**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. 6383 |  | Михайлов Ю.А. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017

**Цель работы.**

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования С++.

**Задание (вариант №4).**

Построить синтаксический анализатор для понятия *скобки.*

*скобки::=*А |(B *скобки  скобки*).

**Основные теоретические положения.**

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.

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

⎧ 1, если *n* = 0;

fact (*n*) = ⎨ (2.3)

⎩ fact (*n* − 1) ⋅ *n*, если *n* > 0;

или в другой форме записи

fact (*n*) ≡ **if** *n* = 0 **then** 1 **else** fact (*n* − 1) ⋅ *n*, (2.4)

где использовано условное выражение **if** *b* **then** *e*1**else** *e*2, означающее, что в том месте, где оно записано, следует читать e1, если выполняется условие b, и следует читать e2, если условие b не выполняется.

Функция, определяемая таким образом, единственна. Действительно, пусть есть две функции, например: fact1 (n) и fact2 (n), удовлетворяющие со-отношениям (2.2) или их эквивалентам (2.3), (2.4). Рассмотрим разность dfact (n) = fact1 (n) − fact2 (n). Очевидно, что, во-первых, в силу соотношения «а» из (2.2) имеем dfact (0) = 0, а, во-вторых, для функции dfact (n) также справедливо соотношение «б». Действительно,

dfact (n) = fact1 (n) − fact2 (n) = fact1 (n − 1) · n − fact2 (n − 1) · n =

= (fact1 (n − 1) − fact2 (n − 1)) · n = dfact (n − 1) · n.

По индукции легко доказывается, что из соотношений dfact (0) = 0 и dfact (n) = dfact (n − 1) · n следует, что dfact (n) = 0 для любого n > 0.

**Спецификация программы.**

*Назначение программы*.

Программа предназначена для определения понятия «скобки».

*Описание программы*.

Программа написана на языке C++. Входными данными для программы являются символы, вводимые пользователем с клавиатуры или из файла. Проверка значений при некорректном вводе количества символов предусмотрена. Выходными данными является ответ на вопрос: «соответствует ли введенная комбинация понятию «скобки»?».

*Пример диалога с пользователем*.

1. Введите комбинацию символов:

(BAA)

Это скобка!

1. Введите комбинацию символов:

AAA

Это не скобка!

*Реализация*.

Функции:

1. bool \_bracketsEnd(std::istream& in)
2. bool \_bracketsCont(std::istream& in)
3. bool \_bracketsBegin(std::istream& in)

Эти три функции принимают на вход строку символов и обрабатывают эту введенную комбинацию для определения понятия «скобки». Они взаимодействуют между собой с помощью рекурсий.

**Тестирование.**

|  |  |
| --- | --- |
| **Комбинация** | **Результат** |
| A | Это скобка! |
| (BAA) | Это скобка! |
| (BAA)(BAA) | Это не скобка! |
| 4 | Это не скобка! |

**Вывод.**

Была разработана программа, которая выясняет, задано ли определение «скобки» пользователем. В результате выполнения работы был получен опыт программирования на языке С++, а также навыки программирования рекурсивных функций.

**Приложение А. Исходный код.**

#include <iostream>

#include <cmath>

#include <Windows.h>

using namespace std;

/\*\*

\* скобки: A | (B скобки скобки)

\*/

bool \_bracketsBegin(std::istream& in);

bool \_bracketsCont(std::istream& in);

int check = 0;

bool \_bracketsEnd(std::istream& in) {

while (true) {

switch (in.peek()) {

case '(':

in.get();

if (!\_bracketsCont(in)) return false;

return true;

break;

case 'A':

check = -1;

in.get();

if (!\_bracketsBegin(in)) return false;

return true;

break;

default:

return false;

break;

}

}

}

bool \_bracketsCont(std::istream& in) {

while (true) {

switch (in.peek()) {

case 'B':

check = -1;

in.get();

if (!\_bracketsEnd(in)) return false;

return true;

break;

default:

return false;

break;

}

}

}

bool \_bracketsBegin(std::istream& in) {

while (true) {

switch (in.peek()) {

case '(':

in.get();

if (!\_bracketsCont(in)) return false;

return true;

break;

case ')':

in.get();

if (in.peek() == ')') return false;

if (!\_bracketsBegin(in)) return true;

break;

case 'A':

in.get();

if (in.peek() != 'A'&& check == 0) return false;

if (!\_bracketsBegin(in)) return true;

return true;

break;

default:

return false;

break;

}

}

return true;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Введите комбинацию символов:" << endl;

if (\_bracketsBegin(cin)) {

cout << "Это скобка!" << endl;

}

else cout << "Это не скобка!" << endl;

system("pause");

return 0;

}