**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Рекурсия

|  |  |  |
| --- | --- | --- |
| Студентка гр. 7382 |  | Головина Е.С. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2018

**Цель работы.**

Изучить понятие рекурсии. Научиться использовать рекурсивные функции в программах.

**Задание.**

Вариант 20

Построить синтаксический анализатор понятия *список\_параметров.*

*список\_параметров::= параметр*|*параметр , список\_параметров*

*параметр::= имя=цифра цифра | имя=*(*список\_параметров*)

*имя::= буква буква буква*

**Постановка задачи.**

Нужно написать программу, которая считывает, заданную пользователем, строку и определяет является ли она понятием «список\_параметров» или нет. Требуется рекурсивная реализация.

**Ход работы.**

Для решения задачи были реализованы следующие функции, которые содержатся в файле functions.c, его содержимое приведено в приложении А. Главная функция (int main()) содержится в файле lab1.c и его содержимое приведено в приложении Б. Для удобства также используется заголовочный файл functions.h, код которого приведен в приложении В.

1. **void recursion\_depth(int k)**

Функция принимает один аргумент – это результат проверки введенной пользователем строки (1 – если строка является понятием «список\_параметров», 0 – если не является). Также функция в своей работе использует ранее созданный текстовый файл sequence.txt, где ранее была сохранена информация о работе программы. Используя эти данные происходит вывод на консоль глубины рекурсии и сообщение о том является ли строка понятием «список\_параметров». Если нет, программа указывает в какой функции произошла ошибка.

1. **int numbers(char \* str, int index)**

Функция принимает на вход два аргумента: строку и индекс текущего элемента, начиная с которого нужно начать проверку на то, является ли текущий и следующий за текущим символ строки цифрами или нет. Возвращаемое значение является результатом этой проверки, 1 – если верно, 0 – если нет.

1. **int name(char \* str, int index)**

Функция принимает на вход два аргумента: строку и индекс текущего элемента, начиная с которого нужно начать проверку на то, является ли текущий и два следующих символа строки цифрами или нет. Проверка начинается с index+2 символа в строке, т.к. таким образом быстрее выявить ошибку. Возвращаемое значение является результатом этой проверки, 1 – если верно, 0 – если нет.

1. **int parametr(char\* str, int \* index, FILE \*\* f)**

Функция запускается только из функции list\_of\_parametrs и принимает на вход три аргумента: строку, указатель на индекс проверяемого символа и указатель на указатель на текстовый файл, куда будет записана информация о вхождении в функцию.

Ход работы функции такой:

- В ранее открытый в главной функции файл записывается информация о вхождении в функцию;

- Так как «параметр=имя=цифра цифра | имя=список\_параметров», в начале функция проверяет соответствует ли начало предполагаемого параметра понятию «имя» (имя=буква буква буква), если да, продолжается проверка строки;

- Проверяется есть ли после понятия «имя» знак «=», если да, продолжается проверка строки;

- Далее переходим ко второй части понятия «параметр», где оно варьируется и поэтому используем конструкцию if .. else. Если следующие после «=» два символа это цифры, то часть проверяемой строки – это параметр и функция возвращает значение = 1;

- Если же продолжение строки после знака «=» соответствует понятию «список\_параметров» (это проверяется в функции list\_of\_parametrs, описанной далее), то часть проверяемой строки – это также «параметр», функция возвращает значение = 1;

- Если ни одно из условий не выполнено, то функция возвращает значение = 0, что сигнализирует о том, что проверяемая часть строки не является понятием «параметр».

5. **int list\_of\_parametrs(char \* str, int \* index, FILE \*\* f)**

Главная функция для проверки строки на соответствие понятию «список\_параметров» принимает на вход три аргумента: строку, указатель на индекс проверяемого символа и указатель на указатель на текстовый файл, куда будет записана информация о вхождении в функцию.

Ход работы функции такой:

- В ранее открытый в главной функции файл записывается информация о вхождении в функцию;

- «Список\_параметров=параметр | параметр ,список\_параметров», поэтому в начале функция проверяет соответствует ли начало предполагаемого списка параметров понятию «параметр» (происходит вход в функцию, проверяющую это понятие), если соответствует, продолжается проверка строки;

- Проверяется идет ли после понятия «параметр» знак конца строки «\0» или запятая, если ничего из этого дальше не стоит, значит строка не является понятием «список\_параметров» и функция завершает свою работу;

- Если стоит знак конец строки, то достигнут конец строки и результат проверки – строка является понятием «список\_параметров»;

- Если стоит запятая, то вновь заходим в функцию list\_of\_parametrs и ход ее работы начинается с начала. Если она вернет значение = 1, то и первая функция list\_of\_parametrs вернет значение = 1.

