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

ОТЧЕТ

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

Студентка гр. 7382 _____ Еременко А.А

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2018

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций.

Постановка задачи.

Вариант №12.

Построить синтаксический анализатор для понятия *скобки*.
скобки::=квадратные | *круглые* | *фигурные*
квадратные::=[*круглые фигурные*] | +
круглые::=(*фигурные квадратные*) | −
фигурные::={*квадратные круглые*} | 0

Основные теоретические положения.

Функция называется рекурсивной, если во время ее обработки возникает ее повторный вызов, либо непосредственно, либо косвенно, путем цепочки вызовов других функций.

Прямой (непосредственной) рекурсией является вызов функции внутри тела этой функции.

```
int a()
{.....a().....}
```

Косвенной рекурсией является рекурсия, осуществляющая рекурсивный вызов функции посредством цепочки вызова других функций. Все функции, входящие в цепочку, тоже считаются рекурсивными.

Например:

```
a(){.....b().....}
b(){.....c().....}
c(){.....a().....} .
```

Все функции a,b,c являются рекурсивными, так как при вызове одной из них, осуществляется вызов других и самой себя.

Одним из примеров программы, основанной на рекурсии является программа, выполняющая синтаксический анализ текста. Задача синтаксического анализатора – проверить правильность записи выражения.

Пусть требуется построить *синтаксический анализатор* понятия *скобки:*
скобки::=квадратные | *круглые*
квадратные::=[*круглые круглые*] | +
круглые::=(*квадратные квадратные*) | –
В этом рекурсивном определении последовательности символов, называемой *скобки*, присутствуют две взаимно-рекурсивные части: *квадратные* определяются через *круглые*, и наоборот, *круглые* – через *квадратные.* В простейшем случае *квадратные* есть символ «+», а *круглые* есть символ «–». Другие примеры последовательностей, порождаемых этим рекурсивным определением:

‘[– –]’, ‘(++)’, ‘[(++)([–(++)][– –])]’, ‘(+[(++)([–(++)][(+[– –])–])])’.

Синтаксическим анализатором назовём программу, которая определяет, является ли заданная (входная) последовательность символов *скобками* или нет. В случае ответа «нет» сообщается место и причина ошибки.

Выполнение работы.

При помощи данной программы можно определить, является ли какая-либо последовательность символов скобками. Исходные данные вводятся с клавиатуры или из файла: в начале работы программы на экран выводится вопрос о том, в каком виде следует принимать данные. (Если был выбран ввод из файла, а самого файла по нужному адресу нет, на экран выводится сообщение об ошибке; если файл пустой – сообщение об этом).

После того, как были получены входные данные, начинается основная часть программы – синтаксический анализ. Т.к. нам известны символы, с которых могут начинаться скобки (‘[’, ‘+’, ‘(’, ‘-’, ‘{’, ‘0’) сначала проходит сравнение первого элемента текста с возможным началом скобок. Если символы совпали, то начинается проход по рекурсивным функциям, которые отвечают за определение того, скобки нам даны или нет. Рассмотрим эти функции подробнее.

1) `int square(FILE *fp, FILE *f, char a, int i)`

Эта функция проверяет, являются ли входные данные квадратными скобками. Данная функция принимает на вход указатели на файлы ввода и вывода (FILE *fp, FILE *f соответственно), переменную char a, которая служит хранилищем для символа, с которым в данный момент проводится работа (анализ, вывод и т.д.) и число int i, которое служит счётчиком для ширины отступа при выводе данных на консоль и в файл, которая соответствует глубине рекурсии.
В ходе работы данной функции возможны изменения всех входных данных кроме файла ввода. Возвращаемое значение равно единице в том случае, если скобки соответствуют символам “+”, “0” или “-”, т.е. для их распознания не требуется вызова других функций. Ноль получается, если функция определения скобок дошла до конца (при этом из этой функции вызывались другие, анализирующие скобки) или если был обнаружен символ, из-за которого входные данные не могут являться скобками.

