**Отчет по лабораторной работе №22** по курсу\_\_\_\_\_\_1\_\_\_\_\_\_\_

Студент группы: **М8О-101Б-22**, **Соколова Виктория Дмитриевна**, № по списку: **20**, Контакты**,** [viktoriyasokolova04@gmail.com](mailto:viktoriyasokolova04@gmail.com) telegram **@Angrytea\_01** Работа выполнена: « » 2023г.

Преподаватель: **каф. 806 Крылов Сергей Сергеевич**, Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет сдан « » 2023 г., итоговая оценка \_\_\_\_\_

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

1. **Тема:** Алгоритмы и структуры данных
2. **Цель работы:** Составить программу выполнения преобразований выражений с применением деревьев.
3. **Задание:** Убрать из частных все делители, равные 1
4. **Оборудование** (лабораторное):

ЭВМ , процессор , имя узла сети с ОП Мб, НМД Мб. Терминал адрес . Принтер Другие устройства

**Оборудование ПЭВМ студента, если использовалось:**

Процессор **Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz** с ОП **16384** Мб, ТТН **524 288** Мб. Встроенный монитор 15.6 дюймов диагональ, разрешение **1920 \* 1080**.

Другие устройства

1. **Программное обеспечение ЭВМ студента, если использовалось:**

Операционная система семейства **Linux** , наименование **Linux Ubuntu** версия 22.04.01 интерпретатор команд **bash** версия **5.1.16** Система программирования версия Редактор текстов **nano** версия **6.2** Утилиты операционной системы

