen = **true**;
864. console.log(pArray);
865. console.log(xArray);
867. **var** pMinW = 500 + **this**.graphWindow.xyMin.x;
868. **var** xMinW = 370 + **this**.graphWindow.xyMin.y;
869. **var** pMaxW = 500 + **this**.graphWindow.xMax.x;
870. **var** xMaxW = 370 + **this**.graphWindow.yMax.y;
872. **var** xMin = xArray[0];
873. **var** xMax = xArray[0];

876. **for** (**var** i = 0 ; i < xArray.length ; ++i) {
877. **if** (xArray[i] < xMin) {
878. xMin = xArray[i];
879. }
880. **if** (xArray[i] > xMax) {
881. xMax = xArray[i];
882. }
883. }
885. **var** dist = Math.abs(xMax - xMin);
886. xMax += dist\*0.15;
888. **if** (dist == 0) {
889. xMax += 0.005;
890. xMin -= 0.005;
891. }
892. **else** **if** (dist < 0.004) {
893. xMax += 0.002;
894. xMin -= 0.002;
895. }
897. **this**.graphWindow.y1.text = Math.round(xMin\*1000)/1000 + "";
899. **this**.graphWindow.y2.text = Math.round((xMin + (xMax - xMin)\*(1/6))\*1000)/1000 + "";
900. **this**.graphWindow.y3.text = Math.round((xMin + (xMax - xMin)\*(2/6))\*1000)/1000 + "";
901. **this**.graphWindow.y4.text = Math.round((xMin + (xMax - xMin)\*(3/6))\*1000)/1000 + "";
902. **this**.graphWindow.y5.text = Math.round((xMin + (xMax - xMin)\*(4/6))\*1000)/1000 + "";
903. **this**.graphWindow.y6.text = Math.round((xMin + (xMax - xMin)\*(5/6))\*1000)/1000 + "";
905. **this**.graphWindow.y7.text = Math.round(xMax\*1000)/1000 + "";
907. **this**.graphWindow.x1.text = Math.round(pMin\*1000)/1000 + "";
909. **this**.graphWindow.x2.text = Math.round((pMin + (pMax - pMin)\*(1/5))\*1000)/1000 + "";
910. **this**.graphWindow.x3.text = Math.round((pMin + (pMax - pMin)\*(2/5))\*1000)/1000 + "";
911. **this**.graphWindow.x4.text = Math.round((pMin + (pMax - pMin)\*(3/5))\*1000)/1000 + "";
912. **this**.graphWindow.x5.text = Math.round((pMin + (pMax - pMin)\*(4/5))\*1000)/1000 + "";
914. **this**.graphWindow.x6.text = Math.round(pMax\*1000)/1000 + "";

917. **for** (**var** i = 0 ; i < xArray.length ; ++i) {
918. **var** point = **new** lib.bigPoint();
919. stage.addChild(point);
921. point.x = transfCD(pArray[i], pMinW, pMaxW, pMin, pMax);
922. point.y = transfCD(xArray[i], xMinW, xMaxW, xMin, xMax);

925. point.visible = **true**;
926. point.alpha = 1;
928. **var** join = **new** lib.line();
929. stage.addChild(join);
931. **if** (i >= 1) {
932. join.x = transfCD(pArray[i-1], pMinW, pMaxW, pMin, pMax);
933. join.y = transfCD(xArray[i-1], xMinW, xMaxW, xMin, xMax);
934. join.endX = transfCD(pArray[i], pMinW, pMaxW, pMin, pMax);
935. join.endY = transfCD(xArray[i], xMinW, xMaxW, xMin, xMax);

938. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
940. join.scaleX = join.len;
942. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
944. join.visible = **true**;
945. join.alpha = 1;
946. join.addEventListener('tick', liveLine);
947. }
949. point.visible = **true**;
950. point.alpha = 1;
952. stage.addChild(point);

955. point.addEventListener('tick', livePoint);
956. }

959. **if** (pMin < 0 && 1.2\*pMax >= 0) {
960. **this**.graphWindow.yAxis.x = transfCD(0, pMinW - 500, pMaxW - 500, pMin, pMax);
961. }
962. **else** {
963. **this**.graphWindow.yAxis.x = -187.35;
964. }

967. **if** (xMin < 0 && 1.2\*xMax >= 0) {
968. **this**.graphWindow.xAxis.y = transfCD(0, xMinW - 370, xMaxW - 370, xMin, xMax);
969. }
970. **else** {
971. **this**.graphWindow.xAxis.y = 203.25;
972. }


976. }
978. }

981. **function** transfCD(x, xStart, xEnd, xMin, xMax) {
982. **var** newX = xStart + ((x-xMin) \* (xEnd - xStart)/(xMax - xMin));
983. **return** newX;
984. }

987. **function** livePoint(e) {
988. **var** object = e.currentTarget;
990. **if** (!graphOpen) {
991. object.alpha = 0;
992. object.visible = **false**;
993. object.removeEventListener('tick', livePoint);
994. stage.removeChild(object);
995. }
997. }

1000. **function** liveLine(e) {
1001. **var** object = e.currentTarget;
1003. **if** (!graphOpen) {
1004. object.alpha = 0;
1005. object.visible = **false**;
1006. object.removeEventListener('tick', liveLine);
1007. stage.removeChild(object);
1008. }
1010. }

1013. **function** podstava(x, p, formule) {
1014. **if** (formule.inner) {
1015. **for** (**var** i = 0 ; i < formule.inner.length ; ++i) {
1016. **if** (formule.inner[i].ist == "root") {
1017. formule.inner[i] = x;
1018. }
1019. **else** **if** (formule.inner[i].ist == "param") {
1020. formule.inner[i] = p;
1021. }
1022. **else** **if** (formule.inner[i].ist == "numb") {
1023. **if** (formule.inner[i].type == "e") {
1024. formule.inner[i] = 2.718281828459;
1025. }
1026. **else** **if** (formule.inner[i].type == "pi") {
1027. formule.inner[i] = 3.14159265359;
1028. }
1029. **else** {
1030. formule.inner[i] = formule.inner[i].value;
1031. }
1032. }
1033. **else** **if** (formule.inner[i].ist == "function") {
1034. **if** (formule.inner[i].type == "log" || formule.inner[i].type == "root") {
1035. formule.inner[i].arg = podstava(x, p, formule.inner[i].arg.inner);
1036. formule.inner[i] = solveFun(formule.inner[i].type, podstava(x, p, formule.inner[i].inner), formule.inner[i].arg);
1037. }
1038. **else** **if** (formule.inner[i].type == "power") {
1039. **if** (formule.inner[i].arg.ist == "numb") {
1040. **if** (formule.inner[i].arg.type == "e") {
1041. formule.inner[i].arg = 2.718281828459;
1042. }
1043. **else** **if** (formule.inner[i].arg.type == "pi") {
1044. formule.inner[i].arg = 3.14159265359;
1045. }
1046. **else** {
1047. formule.inner[i].arg = formule.inner[i].arg.value;
1048. }
1049. }
1050. **else** **if** (formule.inner[i].arg.ist == "param") {
1051. formule.inner[i].arg = p;
1052. }
1053. formule.inner[i] = solveFun(formule.inner[i].type, podstava(x, p, formule.inner[i].inner), formule.inner[i].arg);
1054. }
1055. **else** {
1056. formule.inner[i] = solveFun(formule.inner[i].type, podstava(x, p, formule.inner[i].inner), 0);
1057. }
1058. }
1059. **else** **if** (formule.inner[i].ist == "power") {
1060. formule.inner[i].pow\_to = podstava(x, p, formule.inner[i].pow\_to);
1061. }
1062. }

1065. **for** (**var** i = 1 ; i < formule.inner.length ; ++i) {
1066. **if** (formule.inner[i].ist == "power") {
1067. formule.inner[i-1] = Math.pow(formule.inner[i-1], formule.inner[i].pow\_to);
1068. formule.inner.splice(i, 1);
1069. --i;
1070. }
1071. }
1073. **for** (**var** i = 1 ; i < formule.inner.length ; ++i) {
1074. **if** (formule.inner[i].ist == "operator") {
1075. **if** (formule.inner[i].type == "\*") {
1076. formule.inner[i-1] = formule.inner[i-1] \* formule.inner[i+1];
1077. formule.inner.splice(i, 2);
1078. --i;
1079. }
1080. **else** **if** (formule.inner[i].type == "/") {
1081. formule.inner[i-1] = formule.inner[i-1] / formule.inner[i+1];
1082. formule.inner.splice(i, 2);
1083. --i;
1084. }
1085. }
1086. }
1088. **for** (**var** i = 1 ; i < formule.inner.length ; ++i) {
1089. **if** (formule.inner[i].ist == "operator") {
1090. **if** (formule.inner[i].type == "+") {
1091. formule.inner[i-1] = formule.inner[i-1] + formule.inner[i+1];
1092. formule.inner.splice(i, 2);
1093. --i;
1094. }
1095. **else** **if** (formule.inner[i].type == "-") {
1096. formule.inner[i-1] = formule.inner[i-1] - formule.inner[i+1];
1097. formule.inner.splice(i, 2);
1098. --i;
1099. }
1100. }
1101. }

