|  |  |
| --- | --- |
|  | **Отчёт по лабораторной работе** № 24 по курсу Практикум на ЭВМ ­  студента группы М8О-108Б Жерлыгина Максима Андреевича, № по списку 8  Адреса www, e-mail, jabber, skype mmaxim2710@gmail.com  Работа выполнена: “15“ мая 2019г.  Преподаватель: каф.806  Входной контроль знаний с оценкой  Отчёт сдан “ “ 20 г., итоговая оценка  Подпись преподавателя |

1. **Тема**: Динамические структуры данных. Обработка деревьев
2. **Цель работы**: Составить программу выполнения заданных преобразований арифметических выражений с применением деревьев.
3. **Задание** (*вариант №* 8 ): Упростить выражения, выполнить возведение чисел в степень с целым показателем.
4. **Оборудование** (*лабораторное*):

ЭВМ компьютер , процессор Intel Core2 Duo CPU E8500 @ 3.163GHz , имя узла сети cameron с ОП 16029 МБ

НМД 2 ГБ. Терминал gnome адрес 172.16.80.213 . Принтер Лазерный с технологией pulling

Другие устройства

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Intel Core i5-7200U @ 4x 2.712GHz, ОП 8073 МБ, НМД 464 ГБ. Монитор

Другие устройства

1. **Программное обеспечение** (*лабораторное*):

Операционная система семейства Unix , наименование Ubuntu версия 16.04

Интерпретатор команд bash версия 4.3.48

Система программирования версия 8.0

Редактор текстов VIM версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства Unix , наименование Ubuntu версия 18.04

Интерпретатор команд bash версия 4.4.19

Система программирования версия 8.0

Редактор текстов VIM версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

1. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями.

Строим дерево выражения, и увидев в корне символ степени вызываем функцию возведения в степень левого сына в правого. После удаляем дерево и выводим результат.

1. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

1. Построение дерева выражения.  
2. Увидев в коре символ степни вызываем функцию возведения в степень левого сына в правого.  
3. Удаление дерева.  
4. Вывод результата.

*Пункты 1-7 отчёта составляются* ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)

**Запуск программы**

mmaxim2710@DESKTOP-RDPBU3D:~$ ./a.out

Input expression: 10^4

-----------------------

\_\_4

\_^

\\_10

-----------------------

\_10000

-----------------------

10000

mmaxim2710@DESKTOP-RDPBU3D:~$ ./a.out

Input expression: 2^3

-----------------------

\_\_3

\_^

\\_2

-----------------------

\_8

-----------------------

8

**Код программы**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE\_INPUT 1024

#define SIZE\_STR 40

typedef struct node{

char data[SIZE\_STR];

struct node \*left;

struct node \*right;

} node;

node\* add\_node(char\* dt, node\* l, node\* r);

int priority(char c);

node\* make\_tree(char\* expr, int first, int last);

void print\_tree(node\* tree, int lrc);

node\* zip\_tree(node\* tree);

int check\_int(char\* str);

void tree\_to\_expr(node\* tree, int priority\_node);

void int2str(int n, char\* s);

void reverse\_t(char\* s);

int main(int argc, char\* argv[]){

char input[SIZE\_INPUT];

node\* tree;

printf("Input expression: ");

fgets(input, SIZE\_INPUT-2, stdin);

input[strlen(input)-1] = 0;

printf("-----------------------\n");

tree = make\_tree(input, 0, strlen(input)-1);

print\_tree(tree, 0);

printf("-----------------------\n");

tree = zip\_tree(tree);

print\_tree(tree, 0);

printf("-----------------------\n");

tree\_to\_expr(tree, 1);

printf("\n\n");

return 0;

}

node\* add\_node(char \*dt, node\* l, node\* r){

node\* add = (node\*)malloc(sizeof(node));

strcpy(add->data, dt);

add->left = l;

add->right = r;

return add;

}

int priority(char c){

switch(c){

case '^': case '-': return 1;

case '\*': case '/': return 2;

case '+': return 3;

}

}

node\* make\_tree(char\* expr, int first, int last){

int minprt = 3, nest = 0, i, k, prt;

char temp[SIZE\_STR];

if(first == last){

temp[0] = expr[first];

temp[1] = 0;

return add\_node(temp, 0, 0);

}

for(i = first; i<=last; i++){

if(expr[i] == '(') {nest++; continue;}

if(expr[i] == ')') {nest--; continue;}

if(nest>0) continue;

prt = priority(expr[i]);

if(prt <= minprt) {minprt = prt; k = i;}

}

if(minprt == 3){

if((expr[first] == '(') && (expr[last] == ')'))

return make\_tree(expr, first+1, last-1);

else {

k = last-first+1;

strncpy(temp, expr+first, k);

temp[k] = 0;

return add\_node(temp, 0, 0);

}

}

temp[0] = expr[k];

temp[1] = 0;

return add\_node(temp, make\_tree(expr, first, k-1), make\_tree(expr, k+1, last));

}

void print\_tree(node\* tree, int lrc){

static int level = 0;

int i;

level++;

if (tree){

print\_tree(tree->right, 2);

for (i = 0; i<level; i++) printf(" ");

if(lrc == 1) printf("\\\_%s\n", tree->data);

else if (lrc == 2) printf("\_\_%s\n", tree->data);

else printf("\_%s\n", tree->data);

print\_tree(tree->left, 1);

}

level--;

}

node\* zip\_tree(node\* tree){

int a,b;

int number;

char temp[50];

if(tree){

a = atoi(tree->left->data);

b = atoi(tree->right->data);

number = 1;

for (int i=0 ; i<b ; ++i) {

number \*= a;

}

tree->left = NULL;

tree->right = NULL;

free(tree->left);

free(tree->right);

int2str(number, temp);

strcpy(tree->data, temp);

}

return tree;

}

int check\_int(char\* str){

int i;

for(i = 0; i<strlen(str); i++) if(!(str[i]>='0' && str[i]<='9')) return 0;

return 1;

}

void tree\_to\_expr(node\* tree, int priority\_node){

if(priority\_node == 2) printf("(");

if(tree->left) {

if(!tree->left->left && !tree->left->right) tree\_to\_expr(tree->left, 1);

else tree\_to\_expr(tree->left, priority(tree->data[0]));

}

printf("%s", tree->data);

if(tree->right) {

if(!tree->right->left && !tree->right->right) tree\_to\_expr(tree->right, 1);

else tree\_to\_expr(tree->right, priority(tree->data[0]));

}

if(priority\_node == 2) printf(")");

}

void int2str(int n, char\* s){

int i, sign;

if((sign = n) < 0) n = -n;

i = 0;

do {

s[i++] = n % 10 + '0';

} while ((n /= 10) > 0);

if (sign < 0) s[i++] = '-';

s[i] = '\0';

reverse\_t(s);

}

void reverse\_t(char\* s){

int i, j;

char c;

for (i = 0, j = strlen(s)-1; i<j; i++, j--) {

c = s[i];

s[i] = s[j];

s[j] = c;

}

}

1. **Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб.  или  дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
| 1 | дом | 15.05 | 16:34 | Неправильный формать вывода дерева, т.к. забыл удалить детей. | Удалить детей дерева функцией обнуления и “free” |  |

1. Замечание автора по существу работы
2. Выводы Я получил навыки работы с деревьями, научился писать функции для их реализации. Сделал вывод, что любое математическое можно представить в виде дерева выражений и выполнить над ним действия упрощения.

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом

Подпись студента