**Отчет по лабораторной работе № 24**

по курсу Практикум программирования

Студент группы М8О-111Б-23 Бугренков Владимир Петрович,

№ по списку 4, Контакты e-mail: vladimir.bugrenkov@yandex.ru

Работа выполнена: «18» мая 2023 г.

Преподаватель: доцент каф. 806 Никулин Сергей Петрович

Отчет сдан « » \_\_\_\_\_\_\_\_\_20 \_\_\_ г., итоговая оценка \_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Тема:** Деревья выражений
2. **Цель работы:** составить программу выполнения заданных преобразований арифметических выражений с применением деревьев
3. **Задание** № 24**:**



1. **Оборудование** (лабораторное):

ЭВМ \_\_\_\_\_\_\_\_\_, процессор \_\_\_\_\_\_\_. имя узла сети\_\_\_\_\_\_\_с ОП\_\_\_\_\_ Мб, НМД \_\_\_\_ Мб. Терминал \_\_\_\_\_\_\_\_адрес \_\_\_\_\_\_\_\_. Принтер\_\_\_\_\_\_\_

Другие устройства

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_AMD Ryzen 5 \_ с ОП 16 Гб НМД \_\_\_512 Гб. Монитор 1920x1080~60Hz Другие устройства

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Операционная система семейства \_\_\_\_\_\_\_\_\_\_\_, наименование \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия

\_\_\_\_\_\_\_\_\_\_\_\_\_

интерпретатор команд \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_

Система программирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Редактор текстов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утилиты операционной системы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Прикладные системы и программы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Местонахождение и имена файлов программ и данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства \_Linux\_, наименование \_Ubuntu\_ версия 22.04.2 интерпретатор команд \_GNU bash\_ версия 5.1.16.

Система программирования С.

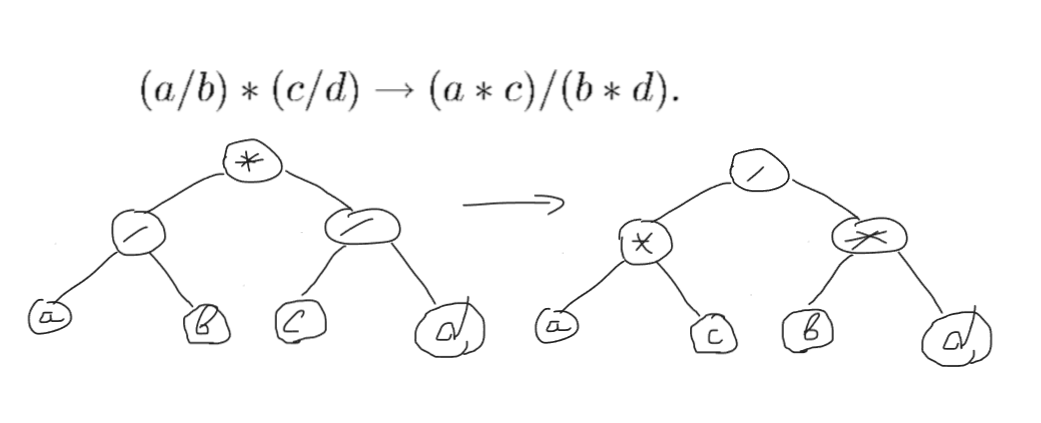
Редактор текстов emacs версия 29.1

Утилиты операционной системы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Прикладные системы и программы Emacs

Местонахождение и имена файлов программ и данных на домашнем компьютере /home/

1. **Идея, метод, алгоритм** решения задачи(в формах:словесной,псевдокода,графической[блок-схема,диаграмма,рисунок,таблица]

Составить программу на языке Си, выполняющую четыре действия: ввод выражения, построение дерева, построение выражения из стэка и функцция по заданию

Понадобятся функции:

