|  |
| --- |
| Московский Государственный университет, факультет вычислительной математики и кибернетики |
| Нахождение производной простейшей функции |
|  |
|  |
| **Владимир Терёшин** |
| **03.03.2012** |

|  |
| --- |
|  |

Оглавление

[Постановка задачи 2](#_Toc318581985)

[Математическое обоснование 2](#_Toc318581986)

[Описание программы 2](#_Toc318581987)

[Тестирование 3](#_Toc318581988)

[Программа на языке СИ 3](#_Toc318581989)

[Использованная литература 6](#_Toc318581990)

# Постановка задачи

Необходимо найти производную введённой функции, представляющую из себя многочлен. При этом необходимо представить данную функцию в виде двоичного дерева. Функции вывода и построения производной должны быть рекурсивными

# Математическое обоснование

Для вычисления производной данного многочлена необходимо применить следующие правила:

# Описание программы

struct formula

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| smb | char | Символ операции, число (от 1 до 9) или ‘x’ |
| left, right | formula \* | Указатели соответственно на левый и правый операнды функции, или NULL, если это число или ‘x’ |

Функции

|  |  |  |
| --- | --- | --- |
| Название | Параметры | Назначение |
| int is\_op | char c | Является ли переданный символ символом арифметической операции |
| void out\_formula | formula \*v | Вывод формулы с последней операцией в v |
| void out\_derivative | formula \*v | Вывод производной функции с последней операцией в v |
| formula \*input\_formula |  | Ввод формулы (a)+(b) из stdin с проверкой корректности выражения |
| formula \*next\_formula\_part |  | Ввод части выражения из stdout с проверкой корректности |
| void free\_tree | formula \*v | Удаление элементов двоичного дерева из динамической памяти |

# Тестирование

Тестирование проверки корректности выражения

|  |  |
| --- | --- |
| Тест | Результат работы программы |
| f=(x-x)(x+x) | Incorrect formula |
| f=(1+(2+(3+0)) | Incorrect formula |
| f=( | Incorrect formula |

Таким образом, проверка входного выражения на корректность вроде работает. Разумеется, здесь приведены не все тесты, но основные вердикты программы выдаются.

Тестирование вычисления производной

|  |  |  |
| --- | --- | --- |
| Тест | Результат работы программы | Примерный правильный ответ |
| *f=((x+1)\*((x-3)\*(x-5+x)))* | f’=((1+0)\*((x-3)\*((x-5)+x))+(x+1)\*((1-0)\*((x-5)+x)+(x-3)\*((1-0)+1))) |  |
| *f=(x\*(x\*(x\*(x\*x)))+x)* | f’=((1\*(x\*(x\*(x\*x)))+x\*(1\*(x\*(x\*x))+x\*(1\*(x\*x)+x\*(1\*x+x\*1))))+1) |  |
|  | f’=((1+1)\*(x-x)+(x+x)\*(1-1)) | f’=0 |

Таким образом, программа правильно считает требуемые производные, за исключением того, что не приводит подобные слагаемые. Из этих трёх тестов и простоты задаваемых функций будем считать, что программа работает правильно.

# Программа на языке СИ

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

typedef struct formula formula;

struct formula {

char smb;

formula \*left, \*right;

};

int is\_op(char c) {

return c == '+' || c == '-' || c == '\*';

}

void out\_formula(formula \*v) {

if (!is\_op(v->smb)) {

printf("%c", v->smb);

return;

}

printf("(");

out\_formula(v->left);

printf("%c", v->smb);

out\_formula(v->right);

printf(")");

}

void out\_derivative(formula \*v) {

if (v->smb == '+' || v->smb == '-') {

printf("(");

out\_derivative(v->left);

printf("%c", v->smb);

out\_derivative(v->right);

printf(")");

}

if (v->smb == '\*') {

printf("(");

out\_derivative(v->left);

printf("\*");

out\_formula(v->right);

printf("+");

out\_formula(v->left);

printf("\*");

out\_derivative(v->right);

printf(")");

}

if (v->smb == 'x') {

printf("1");

return;

}

if (isdigit(v->smb))

printf("0");

}

void free\_tree(formula \*v) {

if (!v)

return;

free\_tree(v->left);

free\_tree(v->right);

free(v);

}

formula \*input\_formula();

formula \*next\_formula\_part() {

formula \*res = NULL;

char c = getchar();

switch (c) {

case '(':

res = input\_formula();

if (getchar() != ')') {

free\_tree(res);

return NULL;

}

break;

case '0': case '1': case '2': case '3': case '4': case '5': case '6': case '7': case '8': case '9': case 'x':

res = (formula \*)malloc(sizeof(formula));

res->smb = c; res->left = res->right = NULL;

break;

default:

break;

}

return res;

}

formula \*input\_formula() {

static int dep = 0;

formula \*res = NULL;

if (!dep++) {

res = next\_formula\_part();

if (!res || getchar() != '\n') {

free\_tree(res);

return NULL;

}

dep = 0;

return res;

}

res = (formula \*)malloc(sizeof(formula)); res->left = res->right = NULL;

res->left = next\_formula\_part();

char c = getchar();

if (c == '+' || c == '-' || c == '\*')

res->smb = c;

else {

free\_tree(res);

return NULL;

}

res->right = next\_formula\_part();

if (!res->left || !res->right) {

free\_tree(res);

return NULL;

}

return res;

}

int main(void) {

printf("f=");

formula \*f\_tree = input\_formula();

if (f\_tree) {

printf("f'="); out\_derivative(f\_tree); printf("\n");

} else

printf("Incorrect formula\n");

free\_tree(f\_tree);

return 0;

}

# Использованная литература

1. Трифонов Н.П., Пильщиков В.Н. Задания практикума на ЭВМ (1 курс)
2. Wikipedia
3. Wolfram|Alpha.