Работа функции основана на следующем принципе: вначале она анализирует текущий символ. Если он является “+”, то работу функции можно завершить, т.к. это квадратные скобки. Если этот символ – “[”, то функция должна обратиться к функции round (см. в п.2), и если та вернула единицу, то затем – к функции figure. Если возвращаемое значение вновь является единицей, то функция должна проверить следующий символ: если тот символ – “]”, то проанализированный текст является квадратными скобками и функция завершает работу и возвращает единицу. Если же при вызове функций round или figure был получен ноль, функция square завершает свою работу со значением 0, что означает, что входные данные не являются скобками.

2) `int round(FILE *fp, FILE *f, char a, int i)` - программа для определения круглых скобок;

3) `int figure(FILE *fp, FILE *f, char a, int i)` - программа для определения фигурных скобок;

Функции round и figure очень похожи на первую описанную функцию. Входные данные, работа с ними и возвращаемое значение – те же. Работа отличается лишь тем, что символы, которым должен соответствовать текущий, другие. Т.е. аналогом “+” для круглых и фигурных скобок являются “-” и “0” соответственно, аналогом “[” для круглых и фигурных скобок являются “(” и “{” соответственно и аналогом “]” для круглых и фигурных скобок являются “)” и “}” соответственно.

Для удобства в программе была введена ещё одна функция - `int znak(FILE *fp, FILE *f, char* a)`, которая отвечает за то, чтобы считывать по символу из

входных данных по мере необходимости и выводить их в файл и на экран. Это позволяет сэкономить место и не прописывать много раз одни и те же строки.

Вся информация о переходах в другие функции выводится по ходу программы на консоль и в файл. Завершает программу сообщение о том, являются ли всё-таки введённые символы скобками или нет. В случае, если это скобки, но в конце приписаны лишние символы, об этом тоже сообщается и за скобки данная конструкция не принимается.

Тестирование.

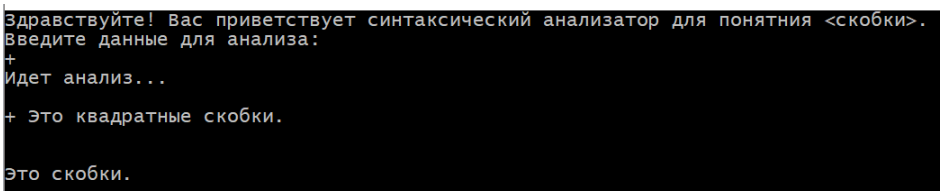
Для начала рассмотрим примеры (см.табл.1), в которых входные данные являются скобками.

Таблица 1 – Корректные данные

Ввод данных	Вывод данных
+	+ Это квадратные скобки. Это скобки.
{[-0](0+)}	{ Это фигурная скобка. Выполняем проверку дальше... [Это квадратная скобка. Выполняем проверку дальше... - Это круглые скобки. 0 Это фигурные скобки.] Это квадратная скобка. (Это круглая скобка. Выполняем проверку дальше... 0 Это фигурные скобки. + Это квадратные скобки.) Это круглая скобка. } Это фигурная скобка. Это скобки.
{+-}	{ Это фигурная скобка. Выполняем проверку дальше... + Это квадратные скобки. - Это круглые скобки. } Это фигурная скобка. Это скобки.

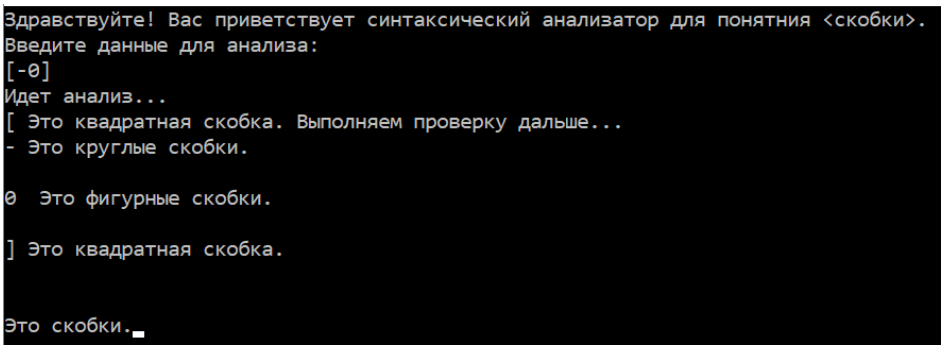
Ввод данных: ‘+’.