6. **int main()**

Головная функция считывает все необходимые для работы данные. В начале узнает у пользователя собирается ли он вводить данные вручную или с помощью текстового файла (текстовый файл должен содержать две строки: максимальную длина строки (>0 и <32767) и саму строку). В зависимости от выбора происходит переключение на соответствующий режим работы.

Если выбран ввод из файла, то необходимо дополнительно написать название файла, который содержит исходные данные, далее происходит проверка на наличие данного файла в текущей директории, если он найден, то происходит считывание максимальной длины строки, выделение памяти под строку и считывание строки, затем происходит закрытие файла (при корректных данных), если не найден программа еще раз попросит ввести имя файла и если он снова не найден, то автоматически переходит в режим ввода данных из консоли. Если файл найден, но данные некорректны, то программа выводит сообщение о некорректных данных и программа завершается.

Если выбран ввод из консоли (или произошло автоматическое переключение на этот режим), то в начале программа считывает максимальную длину строки, корректность полученного числа проверяется, и программа будет просить его ввести до получения корректного результата. Далее происходит выделение памяти под строку, согласно полученному ранее значению и строка считывается. Если строка оказалась больше максимальной длины, то она считается не полностью, поэтому надо с умом выбирать максимум.

Затем, считав все необходимые данные, программа создает файл sequence.txt, куда будет записана информация о ходе работы программы. Указатель на указатель на этот файл будет использоваться в функциях 4 и 5.

Далее проверяем строку, запуская рекурсию с функции 5, результат проверки сохраняем в переменную result и закрываем файл sequence.txt.

Затем программа выводит считанную строку, для наглядности окрашивая верную часть в зеленый цвет, а часть с которой началась ошибка - в красный. Потом выводится глубина рекурсии и результат с помощью функции 1.

В конце освобождается память, выделенная под строку и удаляется файл sequence.txt, т.к. он является вспомогательным и в дальнейшем не нужен.

Тестирование.

Для более наглядной демонстрации работы программы был создан ряд тестов и bash-скрипт, последовательно выводящий содержимое очередного теста и результат работы программы для этого теста. Код bash-скрипта представлен в приложении Г, результат работы скрипта — в приложении Д.

**Вывод**

В ходе лабораторной работы были изучены принципы написания рекурсивных функций.

**ПРИДОЖЕНИЕ А**

**ИСХОДНЫЙ КОД, СОДЕРЖАЩИЙ ИСПОЛЬЗУЕМЫЕ ФУНКЦИИ**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include "functions.h"

// Function, which takes file sequence.txt and use it to show the depth of recursion

void recursion\_depth(int k){

printf("\nRECURSION DEPTH:\n");

char tab[100]="";

char str[20];

FILE \*file = fopen("./sequence.txt","r");

while (1){

fgets(str,20,file);

if (feof(file)) break;

printf("%s%s",tab,str);

strcat(tab," ");

}

str[strlen(str)-1]='\0';

if (!k)

printf("\nMistake in \"%s\". Incorrect data.\nString isn't list of parametrs.\n",str);

else

printf("\nCorrect data.\nString is list of parametrs.\n");

fclose(file);

}

// Function, which check 2 symbols in string, are they digits or not. Yes = 1, No = 0

int numbers(char \* str, int index){

for (int i=index+1;i>index-1;i--){

if (!isdigit(str[i]))

return 0;

}

return 1;

}

// Function, which check 3 symbols in string, are they letters or not. Yes = 1, No = 0

int name(char \* str, int index){

for (int i=index+2;i>index-1;i--){

if (!isalpha(str[i]))

return 0;

}

return 1;

}

// Function, which check if string is parametr

int parametr(char\* str, int \* index, FILE \*\* f){

fprintf(\*f,"parametr\n");

if (name(str,\*index)){

(\*index)+=3;

if (str[(\*index)]=='='){

(\*index)++;

if (numbers(str,\*index)){

(\*index)+=2;

return 1;