Прикладные системы и программы

Местонахождение и имена файлов программ и данных на домашнем компьютере

1. **Идея, метод, алгоритм**  решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиям.
2. Создать структуру дерева, корнем которого будет последняя операция с наименьшим приоритетом
3. Делители будем удалять, заменяя узел деления на левое поддерево узла деления
4. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].
5. #include <stdio.h>
6. #include <stdlib.h>
7. #include <string.h>
8. typedef struct tree{
9. char data[40];
10. struct tree \*left;
11. struct tree \*right;
12. }node\_t;
13. int op(char data[40]) {
14. if (data[0] == '^' || data[0] == '/' || data[0] == '\*' || data[0] == '+' || data[0] == '-') {
15. return 1;
16. }
17. return 0;
18. }
19. int IsNumber(node\_t \*root){
20. int i = 0;
21. if (!root){ //пустое дерево
22. return 0;
23. }
24. while (root->data[i]){ //идем до конца строки
25. if (!strchr("0123456789", root->data[i++]) ){
26. return 0; //выход, если нет цифры
27. }
28. }
29. return 1;
30. }
31. int priority(char c){
32. switch (c){
33. case '+': case '-': return 1;
34. case '\*': case '/': return 2;
35. case '^': return 3;
36. default:
37. return 100;
38. }
39. }
40. // для создания дерева нам необходимо найти последнюю операцию с наименьшим приоритетом (mprt), она будет корнем
41. node\_t \*create(char Ex[], int first, int last){
42. int mprt, i, k, prt;
43. int br = 0; //счетчик скобок
44. node\_t \*root = malloc(sizeof(node\_t));
45. if (root == NULL){
46. return NULL;
47. }
48. mprt = 100;
49. for (i == first + 1; i <= last; i++){
50. if (priority(Ex[i]) != 100 && priority(Ex[i-1]) != 100){
51. printf("Neverno vvedeno\n");
52. root = NULL;
53. return root;
54. }
55. }
56. for (i = first; i <= last; i++){
57. if (Ex[i] == '('){
58. br++;
59. continue;
60. }
61. if (Ex[i] == ')'){
62. br--;
63. continue;
64. }
65. if (br > 0){
66. continue;
67. }
68. prt = priority(Ex[i]);
69. if (prt <= mprt){
70. mprt = prt;
71. k = i;
72. }
73. }
74. if (br != 0){
75. printf("Neverno vvedeno\n");
76. root = NULL;
77. return root;
78. }
79. if (mprt == 100){
80. if (Ex[first] == '(' && Ex[last] == ')'){
81. free(root);
82. return create(Ex, first + 1, last - 1);
83. } else {
84. k = last - first + 1;
85. strncpy(root->data, Ex+first, k);
86. root->data[k] = '\0';
87. root->left = NULL;
88. root->right = NULL;
89. return root;
90. }
91. } else {
92. root -> data[0] = Ex[k];
93. root -> data[1] = '\0';
94. root -> left = create(Ex, first, k-1);
95. root -> right = create(Ex, k+1, last);
96. return root;
97. }
98. }
99. node\_t \*poisk(node\_t \*root){
100. if (root == NULL){
101. return root;
102. }
103. //poisk(root->left, 0);
104. if (root->right->data[0] == '/'){
105. if (root->right->right->data[0] == '1' && root->right->right->data[1] == '\0'){
106. root->right = root->right->left;
107. return root;
108. }
109. } else if(root->left->data[0]== '/'){
110. if (root->left->right->data[0] == '1' && root->left->right->data[1] == '\0'){
111. root->left = root->left->left;
112. return root;
113. }
114. }
115. poisk(root->left);
116. poisk(root->right);
117. }
119. void PrintTree(node\_t \*root, int level) {
120. if (root == NULL){
121. return;
122. }
123. if (root != NULL) {
124. PrintTree(root->right, level + 1);
125. for (int i = 0; i < level; i++) {
126. printf("\t");
127. }
128. for (int i = 0; i < 41; i++){
129. if (root -> data[i] != '\0'){
130. printf("%c", root->data[i]);
131. continue;
132. } else {
133. printf("\n");
134. break;
135. }
136. }
137. PrintTree(root->left, level + 1);
138. }
139. }
140. void print\_expression(node\_t \*root) {
141. if (root == NULL) {
142. return;
143. }
144. if (op(root -> data) && op(root -> left -> data) &&
145. (priority(root -> left -> data[0]) < priority(root -> data[0]) || root -> left -> data[0] == '^' && root -> data[0] == '^')) {
146. printf("(");
147. print\_expression(root -> left);
148. printf(")");
149. } else {
150. print\_expression(root -> left);
151. }
152. for (int i = 0; i <= 40; i++) {
153. if (root->data[i] != '\0') {
154. printf("%c", root->data[i]);
155. continue;
156. }
157. break;
158. }
159. if (op(root -> data) && op(root -> right -> data) &&
160. (priority(root -> right -> data[0]) < priority(root -> data[0]) || root -> right -> data[0] == '^' && root -> data[0] == '^')) {
161. printf("(");
162. print\_expression(root -> right);
163. printf(")");
164. } else {
165. print\_expression(root -> right);
166. }
167. }
168. int main(){
169. char choice[100];
170. int m = 1;
171. int l = 0;
172. node\_t \*root = NULL;
173. while (m){
174. printf("\n");
175. printf("1 - vvesti vyragenie\n");
176. printf("2 - print tree\n");
177. printf("3 - print vyrageniye\n");
178. printf("4 - izmenenie po tasky\n");
179. printf("5 - exit\n");
180. printf("Ny i? ");
181. scanf("%s", &choice);
182. printf("\n");
183. if (strcmp(choice, "1") == 0){
184. int n = 0;
185. char vyrazenie[100];
186. scanf("%s", &vyrazenie);
187. while (vyrazenie[n] != '\0'){
188. n++;
189. }
190. root = create(vyrazenie, 0, n-1);
191. } else if (strcmp(choice, "2") == 0){
192. PrintTree(root, 0);
193. } else if (strcmp(choice, "3") == 0){
194. print\_expression(root);
195. printf("\n");
196. } else if (strcmp(choice, "4") == 0){
197. root = poisk(root);
198. } else if (strcmp(choice, "5") == 0){
199. break;
200. } else{
201. printf("No.\n");
202. }
203. }
204. }

*Допущен к выполнению работы.*  **Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Распечатка протокола:**

PS C:\code\test> ./a

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? авпвап

No.

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 234234234

No.

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 1

dsfghghfgsdasafdfg

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 2

dsfghghfgsdasafdfg

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 1

(a+b)/1/c

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 2

c

/

1

/

b

+

a

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 4

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 2

c

/

b

+

a

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 3

(a+b)/c

1 - vvesti vyragenie

2 - print tree

3 - print vyrageniye

4 - izmenenie po tasky

5 - exit

Ny i? 5

1. **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
2. **Выводы**:.Я составила программу выполнения преобразований выражений с применением деревьев

Недочёты при выполнении задания могут быть устранены следующим образом:

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