Вывод:



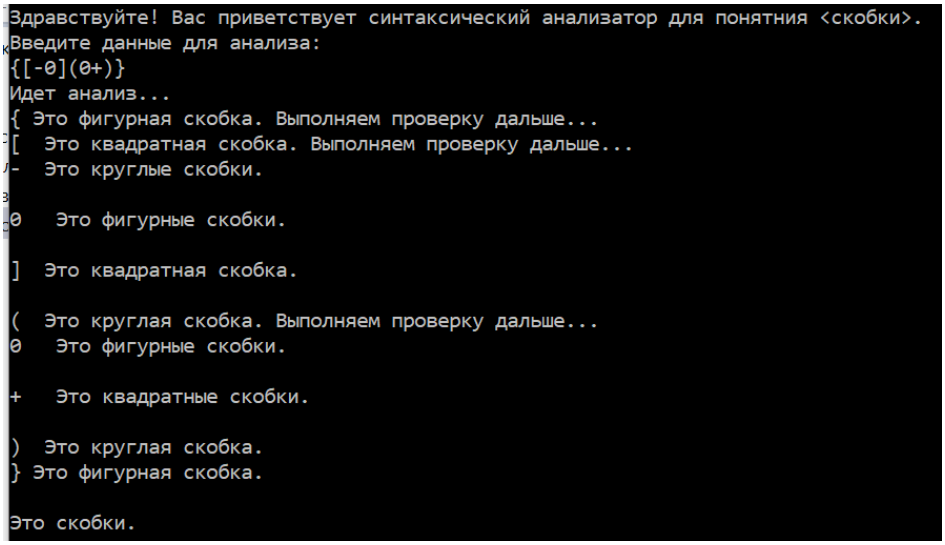
Ввод данных: ‘[-0]’.

Вывод:



Ввод данных:’ {[-0](0+)}’.

Вывод:



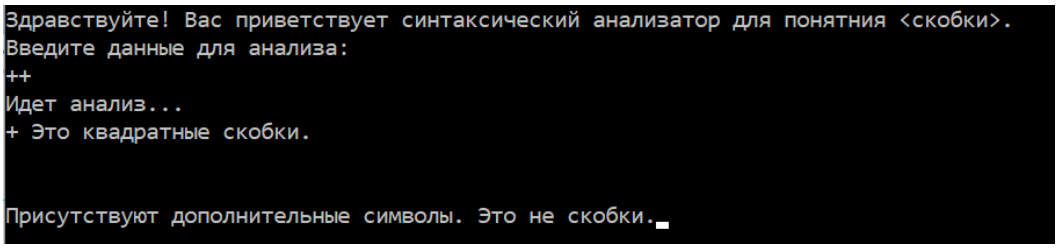
Теперь рассмотрим примеры(см.табл.2), в которых скобок нет.

Ввод данных	Вывод данных
++	+ Это квадратные скобки. Присутствуют дополнительные символы. Это не скобки.
[+-]	[Это квадратная скобка. Выполняем проверку дальше... +

	Это не скобки.
{}	{ Это фигурная скобка. Выполняем проверку дальше... } Это не скобки.

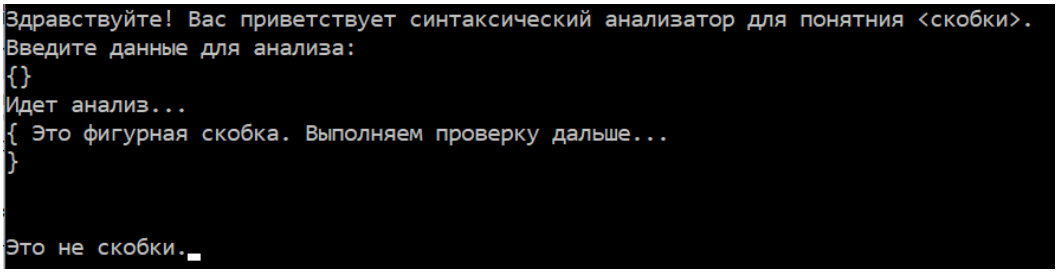
Ввод данных:’++’/.

