МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»
Тема: «Иерархические списки»

Студент гр. 8381 _____ Сосновский Д.Н.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы

Ознакомиться со структурой иерархического списка. Создать программу, позволяющую искать производную выражения по заданной переменной.

Задание

Символьное дифференцирование алгебраического выражения, рассматриваемого как функция от одной из переменных. На входе выражение в виде иерархического списка и переменная, по которой следует дифференцировать. На выходе – производная исходного выражения. После дифференцирования возможно упрощение выражения. В индивидуальном задании набор операций (функций), которые могут входить в выражение, определяется преподавателем.

25) +, -, *, упростить после дифференцирования

Ход работы

Разработка программы производилась на базе ОС Windows 10 в среде разработки QtCreator с использованием фреймворка Qt.

Для удобства использования программы был разработан GUI с помощью редактора UI – форм, встроенного в используемую среду разработки. GUI представляет собой три текстовых поля – одно для ввода текста самой функции, второй для ввода переменной, по которой будет произведено дифференцирование, и третье для вывода производной функции. В дополнение к этому есть элемент CheckBox, позволяющий пользователю отметить, желает он упрощения выражения полученной производной или нет.

Алгоритм дифференцирования

Выражение представляется в виде иерархического списка – дерева с операциями. Создается объект класса Function, имеющий своим полем объект QString с записанной функцией и указателем на корень иерархического списка. Дифференцирование начинается с корня: вызываются функции derivate() дифференцирования его потомков в соответствии с правилом дифференцирования данной операции. И так продолжается спуск вниз по дереву, до тех пор, пока не будут достигнуты «листья» дерева – они являются числами, либо переменными. Их дифференциал определяется в соответствии с тем, чем они являются. Так, например, если дифференцирование происходит по переменной x, то производная от 1 будет равна 0 и т. д.

Далее программа «заворачивает» скобками полученные функции в другой иерархический список, и обращается к флагу об упрощении. Если необходимо упростить полученную функцию, то создается ещё один объект Function, являющийся функцией производной, и вызывается метод simplify(), который возвращает объект класса QString – упрощенное выражение.

Упрощенное выражение представляет собой строку, записанную со всеми раскрытыми скобками и отсутствием нулевых коэффициентов при переменных.

Оценка эффективности

Реализованный алгоритм имеет сложность $O(n)$. Это обусловлено тем, что при дифференцировании мы обращаемся к каждому узлу списка по одному разу.

Тестирование программы

Программа была успешно протестирована на различных тестах, все данные о тестировании собраны в приложении в разделе «Тестирование»

Вывод

В ходе выполнения данной лабораторной работы была изучена структура иерархического списка, а также разработана программа с удобным GUI, позволяющая пользователю получать производные функций с предусмотренной возможностью упрощения полученного выражения.

ПРИЛОЖЕНИЕ

Тестирование

В данном приложении приведена информация о тестировании программы.

№	Входные данные	Выходные данные
1	(x+x) Производная по x без упрощения	(1+1)
2	(x+x) Производная по x с упрощением	2
3	(x*(x*x)) Производная по x без упрощения	((1*(x*x))+(x*((1*x)+(x*1))))
4	(x*(x*x)) Производная по x с упрощением	3*x*x
5	((5+x)*(5+(x*x))) Производная по x с упрощением	5+10*x+3*x*x

6	(x+(y*(r+d))) Производная по y с упрощением	1*d+1*r
7	(x*(x*(x+x))) Производная по x с упрощением	6*x*x