1104. **return** formule.inner[0];
1106. }
1107. **else** {
1108. **return** formule;
1109. }
1110. }



1115. **function** solveFun(fType, num, arg) {
1116. **if** (fType == "bracket") {
1117. **return** num;
1118. }
1119. **else** **if** (fType == "sin") {
1120. **return** Math.sin(num);
1121. }
1122. **else** **if** (fType == "cos") {
1123. **return** Math.cos(num);
1124. }
1125. **else** **if** (fType == "tan") {
1126. **return** Math.tan(num);
1127. }
1128. **else** **if** (fType == "asin") {
1129. **return** Math.asin(num);
1130. }
1131. **else** **if** (fType == "acos") {
1132. **return** Math.acos(num);
1133. }
1134. **else** **if** (fType == "atan") {
1135. **return** Math.atan(num);
1136. }
1137. **else** **if** (fType == "atan") {
1138. **return** Math.atan(num);
1139. }
1140. **else** **if** (fType == "power") {
1141. **return** Math.pow(arg, num);
1142. }
1143. **else** **if** (fType == "log") {
1144. **return** Math.log(num) / Math.log(arg);
1145. }
1146. **else** **if** (fType == "root") {
1147. **return** Math.pow(num, (1/arg));
1148. }
1149. **else** **if** (fType == "ln") {
1150. **return** Math.log(num);
1151. }
1152. **else** **if** (fType == "sqrt") {
1153. **return** Math.pow(num, (1/2));
1154. }
1155. **else** **if** (fType == "cbrt") {
1156. **return** Math.pow(num, (1/3));
1157. }
1158. }



1163. **this**.addMinus.addEventListener("click", addMinus.bind(**this**));
1164. **function** addMinus() {
1165. **if** (!hasMinus) {
1166. hasMinus = **true**;
1167. }
1168. **else** hasMinus = **false**;
1169. }

1172. **function** bll(ss) {
1173. **if** (ss) {
1174. **return** -1;
1175. }
1176. **else** **return** 1;
1177. }
1179. **this**.add\_n.addEventListener("click", add\_nf.bind(**this**));
1180. **function** add\_nf() {
1181. **if** (curPlace.inner.length == 0) {
1182. **if** (isE) {
1183. curPlace.inner.push(**new** numb("e", 1));
1184. removeAllf();
1185. }
1186. **else** **if** (isPI) {
1187. curPlace.inner.push(**new** numb("pi", 1));
1188. removeAllf();
1189. }
1190. **else** {
1191. curPlace.inner.push(**new** numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))));
1192. removeAllf();
1193. }
1194. }
1195. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1196. **if** (isE) {
1197. curPlace.inner.push(**new** numb("e", 1));
1198. removeAllf();
1199. }
1200. **else** **if** (isPI) {
1201. curPlace.inner.push(**new** numb("pi", 1));
1202. removeAllf();
1203. }
1204. **else** {
1205. **if** (curPlace.inner[curPlace.inner.length - 1].type != "/" || toTheNum(custN, custNafter, vedZeros, bll(hasMinus)) != 0) {
1206. curPlace.inner.push(**new** numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))));
1207. removeAllf();
1208. }
1209. }
1210. }
1211. }
1213. **this**.add\_p.addEventListener("click", add\_pf.bind(**this**));
1214. **function** add\_pf() {
1215. **if** (signsNow < 300) {
1216. **if** (curPlace.inner.length == 0) {
1217. curPlace.inner.push(**new** par());
1218. }
1219. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1220. curPlace.inner.push(**new** par());
1221. }
1222. }
1223. }
1225. **this**.add\_x.addEventListener("click", add\_xf.bind(**this**));
1226. **function** add\_xf() {
1227. **if** (signsNow < 300) {
1228. **if** (curPlace.inner.length == 0) {
1229. curPlace.inner.push(**new** root());
1230. }
1231. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1232. curPlace.inner.push(**new** root());
1233. }
1234. }
1235. }
1237. **this**.f\_plus.addEventListener("click", f\_plusf.bind(**this**));
1238. **function** f\_plusf() {
1239. **if** (signsNow < 300) {
1240. **if** (curPlace.inner.length != 0) {
1241. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1242. curPlace.inner.push(**new** operator("+"));
1243. }
1244. }
1245. }
1246. }
1248. **this**.f\_minus.addEventListener("click", f\_minusf.bind(**this**));
1249. **function** f\_minusf() {
1250. **if** (signsNow < 300) {
1251. **if** (curPlace.inner.length != 0) {
1252. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1253. curPlace.inner.push(**new** operator("-"));
1254. }
1255. }
1256. }
1257. }
1259. **this**.f\_mult.addEventListener("click", f\_multf.bind(**this**));
1260. **function** f\_multf() {
1261. **if** (signsNow < 300) {
1262. **if** (curPlace.inner.length != 0) {
1263. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1264. curPlace.inner.push(**new** operator("\*"));
1265. }
1266. }
1267. }
1268. }
1270. **this**.f\_div.addEventListener("click", f\_divf.bind(**this**));
1271. **function** f\_divf() {
1272. **if** (signsNow < 300) {
1273. **if** (curPlace.inner.length != 0) {
1274. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1275. curPlace.inner.push(**new** operator("/"));
1276. }
1277. }
1278. }
1279. }
1281. **this**.powNop.addEventListener("click", powNopf.bind(**this**));
1282. **function** powNopf() {
1283. **if** (signsNow < 300) {
1284. **if** (curPlace.inner.length != 0) {
1285. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1286. **if** (isE) {
1287. curPlace.inner.push(**new** power([**new** numb("e", 1)], curPlace));
1288. removeAllf();
1289. }
1290. **else** **if** (isPI) {
1291. curPlace.inner.push(**new** power([**new** numb("pi", 1)], curPlace));
1292. removeAllf();
1293. }
1294. **else** {
1295. curPlace.inner.push(**new** power([**new** numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus)))], curPlace));
1296. removeAllf();
1297. }
1298. }
1299. }
1300. }
1301. }
1303. **this**.powPop.addEventListener("click", powPopf.bind(**this**));
1304. **function** powPopf() {
1305. **if** (signsNow < 300) {
1306. **if** (curPlace.inner.length != 0) {
1307. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1308. curPlace.inner.push(**new** power([**new** par()], curPlace));
1309. }
1310. }
1311. }
1312. }
1314. **this**.powPPNop.addEventListener("click", powPPNopf.bind(**this**));
1315. **function** powPPNopf() {
1316. **if** (signsNow < 300) {
1317. **if** (curPlace.inner.length != 0) {
1318. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1319. **if** (isE) {
1320. curPlace.inner.push(**new** power([**new** par(), **new** operator("+"), **new** numb("e", 1)], curPlace));
1321. removeAllf();
1322. }
1323. **else** **if** (isPI) {
1324. curPlace.inner.push(**new** power([**new** par(), **new** operator("+"), **new** numb("pi", 1)], curPlace));
1325. removeAllf();
1326. }
1327. **else** {
1328. curPlace.inner.push(**new** power([**new** par(), **new** operator("+"), **new** numb("number", toTheNum(custN, custNafter, vedZeros,bll(hasMinus)))], curPlace));
1329. removeAllf();
1330. }
1331. }
1332. }
1333. }
1334. }
1336. **this**.powPMNop.addEventListener("click", powPMNopf.bind(**this**));
1337. **function** powPMNopf() {
1338. **if** (signsNow < 300) {
1339. **if** (curPlace.inner.length != 0) {
1340. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1341. **if** (isE) {
1342. curPlace.inner.push(**new** power([**new** par(), **new** operator("-"), **new** numb("e", 1)], curPlace));
1343. removeAllf();
1344. }
1345. **else** **if** (isPI) {
1346. curPlace.inner.push(**new** power([**new** par(), **new** operator("-"), **new** numb("pi", 1)], curPlace));
1347. removeAllf();
1348. }
1349. **else** {
1350. curPlace.inner.push(**new** power([**new** par(), **new** operator("-"), **new** numb("number", toTheNum(custN, custNafter, vedZeros,bll(hasMinus)))], curPlace));
1351. removeAllf();
1352. }
1353. }
1354. }
1355. }
1356. }
1358. **this**.powPxNop.addEventListener("click", powPxNopf.bind(**this**));
1359. **function** powPxNopf() {
1360. **if** (signsNow < 300) {
1361. **if** (curPlace.inner.length != 0) {
1362. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1363. **if** (isE) {
1364. curPlace.inner.push(**new** power([**new** par(), **new** operator("\*"), **new** numb("e", 1)], curPlace));
1365. removeAllf();
1366. }
1367. **else** **if** (isPI) {
1368. curPlace.inner.push(**new** power([**new** par(), **new** operator("\*"), **new** numb("pi", 1)], curPlace));
1369. removeAllf();
1370. }
1371. **else** {
1372. curPlace.inner.push(**new** power([**new** par(), **new** operator("\*"), **new** numb("number", toTheNum(custN, custNafter, vedZeros,bll(hasMinus)))], curPlace));
1373. removeAllf();
1374. }
1375. }
1376. }
1377. }
1378. }
1380. **this**.powPdNop.addEventListener("click", powPdNopf.bind(**this**));
1381. **function** powPdNopf() {
1382. **if** (signsNow < 300) {
1383. **if** (curPlace.inner.length != 0) {
1384. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1385. **if** (isE) {
1386. curPlace.inner.push(**new** power([**new** par(), **new** operator("/"), **new** numb("e", 1)], curPlace));
1387. removeAllf();
1388. }
1389. **else** **if** (isPI) {
1390. curPlace.inner.push(**new** power([**new** par(), **new** operator("/"), **new** numb("pi", 1)], curPlace));
1391. removeAllf();
1392. }
1393. **else** {
1394. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1395. curPlace.inner.push(**new** power([**new** par(), **new** operator("/"), **new** numb("number", toTheNum(custN, custNafter, vedZeros,bll(hasMinus)))], curPlace));
1396. removeAllf();
1397. }
1398. }
1399. }
1400. }
1401. }
1402. }
1404. **this**.nPowf.addEventListener("click", nPowff.bind(**this**));
1405. **function** nPowff() {
1406. **if** (signsNow < 300) {
1407. **if** (curPlace.inner.length == 0) {
1408. **if** (isE) {
1409. curPlace.inner.push(**new** funct("power", **new** numb("e", 1), curPlace));
1410. removeAllf();
1411. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1412. }
1413. **else** **if** (isPI) {
1414. curPlace.inner.push(**new** funct("power", **new** numb("pi", 1), curPlace));
1415. removeAllf();
1416. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1417. }
1418. **else** {
1419. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1420. curPlace.inner.push(**new** funct("power", **new** numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))),curPlace));
1421. removeAllf();
1422. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1423. }
1424. }
1425. }
1426. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1427. **if** (isE) {
1428. curPlace.inner.push(**new** funct("power", **new** numb("e", 1), curPlace));
1429. removeAllf();
1430. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1431. }
1432. **else** **if** (isPI) {
1433. curPlace.inner.push(**new** funct("power", **new** numb("pi", 1), curPlace));
1434. removeAllf();
1435. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1436. }
1437. **else** {
1438. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1439. curPlace.inner.push(**new** funct("power", **new** numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))),curPlace));
1440. removeAllf();
1441. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1442. }
1443. }
1444. }
1445. }
1446. }

