**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**

**Вариант №17**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 7381 |  | Судакова П.С. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2018

**Цель работы.**

Ознакомление с парадигмой функционального программирования. Получение навыков реализации рекурсивных функций на языке программирования C.

**Постановка задачи.**

Разработать программу для выполнения задания с использованием функций языка С.

*Задание.*

Функция Ф преобразования текста определяется следующим образом (аргумент функции – это текст, т. е. последовательность символов)

Φ(α)=

Например: Ф(«ла/ска») = «скала», Ф(«б/ру/с») = «сруб», Ф(«ца/ри/ца») = «царица», Ф(«ум/ри/ва/к/а») = «аквариум». Реализовать функцию Ф рекурсивно.

**Основные теоретические положения.**

**Рекурсия** — вызов функции из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия).

Прямой (непосредственной) рекурсией является вызов функции внутри тела этой функции (см.рис.1). Косвенной же рекурсией является рекурсия, осуществляющая рекурсивный вызов функции посредством цепочки вызова других функций. Все функции, входящие в цепочку, тоже считаются рекурсивными.

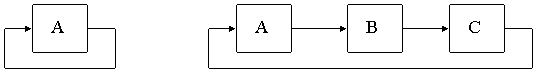
Рассмотрим косвенно рекурсивную функцию: функция A вызывает функцию B, а функция B — функцию С, а С— функцию А. Все функции A, B, C являются рекурсивными, так как при вызове одной из них, осуществляется вызов других и самой себя (см.рис.1).

Рис.1.Типы рекурсии

Рассмотрим пример, который присутствует, по-видимому, во всех учебниках по программированию. Функция *факториал* натурального аргумента *n* обозначается как *n*! и определяется соотношением

*n*!=1⋅2⋅3...(*n* – 1)⋅*n* . (2.1)

Удобно доопределить 0!=1 и считать, что *n* – целое неотрицательное число.

Некоторым недостатком определения (2.1) является наличие в нём многоточия «...», передающего речевой оборот «и так далее» и имеющего интуитивно понятный читателю смысл. Можно дать точное, так называемое *рекурсивное* определение функции *n*!, лишенное этого недостатка, т. е. не апеллирующее к нашей интуиции. Определим:

а) 0! = 1 , (2.2)

б) *n*! *=* (*n −* 1)!⋅*n* при *n* > 0.

Соотношения (2.2) можно рассматривать как свойства ранее определенной функции, а можно (как в данном случае) использовать их для определения этой функции.

Пример кода:

int factr(int n) /\* рекурсивно \*/  
{  
int answer;  
if(n==1) return(1);  
answer = factr(n-1)\*n;  
return(answer);  
}

**Спецификация программы.**

Данная программа предназначена для рекурсивной реализации функции Ф преобразования текста, которая определена в постановке задачи. Помимо этого, промежуточные данные должны выводиться с отступами, соответствующими глубине рекурсии. Должна выводиться информация о вызовах и завершениях рекурсивных функций. Желательно также явно выводить глубину рекурсии.

**Описание алгоритма.**

Для решения данной задачи используются два символьных массива – массив исходных данных и массив выходных данных(результата). Программа обрабатывает входную строку до тех пор, пока не встретит символ «/». Если этот символ не найден, то программа выводит исходные данные. При обнаружении «/» программа копирует символы, идущие до него, удаляет эти символы и «/» из исходных данных и вставляет их в начало массива результата. Этот процесс продолжается до тех пор, пока исходная строка не окажется пустой.

1.Ищем в исходной строке символ «/».

2.Если требуемый символ найден, то в массив результата копируются символы массива исходных данных, идущие до знака «/», а из последнего они удаляются вместе с «/».

3.Если требуемый символ не найден, то выводим содержимое конкатенации массива исходных данных и массива результат (завершение алгоритма).

4.Повторяем п.1 – п.3 для преобразованной строки исходной данных.

**Описание функций.**

1. void menu()

Функция menu предназначена для возможности выбора ввода данных из файла или с клавиатуры.