* main: Главная функция программы, обрабатывает ввод пользователя и управляет программой.
* print\_menu: Выводит меню программы.
* print\_tree: Выводит дерево выражений в консоль.
* print\_expr: Выводит выражение в консоль.
* build\_tree: Строит дерево выражений из стека.
* next\_char: Считывает следующий непробельный символ из ввода.
* next\_symbol: Считывает следующий символ и определяет его тип.
* clearInputBuffer: Очищает буфер ввода.
* simplify\_expression2: Упрощает выражение в дереве.
* stack\_create: Создает новый стек с начальной емкостью.
* stack\_delete: Освобождает память, выделенную под стек.
* stack\_empty: Проверяет, пуст ли стек.
* stack\_push: Добавляет элемент в стек.
* stack\_pop: Удаляет и возвращает последний элемент стека.
* stack\_peek: Возвращает последний элемент стека без его удаления.

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

main.c:

#include <stdio.h>  
#include <stdlib.h>  
#include <stdbool.h>  
#include <string.h>  
  
#include "symbol.h"  
#include "stack.c"  
#include "tree.h"  
  
int op\_priority(char op) {  
 switch (op) {  
 case OP\_MINUS:  
 case OP\_PLUS:  
 return 1;  
 case OP\_MULT:  
 case OP\_DIVIDE:  
 return 2;  
 case OP\_POW:  
 return 3;  
 case OP\_UNARY\_MINUS:  
 return 4;  
 }  
 return -1;  
}  
  
  
  
OP\_ASSOC op\_assoc(OP op) {  
 switch (op) {  
 case OP\_MINUS:  
 case OP\_PLUS:  
 case OP\_MULT:  
 case OP\_DIVIDE:  
 return ASSOC\_LEFT;  
 case OP\_UNARY\_MINUS:  
 case OP\_POW:  
 return ASSOC\_RIGHT;  
 }  
 return -1;  
}  
  
char op\_to\_char(OP op) {  
 switch (op) {  
 case OP\_MINUS:  
 case OP\_PLUS:  
 case OP\_MULT:  
 case OP\_DIVIDE:  
 case OP\_POW:  
 return op;  
 case OP\_UNARY\_MINUS:  
 return '-';  
 }  
 return -1;  
}  
  
bool is\_space(int c) {  
 return (c == ' ') || (c == '\n') || (c == '\t');  
}  
  
int next\_char() {  
 int c;  
 while (is\_space(c = getchar())) {  
 }   
 return c;  
}  
  
bool next\_symbol(symbol \*out) {  
   
 static symb\_TYPE prev\_type = symb\_NONE;  
  
 int c = next\_char();  
 // сменил EOF на вопросик  
 if (c == '?') {  
 out->type = symb\_NONE;   
 prev\_type = symb\_NONE;   
 return false;   
 } else if (c == '.' || (c >= '0' && c <= '9')) {  
 // для работы с float: .9 == 0.9  
 ungetc(c, stdin);  
 out->type = symb\_NUMBER;  
 scanf("%f", &(out->data.number));   
 } else if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) {  
 ungetc(c, stdin);  
 out->type = symb\_VAR;  
 // scanf("%[^\n\t+-\*/^()]", out->data.var);  
 // не английские буквы - нафиг  
 scanf("%[a-zA-Z]", out->data.var); // добавляем буквы в переменную  
 } else if (c == '(') {  
 out->type = symb\_LEFT\_BR;  
 } else if (c == ')') {  
 out->type = symb\_RIGHT\_BR;  
 } else if (c == '+' || c == '\*' || c == '/' || c == '^') {  
 out->type = symb\_OP;  
 out->data.op = c;  
 } else if (c == '-') {  
 out->type = symb\_OP;   
 if (prev\_type == symb\_OP || prev\_type == symb\_NONE) {  
 out->data.op = OP\_UNARY\_MINUS;  
 } else {  
 out->data.op = OP\_MINUS;  
 }  
 } else {  
 out->type = symb\_NONE;  
 out->data.c = c;  
 }  
  
 prev\_type = out->type;  
  
 return true;  
}  
  
bool build\_tree(TN \*\*tree, STACK \*rev) {  
 if (stack\_empty(rev)) {  
 return false;  
 }  
 symbol t = stack\_pop(rev);  
 (\*tree) = (TN \*) malloc(sizeof(TN));  
 (\*tree)->t = t;  
  