1449. **this**.pPowf.addEventListener("click", pPowff.bind(**this**));
1450. **function** pPowff() {
1451. **if** (signsNow < 300) {
1452. **if** (curPlace.inner.length == 0) {
1453. curPlace.inner.push(**new** funct("power", **new** par(), curPlace));
1454. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1455. }
1456. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1457. curPlace.inner.push(**new** funct("power", **new** par(), curPlace));
1458. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1459. }
1460. }
1461. }

1464. **this**.sinf.addEventListener("click", sinff.bind(**this**));
1465. **function** sinff() {
1466. **if** (signsNow < 300) {
1467. **if** (curPlace.inner.length == 0) {
1468. curPlace.inner.push(**new** funct("sin", 0, curPlace));
1469. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1470. }
1471. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1472. curPlace.inner.push(**new** funct("sin", 0, curPlace));
1473. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1474. }
1475. }
1476. }

1479. **this**.asinf.addEventListener("click", asinff.bind(**this**));
1480. **function** asinff() {
1481. **if** (signsNow < 300) {
1482. **if** (curPlace.inner.length == 0) {
1483. curPlace.inner.push(**new** funct("asin", 0, curPlace));
1484. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1485. }
1486. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1487. curPlace.inner.push(**new** funct("asin", 0, curPlace));
1488. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1489. }
1490. }
1491. }

1494. **this**.cosf.addEventListener("click", cosff.bind(**this**));
1495. **function** cosff() {
1496. **if** (signsNow < 300) {
1497. **if** (curPlace.inner.length == 0) {
1498. curPlace.inner.push(**new** funct("cos", 0, curPlace));
1499. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1500. }
1501. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1502. curPlace.inner.push(**new** funct("cos", 0, curPlace));
1503. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1504. }
1505. }
1506. }

1509. **this**.acosf.addEventListener("click", acosff.bind(**this**));
1510. **function** acosff() {
1511. **if** (signsNow < 300) {
1512. **if** (curPlace.inner.length == 0) {
1513. curPlace.inner.push(**new** funct("acos", 0, curPlace));
1514. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1515. }
1516. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1517. curPlace.inner.push(**new** funct("acos", 0, curPlace));
1518. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1519. }
1520. }
1521. }

1524. **this**.tanf.addEventListener("click", tanff.bind(**this**));
1525. **function** tanff() {
1526. **if** (signsNow < 300) {
1527. **if** (curPlace.inner.length == 0) {
1528. curPlace.inner.push(**new** funct("tan", 0, curPlace));
1529. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1530. }
1531. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1532. curPlace.inner.push(**new** funct("tan", 0, curPlace));
1533. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1534. }
1535. }
1536. }

1539. **this**.atanf.addEventListener("click", atanff.bind(**this**));
1540. **function** atanff() {
1541. **if** (signsNow < 300) {
1542. **if** (curPlace.inner.length == 0) {
1543. curPlace.inner.push(**new** funct("atan", 0, curPlace));
1544. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1545. }
1546. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1547. curPlace.inner.push(**new** funct("atan", 0, curPlace));
1548. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1549. }
1550. }
1551. }


1555. **this**.ex\_form.addEventListener("click", ex\_formf.bind(**this**));
1556. **function** ex\_formf() {
1557. **if** (signsNow < 300) {
1558. **if** (curPlace.outer != **null** && curPlace.inner.length > 0) {
1559. **if** (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1560. curPlace.outer.inner[curPlace.outer.inner.length - 1].exit = **true**;
1561. curPlace = curPlace.outer;
1562. }
1563. }
1564. }
1565. }

1568. **this**.brac.addEventListener("click", bracf.bind(**this**));
1569. **function** bracf() {
1570. **if** (signsNow < 300) {
1571. **if** (curPlace.inner.length == 0) {
1572. curPlace.inner.push(makeBracket([], curPlace));
1573. curPlace.inner[curPlace.inner.length - 1].exit = **false**;
1574. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1575. }
1576. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1577. curPlace.inner.push(makeBracket([], curPlace));
1578. curPlace.inner[curPlace.inner.length - 1].exit = **false**;
1579. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1580. }
1581. }
1582. }

1585. **this**.epowf.addEventListener("click", epowff.bind(**this**));
1586. **function** epowff() {
1587. **if** (signsNow < 300) {
1588. **if** (curPlace.inner.length == 0) {
1589. curPlace.inner.push(**new** funct("power", **new** numb("e", 1), curPlace));
1590. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1591. }
1592. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1593. curPlace.inner.push(**new** funct("power", **new** numb("e", 1), curPlace));
1594. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1595. }
1596. }
1597. }

1600. **this**.logPPNf.addEventListener("click", logPPNff.bind(**this**));
1601. **function** logPPNff() {
1602. **if** (signsNow < 300) {
1603. **if** (curPlace.inner.length == 0) {
1604. **if** (isE) {
1605. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("+"), **new** numb("e", 1)], curPlace), curPlace));
1606. removeAllf();
1607. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1608. }
1609. **else** **if** (isPI) {
1610. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("+"), **new** numb("pi", 1)], curPlace), curPlace));
1611. removeAllf();
1612. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1613. }
1614. **else** {
1615. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1616. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("+"), **new** numb("number", toTheNum(custN,custNafter, vedZeros, bll(hasMinus)))], curPlace), curPlace));
1617. removeAllf();
1618. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1619. }
1620. }
1621. }
1622. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1623. **if** (isE) {
1624. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("+"), **new** numb("e", 1)], curPlace), curPlace));
1625. removeAllf();
1626. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1627. }
1628. **else** **if** (isPI) {
1629. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("+"), **new** numb("pi", 1)], curPlace), curPlace));
1630. removeAllf();
1631. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1632. }
1633. **else** {
1634. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1635. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("+"), **new** numb("number", toTheNum(custN,custNafter, vedZeros, bll(hasMinus)))], curPlace), curPlace));
1636. removeAllf();
1637. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1638. }
1639. }
1640. }
1641. }
1642. }