Вывод:



Ввод данных:’{}’.

Вывод:



Выводы.

В ходе выполнения данной лабораторной работы были закреплены знания о рекурсии и освоено создание синтаксического анализатора на примере распознавания скобок.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
int square(FILE *fp, FILE *f, char a, int i);// Функция для идентификации квадратных скобок
int round(FILE *fp, FILE *f, char a, int i);// -//- круглых скобок
int figure(FILE *fp, FILE *f, char a, int i);// -//- фигурных скобок
int znak(FILE *fp, FILE *f, char* a); //Функция для считывания и вывода данных посимвольно
int brackets(FILE *fp, FILE *f); // функция, которая проверяет является ли введенная последовательность скобками
int main() {

	FILE *fp; FILE *f;  char arr[10000];
	setlocale(LC_ALL, "rus");
	//Считываем данные с клавиатуры и записываем в файл input для дальнейшего использования
	fp = fopen("input.txt", "w"); //открыли файл ввода, для записи туда данные с консоли
	printf("Здравствуйте! Вас приветствует синтаксический анализатор для понятния <скобки>.\n");
	printf("Введите данные для анализа:\n");
	scanf("%s", arr);
	fputs(arr, fp);
	fclose(fp);

	fp = fopen("input.txt", "r");//открыли файл ввода
	if (fp == NULL) {
		fprintf(stderr, "Error: file 'input.txt' is not opened!\n");
		exit(EXIT_FAILURE);
	}
	f = fopen("output.txt", "w");//открыли файл вывода
	if (f == NULL) {
		fprintf(stderr, "Error: You can`t create 'output.txt' file!\n");
		exit(EXIT_FAILURE);
	}
	printf("Идет анализ...\n");
	getchar();
	brackets(fp, f);
	fclose(fp);//закрытие файла ввода
	fclose(f);//и вывода
	return 0;
}

int brackets(FILE *fp, FILE *f) {

	char a; int b = 0; int i = 0;
	znak(fp, f, &a);
	if (a == EOF) {//Проверка на наличие данных в файле
		printf("The file is empty");
		fprintf(f, "The file is empty");
		return 0;
	}
```

```
	//Проверка первого символа. Если он не соответствует возможному началу скобок, программа завершает работу, иначе - переходит в функцию для дальнейшей проверки на скобки
	if (a == '+' || a == '[')
		b = square(fp, f, a, i);
	if (a == '-' || a == '(')
		b = round(fp, f, a, i);
	if (a == '0' || a == '{')
		b = figure(fp, f, a, i);
	if (b == 1 && fscanf(fp, "%c", &a) == 1) {//Случай, если данные - скобки, но в конце присутствуют лишние символы
		printf("\nПрисутствуют дополнительные символы. Это не скобки.");
		fprintf(f, "\nПрисутствуют дополнительные символы. Это не скобки.");
		getchar();
		return 0;
	}
	if (b != 1) {
		printf("\nЭто не скобки.");
		fprintf(f, "\nЭто не скобки.");
	}
	if (b == 1) {
		printf("\nЭто скобки.");
		fprintf(f, "\nЭто скобки.");
	}
	getchar();
	return 0;
}