*Предусловие*: функция не принимает никаких переменных.

*Постусловие*: вывод меню в стандартный поток вывода.

2. void insert\_before(char\*s1, char\*s2, int count)

Функция insert\_before предназначена для вставки в s1 обработанной подстроки функцией Ф строки s2.

*Предусловие:*

char\*s1– массив символов, в который вставляется подстрока

char\*s2– массив символов, обрабатываемый функцией Ф

int count – целочисленная переменная, обозначающая количество символов, которые нужно скопировать

int i – переменная-счетчик, необходимая для прохода по массиву s1 с целью вставки в него подстроки (см. Приложение А)

*Постусловие*: вставка в s1 обработанной подстроки функцией Ф строки s2.

3. void erase(char\*s)

Функция erase предназначена для удаления обработанной подстроки функцией Ф строки s.

*Предусловие:*

char\*s – строка, подстроку которой требуется удалить

int m – длина оставшейся строки

int kol – длина удаляемой подстроки

char\*p – вспомогательная переменная, необходимая для реализации удаления полстроки

int i – переменная-счетчик, необходимая для сдвига массива влево на kol+1 символов

*Постусловие*: удаление обработанной подстроки функцией Ф строки s.

4. void empty(char\*s, char\*res)

Функция empty предназначена для инициализации массивов s и res нуль-терминаторами.

*Предусловие:*

char\*s – массив символов(строка)

char\*res – массив символов(строка)

int i – переменная-счетчик, необходимая для прохода по массивам

*Постусловие*: инициализация массивов s и res нуль-терминаторами.

5. void rekurs(char\*s, char\*res, FILE\*out)

Функция empty предназначена для реализации рекурсивной функции Ф.

char\*s – массив символов (исходная строка)

char\*res – массив символов (обработанная строка)

FILE\*out – файл вывода

int kol – длина обработанной подстроки функцией Ф строки s

int i – переменная-счетчик, необходимая для вывода глубины рекурсии

**Тестирование.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| ска/ла | ласка |
| с/ру/б | брус |
| сруб | сруб |
| т/е/в/и/р/п | привет |
| ум/////ри/ва////к/////а | аквариум |
| ла//////////////////ска | скала |
| привет | привет |
| ш/а/л/а/ш | шалаш |
| ///////////////////// |  |
| hello/bye//yes | yesbyehello |

*Пример работы программы.*

Введите строку:тм/ри/го/л/а

Анализ полученной строки...

Найдено первое вхождение символа "/", до него стоит подстрока [тм], вставляем её в конец еще не обработанной строки

ри/го/л/атм

Проверяем на следующее вхождение символа "/"

Найдено первое вхождение символа "/", до него стоит подстрока [ри], вставляем её в конец еще не обработанной строки

го/л/аритм

Проверяем на следующее вхождение символа "/"

Найдено первое вхождение символа "/", до него стоит подстрока [го], вставляем её в конец еще не обработанной строки

л/агоритм

Проверяем на следующее вхождение символа "/"

Найдено первое вхождение символа "/", до него стоит подстрока [л], вставляем её в конец еще не обработанной строки

алгоритм

Проверяем на следующее вхождение символа "/"

Текущая рассматриваемая строка не содержит символа "/"

Анализ завершон!

Полученное преобразование: алгоритм

**Выводы.**

Таким образом, в ходе выполнения лабораторной работы были получены и закреплены теоретические знания о рекурсии, а навыки реализации рекурсивных функций, а также получен опыт реализации рекурсивной функции преобразования текста.