1645. **this**.logPxNf.addEventListener("click", logPxNff.bind(**this**));
1646. **function** logPxNff() {
1647. **if** (signsNow < 300) {
1648. **if** (curPlace.inner.length == 0) {
1649. **if** (isE) {
1650. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("\*"), **new** numb("e", 1)], curPlace), curPlace));
1651. removeAllf();
1652. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1653. }
1654. **else** **if** (isPI) {
1655. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("\*"), **new** numb("pi", 1)], curPlace), curPlace));
1656. removeAllf();
1657. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1658. }
1659. **else** {
1660. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1661. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("\*"), **new** numb("number", toTheNum(custN,custNafter, vedZeros, bll(hasMinus)))], curPlace), curPlace));
1662. removeAllf();
1663. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1664. }
1665. }
1666. }
1667. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1668. **if** (isE) {
1669. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("\*"), **new** numb("e", 1)], curPlace), curPlace));
1670. removeAllf();
1671. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1672. }
1673. **else** **if** (isPI) {
1674. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("\*"), **new** numb("pi", 1)], curPlace), curPlace));
1675. removeAllf();
1676. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1677. }
1678. **else** {
1679. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1680. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** operator("\*"), **new** numb("number", toTheNum(custN,custNafter, vedZeros, bll(hasMinus)))], curPlace), curPlace));
1681. removeAllf();
1682. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1683. }
1684. }
1685. }
1686. }
1687. }

1690. **this**.lnf.addEventListener("click", lnff.bind(**this**));
1691. **function** lnff() {
1692. **if** (signsNow < 300) {
1693. **if** (curPlace.inner.length == 0) {
1694. curPlace.inner.push(**new** funct("ln", 1, curPlace));
1695. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1696. }
1697. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1698. curPlace.inner.push(**new** funct("ln", 1, curPlace));
1699. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1700. }
1701. }
1702. }

1705. **this**.lognf.addEventListener("click", lognff.bind(**this**));
1706. **function** lognff() {
1707. **if** (signsNow < 300) {
1708. **if** (curPlace.inner.length == 0) {
1709. **if** (isE) {
1710. curPlace.inner.push(**new** funct("ln", 1, curPlace));
1711. removeAllf();
1712. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1713. }
1714. **else** **if** (isPI) {
1715. curPlace.inner.push(**new** funct("log", makeBracket([**new** numb("pi", 1)], curPlace), curPlace));
1716. removeAllf();
1717. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1718. }
1719. **else** {
1720. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) > 0 && (toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) !=1) {
1721. curPlace.inner.push(**new** funct("log", makeBracket([**new** numb("number", toTheNum(custN, custNafter, vedZeros,bll(hasMinus)))], curPlace), curPlace));
1722. removeAllf();
1723. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1724. }
1725. }
1726. }
1727. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1728. **if** (isE) {
1729. curPlace.inner.push(**new** funct("ln", 1, curPlace));
1730. removeAllf();
1731. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1732. }
1733. **else** **if** (isPI) {
1734. curPlace.inner.push(**new** funct("log", makeBracket([**new** numb("pi", 1)], curPlace), curPlace));
1735. removeAllf();
1736. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1737. }
1738. **else** {
1739. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1740. curPlace.inner.push(**new** funct("log", makeBracket([**new** numb("number", toTheNum(custN, custNafter, vedZeros,bll(hasMinus)))], curPlace), curPlace));
1741. removeAllf();
1742. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1743. }
1744. }
1745. }
1746. }
1747. }

1750. **this**.logpf.addEventListener("click", logpff.bind(**this**));
1751. **function** logpff() {
1752. **if** (signsNow < 300) {
1753. **if** (curPlace.inner.length == 0) {
1754. curPlace.inner.push(**new** funct("log", makeBracket([**new** par()], curPlace), curPlace));
1755. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1756. }
1757. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1758. curPlace.inner.push(**new** funct("log", makeBracket([**new** par()], curPlace), curPlace));
1759. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1760. }
1761. }
1762. }

1765. **this**.loppownf.addEventListener("click", loppownff.bind(**this**));
1766. **function** loppownff() {
1767. **if** (signsNow < 300) {
1768. **if** (curPlace.inner.length == 0) {
1769. **if** (isE) {
1770. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** power([**new** numb("e", 1)], curPlace)], curPlace), curPlace));
1771. removeAllf();
1772. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1773. }
1774. **else** **if** (isPI) {
1775. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** power([**new** numb("pi", 1)], curPlace)], curPlace),curPlace));
1776. removeAllf();
1777. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1778. }
1779. **else** {
1780. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1781. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** power([**new** numb("number", toTheNum(custN, custNafter,vedZeros, bll(hasMinus)))], curPlace)], curPlace), curPlace));
1782. removeAllf();
1783. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1784. }
1785. }
1786. }
1787. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1788. **if** (isE) {
1789. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** power([**new** numb("e", 1)], curPlace)], curPlace), curPlace));
1790. removeAllf();
1791. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1792. }
1793. **else** **if** (isPI) {
1794. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** power([**new** numb("pi", 1)], curPlace)], curPlace),curPlace));
1795. removeAllf();
1796. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1797. }
1798. **else** {
1799. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1800. curPlace.inner.push(**new** funct("log", makeBracket([**new** par(), **new** power([**new** numb("number", toTheNum(custN, custNafter,vedZeros, bll(hasMinus)))], curPlace)], curPlace), curPlace));
1801. removeAllf();
1802. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1803. }
1804. }
1805. }
1806. }
1807. }



1812. **this**.sqrtf.addEventListener("click", sqrtff.bind(**this**));
1813. **function** sqrtff() {
1814. **if** (signsNow < 300) {
1815. **if** (curPlace.inner.length == 0) {
1816. curPlace.inner.push(**new** funct("sqrt", 1, curPlace));
1817. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1818. }
1819. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1820. curPlace.inner.push(**new** funct("sqrt", 1, curPlace));
1821. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1822. }
1823. }
1824. }



1829. **this**.cbrtf.addEventListener("click", cbrtff.bind(**this**));
1830. **function** cbrtff() {
1831. **if** (signsNow < 300) {
1832. **if** (curPlace.inner.length == 0) {
1833. curPlace.inner.push(**new** funct("cbrt", 1, curPlace));
1834. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1835. }
1836. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1837. curPlace.inner.push(**new** funct("cbrt", 1, curPlace));
1838. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1839. }
1840. }
1841. }

1844. **this**.nrootf.addEventListener("click", nrootff.bind(**this**));
1845. **function** nrootff() {
1846. **if** (signsNow < 300) {
1847. **if** (curPlace.inner.length == 0) {
1848. **if** (isE) {
1849. curPlace.inner.push(**new** funct("root", makeBracket([**new** numb("e", 1)], curPlace), curPlace));
1850. removeAllf();
1851. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1852. }
1853. **else** **if** (isPI) {
1854. curPlace.inner.push(**new** funct("root", makeBracket([**new** numb("pi", 1)], curPlace), curPlace));
1855. removeAllf();
1856. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1857. }
1858. **else** {
1859. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1860. curPlace.inner.push(**new** funct("root", makeBracket([**new** numb("number", toTheNum(custN, custNafter, vedZeros,bll(hasMinus)))], curPlace), curPlace));
1861. removeAllf();
1862. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1863. }
1864. }
1865. }
1866. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1867. **if** (isE) {
1868. curPlace.inner.push(**new** funct("root", makeBracket([**new** numb("e", 1)], curPlace), curPlace));
1869. removeAllf();
1870. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1871. }
1872. **else** **if** (isPI) {
1873. curPlace.inner.push(**new** funct("root", makeBracket([**new** numb("pi", 1)], curPlace), curPlace));
1874. removeAllf();
1875. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1876. }
1877. **else** {
1878. **if** ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1879. curPlace.inner.push(**new** funct("root", makeBracket([**new** numb("number", toTheNum(custN, custNafter, vedZeros,bll(hasMinus)))], curPlace), curPlace));
1880. removeAllf();
1881. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1882. }
1883. }
1884. }
1885. }
1886. }

1889. **this**.prootf.addEventListener("click", prootff.bind(**this**));
1890. **function** prootff() {
1891. **if** (signsNow < 300) {
1892. **if** (curPlace.inner.length == 0) {
1893. curPlace.inner.push(**new** funct("root", makeBracket([**new** par()], curPlace), curPlace));
1894. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1895. }
1896. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1897. curPlace.inner.push(**new** funct("root", makeBracket([**new** par()], curPlace), curPlace));
1898. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1899. }
1900. }
1901. }

1904. **this**.makeDer.addEventListener("click", makeDerf.bind(**this**));
1905. **function** makeDerf() {
1906. **if** (signsNow < 300) {
1907. **for** (**var** i = 0 ; i < 10 ; ++i) {
1908. formula.inner = optimize(formula.inner, 0);
1909. }
1910. formula = findDerivative(formula, formula.outer);
1911. **for** (**var** i = 0 ; i < 10 ; ++i) {
1912. formula.inner = optimize(formula.inner, 0);
1913. }
1914. curPlace = formula;
1915. }
1916. }



