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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия

Студент гр. 8304 _____ Перелыгин Д.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Освоение рекурсивного метода решения задач.

Задание.

Вариант 17.

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

Ф(γ)β, если α = β/γ и текст β не содержит вхождений символа «/»,
Ф(α)=
α, если в α нет вхождений символа «/».

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.

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

$$n!=1\cdot 2\cdot 3...(n-1)\cdot n\ .$$

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) $0! = 1\ ,$

б) $n! = (n-1)!\cdot n$ 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.

Далее для функции *n*! будем использовать привычное «функциональное» (префиксное) обозначение fact (*n*), указывая имя функции и за ним в скобках – аргумент. Тогда (2.2) можно записать в виде

$$\text{fact}\ (n) = \begin{cases} 1, & \text{если } n = 0; \\ \text{fact}\ (n-1)\cdot n, & \text{если } n > 0; \end{cases}$$

или в другой форме записи

$$\text{fact}\,(n) \equiv \textbf{if}\ n = 0\ \textbf{then}\ 1\ \textbf{else}\ \text{fact}\,(n-1)\cdot n, \qquad (2.4)$$

где использовано условное выражение **if** *b* **then** *e*1 **else** *e*2, означающее, что в том месте, где оно записано, следует читать *e*1, если выполняется условие *b*, и следует читать *e*2, если условие *b* не выполняется.

Функция, определяемая таким образом, *единственна.*

Как происходит выполнение таких функций и процедур? Рассмотрим следующую наглядную *модель* исполнения рекурсивных алгоритмов вычислительной машиной на примере функции fact. Пусть в программе имеется вызов описанной ранее рекурсивной функции, например z:=fact(k) с фактическим параметром (аргументом) k>0. Будем считать, что этот вызов запускает первый (стартовый) *экземпляр* функции fact. Каждый очередной рекурсивный вызов этой же функции (в данном примере это fact(k–1), fact(k–2) и т. д.) приводит к запуску нового экземпляра функции. При этом предыдущий экземпляр *откладывается,* и так происходит до тех пор, пока не будет вызван последний (*терминальный*) экземпляр функции, т. е. экземпляр, не содержащий рекурсивного вызова (в данном примере fact(0)). Этот терминальный экземпляр порождает значение функции (fact(0)=1) и завершает свою работу. Затем *восстанавливается* предыдущий (отложенный последним) экземпляр. Он, в свою очередь, порождает новое значение функции и завершает работу. Затем восстанавливается предыдущий экземпляр и т. д., до тех пор, пока не будет восстановлен и завершен экземпляр, соответствующий стартовому запуску (и тем самым закончится процесс вычисления fact(k)). Очевидно, память машины, используемая для хранения отложенных экземпляров, должна быть устроена таким образом, чтобы обеспечить восстановление первым того экземпляра, который был отложен последним.

Такой способ организации и функционирования памяти известен как *стек (Stack),* или *магазин* (по аналогии с магазином огнестрельного оружия), или *очередь типа LIFO*, т. е. Last In First Out (последним пришёл – первым ушёл).

Ход работы.

1. Программа использует следующий алгоритм для выполнения задачи: после считывания текста программа применяет к нему функцию, которая отделяет все символы, что находятся до первого знака ‘/’ и рекурсивно запускает саму себя, но уже с оставшейся частью строки. Так происходит до тех пор, пока в строке не остается символов ‘/’. В этот момент начинается «свёртывание» рекурсии с записью конечного результата.

2. Написана функция int main() со стандартной сигнатурой, отвечающая за открытие необходимых для ввода-вывода файлов, общение с пользователем и запуск функции-обработчика.

3. Основная функция string transmute(string out, string & text, int& deep) производит обрезание строки по первому встреченному символу «/», запуская саму себя от оставшейся части. При сворачивании рекурсии происходит добавление обрезанных частей слова к конечному результату.

Тестирование.

Входные данные	Ответ программы	Ожидаемый ответ
б\сру	**сруб**	**сруб**
ум/ри/ва/к/а	**аквариум**	**аквариум**
ца/ри/ца	**царица**	**царица**

Выводы.

В ходе выполнения работы были отработаны навыки работы с рекурсивными алгоритмами и их применение. Была написана программа с простейшим пользовательским интерфейсом. Был реализован выбор ввода и вывода в файл и с консоли.

Приложение А.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <clocale>
#include <windows.h>
using namespace std;

string transmute(string & out, string & text, int& deep)
{
	string buff;
	int len;
	int n = 0;
	n = text.find('/');

	if (n == -1)
	{
		cout << "на выход - " << text << '\n';
```

```
			return text;
		}
		else
		{
			deep++;
			buff = text.substr(0, n);
			cout << "Буфер - " << buff << " Глубина - " << deep <<'\n';
			len = text.length();
			text = text.substr(n + 1, len);
			cout << "Остаток - " << text << " Глубина - " << deep << '\n';
			out = transmute(out, text, deep);
			out += buff;
			cout << "получилось - " << out << '\n';
			return out;
		}
}

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	setlocale(LC_ALL, "Russian");
	int deep = 0;
	int choice;
	string out;
	string text;
	cout << "Выберите метод ввода: Из файла нажмите 0, с консоли нажмите 1\n";
	cin >> choice;
	if (!choice)
	{
		ifstream fin;
		fin.open("input.txt");
		getline (fin,text);
		fin.close();
	}
	else
		if (choice == 1)
	{
		getline(cin, text);
		cout << "Введите текст.\n";
		getline(cin, text);
	}
	ofstream fout;
	fout.open("out.txt");

	transmute(out, text, deep);
	cout << "итог - " << out << '\n';
	fout << out;
	fout.close();

	}
```