 bool res = true;  
 if (t.type == symb\_OP) {  
 if (t.data.op == OP\_UNARY\_MINUS) {  
 (\*tree)->l = NULL;  
 // TODO: исправлял! может ломаться  
// res = res && build\_tree(&((\*tree)->r), rev);  
 res = build\_tree(&((\*tree)->r), rev);  
 } else {  
// res = res && build\_tree(&((\*tree)->r), rev);  
 res = build\_tree(&((\*tree)->r), rev);   
 res = res && build\_tree(&((\*tree)->l), rev);  
 }  
 }  
  
 return res;  
}  
  
void print\_tree(TN \*tree, int lev) {  
  
 if (tree->t.type == symb\_OP) {  
 print\_tree(tree->r, lev + 1);  
 }  
 for (int i = 0; i < lev; i++) {  
 printf("\t");  
 }  
 switch (tree->t.type) {  
 case symb\_NUMBER:  
 printf("%.2lf\n", tree->t.data.number);  
 break;  
 case symb\_VAR:  
 printf("%s\n", tree->t.data.var);  
 break;  
 case symb\_OP:  
 if (tree->t.data.op == OP\_UNARY\_MINUS) {  
 printf("-\n");  
 print\_tree(tree->r, lev + 1);  
 } else {  
 printf("%c\n", op\_to\_char(tree->t.data.op));  
   
 }  
 break;  
 default:  
 fprintf(stderr, "This symlbol must not be in the tree already");  
 return;  
 }  
 if (tree->t.type == symb\_OP) {  
 print\_tree(tree->l, lev + 1);  
 }  
}  
  
void print\_expr(TN \*tree) {  
 switch (tree->t.type) {  
 case symb\_NUMBER:  
 printf("%.2lf", tree->t.data.number);  
 break;  
 case symb\_VAR:  
 printf("%s", tree->t.data.var);  
 break;  
 case symb\_OP:  
 if (tree->t.data.op == OP\_UNARY\_MINUS) {  
 printf("-");  
 print\_expr(tree->r);  
 } else {  
 if (tree->l->t.type == symb\_OP && op\_priority(tree->t.data.op) > op\_priority(tree->l->t.data.op)) {  
 printf("(");  
 print\_expr(tree->l);  
 printf(")");  
 } else {  
 print\_expr(tree->l);  
 }  
 // printf("(");  
 // print\_expr(tree->l); 2+3) -> 2+3  
 printf("%c", op\_to\_char(tree->t.data.op));  
 // print\_expr(tree->r);// (2+3) -> 2+3  
 // printf(")");  
// if (tree->r->t.type == symb\_OP && op\_priority(tree->t.data.op) > op\_priority(tree->r->t.data.op))  
 if (tree->r != NULL && tree->r->t.type == symb\_OP &&  
 op\_priority(tree->t.data.op) > op\_priority(tree->r->t.data.op)) {  
 printf("(");  
 print\_expr(tree->r);  
 printf(")");  
 } else if (tree->r != NULL) {  
 print\_expr(tree->r);  
 }  
 }  
 break;  
 default:  
 fprintf(stderr, "This symlbol must not be in the tree already");  
 return;  
 }  
}  
  
void clearInputBuffer() {  
 int c;  
 while ((c = getchar()) != '\n' && c != EOF);  
}  
  
void print\_menu() {  
 puts("---------------------------------------------");  
 puts("█ 1) Ввести выражение.");  
 puts("█ 2) Распечатать дерево.");  
 puts("█ 3) Вывести выражение.");  
 puts("█ 4) Упростить выражение.");  
 puts("█ 0) Выход.");  
 puts("---------------------------------------------");  
 printf("▷▷▷ ");  
  
}  
  
  
//(a/b)\*(c/d)->(a\*c)/(b\*d)  
void simplify\_expression2(TN \*node) {  
 if (node == NULL) {  
 return;  
 }  
  