1921. **function** findDerivative(formula, theOuter) {
1922. **var** index = 0;
1923. **var** total = **new** theFormula([], theOuter);
1924. **if** (formula.inner.length == 2 && formula.inner[1].ist == "power") {
1925. **return** getDer(formula, theOuter);
1926. }
1927. **else** **if** (formula.inner.length > 0) {
1928. while (index < formula.inner.length) {
1929. **var** partArray = **new** theFormula([], theOuter);
1930. **var** reversed = **false**;
1931. while (index < formula.inner.length && (formula.inner[index].ist != "operator" || (formula.inner[index].type != "+" &&formula.inner[index].type != "-"))) {
1932. **if** (formula.inner[index].ist == "operator" && formula.inner[index].type == "\*") {
1933. partArray.inner.push(formula.inner[index]);
1934. }
1935. **else** **if** (formula.inner[index].ist == "operator" && formula.inner[index].type == "/") {
1936. partArray.inner.push(**new** operator("\*"));
1937. reversed = **true**;
1938. }
1939. **else** **if** (formula.inner[index].ist == "power") {
1940. partArray.inner.push(formula.inner[index]);
1941. }
1942. **else** **if** (reversed) {
1943. **if** (formula.inner[index].ist == "root") {
1944. partArray.inner.push(formula.inner[index]);
1945. partArray.inner.push(**new** power([**new** numb("number", -1)], partArray));
1946. reversed = **false**;
1947. }
1948. **else** **if** (formula.inner[index].ist == "param") {
1949. partArray.inner.push(formula.inner[index]);
1950. partArray.inner.push(**new** power([**new** numb("number", -1)], partArray));
1951. reversed = **false**;
1952. }
1953. **else** **if** (formula.inner[index].ist == "numb") {
1954. **if** (formula.inner[index].type == "e") {
1955. partArray.inner.push(**new** numb("number", (1/2.718281828459)));
1956. partArray.inner.push(**new** power([**new** numb("number", 1)], partArray));
1957. reversed = **false**;
1958. }
1959. **else** **if** (formula.inner[index].type == "pi") {
1960. partArray.inner.push(**new** numb("number", (1/3.141592653589)));
1961. partArray.inner.push(**new** power([**new** numb("number", 1)], partArray));
1962. reversed = **false**;
1963. }
1964. **else** **if** (formula.inner[index].type == "number") {
1965. partArray.inner.push(**new** numb("number", (1/formula.inner[index].value)));
1966. partArray.inner.push(**new** power([**new** numb("number", 1)], partArray));
1967. reversed = **false**;
1968. }
1969. }
1970. **else** **if** (formula.inner[index].ist == "function") {
1971. partArray.inner.push(formula.inner[index]);
1972. partArray.inner.push(**new** power([**new** numb("number", -1)], partArray));
1973. reversed = **false**;
1974. }
1975. }
1976. **else** {
1977. partArray.inner.push(formula.inner[index]);
1978. partArray.inner.push(**new** power([**new** numb("number", 1)], partArray));
1979. }
1980. ++index;
1981. }
1983. *//shrink double powers*
1984. **for** (**var** i = 1 ; i < partArray.inner.length ; ++i) {
1985. **if** (partArray.inner[i-1].ist == "power" && partArray.inner[i].ist == "power") {
1986. **var** newPower = **new** power([], partArray);
1988. **var** part1 = partArray.inner[i-1].pow\_to.inner;
1989. **var** brPart1 = makeBracket(part1, partArray);
1991. newPower.pow\_to.inner.push(brPart1);
1992. newPower.pow\_to.inner.push(**new** operator("\*"));
1994. **var** part2 = partArray.inner[i].pow\_to.inner;
1995. **var** brPart2 = makeBracket(part2, partArray);
1997. newPower.pow\_to.inner.push(brPart2);
1998. partArray.inner[i-1] = newPower;
1999. partArray.inner.splice(i, 1);
2000. --i;
2001. }
2002. }
2004. **var** derivResult = **new** theFormula([], total);
2005. **var** firstVal = **new** theFormula([], derivResult);
2006. **var** firstDer = **new** theFormula([], derivResult);
2007. **var** secondVal = **new** theFormula([], derivResult);
2008. **var** secondDer = **new** theFormula([], derivResult);
2010. firstVal.inner.push(partArray.inner[0]);
2012. **for** (**var** i = 0 ; i < 5 ; ++i) {
2013. optimize(partArray.inner[1], 0);
2014. }
2015. firstVal.inner.push(partArray.inner[1]);
2017. partArray.inner.splice(0, 3);
2019. secondVal = partArray;
2021. **if** (partArray.inner.length == 2) {
2022. **for** (**var** i = 0 ; i < 5 ; ++i) {
2023. optimize(partArray.inner[1].pow\_to.inner, 0);
2024. }
2025. }

2028. firstDer = getDer(firstVal, derivResult);
2029. secondDer = getDer(secondVal, derivResult);

2032. **for** (**var** i = 0 ; i < 5 ; ++i) {
2033. optimize(firstDer.inner, 0);
2034. optimize(secondDer.inner, 0);
2035. }

2038. **if** (secondVal.inner.length > 0) {
2039. derivResult.inner.push(makeBracket(firstDer.inner, derivResult));
2040. derivResult.inner.push(**new** operator("\*"));
2041. derivResult.inner.push(makeBracket(secondVal.inner, derivResult));
2042. derivResult.inner.push(**new** operator("+"));
2043. derivResult.inner.push(makeBracket(secondDer.inner, derivResult));
2044. derivResult.inner.push(**new** operator("\*"));
2045. derivResult.inner.push(makeBracket(firstVal.inner, derivResult));
2046. }
2047. **else** {
2048. derivResult.inner.push(makeBracket(firstDer.inner, derivResult));
2049. }

2052. **for** (**var** i = 0 ; i < 10 ; ++i) {
2053. *//optimize(derivResult.inner, 0);*
2054. }

2057. total.inner.push(makeBracket(derivResult.inner, total));
2059. **if** (index < formula.inner.length) {
2060. **if** (formula.inner[index].ist == "operator" && formula.inner[index].type == "+") {
2061. total.inner.push(**new** operator("+"));
2062. }
2063. **else** **if** (formula.inner[index].ist == "operator" && formula.inner[index].type == "-") {
2064. total.inner.push(**new** operator("-"));
2065. }
2066. }
2068. ++index;
2070. }
2072. **return** total;
2074. }
2075. }




2081. **function** getDer(formula, out) {
2082. **if** (formula.inner.length == 2) {
2083. **var** result = **new** theFormula([], out);
2085. **var** powFormula = copyF(formula.inner[1].pow\_to);
2087. **var** inBracket1 = makeBracket(copyF(powFormula).inner, result);

2090. **for** (**var** i = 0; i < 10 ; ++i) {
2091. inBracket1.inner.inner = optimize(inBracket1.inner.inner, 0);
2092. }

2095. result.inner.push(inBracket1);
2096. result.inner.push(**new** operator("\*"));

2099. result.inner.push(copyThis(formula.inner[0], result));


2103. **var** inBracket2 = makeBracket(copyF(powFormula).inner, result);

2106. **var** newOpted = [inBracket2, **new** operator("-"), **new** numb("number", 1)]
2108. **for** (**var** i = 0; i < 10 ; ++i) {
2109. newOpted = optimize(newOpted, 0);
2110. }
2111. result.inner.push(**new** power(newOpted, result));
2113. **if** (formula.inner[0].ist == "function") {
2114. result.inner.push(**new** operator("\*"));
2115. **var** brDerInner = makeBracket(findDerivative(formula.inner[0].inner, result).inner, result);
2116. result.inner.push(brDerInner);
2117. }
2118. result.inner.push(**new** operator("\*"));
2120. result.inner.push(funcDer(formula.inner[0], result));
2122. **return** result;
2123. }
2124. **else** **if** (formula.inner.length < 2) {
2125. **return** **new** theFormula([**new** numb("number", 0)], **null**);
2126. }
2127. **else** {
2128. **return** findDerivative(formula, formula.outer);
2129. }
2130. }