**Приложение А. Исходный код**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

#define MAX 100

void menu();

void insert\_before(char\*s1, char\*s2, int count);

void erase(char\*s);

void rekurs(char\*s, char\*res,FILE\*out);

void empty(char\*s, char\*res);

int main() {

SetConsoleCP(1251);//русский язык

SetConsoleOutputCP(1251);

FILE \* in;

FILE\* out;

int cmd,count=0;

in = fopen("input.txt", "r");

out = fopen("output.txt", "w");

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

char\* res = (char\*)malloc(MAX \* sizeof(char));

menu();

printf("Выберите команду:");

if (!scanf("%d", &cmd)) {

printf("Неккоректный ввод.Попробуйте ввести цифру.\n");

exit(EXIT\_FAILURE);

}

switch (cmd) {

case 1: {

if (in == NULL) {

printf("Не удалось открыть файл input.txt");

exit(EXIT\_FAILURE);

}

if (out == NULL) {

printf("Не удалось открыть файл output.txt");

exit(EXIT\_FAILURE);

}

empty(s, res);

fprintf(out, "Входные данные(аргумент функции): ");

printf("Входные данные(аргумент функции): ");

fscanf(in, "%s", s);

fprintf(out, "%s\n", s);

printf("%s\n", s);

printf("\nАнализирование полученной строки...\n\n");

fprintf(out, "\nАнализирование полученной строки...\n\n");

rekurs(s, res,out);

printf("\nАнализирование завершено!\n");

printf("Полученное значение: ");

fprintf(out,"Полученное значение: ");

printf("%s%s\n\n", s, res);

fprintf(out, "%s%s\n", s, res);

break; }

case 2:

{

empty(s, res);

printf("Введите строку:");

scanf("%s", s);

printf("\nАнализирование полученной строки...\n\n");

fprintf(out, "\nАнализирование полученной строки...\n\n");

rekurs(s, res, out);

printf("\nАнализирование завершено!\n");

printf("Полученное преобразование:");

printf("%s%s\n\n", s, res);

fprintf(out,"Полученное преобразование:");

fprintf(out,"%s%s\n\n", s, res);

break;

}

case 3: {

exit(EXIT\_SUCCESS);

break; }

default: {

printf("Некорректная команда\n");

fprintf(out, "Некорректная команда\n");

break;

}

}

free(res);

free(s);

fclose(in);

fclose(out);

system("pause");

return 0;

}

void rekurs(char\*s, char\*res,FILE\*out) {

if (strchr(s, '/')) {

printf("Найдено первое вхождение символа \"/\"");

int kol = strlen(s) - strlen(strchr(s, '/'));

insert\_before(res, s, kol);

erase(s);

printf("%s%s", s, res);

fprintf(out,"%s%s", s, res);

fprintf(stdout, "\nПроверяем на следующее вхождение символа \"/\"\n");

fprintf(out, "\nПроверяем на следующее вхождение символа \"/\"\n");

rekurs(s, res,out);

}

else {

printf("Текущая рассматриваемая строка не содержит символа \"/\"\n");

}

}

void erase(char\*s) {

int m = strlen(strchr(s, '/'));

int kol = strlen(s) - strlen(strchr(s, '/'));

char\*p = s;

if (kol) {

for (int i = 0; i < m; i++) {

p[i] = p[i + kol];

p[i + kol] = '\0';

}

p[m] = '\0';

}

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

p[i] = p[i + 1];

}

void empty(char\*s, char\*res) {

for (int i = 0; i < MAX; i++) {

res[i] = '\0';

s[i] = '\0';

}

}

void insert\_before(char\*s1, char\*s2, int count) {

if (s1[0] == '\0') {

strncpy(s1, s2, count);

printf(", до него стоит подстрока [");

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

{

printf("%c",s1[i]);

}

printf("], вставляем её в конец еще не обработанной строки\n");

return;

}

for (int i = strlen(s1) - 1; i >= 0; i--)

s1[i + count] = s1[i];

strncpy(s1, s2, count);

printf(", до него стоит подстрока [");

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

{

printf("%c", s1[i]);

}

printf("], вставляем её в конец еще не обработанной строки\n");

}

void menu()

{

printf("------------------------------\n");

printf("Судакова Полина группа 7381,лаб. работа 1\n");

printf("------------------------------\n");

printf("1.Ввести строку из файла\n");

printf("2.Ввести строку с клавиатуры\n");

printf("3.Выход из меню\n");

printf("------------------------------\n");

}