 // Рекурсивно обходим левое поддерево  
 simplify\_expression(node->l);  
  
 // Рекурсивно обходим правое поддерево  
 simplify\_expression(node->r);  
  
 // Проверка корня  
 if (node->t.type == symb\_OP && node->t.data.op == OP\_MULT) {  
 // Проверки  
 if (node->l != NULL && node->l->t.type == symb\_OP && node->l->t.data.op == OP\_DIVIDE &&  
 node->r != NULL && node->r->t.type == symb\_OP && node->r->t.data.op == OP\_DIVIDE) {  
  
 // Создаём ноду деления  
 TN \*newOpNode = malloc(sizeof(TN));  
 newOpNode->t.type = symb\_OP;  
 newOpNode->t.data.op = OP\_DIVIDE;  
  
 // Нода числителя  
 TN \*newNum = malloc(sizeof(TN));  
 newNum->t.type = symb\_OP;  
 newNum->t.data.op = OP\_MULT;  
 newNum->l = node->l->l; // Числитель левого деления  
 newNum->r = node->r->l; // Числитель правого деления  
  
 // Нода знаменателя  
 TN \*newDen = malloc(sizeof(TN));  
 newDen->t.type = symb\_OP;  
 newDen->t.data.op = OP\_MULT;  
 newDen->l = node->l->r; // Знаменатель левого деления  
 newDen->r = node->r->r; // Знаменатель правого деления  
  
 // Линкуем ноды  
 newOpNode->l = newNum;  
 newOpNode->r = newDen;  
  
 // Заменяем текущий узел на новый узел деления  
 \*node = \*newOpNode;  
 }  
 }  
}  
  
/\*\*  
 \* точка старта  
\*/  
int main(void) {  
 STACK \*s, \*rev;  
 symbol t;  
 TN \*root = NULL;  
 print\_menu();  
 char choise;  
 scanf("%c", &choise);  
 while (choise != '0') {  
 if (choise == EOF) {  
 printf("Bye-bye!");  
 break;  
 }  
 switch (choise) {  
 case '1':  
 printf("Введи выражение: \n");  
  
 // ============== Ввод выражения ==============  
 s = stack\_create();  
 rev = stack\_create();  
  
 while (next\_symbol(&t)) {  
 switch (t.type) {  
 case symb\_NONE:  
 fprintf(stderr, "Error: symbol %c not recognized\n", t.data.c);  
 return 1;  
  
 case symb\_OP:  
 for (;;) {  
 if (stack\_empty(s))  
 break;  
 symbol top = stack\_peek(s);  
 if (top.type != symb\_OP)  
 break;  
  
 if ((op\_assoc(t.data.op) == ASSOC\_LEFT &&  
 op\_priority(t.data.op) <= op\_priority(top.data.op)) ||  
 (op\_assoc(t.data.op) == ASSOC\_RIGHT &&  
 op\_priority(t.data.op) < op\_priority(top.data.op))) {  
   
 stack\_pop(s);  
 stack\_push(rev, top);  
 } else {  
 break;  
 }  
 }  
  
   
 stack\_push(s, t);  
  
 char \*command\_user = NULL;  
 if (scanf("%20s", command\_user) == EOF) {  
 printf("Woops");  
 }  
 break;  
  
 case symb\_NUMBER:  
 case symb\_VAR:  
  
 stack\_push(rev, t);  
 break;  
  
 case symb\_LEFT\_BR:  
 stack\_push(s, t);  
 break;  
  
 case symb\_RIGHT\_BR:  
 for (;;) {  
 if (stack\_empty(s)) {  
 fprintf(stderr, "Error: closing bracket hasn't pair");  
 return 2;  
 }  
 symbol top = stack\_peek(s);  
 if (top.type == symb\_LEFT\_BR) {  
 stack\_pop(s);  
 break;  
 } else {  
 stack\_pop(s);  
 stack\_push(rev, top);  
 }  
 }  
 break;  
 }  
 }  
  
 while (!stack\_empty(s)) {  
 t = stack\_pop(s);  
 if (t.type == symb\_LEFT\_BR) {  
 fprintf(stderr, "Error: opening bracket hasn't pair");  
 return 2;  
 }  
 stack\_push(rev, t);  
 }  
 // ============== Конец выражения ==============  
  
 printf("ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО! \n");  
 break;  
  
 case '2':  
 // Распечатать дерево  
 if (rev == NULL || stack\_empty(rev)) {  
 fprintf(stderr, "Error: expression is empty");  
 return 3;  
 }  
  