2136. **function** funcDer(object, nOuter) {
2137. **if** (object.ist == "root") {
2138. **return** **new** numb("number", 1);
2139. }
2140. **else** **if** (object.ist == "param") {
2141. **return** **new** numb("number", 0);
2142. }
2143. **else** **if** (object.ist == "numb") {
2144. **return** **new** numb("number", 0);
2145. }*//производные функций*
2146. **else** **if** (object.ist == "function") {
2147. **if** (object.type == "bracket") {
2148. **return** **new** numb("number", 1);
2149. }
2150. **else** **if** (object.type == "sin") {
2151. **var** newBRF = makeBracket([], nOuter);
2152. newBRF.exit = **true**;
2153. **var** newFF = **new** funct("cos", 0, newBRF.inner);
2154. newFF.inner.inner = copyArr(object.inner.inner, newFF.inner);
2155. newFF.exit = **true**;
2156. newBRF.inner = **new** theFormula([**new** numb("number", 1), **new** operator("\*"), newFF], nOuter);
2157. newFF.inner.outer = newBRF.inner;
2159. **return** newBRF;
2160. }
2161. **else** **if** (object.type == "cos") {
2162. **var** newBRF = makeBracket([], nOuter);
2163. newBRF.exit = **true**;
2164. **var** newFF = **new** funct("sin", 0, newBRF.inner);
2165. newFF.inner.inner = copyArr(object.inner.inner, newFF.inner);
2166. newFF.exit = **true**;
2167. newBRF.inner = **new** theFormula([**new** numb("number", -1), **new** operator("\*"), newFF], nOuter);
2168. newFF.inner.outer = newBRF.inner;
2170. **return** newBRF;
2171. }
2172. **else** **if** (object.type == "tan") {
2173. **var** newBRF = makeBracket([], nOuter);
2174. newBRF.exit = **true**;
2175. **var** newFF = **new** funct("cos", 0, newBRF.inner);
2176. newFF.inner.inner = copyArr(object.inner.inner, newFF.inner);
2177. newFF.exit = **true**;
2178. newBRF.inner = **new** theFormula([**new** numb("number", 1), **new** operator("/"), makeBracket([newFF, **new** power([**new** numb("number", 2)],newBRF.inner)])], nOuter);
2179. newFF.inner.outer = newBRF.inner;
2181. **return** newBRF;
2182. }
2183. **else** **if** (object.type == "asin") {
2184. **var** newBRF = makeBracket([], nOuter);
2185. newBRF.exit = **true**;
2186. **var** newInner2 = **new** theFormula([], nOuter)
2187. **var** newInner = **new** theFormula([], newInner2)
2188. newInner.inner = [**new** numb("number", 1), **new** operator("-"), makeBracket(copyArr(object.inner.inner, newInner), newInner), **new**power([**new** numb("number", 2)], 1)];
2189. **var** newFF = **new** funct("sqrt", 0, newInner.inner);
2190. newFF.exit = **true**;
2192. newFF.inner.inner = copyArr(newInner.inner, newFF.inner);
2193. newInner2.inner = [**new** numb("number", 1), **new** operator("/"), newFF];
2194. newBRF.inner = copyF(newInner2);
2196. **return** newBRF;
2197. }
2198. **else** **if** (object.type == "acos") {
2199. **var** newBRF = makeBracket([], nOuter);
2200. newBRF.exit = **true**;
2201. **var** newInner2 = **new** theFormula([], nOuter)
2202. **var** newInner = **new** theFormula([], newInner2)
2203. newInner.inner = [**new** numb("number", 1), **new** operator("-"), makeBracket(copyArr(object.inner.inner, newInner), newInner), **new**power([**new** numb("number", 2)], 1)];
2204. **var** newFF = **new** funct("sqrt", 0, newInner.inner);
2205. newFF.exit = **true**;
2207. newFF.inner.inner = copyArr(newInner.inner, newFF.inner);
2208. newInner2.inner = [**new** numb("number", -1), **new** operator("/"), newFF];
2209. newBRF.inner = copyF(newInner2);
2211. **return** newBRF;
2212. }
2213. **else** **if** (object.type == "atan") {
2214. **var** newBRF = makeBracket([], nOuter);
2215. newBRF.exit = **true**;
2216. **var** newInner2 = **new** theFormula([], nOuter)
2217. **var** newInner = **new** theFormula([], newInner2)
2218. newInner.inner = [**new** numb("number", 1), **new** operator("+"), makeBracket(copyArr(object.inner.inner, newInner), newInner), **new**power([**new** numb("number", 2)], 1)];
2219. **var** newFF = makeBracket([], newInner.inner);
2220. newFF.inner.inner = copyArr(newInner.inner, newFF.inner);
2221. newInner2.inner = [**new** numb("number", 1), **new** operator("/"), newFF];
2222. newBRF.inner = copyF(newInner2);
2223. **return** newBRF;
2224. }
2225. **else** **if** (object.type == "power") {
2226. **var** newBRF = makeBracket([], nOuter);
2227. newBRF.exit = **true**;
2229. **var** objCopy = copyThis(object, newBRF.inner);
2231. **var** lnFunc = **new** funct("ln", 1, newBRF.inner);
2232. lnFunc.inner.inner = [object.arg];
2233. lnFunc.exit = **true**;
2235. newBRF.inner.inner = [copyThis(objCopy, newBRF.inner), **new** operator("\*"), lnFunc];

2238. **return** newBRF;
2239. }
2240. **else** **if** (object.type == "ln") {
2241. **var** newBRF = makeBracket([], nOuter);
2242. newBRF.exit = **true**;
2243. **var** newInner = **new** theFormula([], nOuter);
2245. **var** inF = copyF(object.inner);
2246. **var** newBr = makeBracket(inF.inner, newBRF.inner);
2248. newInner.inner = [**new** numb("number", 1), **new** operator("/"), newBr];
2250. newBRF.inner = copyF(newInner);
2252. **return** newBRF;
2253. }
2254. **else** **if** (object.type == "log") {
2255. **var** newBRF = makeBracket([], nOuter);
2256. newBRF.exit = **true**;
2257. **var** newInner = **new** theFormula([], nOuter);
2259. **var** inF = copyF(object.inner);
2260. **var** newBr = makeBracket(copyArr(inF.inner), newBRF.inner);
2262. **var** brackUnder = makeBracket([], newBRF.inner);
2264. **var** dopLn = **new** funct("ln", 0, brackUnder.inner);
2266. dopLn.inner = copyF(object.arg.inner);
2268. brackUnder.inner.inner = [newBr, **new** operator("\*"), dopLn];
2270. newInner.inner = [**new** numb("number", 1), **new** operator("/"), brackUnder];
2272. newBRF.inner = copyF(newInner);
2274. **return** newBRF;
2275. }
2276. **else** **if** (object.type == "sqrt") {
2277. **var** newBRF = makeBracket([], nOuter);
2278. newBRF.exit = **true**;
2280. **var** objCopy = copyThis(object, newBRF.inner);
2282. **var** brack = makeBracket([], newBRF.inner);
2284. brack.inner.inner = [**new** numb("number", 2), **new** operator("\*"), copyThis(objCopy, brack.inner)];
2286. newBRF.inner.inner = [**new** numb("number", 1), **new** operator("/"), copyThis(brack, newBRF.inner)];

2289. **return** newBRF;
2290. }
2291. **else** **if** (object.type == "cbrt") {
2292. **var** newBRF = makeBracket([], nOuter);
2293. newBRF.exit = **true**;

2296. **var** objCopy = copyThis(object, newBRF.inner);
2297. **var** powBracket = makeBracket([], objCopy.inner);
2299. powBracket.inner = copyF(objCopy.inner);
2300. objCopy.inner.inner = [copyThis(powBracket), **new** power([**new** numb("number", 2)], objCopy.inner)];
2302. **var** brack = makeBracket([], newBRF.inner);
2304. brack.inner.inner = [**new** numb("number", 3), **new** operator("\*"), copyThis(objCopy, brack.inner)];
2306. newBRF.inner.inner = [**new** numb("number", 1), **new** operator("/"), copyThis(brack, newBRF.inner)];

2309. **return** newBRF;
2310. }
2311. **else** **if** (object.type == "root") {
2312. **var** newBRF = makeBracket([], nOuter);
2313. newBRF.exit = **true**;

2316. **var** objCopy = copyThis(object, newBRF.inner);
2317. **var** powBracket = makeBracket([], objCopy.inner);
2319. powBracket.inner = copyF(objCopy.inner);
2320. objCopy.inner.inner = [copyThis(powBracket), **new** power([copyThis(object.arg.inner.inner[0]), **new** operator("-"), **new**numb("number", 1)], objCopy.inner)];
2322. **var** brack = makeBracket([], newBRF.inner);
2324. brack.inner.inner = [copyThis(object.arg.inner.inner[0]), **new** operator("\*"), copyThis(objCopy, brack.inner)];
2326. newBRF.inner.inner = [**new** numb("number", 1), **new** operator("/"), copyThis(brack, newBRF.inner)];

2329. **return** newBRF;
2330. }
2331. }
2332. }




2338. **this**.deleteOne.addEventListener("click", deleteOnef.bind(**this**));
2339. **function** deleteOnef() {
2340. **if** (curPlace.inner.length > 0) {
2341. **if** (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
2342. curPlace.inner.pop();
2343. }
2344. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "root") {
2345. curPlace.inner.pop();
2346. }
2347. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "param") {
2348. curPlace.inner.pop();
2349. }
2350. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "numb") {
2351. curPlace.inner.pop();
2352. }
2353. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "power") {
2354. curPlace.inner.pop();
2355. }
2356. **else** **if** (curPlace.inner[curPlace.inner.length - 1].ist == "function") {
2357. curPlace.inner[curPlace.inner.length - 1].exit = **false**;
2358. curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
2359. }
2360. }
2361. **else** **if** (curPlace.inner.length == 0 && curPlace.outer != **null**) {
2362. curPlace = curPlace.outer;
2363. curPlace.inner.pop();
2364. }
2366. *//Я без понятия, как фиксить эту х\*йню с выходами из функций*
2367. *//Уже и так несколько часов сижу, но иногда норм не удаляются*
2368. *//Легче нажать clear и ввести формулу заново*
2369. *//А вообще лучше не преобразовывать функцию в производную*
2370. *//От этого одни проблемы*
2372. *//Ибо эта проблема встречается после преобразования в производную*
2373. *//Если сам вводил формулу, то все норм удаляется*
2375. }
2377. **this**.clearAll.addEventListener("click", clearAllf.bind(**this**));
2378. **function** clearAllf() {
2379. formula = **new** theFormula([], **null**);
2380. curPlace = formula;
2381. }


