**Отчет по лабораторной работе №** 24 по курсу 1

Студент группы М80-108Б-18, Коростелев Д.В. № по списку 12

E-mail [dmitry.k48@yandex.ru](mailto:dmitry.k48@yandex.ru)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа выполнена: « 25 » мая 2019 г.

Преподаватель: Поповкин А.В. каф.806 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет сдан « » \_\_\_\_\_\_\_\_\_ 201\_\_ г., итоговая оценка \_\_\_\_\_\_

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

1. **Тема:** Языки и методы программирования, алгоритмы и структуры данных.
2. **Цель работы:** составить программу выполнения заданных преобразований арифметических выражений с применением деревьев.
3. **Задание (***вариант №* **):** вынести общие сомножители (переменные и константы) из суммы.
4. **Оборудование (лабораторное):**

ЭВМ Cameron, процессор Intel Core2 Duo CPU E8500 3.16ГГц, имя узла сити cameron с ОП 16029 Мб, НДМ \_\_\_\_\_\_\_\_\_ Гб. Терминал gnome, адрес 172.16.80.213. Принтер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Другие устройства: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Intel Core i7-7700HQ 4x 2.808ГГц с ОП 3945 Мб, НДМ 48.9 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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Утилиты операционной системы: cd, cat, nano, cc, make \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Прикладные системы и программы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Местонахождение 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\_\_\_\_\_\_4.3.48\_\_\_\_\_\_\_

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

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

Утилиты операционной системы: cd, cat, nano, cc. make\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 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 постусловием)

Программа на вход получает строку, без пробелов, кроме того, последовательность арифметических выражений задает пользователь посредством круглых скобок, в итоге, получаем, что число скобок должно быть ровно в два раза больше знаков арифметических выражений.

Первый шаг, создать бинарное дерево, узел которого будет являться структурным типом данных, в котором хранится значение самого узла и указатели на родителя, левого и правого потомка, создавать бинарное дерево будем итерационно, следующему алгоритму:

1. Если встречаем «(«, то создаем левого потомка у текущего узла и переходим в него
2. Если встречаем число или константу, то записываем ее в текущий узел, и возвращаемся в родительский узел
3. Если встречаем «)», то возвращаемся в предыдущий узел
4. Если встречаем арифметическое действие, записываем его в текущий узел, создаем правого потомка и переходим в него.

Стоит отметить, что последую скобу нужно игнорировать, так как в процессе выполнения, программа попадает в корневой узел на предпоследнем шаге.

Далее, нужно преобразовать дерево, находим в дереве поддерево с корнем «\*» и его потомками со значениями ‘-‘ , далее, так как у потомков потомков есть 4 ветки, существуют 4 варианта преобразования дерева, каждое из которое отличается от другого на 1 или 2 действия, но суть у этих преобразований одна и та же, у преобразуемого поддерева, корень заменяем на «\*» значение правого потомка делаем равным «-», его потомки - два отличающихся множителя, а левый потомок «\*» - общий сомножитель

Далее с помощью специальной функции, которая по бинарному дереву образует строку выражения выводим арифметическое выражение.

1. **Сценарий выполнения работы** (план работы, первоначальный текст программы на черновике или отдельном листе, тесты либо соображения по тестированию)

*Функции*

|  |  |  |
| --- | --- | --- |
| **№** | **Функция** | **Описание** |
| 1. | void printpt(PTREE\* root) | Вывести арифметическое выражение используя бинарное дерево выражения. Принимает указатель на корень бинарного дерева |
| 2. | void printt(PTREE\* root, int counter) | Вывести бинарное дерево арифметического выражения  Принимает указатель на корень бинарного дерева, и кол-во отступов от левого края |
| 3. | PTREE\* rbuildt(PTREE\* root) | Выполнить задание варианта. Принимает указатель на корень бинарного дерева |
| 4. | PTREE\* getptree(PTREE\* root, char\* c, int len) | Построить бинарное дерево арифметического выражения. Принимает указатель на корень бинарного дерева, указатель на массив выражения и его длину. |

