**Отчет по лабораторной работе № 24** по курсу “Языки и методы программирования”

Студент группы М80-103Б-21 Зверева Елизавета Леонидовна, № по списку 11

e-mail: [elizavetka.zvereva.2003@mail.ru](mailto:elizavetka.zvereva.2003@mail.ru) , telegram: @banshee

Работа выполнена: «» сентября 2021г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

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

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

* 1. **Тема:** Деревья выражений.

1. **Цель работы:** Составить программу выполнения заданных преобразований арифметических выражений с применением деревьев.
2. **Задание (*вариант № 25* ): Заменить степень с суммой в показателе на произведение степеней.**
3. **a^(b+c) = a^b\*a^c**
4. **Оборудование** (студента):

Процессор *AMD A9-9420 RADEON R5, 5 COMPUTE CORES 2C+3G 3.00 GHz* с ОП *8* Гб, НМД *512* Гб. Монитор *1920x1080*

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

Операционная система семейства: *linux*, наименование: *ubuntu*, версия *20.04 focal*

интерпретатор команд: *bash* версия *5.0.17*

Редактор текстов *emacs* версия *3.24.14*

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

При запуске программы, выводится меню, где пользователь может выбрать одно из следующих действий:

* Ввести выражение и сгенерировать дерево выражений на его основе. Выполняется с помощью функции MakeTree, получает на вход выражение и индексы начала и конца части, которую необходимо обработать. Создает пустое дерево, далее ищет элемент с минимальным приоритетом и придает содержимому узла его значение. Если в выражении присутствуют скобки, то все операции в них пропускаются, как имеющие больший приоритет.
* Изменить выражение. Изменение производится в соответствии с вариантом, то есть степень с суммой в показателе заменяется на произведение степеней. Производится следующим образом: функция transformation производит рекурсивный обход дерева, и если обнаруживает значение одного из узлов равным '^', а его правого подузла равным '+', то заменяет значение первого на знак умножения, далее правый и левый подузлы принимают значение '^'
* Распечатать дерево. Производится с помощью рекурсивного обхода всех вершин дерева.
* Распечатать выражение. Последовательно выводит каждый элемент дерева.

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

1) реализовать создание дерева

2)трансформацию выражения

3)вывод выражения

4)вывод деревьев

gcc main.c

./a.out

Choose an action:

1. Enter an expression and create tree.

2. Transforn expression.

3. Print tree.

4. Print expression.

5. Exit

1

Please, enter an expression: a^(b+c)

Choose an action:

1. Enter an expression and create tree.

2. Transforn expression.

3. Print tree.

4. Print expression.

5. Exit

3

c

+

b

^

a

Choose an action:

1. Enter an expression and create tree.

2. Transforn expression.

3. Print tree.

4. Print expression.

5. Exit

4

a^(b+c)

Choose an action:

1. Enter an expression and create tree.

2. Transforn expression.

3. Print tree.

4. Print expression.

5. Exit

2

Choose an action:

1. Enter an expression and create tree.

2. Transforn expression.

3. Print tree.

4. Print expression.

5. Exit

4

a^b\*a^c

**8. Распечатка протокола**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**struct StrNode {**

**char data [50];**

**struct StrNode \*left;**

**struct StrNode \*right;**

**};**

**typedef struct StrNode \*node;**

**int define\_priority (char a) {**

**switch (a){**

**case '-':**

**case '+':**

**return 1;**

**case '\*':**

**return 2;**

**case '/':**

**return 3;**

**case '^':**

**return 4;**

**default:**

**return 100;**

**}**

**}**

**node MakeTree (char expr[], int first, int last) {**

**int prior, MinPrior = 100, k, depth = 0;**

**node Tree = (node) malloc (sizeof(struct StrNode));**

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

**if (expr[i] == '(') {**

**depth++;**

**continue;**

**}**

**if (expr[i] == ')') {**

**depth--;**

**continue;**

**}**

**if (depth>0){**

**continue;**

**}**

**prior = define\_priority(expr[i]);**

**if (prior <= MinPrior) {**

**MinPrior = prior;**

**k = i;**

**}**

**}**

**if (depth !=0) {**

**printf("Wrong expression!\n");**

**exit (1);**

**}**

**int l;**

**if (MinPrior == 100) {**

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

**free(Tree);**

**return MakeTree(expr, first +1, last - 1);**

**}**

**else {**

**l = last - first + 1;**

**for (int i = 0; i < l; i++ ) {**

**Tree->data[i] = expr[first+i];**

**}**

**Tree->data[l] = '\n';**

**Tree->left = NULL;**

**Tree->right = NULL;**

**return Tree;**

**}**

**}**

**Tree->data[0] = expr[k];**

**Tree->data[1] = '\n';**

**Tree->left = MakeTree(expr, first, k-1);**

**Tree->right = MakeTree(expr, k+1, last);**

**return Tree;**

**}**

**void print\_tree(node tr, int l) {**

**if (tr->right != NULL) print\_tree(tr->right, l+1);**

**for(int i = 0; i < l; ++i) {**

**printf(" ");**

**}**

**printf("%5s", tr->data);**

**if (tr->left != NULL) print\_tree(tr->left, l+1);**

**}**

**void print\_expression(node t) {**

**if (t==NULL) {**

**return;**

**}**

**if (define\_priority(t->data[0])!=100 && define\_priority(t->left->data[0])!=100 && define\_priority(t->data[0])**

**> define\_priority(t->left->data[0]) || t->data[0] == '^' && t->left->data[0] == '^' ){**

**printf("(");**

**print\_expression(t->left);**

**printf(")");**

**} else print\_expression(t->left);**

**for (int i = 0; i < 50; ++i) {**

**if (t->data[i] == '\n') {**

**break;**

**}**

**printf("%c", t->data[i]);**

**}**

**if (define\_priority(t->data[0])!=100 && define\_priority(t->right->data[0])!=100 && define\_priority(t->data[0]**

**) > define\_priority(t->right->data[0]) || t->data[0] == '^' && t->right->data[0] == '^' ){**

**printf("(");**

**print\_expression(t->right);**

**printf(")");**

**} else print\_expression(t->right);**

**}**

**node copy (node t) {**

**if (t == NULL) {**

**return NULL;**

**}**

**node new\_t = (node) malloc (sizeof(struct StrNode));**

**for (int i = 0; i<50; ++i) {**

**new\_t->data[i] = t->data[i];**

**}**

**new\_t ->left = copy(t->left);**

**new\_t->right = copy(t->right);**

**return new\_t;**

**}**

**node transformation (node \* t) {**

**if (\*t == NULL) {**

**return NULL;**

**}**

**if ((\*t)->data[0] == '^' && (\*t)->right->data[0] == '+') {**

**(\*t)->data[0] = '\*';**

**node base1 = copy((\*t)->left);**

**node base2 = copy(base1);**

**node degree = copy((\*t)->right);**

**char deg [50];**

**deg[0] = '^';**

**deg[1] = '\n';**

**for (int i = 0; i < 50; ++i) {**

**(\*t)->left->data[i] = deg[i];**

**}**

**(\*t)->left->left = base1;**

**(\*t)->left->right = (degree)->left;**

**\*(\*t)->right->data= \*deg;**

**(\*t)->right->left = base2;**

**(\*t)->right->right = (degree)->right;**

**}**

**(\*t)->left = transformation(&((\*t)->left));**

**(\*t)->right = transformation(&((\*t)->right));**

**return \*t;**

**}**

**int main(void) {**

**node t = NULL;**

**int opt = -1;**

**while (opt!=5) {**

**printf("Choose an action:\n 1. Enter an expression and create tree.\n 2. Transforn expression.\n 3. Print tree.\n 4. Print expression.\n 5. Exit\n");**

**scanf("%d", &opt);**

**switch (opt) {**

**case 1: {**

**printf("Please, enter an expression: ");**

**char expression[1000];**

**scanf("%s", expression);**

**int n = 0;**

**while (expression[n] != '\0') {**

**n++;**

**}**

**t = MakeTree(expression, 0, n-1);**

**break;**

**}**

**case 2: {**

**t = transformation(&t);**

**break;**

**}**

**case 3: {**

**printf("\n");**

**print\_tree(t, 0);**

**break;**

**}**

**case 4: {**

**printf("\n");**

**print\_expression(t);**

**printf("\n");**

**break;**

**}**

**}**

**}**

**return 0;**

**}**

**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. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
| 1 | - |  |  | - | - | - |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора.** Нет.

**Выводы.** Результатом выполнения работы стало глубокое изучение работы с выражениями и использование деревьев. Реализация оказалась не такой уж простой задачей.

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