 // TN \*root = NULL;  
 if (!build\_tree(&root, rev)) {  
 fprintf(stderr, "Error while building tree: don't find one of operands");  
 return 4;  
 }  
 if (!stack\_empty(rev)) {  
 fprintf(stderr, "Error while building tree: extra operands or opetators");  
 return 4;  
 }  
 if (root != NULL) {  
 print\_tree(root, 0);  
 } else {  
 fprintf(stderr, "Дерево не создано");  
 return 3;  
 }  
 break;  
  
 case '3':  
 if (root != NULL) {  
 print\_expr(root);  
 printf("\n");  
 } else {  
 printf("Выражение не создано.\n");  
 }  
 break;  
  
 case '4':  
 if (stack\_empty(rev)) {  
 fprintf(stderr, "Error: expression is empty");  
 return 3;  
 }  
  
 // TN \*root = NULL;  
 if (!build\_tree(&root, rev)) {  
 fprintf(stderr, "Error while building tree: don't find one of operands");  
 return 4;  
 }  
 if (!stack\_empty(rev)) {  
 fprintf(stderr, "Error while building tree: extra operands or opetators");  
 return 4;  
 }  
 // Вызов функции упрощения выражения  
 simplify\_expression2(root);  
 if (root != NULL) {  
 printf("Упрощенное выражение: \n");  
 print\_expr(root);  
 printf("\n========\n");  
 print\_tree(root, 0);  
 stack\_delete(rev);  
 stack\_delete(s);  
 free(root);  
 root = NULL;  
 } else {  
 fprintf(stderr, "Пуста!!!");  
 return 3;  
 }  
 break;  
  
 case '0':  
 printf("Выход из программы.\n");  
 exit(0);  
 break;  
  
 default:  
 printf("Неверный выбор. Попробуйте снова.\n");  
 }  
 choise = getchar();  
 clearInputBuffer();  
 }  
  
 printf("Конец работы программы!\n");  
  
 // // УБИРАЕМ НЕНУЖНОЕ  
 stack\_delete(rev);  
 stack\_delete(s);  
 // free(&t);  
  
 return 0;  
}

stack.h

#ifndef \_\_stack\_h\_\_  
#define \_\_stack\_h\_\_  
  
#include <stdbool.h>  
  
#include "symbol.h"  
#include "utils.h"  
  
typedef struct {  
 symbol \*body;  
 int size; // текущий размер  
 int cap; // capacity - вместимость  
} STACK;  
  
STACK \*stack\_create();  
void stack\_delete(STACK \*stack);  
bool stack\_empty(STACK \*stack);  
void stack\_push(STACK \*stack, symbol t);  
symbol stack\_pop(STACK \*stack);  
symbol stack\_peek(STACK \*stack);  
  
#endif

stack.c:

#include <stdlib.h>  
#include <stdbool.h>  
  
#include "symbol.h"  
#include "stack.h"  
  
#define MINSIZE 4  
  
STACK \*stack\_create() {  
 STACK \*stack = (STACK\*)malloc(sizeof(STACK));  
 stack->cap = MINSIZE;  
 stack->size = 0;  
 stack->body = (symbol\*)malloc(sizeof(symbol) \* stack->cap);  
 return stack;  
}  
  
STACK \*stack\_copy(STACK \*original) {  
 // Создаем новый стек  
 STACK \*copy = stack\_create();  
  
 // Копируем элементы из исходного стека в новый  
 for (int i = 0; i < original->size; i++) {  
 stack\_push(copy, original->body[i]);  
 }  
  
