МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

Тема: Иерархические списки

Студентка гр. 7383 _____ Маркова А. В.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

Содержание

Цель работы

Ознакомиться с основными понятиями иерархического списка и научиться реализовывать его на языке программирования C.

Формулировка задачи: вычислить глубину (число уровней вложения) иерархического списка как максимальное число одновременно открытых левых скобок в сокращенной скобочной записи списка; принять, что глубина пустого списка и глубина атомарного S – выражения равны нулю; например, глубина списка (a (b () c) d) равна двум.

Реализация задачи

В данной работе используется главная функция `main()` и дополнительные функции `El* CreateEl(int tag, char* sym, El* up, El* prev)`, `El* CreateList(FILE** test, El* UpEl, int* count, int* countmax)`, `void clearList(El* element)`, `void check(FILE** test, int* c, int* cm)`, а также структура `Element`.

После запуска программы функция `main()` выводит меню на консоль, где можно выбрать пункт, соответствующий нужной операции. Считывается целое число и при помощи оператора `switch()`, выполняется необходимое действие. При нажатие «1», программа считает глубину вложенности списка, считанного с файла, при нажатие «2» пользователь сам вводит нужный иерархический список для проверки. При выборе «3» функция завершает свою работу. Если было введено другое значение, отличное от стандартных, то программа выведет сообщение об ошибке: «Некорректный выбор!» и завершит работу. Программа завершается при выборе «3» или введение неизвестной команды, в противном случае – ожидает дальнейших указаний. Если была выбрана опция вычисления глубины списка в файле, то функция `void check (FILE** test, int* c, int* cm)` проверяет существование этого файла и возможность его прочесть, а затем вызывается функция `El* CreateList (FILE** test, El* UpEl, int* count, int* countmax)`. Если пользователь ввел список сам, то `main()` сначала создаст текстовый файл, куда будет записана считанная с консоли строка, а затем функция обратится к `void check (FILE** test, int* c, int* cm)`. По завершении проверки созданный документ будет удален.

Рекурсивная функция `CreateList (FILE** test, El* UpEl, int* count, int* countmax)` получает на вход файл, с которым нужно работать, голову текущего списка, а так же значения глубины. В цикле

оператора `while` обрабатывается входная строка до тех пор пока все элементы заданного уровня вложенности не закончатся. Функция находит открывающуюся скобку и начинает заполнять элементы одного уровня, которые разделяются пробелами, с помощью вызова функции `El* CreateEl(int tag, char* sym, El* up, El* prev)`, когда вновь появляется ‘(’ программа рекурсивно обращается сама к себе и обрабатывает строку до ‘)’, а затем поднимается на предыдущий уровень вложенности. По мере того как функция встречает левые скобки счетчик вложенности растет, а при встрече правых убывает, сравнивая получившиеся значения глубины вложенности, определятся максимальный.

Функция `El* CreateEl(int tag, char* sym, El* up, El* prev)` выделяет память под элемент списка и заполняет его значениями. Так как при создании не известны следующие элементы и их адреса, то функция хранит текущие и записывает их в предыдущие.

При выполнении `check(FILE** test, int* c, int* cm)` просчитывается максимальная глубина вложенности, если `cm` равно нулю, то функцию выведет сообщение: «Список пуст!», в противном случае – «Максимальное число уровней вложенности `cm`».

Функция `void clearList(El* element)` очищает память, которая была выделена в ходе работы программы под иерархический список. Сначала функция спускается вниз на максимальный уровень вложенности, а потом начинает удалять элементы на данном уровне, поднимаясь рекурсивно вверх.

Структура `Element` имеет несколько полей: поле `tag`, которое указывает на углубление уровней вложенности, поле `char* sym` хранит в себе значение элементов, а так же есть поля, хранящие указатели вниз и в сторону по ходу движения в иерархическом списке.

Разберем для примера работы программы строку (v (d) c):

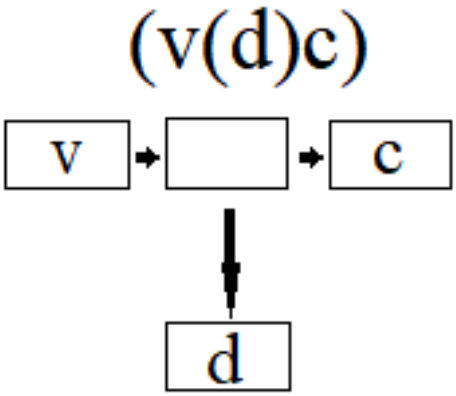

Рисунок 1 – пример работы программы

Сначала программа заполнит элемент с «v», затем встретив левую скобку она создаст пустой элемент, который будет содержать указатель на следующий уровень вложенности, вызов будет являться рекурсивным. После того как элемент «d» будет создан, программа продолжит обрабатывать строку и когда считает правую скобку, то поднимется на уровень выше и продолжит работу с уровнем, пока не закончится строка. Глубина вложенности в данном примере равна двум.

Тестирование

Программа собрана в операционной системе Ubuntu 17.04, с использованием компилятора gcc версии 5.4.0 20160609. В других ОС и компиляторах тестирование не проводилось.

Программа может быть скомпилирована с помощью команды:

gcc - Wall <имя файла>.c

Тестовые случаи представлены в Приложении А.

Исходя из тестовых случаев можно заметить, что во втором тесте программа ведет себя неверно, поэтому была исправлена программа: добавлено в `default` изменение значения флага на `false`, чтобы функция не зацикливалась.

Так же была выявлена ошибка в пятом тесте, в связи с этим в программу добавилась строка, которая обнуляла значения счетчика.