2385. **this**.optimizeB.addEventListener("click", optimizeBf.bind(**this**));
2386. **function** optimizeBf() {
2387. formula.inner = optimize(formula.inner, 0);
2388. }


2392. **function** optimize(thisArr, startPos) {
2393. **if** (thisArr.length >= 3) {
2394. **for** (**var** i = 2 + startPos ; i < thisArr.length ; ++i) {
2395. **var** prev2 = thisArr[i-2];
2396. **var** prev = thisArr[i-1];
2397. **var** thisX = thisArr[i];
2399. **if** (prev.ist == "operator") {
2400. **if** (prev.type == "\*") {
2401. **if** (thisArr.length <= i+1) {
2402. **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2403. **if** (prev2.type == "number" && thisX.type == "number") {
2404. prev2.value = Math.round(prev2.value \* thisX.value\*1000000000)/1000000000;
2405. thisArr.splice(i-1, 2);
2406. i -= 1;
2407. }
2408. }
2409. }
2410. **else** **if** (thisArr[i+1].ist != "power") {
2411. **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2412. **if** (prev2.type == "number" && thisX.type == "number") {
2413. prev2.value = Math.round(prev2.value \* thisX.value\*1000000000)/1000000000;
2414. thisArr.splice(i-1, 2);
2415. i -= 1;
2416. }
2417. }
2418. }
2419. }
2420. **else** **if** (prev.type == "/") {
2421. **if** (thisArr.length <= i+1) {
2422. **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2423. **if** (prev2.type == "number" && thisX.type == "number") {
2424. prev2.value = Math.round(prev2.value / thisX.value\*1000000000)/1000000000;
2425. thisArr.splice(i-1, 2);
2426. i -= 1;
2427. }
2428. }
2429. }
2430. **else** **if** (thisArr[i+1].ist != "power") {
2431. **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2432. **if** (prev2.type == "number" && thisX.type == "number") {
2433. prev2.value = Math.round(prev2.value / thisX.value\*1000000000)/1000000000;
2434. thisArr.splice(i-1, 2);
2435. i -= 1;
2436. }
2437. }
2438. }
2439. }
2440. }
2441. }


2445. **for** (**var** i = 2 + startPos ; i < thisArr.length ; ++i) {
2446. **var** prev2 = thisArr[i-2];
2447. **var** prev = thisArr[i-1];
2448. **var** thisX = thisArr[i];
2449. **if** (prev.ist == "operator") {
2450. **if** (prev.type == "+") {
2451. **if** (thisArr.length <= i+1) {
2452. **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2453. **if** (prev2.type == "number" && thisX.type == "number") {
2454. prev2.value = Math.round((prev2.value + thisX.value)\*1000000000)/1000000000;
2455. thisArr.splice(i-1, 2);
2456. i -= 1;
2457. }
2458. }
2459. }
2460. **else** **if** (thisArr[i+1].ist != "power") {
2461. **if** (thisArr[i+1].ist == "operator") {
2462. **if** (thisArr[i+1].type != "\*" && thisArr[i+1].type != "/") {
2463. **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2464. **if** (prev2.type == "number" && thisX.type == "number") {
2465. prev2.value = Math.round((prev2.value + thisX.value)\*1000000000)/1000000000;
2466. thisArr.splice(i-1, 2);
2467. i -= 1;
2468. }
2469. }
2470. }
2471. }
2472. **else** **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2473. **if** (prev2.type == "number" && thisX.type == "number") {
2474. prev2.value = Math.round((prev2.value + thisX.value)\*1000000000)/1000000000;
2475. thisArr.splice(i-1, 2);
2476. i -= 1;
2477. }
2478. }
2479. }
2480. }
2481. **else** **if** (prev.type == "-") {
2482. **if** (thisArr.length <= i+1) {
2483. **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2484. **if** (prev2.type == "number" && thisX.type == "number") {
2485. prev2.value = Math.round((prev2.value - thisX.value)\*1000000000)/1000000000;
2486. thisArr.splice(i-1, 2);
2487. i -= 1;
2488. }
2489. }
2490. }
2491. **else** **if** (thisArr[i+1].ist != "power") {
2492. **if** (thisArr[i+1].ist == "operator") {
2493. **if** (thisArr[i+1].type != "\*" && thisArr[i+1].type != "/") {
2494. **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2495. **if** (prev2.type == "number" && thisX.type == "number") {
2496. prev2.value = Math.round((prev2.value - thisX.value)\*1000000000)/1000000000;
2497. thisArr.splice(i-1, 2);
2498. i -= 1;
2499. }
2500. }
2501. }
2502. }
2503. **else** **if** (prev2.ist == "numb" && thisX.ist == "numb") {
2504. **if** (prev2.type == "number" && thisX.type == "number") {
2505. prev2.value = Math.round((prev2.value - thisX.value)\*1000000000)/1000000000;
2506. thisArr.splice(i-1, 2);
2507. i -= 1;
2508. }
2509. }
2510. }
2511. }
2512. }
2513. }

2516. **for** (**var** k = 0 ; k < 5 ; ++k) {
2518. **for** (**var** i = 1 + startPos ; i < thisArr.length ; ++i) {
2519. **var** prev = thisArr[i-1];
2520. **var** thisX = thisArr[i];
2521. **if** (prev.ist == "numb") {
2522. **if** (prev.type == "number") {
2523. **if** (thisX.ist == "power") {
2524. **if** (thisX.pow\_to.inner.length == 1 && thisX.pow\_to.inner[0].ist == "numb") {
2525. **if** (thisX.pow\_to.inner[0].type == "number") {
2526. prev.value = Math.round(Math.pow(prev.value, thisX.pow\_to.inner[0].value)\*1000000000)/1000000000;
2527. thisArr.splice(i, 1);
2528. i -= 1;
2529. }
2530. }
2531. }
2532. }
2533. }
2534. }
2536. **for** (**var** i = 1 + startPos ; i < thisArr.length ; ++i) {
2537. **var** prev = thisArr[i-1];
2538. **var** thisX = thisArr[i];
2539. **var** num0 = **new** numb("number", 0);
2540. **var** num1 = **new** numb("number", 1);
2542. **if** (thisX.ist == "power" && thisX.pow\_to.inner.length == 1 && thisX.pow\_to.inner[0].ist == "numb" &&thisX.pow\_to.inner[0].type == "number" && thisX.pow\_to.inner[0].value == 0) {
2543. **if** (prev.ist != "power") {
2544. thisArr[i-1] = copyThis(num1);
2545. thisArr.splice(i, 1);
2546. }
2547. **else** {
2548. thisArr.splice(i-1, 1);
2549. }
2550. --i;
2551. }
2552. **else** **if** (thisX.ist == "power" && thisX.pow\_to.inner.length == 1 && thisX.pow\_to.inner[0].ist == "numb" &&thisX.pow\_to.inner[0].type == "number" && thisX.pow\_to.inner[0].value == 1) {
2553. thisArr.splice(i, 1);
2554. --i;
2555. }
2557. prev = thisArr[i-1];
2558. thisX = thisArr[i];
2560. **if** (thisX.ist == "power" && prev.ist == "numb" && prev.type == "number" && (prev.value == 0 || prev.value == 1)) {
2561. thisArr.splice(i, 1);
2562. --i;
2563. }
2565. }
2567. }



2572. **for** (**var** i = 2 + startPos ; i < thisArr.length ; ++i) {
2573. **var** prev2 = thisArr[i-2];
2574. **var** prev = thisArr[i-1];
2575. **var** thisX = thisArr[i];

2578. **if** (prev2.ist == "numb") {
2579. **if** (prev2.type == "number") {
2580. **if** (prev2.value == 0) {
2581. **if** (prev.ist == "operator") {
2582. **if** (prev.type == "+") {
2583. thisArr.splice(i-2, 2);
2584. i -= 1;
2585. }
2586. **else** **if** (prev.type == "-") {
2587. prev2.value = -1;
2588. prev.type = "\*";
2589. }
2590. **else** **if** (prev.type == "\*") {
2591. thisArr.splice(i-1, 2);
2592. i -= 1;
2593. }
2594. **else** **if** (prev.type == "/") {
2595. thisArr.splice(i-1, 2);
2596. i -= 1;
2597. }
2598. }
2599. }
2600. **else** **if** (prev2.value == 1) {
2601. **if** (prev.ist == "operator") {
2602. **if** (prev.type == "\*") {
2603. thisArr.splice(i-2, 2);
2604. i -= 1;
2605. }
2606. }
2607. }
2608. }
2609. }
2610. }

2613. **for** (**var** i = 2 + startPos ; i < thisArr.length ; ++i) {
2614. **var** prev2 = thisArr[i-2];
2615. **var** prev = thisArr[i-1];
2616. **var** thisX = thisArr[i];