 // Возвращаем копию стека  
 return copy;  
}  
  
void stack\_delete(STACK \*stack) {  
 // free(stack->body);  
 // free(stack);  
 FREE\_AND\_NULL(stack);  
  
}  
bool stack\_empty(STACK \*stack) {  
 return stack->size == 0;  
}  
  
void stack\_push(STACK \*stack, symbol t) {  
 if(stack->size <= stack->cap) {  
 stack->cap \*= 2;  
 stack->body = (symbol\*)realloc(stack->body, sizeof(symbol) \* stack->cap);  
 }  
  
 stack->body[stack->size] = t;  
 stack->size++;  
}  
  
symbol stack\_pop(STACK \*stack) {  
 symbol res = stack->body[stack->size - 1];  
 stack->size--;  
  
 if(stack->size \* 2 < stack->cap && stack->cap > MINSIZE) {  
 stack->cap /= 2;  
 stack->body = (symbol\*)realloc(stack->body, sizeof(symbol) \* stack->cap);  
 }  
  
 return res;  
}  
  
symbol stack\_peek(STACK \*stack) {  
 return stack->body[stack->size - 1];  
}

Symbol.h:

#ifndef \_\_symbol\_h\_\_  
#define \_\_symbol\_h\_\_  
  
  
typedef enum \_OP\_ASSOC {  
 ASSOC\_LEFT, ASSOC\_RIGHT  
} OP\_ASSOC  
  
  
  
#define VARNAME\_LEN 10  
  
typedef enum \_symb\_TYPE {  
 symb\_NONE, //   
 symb\_NUMBER, //   
 symb\_VAR, //   
 symb\_OP, //   
 symb\_LEFT\_BR, //   
 symb\_RIGHT\_BR //   
} symb\_TYPE;  
  
typedef enum \_OP {  
 OP\_MINUS = '-',  
 OP\_PLUS = '+',  
 OP\_MULT = '\*',  
 OP\_DIVIDE = '/',  
 OP\_POW = '^',  
 OP\_UNARY\_MINUS = '!'   
} OP;  
  
typedef struct {  
 symb\_TYPE type;  
 union {  
 float number;   
 char var[VARNAME\_LEN]   
 OP op;   
 char c;   
 } data;  
} symbol;  
  
#endif

token.h

#ifndef \_\_token\_h\_\_  
#define \_\_token\_h\_\_  
  
  
typedef enum \_OP\_ASSOC {  
 ASSOC\_LEFT, ASSOC\_RIGHT  
} OP\_ASSOC;  
  
  
  
#define VARNAME\_LEN 10  
  
typedef enum \_TOK\_TYPE {  
 TOK\_NONE,   
 TOK\_NUMBER,   
 TOK\_VAR,   
 TOK\_OP, TOK\_LEFT\_BR,   
 TOK\_RIGHT\_BR   
} TOK\_TYPE;  
  
typedef enum \_OP {  
 OP\_MINUS = '-',  
 OP\_PLUS = '+',  
 OP\_MULT = '\*',  
 OP\_DIVIDE = '/',  
 OP\_POW = '^',  
 OP\_UNARY\_MINUS = '!'   
} OP;  
  
typedef struct {  
 TOK\_TYPE type;  
 union {  
 float number;   
 char var[VARNAME\_LEN];   
 OP op;   
 char c;   
 } data;  
} TOKEN;  
  
#endif

Tree.h

#ifndef \_\_tree\_h\_\_  
#define \_\_tree\_h\_\_  
  
#include <stdbool.h>  
  
#include "symbol.h"  
  
  
  
//typedef struct \_SN SN;  
  
//struct \_TN {  
typedef struct SN {  
 symbol t;  
  
 struct SN\* r;  
 struct SN\* l;  
} TN;  
  
#endif

Utils.h:

#define FREE\_AND\_NULL(p) do { free(p); (p) = NULL; } while (0)

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика$ cd '2 Семестр'

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика/2 Семестр$ cd ЛР24

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика/2 Семестр/ЛР24$ gcc main.c

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика/2 Семестр/ЛР24$ ./a.out

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

█ 1) Ввести выражение.

