1 Схема программы

Сначало в оперативную память загружается, путем ручного ввода, автоматического построения, или считывания из файла, некое алгебраическое выражение.
Его превращают в дерево
Далее создается новое дерево так, чтобы после превращения его обратно в алгебраическое выражение, оно соответствовало результату дифференцирования.

2 Таблица

type	метод	value			
		0	1	2	3
0	цифры	значение числа			
1	операция	$+$	$-$	$*$	$/$
2	переменная	x_1	x_2	x_3	x_4
3	функция	ln	sin	cos	exp

3 Принцип построения

Аргумент функции в её левом поддереве а правое НУЛЛ
Числитель в левом, а знаменатель в правом
Иные случаи: $a \cdot b$ или $a + b$, то a в левом, а b в правом

4 Методы class node (все публичные)

4.1 Конструктор

Принемает параметры int тип , и int значение
yzel(int tip, int znach){
type = tip;
value = znach;
left = NULL;
right = NULL;
std::cerr«"New element created.«<std::endl;

}

4.2 Диструктор

Выводит сообщение:
std::cerr«"One element deleted.«<std::endl;

5 Публичные Методы class sTree

5.1 Конструктор

Принемает параметры int тип , и int значение
dif(int tip, int znach){
root = new yzel(tip, znach);
std::cerr«"Vyzvan konstryktor dly classa dif.«<std::endl;
}

5.2 Диструктор

Вызывает рекурсивное удаление дерева, отсылая адрес корня dif()
delete_tree(this->root);
std::cerr«"Vyzvan distryktor dly classa dif.«<std::endl;
}

5.3 print_dif()

Выполняет вызывает функцию view, которая выполняет рекурсивное распечатка всего дерева Тип void

5.4 sozdanye_bazovogo_dereva()

void строит $((ln(2)) * x) + (sin(x))/48))$

5.5 sTree* differentiate()

Возвращает адресс дифференцированного дерева сама же создаёт корневой элемент , и вызывает рекурсию (но это пока еще не решено

Внимание к ней требуется в конце программы вызвать delete (адрес который она вернула)

6 Частные Методы class sTree

6.1 delete_tree(yzel* cur)

Выполняет рекурсивное удаление всего дерева Тип void

6.2 view()

Выполняет рекурсивное распечатка всего дерева в графическом представлении Тип void

6.3 void rek_d()

просит команды дифференцирования, ведя его рекурсивно, к концу работы создаёт полностью готовое дерево

6.4 void copy(yzel* stemp, yzel* ftemp)

получает корень дерева(поддерева), которое нужно подшить и корень тоже к некому корню, болванке

6.5 void x_search(yzel* tempr, int* counter)

получает адрес некоторыей узел и адрес некоторого числа типа инт, которое перед передачей присваивают значение нуль,и обыскивает его и его подузлы на наличиче переменных

6.6 void view_helper(yzel* cur);

Выводит на экран некоторые значения класса текущего узла, преобразовав их графический вид делает чтение графического дерева более понятным