Переменные

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Переменная** | **Тип** | **Назначение** |
| 1. | left | struct parstree\* | Указатель на левого потомка, часть структуры parstree |
| 2. | right | Указатель на правого потомка, часть структуры parstree |
| 3. | prev | Указатель на родителя, часть структуры parstree |
| 4. | temp | struct parstree\* | Буферный узел |
| 5. | value | char | Значение узла дерева, часть структуры parstree |
| 6. | tempc | Временное хранилище символьных значений |
| 7. | c | char\* | Указатель на символьный массив арифметического выражения |
| 8. | len | int | Кол-во символов в арифметическом выражений |

*Тесты*

|  |  |  |
| --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** |
| 1. | (1+2) | (1+2) |
| 2. | ((1+2)\*3) | ((1+2)\*3) |
| 3. | ((1\*3)-(4\*3)) | (3\*(1-4)) |
| 4. | (1\*((3\*4)-(4\*a))) | (1\*(4\*(3-a))) |
| 5. | ((((3\*a)-(b\*3))^3)\*5) | (((3\*(a-b))^3)\*5) |

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

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

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

|  |
| --- |
| dmitry@dmitry-VirtualBox:~/lubs/24$ cat 24.c  #include<stdio.h>  #include<stdlib.h>  #include<malloc.h>  struct parstree{  char value;  struct parstree\* left;  struct parstree\* right;  struct parstree\* prev;  };  typedef struct parstree PTREE;  void printpt(PTREE\* root){  if(root->left != 0){  printf("(");  printpt(root->left);  }  printf("%c",root->value);  if(root->right != 0){  printpt(root->right);  printf(")");  }  }  void printt(PTREE\* root, int counter){  if(root->left !=0)printt(root->left, counter + 1);  for(int i = 0;i<counter;i++){  printf(" ");  }  printf("\_%c\n",root->value);  if(root->right != 0)printt(root->right, counter + 1);  }  PTREE\* rbuildt(PTREE\* root){  PTREE\* temp;  temp = root;  char general;  char tempc;  if(temp->value == '-')  if(temp->left != 0 && temp->right != 0 && temp->left->value == temp->right->value && temp->left->value == '\*')  if(temp->left->left->left == 0 && temp->left->left->right ==0 && temp->right->left->left == 0 && temp->right->left->right == 0  && temp->right->right->left == 0 && temp->right->right->right == 0 && temp->left->right->left == 0 && temp->left->right->right == 0){  if(temp->left->left->value == temp->right->left->value){  general = temp->left->left->value;  tempc = temp->value;  temp->value = temp->right->value;  temp->right->value = tempc;  temp->left->value = general;//  temp->right->left->value = temp->left->right->value;  free(temp->left->left);  temp->left->left = 0;  free(temp->left->right);  temp->left->right = 0;  return (temp);  }  if(temp->left->left->value == temp->right->right->value){  general = temp->left->left->value;  tempc = temp->value;  temp->value = temp->right->value;  temp->right->value = tempc;  temp->left->value = general;//  temp->right->right->value = temp->right->left->value;  temp->right->left->value = temp->left->right->value;  free(temp->left->left);  temp->left->left = 0;  free(temp->left->right);  temp->left->right = 0;  return (temp);  }  if(temp->left->right->value == temp->right->left->value){  general = temp->left->right->value;  tempc = temp->value;  temp->value = temp->right->value;  temp->right->value = tempc;  temp->left->value = general;//  temp->right->left->value = temp->left->left->value;  free(temp->left->left);  temp->left->left = 0;  free(temp->left->right);  temp->left->right = 0;  return (temp);  }  if(temp->left->right->value == temp->right->right->value){  general = temp->left->right->value;  tempc = temp->value;  temp->value = temp->right->value;  temp->right->value = tempc;  temp->left->value = general;//  temp->right->right->value = temp->right->left->value;  temp->right->left->value = temp->left->left->value;  free(temp->left->left);  temp->left->left = 0;  free(temp->left->right);  temp->left->right = 0;  return (temp);  }  }  if(temp->left != 0) temp->left = rbuildt(temp->left);  if(temp->right != 0) temp->right = rbuildt(temp->right);  return (temp);  }  PTREE\* getptree(PTREE\* root, char\* c, int len){  PTREE \* temp;  temp = root;  for(int i= 0; i < len-1;i++){  if(c[i] == '('){  temp->left = malloc(sizeof(PTREE));  temp->left->prev = temp;  temp= temp->left;  temp->left = 0;  temp->right = 0;  continue;  }  if(c[i] == ')'){  temp= temp->prev;  continue;  }  if(c[i] == '+' || c[i] == '-' || c[i] == '/' || c[i] == '\*' || c[i] == '^'){  temp->value = c[i];  temp->right = malloc(sizeof(PTREE));  temp->right->prev = temp;  temp = temp->right;  temp->left = 0;  temp->right = 0;  continue;  }  else{  temp->value = c[i];  temp = temp->prev;  continue;  }  }  return (temp);  }  int main(){  char\* c;  c = malloc(sizeof(char));  char temp;  int len = 0;  printf("Enter arithmetical expression: ");  while(1){  scanf("%c", &temp);  if(temp == '\n') break;  len++;  c = realloc(c, len\*sizeof(char));  c[len-1] = temp;  }  PTREE\* root;  root = malloc(sizeof(PTREE));  root->prev = 0;  root->left = 0;  root->right = 0;  printf("[%s], %d\n", c, len);  printf("Creating of tree...\n");  root = getptree(root, c, len);  printf("Created parsing tree: \n");  printt(root, 0);  root = rbuildt(root);  printf("\n");  printf("Changed Tree: \n");  printt(root, 0);  // printf("|%c",root->value);  printf("Changed expression: ");  printpt(root);  printf("\n");  return 0;  }  dmitry@dmitry-VirtualBox:~/lubs/24$ cc -o 24.out 24.c  dmitry@dmitry-VirtualBox:~/lubs/24$ ./24.out  Enter arithmetical expression: (1+2)  [(1+2)], 5  Creating of tree...  Created parsing tree:  \_1  \_+  \_2  Changed Tree:  \_1  \_+  \_2  Changed expression: (1+2)  dmitry@dmitry-VirtualBox:~/lubs/24$ ./24.out  Enter arithmetical expression: ((1+2)\*3)  [((1+2)\*3)], 9  Creating of tree...  Created parsing tree:  \_1  \_+  \_2  \_\*  \_3  Changed Tree:  \_1  \_+  \_2  \_\*  \_3  Changed expression: ((1+2)\*3)  dmitry@dmitry-VirtualBox:~/lubs/24$ ./24.out  Enter arithmetical expression: ((1\*3)-(4\*3))  [((1\*3)-(4\*3))], 13  Creating of tree...  Created parsing tree:  \_1  \_\*  \_3  \_-  \_4  \_\*  \_3  Changed Tree:  \_3  \_\*  \_1  \_-  \_4  Changed expression: (3\*(1-4))  dmitry@dmitry-VirtualBox:~/lubs/24$ ./a.out  bash: ./a.out: No such file or directory  dmitry@dmitry-VirtualBox:~/lubs/24$ ./24.out  Enter arithmetical expression: (1\*((3\*4)-(4\*a)))  [(1\*((3\*4)-(4\*a)))], 17  Creating of tree...  Created parsing tree:  \_1  \_\*  \_3  \_\*  \_4  \_-  \_4  \_\*  \_a  Changed Tree:  \_1  \_\*  \_4  \_\*  \_3  \_-  \_a  Changed expression: (1\*(4\*(3-a)))  dmitry@dmitry-VirtualBox:~/lubs/24$ ./24.out  Enter arithmetical expression: ((((3\*a)-(b\*3)^3)\*5)-6)  [((((3\*a)-(b\*3)^3)\*5)-6)], 23  Creating of tree...  Created parsing tree:  \_3  \_\*  \_a  \_^  \_3  \_\*  \_5  \_-  \_6  Changed Tree:  \_3  \_\*  \_a  \_^  \_3  \_\*  \_5  \_-  \_6  Changed expression: ((((3\*a)^3)\*5)-6) |

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:** данная лабораторная работа учит студента основным методам построения и обработки бинарных деревьев арифметических выражений и их паркинга, а также учит способам преобразования этих деревьев, а в последствии и самих арифметических выражений.

Недочеты при выполнении задания могут быть устранены следующим образом: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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