█ 2) Распечатать дерево.

█ 3) Вывести выражение.

█ 4) Упростить выражение.

█ 0) Выход.

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

▷▷▷ 1

Введи выражение:

(a/b)\*(c/d)

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷ 2

d

/

c

\*

b

/

a

▷▷▷ 3

a/b\*c/d

▷▷▷ 1

Введи выражение:

(a/b)\*(c/d)

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷4

Упрощенное выражение:

a\*c/b\*d

========

d

\*

b

/

c

\*

a

▷▷▷ 0

Конец работы программы!

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика/2 Семестр/ЛР24$ gcc main.c

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика/2 Семестр/ЛР24$ ./a.out

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

█ 1) Ввести выражение.

█ 2) Распечатать дерево.

█ 3) Вывести выражение.

█ 4) Упростить выражение.

█ 0) Выход.

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

▷▷▷ 1

Введи выражение:

(1/2)\*(3/4)

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷2

4.00

/

3.00

\*

2.00

/

1.00

▷▷▷1

Введи выражение:

(1/2)\*(3/4)

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷4

Упрощенное выражение:

1.00\*3.00/2.00\*4.00

========

4.00

\*

2.00

/

3.00

\*

1.00

▷▷▷ 1

Введи выражение:

(5/v)\*(7/s)

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷2

s

/

7.00

\*

v

/

5.00

▷▷▷1

Введи выражение:

(5/v)\*(7/s)

▷▷▷3

5.00/v\*7.00/s

▷▷▷4

Упрощенное выражение:

5.00\*7.00/v\*s

========

s

\*

v

/

7.00

\*

5.00

▷▷▷ 0

Конец работы программы!

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика/2 Семестр/ЛР24$ gcc main.c

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика/2 Семестр/ЛР24$ ./a.out

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

█ 1) Ввести выражение.

█ 2) Распечатать дерево.

█ 3) Вывести выражение.

█ 4) Упростить выражение.

█ 0) Выход.

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

▷▷▷ 1

Введи выражение:

((a/b)\*(c/d))+((g/h)\*(j/k))

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷2

k

\*

j

/

h

\*

g

+

d

\*

c

/

b

\*

a

▷▷▷ 3

a/b\*c/d+g/h\*j/k

▷▷▷ 5

Неверный выбор. Попробуйте снова.

▷▷▷1

Введи выражение:

1

((a/b)\*(c/d))+((g/h)\*(j/k))

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷4

Упрощенное выражение:

a\*c/b\*d+g\*j/h\*k

========

k

/

h

\*

j

/

g

+

d

/

b

\*

c

/

a

▷▷▷ 1

Введи выражение:

((1/2)\*(3/4))-((5/6)\*(7/8))

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷2

8.00

/

7.00

\*

6.00

/

5.00

-

4.00

/

3.00

\*

2.00

/

1.00

▷▷▷3

3

1.00/2.00\*3.00/4.00-5.00/6.00\*7.00/8.00

▷▷▷ 1

Введи выражение:

((1/2)\*(3/4))-((5/6)\*(7/8))

ВЫРАЖЕНИЕ БЫЛО ВВЕДЕНО!

▷▷▷4

Упрощенное выражение:

1.00\*3.00/2.00\*4.00-5.00\*7.00/6.00\*1.00

========

8.00

/

6.00

\*

7.00

/

5.00

-

4.00

/

2.00

\*

3.00

/

1.00

▷▷▷ 0

Конец работы программы!

qwental@DESKTOP-NKF1EUK:/mnt/c/Users/Holiday/Desktop/Мои материалы/Информатика/2 Семестр/ЛР24$ exit

**9 Дневник отладки** должен содержать дату и время сеансов отладки и основные события(ошибки в сценарии и программе,нестандартные ситуации) 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. **Замечания автора** по существу работы

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Выводы**

Научился работать с деревом выражений, реализовал алгоритм решения предложенной задачи.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_