После, тестовые случаи не выявили неправильного поведения программы, что говорит о том, что по результатам тестирование было показано, поставленная задача была выполнена.

Выводы

В ходе лабораторной работы были изучены основные свойства и понятия иерархического списка. Была написана программа реализующая рекурсивную функцию `CreateList`, а так же создана программа для подсчета максимальной глубины вложенности на языке программирования C.

ПРИЛОЖЕНИЕ А

Тестовые случаи

Ввод	Вывод	Верно?
1 В файле: «( )»	Список пуст!	Да
4	Некорректный выбор! Некорректный выбор! …… Некорректный выбор!	Нет
8	Некорректный выбор! До свидания!	Да
1 (файл не был создан)	Файл не может быть открыт!	Да
2 (a ( )) 2 ( ) 2 (d f() d(d))	Максимальный уровень вложенности 1 Максимальный уровень вложенности 1 Максимальный уровень вложенности 1	Нет
2 (F f() f(d(d))) 3	Максимальный уровень вложенности 3 До свидания!	Да
2 (D s)	Максимальный уровень вложенности 1	Да

ПРИЛОЖЕНИЕ Б

Код программы

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#define N 50

typedef struct Element{
	struct Element* next;
	struct Element* down;
	int tag;
	char* sym;
} El;
El* CreateEl(int tag, char* sym, El* up, El* prev){ //создание элемента
списка
	El* new_element = (El*)malloc(sizeof(El));
	new_element -> tag = tag;
	if(tag){
		new_element -> sym = (char*)malloc(sizeof(char)*N);
		strcpy(new_element -> sym, sym);
	}
	if(up) up -> down = new_element; //поднялись на элемент выше и зполнили
поле, указывающее на след.элемент
	if(prev) prev -> next = new_element;
	new_element -> down = NULL;
	new_element -> next = NULL;
	return new_element;
}

El* CreateList(FILE** test, El* UpEl, int* count, int* countmax){ //создает
иерархический список
	int tag, index=0, run=1;
	El *up=UpEl, *prev=NULL, *Head_El, *El;
	char osym; //one symbol
	fpos_t pos;
	char* sym = (char*)malloc(sizeof(char)*N);
	while(run){
		sym[0]=0; //обнуление строки
		if (index) up=NULL;//для того чтобы указывал на первый нижний
элемент, не меняясь
		osym=fgetc(*test);
		if(osym == '\n' || osym == '\0') break;
```

```
            if(osym == ' ') continue; // пропуск пробелов
            if(osym == '(' && *count == 0 && index == 0){
                  (*count)++;
                  continue; //пропуск первой скобки
            }
            if(index==0 && osym == ')') (*count)--; //пустой уровень
            while(isalpha(osym)){
                  strncat(sym, &osym,1);
                  fgetpos(*test, &pos); //сохранение позиции
                  osym=fgetc(*test); //файловый поток
            }
            if(sym[0]){
                  osym='0';
                  fsetpos(*test, &pos); //возвращение на пред.символ, чтобы
не потерять скобочки
                  tag=1;
            }
            else tag=0;
            if(osym==')'){
                  if(*count>*countmax) *countmax=*count; //поиск
максимального уровня вложенности
                  (*count)--;
                  return Head_El;
            }
            if(!index){ //проверка на нулевую позицию - голову
                  Head_El=CreateEl(tag, sym, up, prev);
                  prev=Head_El;
                  El=Head_El;
            }
            else{
                  El=CreateEl(tag, sym, up, prev); //создание элементов
                  prev=El;
            }
            index++;
            if(osym=='('){
                  (*count)++;
                  CreateList(test, El, count, countmax);
            }
      }
return Head_El;
}

void clearList(El* element){ //очистка памяти
      El* temp;
      temp=element;
      while(temp!=NULL){
            if(temp->down!=NULL){
```

```
                clearList(temp->down);
            }
            temp=temp->next;
      }
      free(temp);
}

void check(FILE** test, int* c, int* cm){ //работа с файлом
      if(!(*test)){printf("\033[31mФайл не может быть открыт!\n\033[0m");
return;}
      El* Head = CreateList(test, NULL, c, cm);
      if(!(*cm)) printf("Список пуст!\n");
      else printf("Максимальное число уровней вложенности %d\n", *cm);
      clearList(Head);
      fclose(*test);
      return;
}

int main(){
      char* str;
      FILE* test;
      int c=0, cm=0, flag=1, var; //c-счетчик, cm-максимальный счетчик, т.е
показывает уровень вложенности
            printf("\t\033[34mЗдравствуйте! Выберите что вы хотите:\033[0m\n
1) Нажмите 1, чтобы посчитать уровень вложенности списка в файле\n 2) Нажмите
2, чтобы посчитать уровень вложенности списка с консоли\n 3) Нажмите 3, чтобы
выйти из программы\n");
            while(flag){
            scanf("%d", &var);
            switch(var){
                  case 1:
                        test=fopen("test.txt", "r");
                        c=0;
                        cm=0;
                        check(&test, &c, &cm);
                        break;
                  case 2:
                        scanf("\n");
                        str=(char*)malloc(sizeof(char)*N);
                        fgets(str, N, stdin);
                        test=fopen("test1.txt", "w+");
                        fputs(str, test);
                        fclose(test);
                        test=fopen("test1.txt", "r");
                        c=0;
                        cm=0;
                        check(&test, &c, &cm);
```

```
                        remove("test1.txt");
                        free(str);
                        break;
                  case 3:
                        flag=0;
                        break;
                  default:
                        flag=0;
                        printf("\033[31mНекорректный выбор!\033[0m\n");
                        break;
            }
      }
      printf("\033[34mДо свидания!\033[0m\n");
return 0;
}
```