}

else{

if (list\_of\_parametrs(str,index,f))

return 1;

}

}

}

return 0;

}

// Function, which check if string is list\_of\_parametr

int list\_of\_parametrs(char \* str, int \* index, FILE \*\* f){

fprintf(\*f,"list\_of\_parametrs\n");

if (parametr(str,index,f)){

if (str[\*index]=='\0'){

return 1;

}

else{

if (str[\*index]==','){

(\*index)++;

return list\_of\_parametrs(str,index,f);

}

}

}

return 0;

}

**ПРИЛОЖЕНИЕ Б**

**ИСХОДНЫЙ КОД, СОДЕРЖАЩИЙ ГОЛОВНУЮ ФУНКЦИЮ**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "functions.h"

#define INT\_MAX 32767

int main(){

//declaration of variables

char name[22]="./"; //for the name of text file

int input\_format=0; //0 - Terminal input, 1 - Input from file

int index=0; //current index of symbol in string, which is checking/going to check

int result; // result of program, is string "list of parametrs" or not

long int length=100; //length of the string

char \* s; //pointer to the string

char l[20]; //using to read length as string

FILE \*f=NULL; //pointer to the text file

//check if person want to type text in terminal or use <file>.txt

fputs("If you want to type input in terminal - type 0.\nIf you want to use input from file - type 1.\n",stdout);

scanf("%d",&input\_format);

//asking for name of <file>.txt - 2 times. If still wrong - terminal mode is on

if (input\_format==1){

fputs("Input from file is on.\n",stdout);

fputs("Please, type the name of the file in format <name>.txt\n",stdout);

scanf("\n");

fgets(l,20,stdin);

l[strlen(l)-1]='\0';

strcat(name,l);

f = fopen(name,"r");

//check if pointer to the file isn't NULL

if (!f){

fputs("Error in opening file. There is no such file in directory or name of it was written incorrectly.\nPlease, try again.\n",stdout);

// scanf("\n");

fgets(l,20,stdin);

l[strlen(l)-1]='\0';

strcpy(name,"./\0");

strcat(name,l);

f = fopen(name,"r");

}

//check if pointer to the file isn't NULL - 2 time

if (!f){

fputs("Error in opening file.\n",stdout);

input\_format=0;

}

//if everything correct - read maximum length and string from <file>.txt

if (input\_format==1){

fgets(l,20,f);

length=strtol(l,&s,10);

//if data is written incorrectly in file - termination of the program

if (length<0 || length>INT\_MAX || !length){

fputs("Incorrect data. Program is terminated.",stdout);

return 0;

}

//if data is correct - allocation memory and reading string from file

s=(char\*)malloc(sizeof(char)\*length);

fgets(s,length,f);

}

fclose(f);

}

// Terminal mode, reading length of the string (and checking if it correct) and string

if (input\_format!=1){

fputs("Terminal input is on.\n",stdout);

fputs("Please, type the maximum length of the string: \n\0",stdout);

scanf("\n");

//Check if maximum length is number and it's value in between 0 and 32767

while (1){

fgets(l,20,stdin); //read length as string

length=strtol(l,&s,10); //transformation string to integer; length=0, if it fail to transformate

if (!length)

fputs("Incorrect data. Please, type number: \n\0",stdout);

else{

if (length<INT\_MAX && length>0)

break;

else

fputs("Incorrect data. Please, type number < 32767 and > 0: \n\0",stdout);

}

scanf("\n");

}

//allocates memory for string accordiing to the maximum length of it

s=(char\*)malloc(sizeof(char)\*length);

fputs("Please, type the string:\n\0",stdout);

scanf("\n");

fgets(s,length,stdin);

}

//creating of file sequence.txt, where will be the information about what function (recursive) was on

f= fopen("./sequence.txt","w+");

//deleting sign of next line

if (s[strlen(s)-1]=='\n') //maybe unnecessarily check

s[strlen(s)-1]='\0';

//start of recursive functions

result=list\_of\_parametrs(s,&index,&f);

fclose(f);

//output of the string, green - correct part, red - incorrect

printf("\nINPUT STRING:\n");

for (int i=0;i<index;i++)

printf(ANSI\_COLOR\_GREEN "%c" ANSI\_COLOR\_RESET,s[i]);

for (int i=index;i<strlen(s);i++)

printf(ANSI\_COLOR\_RED "%c" ANSI\_COLOR\_RESET, s[i]);

printf("\n");