int square(FILE *fp, FILE *f, char a, int i) {
	for (int k = 0; k < i; k++) { //печать отступа, соответствующего глубине рекурсии. i - счётчик, который увеличивается при новом входе в функцию проверки скобок и уменьшается при выходе
		printf(" ");
		fprintf(f, " ");
	}
	i++;
	if (a == '+') {//первый из допустимых вариантов символа для данных скобок, дальнейшая проверка для них не требуется
		i--;
		printf(" Это квадратные скобки.\n");
		fprintf(f, " Это квадратные скобки.\n");
		getchar();
		return 1;
	}
	if (a == '[') {//второй из допустимых вариантов символа для данных скобок
		printf(" Это квадратная скобка. Выполняем проверку дальше...\n");
		fprintf(f, " Это квадратная скобка. Выполняем проверку дальше...\n");
		znak(fp, f, &a);
		if (round(fp, f, a, i) == 1) {//проверка на то, находятся ли внутри круглые скобки. Если так, функция продолжает работу
			znak(fp, f, &a);
			if (figure(fp, f, a, i) == 1) {//проверка на то, находятся ли внутри фигурные скобки. Если так, то далее потребуется лишь закрывающий символ квадратных скобок
				znak(fp, f, &a);
				if (a == ']') {//проверка на завершающий символ данных скобок
					i--;
					for (int k = 0; k < i; k++) {
						printf(" ");
						fprintf(f, " ");
					}
```

```
                                    fprintf(f, " Это квадратная скобка.\n");
                                    printf(" Это квадратная скобка.\n");
                                    getchar();
                                    return 1;

                              }
                        }
                  }
            }
            getchar();
            return 0;
}

int round(FILE *fp, FILE *f, char a, int i) {//функция похожа на предыдущую, поэтому
комментарии будут вставлены только там, где есть какие-либо различия
            for (int k = 0; k < i; k++) {
                  printf(" ");
                  fprintf(f, " ");
            }
            i++;
            if (a == '-') {
                  i--;
                  printf("Это круглые скобки.\n");
                  fprintf(f, "Это круглые скобки.\n");
                  getchar();
                  return 1;
            }
            if (a == '(') {
                  printf(" Это круглая скобка. Выполняем проверку дальше...\n");
                  fprintf(f, " Это круглая скобка. Выполняем проверку дальше...\n");
                  znak(fp, f, &a);
                  if (figure(fp, f, a, i) == 1) { //проверка на то, находятся ли внутри
фигурные скобки. Если так, функция продолжает работу
                        znak(fp, f, &a);
                        if (square(fp, f, a, i) == 1) {//проверка на то, находятся ли внутри
квадратные скобки. Если так, то далее потребуется лишь закрывающий символ квадратных
скобок
                              znak(fp, f, &a);
                              if (a == ')') {
                                    i--;
                                    for (int k = 0; k < i; k++) {
                                          printf(" ");
                                          fprintf(f, " ");
                                    }
                                    printf(" Это круглая скобка.\n");
                                    fprintf(f, " Это кругая скобка.\n");
                                    return 1;
                              }
                        }
                  }
            }
            getchar();
            return 0;
}

int figure(FILE *fp, FILE *f, char a, int i) {//функция похожа на предыдущую, поэтому
комментарии будут вставлены только там, где есть какие-либо различия
            for (int k = 0; k < i; k++) {
                  printf(" ");
                  fprintf(f, " ");
            }
```

```
	i++;
	if (a == '0') {
		i--;
		printf(" Это фигурные скобки.\n");
		fprintf(f, " Это фигурные скобки.\n");
		getchar();
		return 1;
	}
	if (a == '{') {
		printf(" Это фигурная скобка. Выполняем проверку дальше...\n");
		fprintf(f, " Это фигурная скобка. Выполняем проверку дальше...\n");
		znak(fp, f, &a);
		if (square(fp, f, a, i) == 1) {//проверка на то, находятся ли внутри
квадратные скобки. Если так, функция продолжает работу
			znak(fp, f, &a);
			if (round(fp, f, a, i) == 1) {//проверка на то, находятся ли внутри
круглые скобки. Если так, то далее потребуется лишь закрывающий символ квадратных
скобок
				znak(fp, f, &a);
				if (a == '}') {
					i--;
					for (int k = 0; k < i; k++) {
						printf(" ");
						fprintf(f, " ");
					}
					printf(" Это фигурная скобка.\n");
					fprintf(f, " Это фигурная скобка.\n");
					return 1;
				}
			}
		}
	}
	getchar();
	return 0;
}

int znak(FILE *fp, FILE *f, char* a) {
	*a = fgetc(fp);
	printf("%c", *a);
	fputc(*a, f);
	return 0;
}
```