2619. **if** (thisX.ist == "numb") {
2620. **if** (thisX.type == "number") {
2621. **if** (thisX.value == 0) {
2622. **if** (prev.ist == "operator") {
2623. **if** (prev.type == "+") {
2624. thisArr.splice(i-1, 2);
2625. i -= 1;
2626. }
2627. **else** **if** (prev.type == "-") {
2628. thisArr.splice(i-1, 2);
2629. i -= 1;
2630. }
2631. **else** **if** (prev.type == "\*") {
2632. **if** (i <= 2) {
2633. thisArr.splice(i-2, 2);
2634. i -= 1;
2635. }
2636. **else** **if** (thisArr[i-3].ist != "operator" && thisArr[i-2].ist != "power") {
2637. thisArr.splice(i-2, 2);
2638. i -= 1;
2639. }
2640. **else** **if** (thisArr[i-3].ist == "operator" && thisArr[i-3].type != "/" && thisArr[i-2].ist != "power") {
2641. thisArr.splice(i-2, 2);
2642. i -= 1;
2643. }
2644. }
2645. }
2646. }
2647. **else** **if** (thisX.value == 1) {
2648. **if** (prev.ist == "operator") {
2649. **if** (prev.type == "\*") {
2650. **if** (i <= 2) {
2651. thisArr.splice(i-1, 2);
2652. i -= 1;
2653. }
2654. **else** **if** (thisArr[i-3].ist != "operator") {
2655. thisArr.splice(i-1, 2);
2656. i -= 1;
2657. }
2658. **else** **if** (thisArr[i-3].type != "/") {
2659. thisArr.splice(i-1, 2);
2660. i -= 1;
2661. }
2662. }
2663. }
2664. }
2665. }
2666. }
2667. }
2668. }


2672. **for** (**var** k = 0 ; k < 5 ; ++k) {
2674. **for** (**var** i = 1 + startPos ; i < thisArr.length ; ++i) {
2675. **var** prev = thisArr[i-1];
2676. **var** thisX = thisArr[i];
2677. **if** (prev.ist == "numb") {
2678. **if** (prev.type == "number") {
2679. **if** (thisX.ist == "power") {
2680. **if** (thisX.pow\_to.inner.length == 1 && thisX.pow\_to.inner[0].ist == "numb") {
2681. **if** (thisX.pow\_to.inner[0].type == "number") {
2682. prev.value = Math.round(Math.pow(prev.value, thisX.pow\_to.inner[0].value)\*1000000000)/1000000000;
2683. thisArr.splice(i, 1);
2684. i -= 1;
2685. }
2686. }
2687. }
2688. }
2689. }
2690. }
2692. **for** (**var** i = 1 + startPos ; i < thisArr.length ; ++i) {
2693. **var** prev = thisArr[i-1];
2694. **var** thisX = thisArr[i];
2695. **var** num0 = **new** numb("number", 0);
2696. **var** num1 = **new** numb("number", 1);
2698. **if** (thisX.ist == "power" && thisX.pow\_to.inner.length == 1 && thisX.pow\_to.inner[0].ist == "numb" &&thisX.pow\_to.inner[0].type == "number" && thisX.pow\_to.inner[0].value == 0) {
2699. **if** (prev.ist != "power") {
2700. thisArr[i-1] = copyThis(num1);
2701. thisArr.splice(i, 1);
2702. }
2703. **else** {
2704. thisArr.splice(i-1, 1);
2705. }
2706. --i;
2707. }
2708. **else** **if** (thisX.ist == "power" && thisX.pow\_to.inner.length == 1 && thisX.pow\_to.inner[0].ist == "numb" &&thisX.pow\_to.inner[0].type == "number" && thisX.pow\_to.inner[0].value == 1) {
2709. thisArr.splice(i, 1);
2710. --i;
2711. }
2713. prev = thisArr[i-1];
2714. thisX = thisArr[i];
2716. **if** (thisX.ist == "power" && prev.ist == "numb" && prev.type == "number" && (prev.value == 0 || prev.value == 1)) {
2717. thisArr.splice(i, 1);
2718. --i;
2719. }
2721. }
2723. }

2726. **for** (**var** i = 1 + startPos ; i < thisArr.length ; ++i) {
2727. **var** prev = thisArr[i-1];
2728. **var** thisX = thisArr[i];
2729. **if** (prev.ist == "operator") {
2730. **if** (prev.type == "+") {
2731. **if** (thisX.ist == "numb") {
2732. **if** (thisX.type == "number") {
2733. **if** (thisX.value < 0) {
2734. prev.type = "-";
2735. thisX.value \*= -1;
2736. }
2737. }
2738. }
2739. }
2740. **else** **if** (prev.type == "-") {
2741. **if** (thisX.ist == "numb") {
2742. **if** (thisX.type == "number") {
2743. **if** (thisX.value < 0) {
2744. prev.type = "+";
2745. thisX.value \*= -1;
2746. }
2747. }
2748. }
2749. }
2750. }
2751. }


2755. **for** (**var** i = startPos ; i < thisArr.length ; ++i) {
2756. **if** (thisArr[i].ist == "function") {
2757. thisArr[i].inner.inner = optimize(thisArr[i].inner.inner, 0);
2758. }
2759. }
2761. **for** (**var** i = startPos ; i < thisArr.length ; ++i) {
2762. **if** (thisArr[i].ist == "power") {
2763. thisArr[i].pow\_to.inner = optimize(thisArr[i].pow\_to.inner, 0);
2764. }
2765. }
2767. **for** (**var** i = startPos ; i < thisArr.length ; ++i) {
2768. **if** (thisArr[i].ist == "function" && thisArr[i].type == "bracket") {
2769. **if** (thisArr[i].inner.inner.length == 1) {
2770. **var** outerNew;
2771. **if** (thisArr[i].inner.inner[0].ist == "function") {
2772. outerNew = thisArr[i].inner.outer;
2773. thisArr[i].inner.inner[0].inner.outer = thisArr[i].inner.outer;
2774. }
2775. thisArr[i] = thisArr[i].inner.inner[0];
2776. **if** (thisArr[i].ist == "function") {
2777. thisArr[i].inner.outer = outerNew;
2778. }
2779. }
2780. }
2781. }
2783. **for** (**var** i = startPos ; i < thisArr.length ; ++i) {
2784. **if** (thisArr[i].ist == "function" && thisArr[i].type == "ln") {
2785. **if** (thisArr[i].inner.inner.length == 1 && thisArr[i].inner.inner[0].ist == "numb" && thisArr[i].inner.inner[0].type == "e") {
2786. thisArr[i] = **new** numb("number", 1);
2787. }
2788. **else** **if** (thisArr[i].inner.inner.length == 1 && thisArr[i].inner.inner[0].ist == "numb" && thisArr[i].inner.inner[0].type =="number" && thisArr[i].inner.inner[0].value == 1) {
2789. thisArr[i] = **new** numb("number", 0);
2790. }
2791. }
2792. }
2794. **return** thisArr;
2795. }
2797. **this**.prestigeB.addEventListener("click", pviss.bind(**this**));
2798. **function** pviss() {
2799. **if** (!pVis) {
2800. pVis = **true**;
2801. }
2802. **else** pVis = **false**;
2803. }

2806. **function** sign(x) {
2807. **return** (x>0 ? 1 : -1);
2808. }
2810. **function** short(num) {
2811. **if** (num < 1000) {
2812. **return** Math.round(num);
2813. }
2814. **else** **if** (num < 1e6) {
2815. **return** Math.round(num/100)/10 + " K";
2816. }
2817. **else** **if** (num < 1e9) {
2818. **return** Math.round(num/1e5)/10 + " M";
2819. }
2820. **else** **if** (num < 1e12) {
2821. **return** Math.round(num/1e8)/10 + " B";
2822. }
2823. **else** **if** (num < 1e15) {
2824. **return** Math.round(num/1e11)/10 + " T";
2825. }
2826. **else** **if** (num < 1e18) {
2827. **return** Math.round(num/1e14)/10 + " Qa";
2828. }
2829. **else** **if** (num < 1e21) {
2830. **return** Math.round(num/1e17)/10 + " Qi";
2831. }
2832. **else** **if** (num < 1e24) {
2833. **return** Math.round(num/1e20)/10 + " Sx";
2834. }
2835. **else** **if** (num < 1e27) {
2836. **return** Math.round(num/1e23)/10 + " Sp";
2837. }
2838. **else** **if** (num < 1e30) {
2839. **return** Math.round(num/1e26)/10 + " Oc";
2840. }
2841. **else** **if** (num < 1e33) {
2842. **return** Math.round(num/1e29)/10 + " No";
2843. }
2844. **else** **if** (num < 1e36) {
2845. **return** Math.round(num/1e32)/10 + " Dc";
2846. }
2847. **else** {
2848. **var** numPow = Math.floor((Math.log(num) / Math.log(10)))
2849. **return** Math.round(num/Math.pow(10,numPow-1))/10 + "e" + numPow;
2850. }
2851. }