//output the depth of recursion

recursion\_depth(result);

remove("sequence.txt");

free(s);

return 0;

}

**ПРИЛОЖЕНИЕ В**

**СОДЕРЖАНИЕ ЗАГОЛОВОЧНОГО ФАЙЛА**

//constants for color output in function printf

#define ANSI\_COLOR\_GREEN "\x1b[32m"

#define ANSI\_COLOR\_RED "\x1b[31m"

#define ANSI\_COLOR\_RESET "\x1b[0m"

// Function, which takes file sequence.txt and use it to show the depth of recursion

void recursion\_depth(int k);

// Function, which check 2 symbols in string, are they digits or not. Yes = 1, No = 0

int numbers(char \* str, int index);

// Function, which check 3 symbols in string, are they letters or not. Yes = 1, No = 0

int name(char \* str, int index);

// Function, which check if string is parametr

int parametr(char\* str, int \* index, FILE \*\* f);

// Function, which check if string is list\_of\_parametr

int list\_of\_parametrs(char\*,int\*,FILE\*\*);

**ПРИЛОЖЕНИЕ Г**

**ИСХОДНЫЙ КОД СКРИПТА**

#!/bin/bash

make Makefile

echo " "

echo "Tests with correct data:"

echo " "

echo "Test 1"

echo " "

./a.out <./Tests/test1.txt

echo " "

echo "Test 2"

echo " "

./a.out <./Tests/test2.txt

echo " "

echo "Test 3"

echo " "

./a.out <./Tests/test3.txt

echo " "

echo "Tests with incorrect data:"

echo " "

echo "Test 4"

echo " "

./a.out <./Tests/test4.txt

echo " "

echo "Test 5"

echo " "

./a.out <./Tests/test5.txt

echo " "

echo "Test 6"

echo " "

./a.out <./Tests/test6.txt

make -f Makefile clean

**ПРИЛОЖЕНИЕ Д**

**РЕЗУЛЬТАТ РАБОТЫ СКРИПТА**

Tests with correct data:

Test 1

If you want to type input in terminal - type 0.

If you want to use input from file - type 1.

Terminal input is on.

Please, type the maximum length of the string:

Please, type the string:

INPUT STRING:

dfg=87

RECURSION DEPTH:

list\_of\_parametrs

parametr

Correct data.

String is list of parametrs.

Test 2

If you want to type input in terminal - type 0.

If you want to use input from file - type 1.

Terminal input is on.

Please, type the maximum length of the string:

Please, type the string:

INPUT STRING:

asd=12,FGT=67,juy=78,ERT=87,asd=12,FGT=67,juy=78,ERT=87,asd=12,FGT=67,juy=78,ERT=87,asd=12,FGT=67,juy=78,ERT=87

RECURSION DEPTH:

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

Correct data.

String is list of parametrs.

Test 3

If you want to type input in terminal - type 0.

If you want to use input from file - type 1.

Terminal input is on.

Please, type the maximum length of the string:

Please, type the string:

INPUT STRING:

sdf=78,asd=Gth=Jyy=99

RECURSION DEPTH:

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

Correct data.

String is list of parametrs.

Tests with incorrect data:

Test 4

If you want to type input in terminal - type 0.

If you want to use input from file - type 1.

Terminal input is on.

Please, type the maximum length of the string:

Please, type the string:

INPUT STRING:

fdydffghfhfhot

RECURSION DEPTH:

list\_of\_parametrs

parametr

Mistake in "parametr". Incorrect data.

String isn't list of parametrs.

Test 5

If you want to type input in terminal - type 0.

If you want to use input from file - type 1.

Terminal input is on.

Please, type the maximum length of the string:

Please, type the string:

INPUT STRING:

asd=12,FGT=67,juy=78,ERT=87,dfg=sfg=67,ty=78

RECURSION DEPTH:

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

list\_of\_parametrs

parametr

Mistake in "parametr". Incorrect data.

String isn't list of parametrs.

Test 6

If you want to type input in terminal - type 0.

If you want to use input from file - type 1.

Terminal input is on.

Please, type the maximum length of the string:

Please, type the string:

INPUT STRING:

RECURSION DEPTH:

list\_of\_parametrs

parametr

Mistake in "parametr". Incorrect data.

String isn't list of parametrs.