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

Студент гр. 9382 _____ Кузьмин Д. И.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург
2020

Цель работы.

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

Основные теоретические положения.

Рекурсия — определение, описание какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, когда объект является частью самого себя.

Задание.

Вариант 12. Построить синтаксический анализатор для понятия *скобки*.

скобки::=квадратные | *круглые* | *фигурные*
квадратные::=[*круглые фигурные*] | +
круглые::=(*фигурные квадратные*) | -
фигурные::={*квадратные круглые*} | 0

Описание алгоритма

1) Сначала проверяется первый символ последовательности. Затем в зависимости от него проверяется второй, затем в зависимости от него третий и тд. Проверка осуществляется рекурсивно. В случае, если открывающий символ, символы внутри и закрывающий символ совпадают с каким-либо типом скобок, то последовательность считается скобками, в противном случае – нет. Также проверяется нет ли в последовательности лишних символов.

Описание функций и структур данных

1) bool Square(char c, FILE* F) – проверка является ли последовательность символов квадратными скобками, char c – это символ, с которого начинается проверка, FILE* F – это файл с которого производится считывание. Возвращает true или false.

2) bool Circle(char c, , FILE* F) – проверка, является ли последовательность символов круглыми скобками. char c – это символ, с

которого начинается проверка, FILE* F – это файл с которого производится считывание. Возвращает true или false.

3) bool Figure(char c, , FILE* F) – проверка, является ли последовательность символов фигурными скобками. char c – это символ, с которого начинается проверка, FILE* F – это файл с которого производится считывание. Возвращает true или false

4) void Bracket(FILE* F = nullptr) – проверка, является последовательность символов скобками в принципе. FILE* F – файл, с которого считывается последовательность.

Исходный код см. в приложении А.

Тестирование.

Результаты тестирование представлены в табл. 1.

Таблица 1 — результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарий
0	[-{[-0](0+)}]	Это скобки	Действительно, это скобки. Такие скобки можно создать из первого символа +.
1	[-{[-0](0-)}]	Это не скобки	Действительно, В третьей с начала скобки второй символ – 0, а должен быть +.
2	+	Это скобки	Отдельный символ тоже может быть скобками
3	{[-0](0[-0])}	Это скобки	Такие скобки можно создать из первого символа 0
4	{+(0[-0])]}	Это не скобки	Предпоследний символ “]” здесь лишний

Выводы.

Был изучен принцип рекурсии. Получены навыки программирования рекурсивных функций. Разработана программа-синтаксический анализатор, определяющая является ли последовательность скобками или нет.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <cstdlib>
using namespace std;

/*
   + = [-0] (квадратные скобки)
   - = (0+) (круглые скобки)
   0 = {+-} (фигурные скобки)
*/

bool Square(char c, FILE* F = nullptr);
bool Round(char c, FILE* F = nullptr);
bool Figure(char c, FILE* F = nullptr);
void Bracket(FILE* F = nullptr);

int main() {
      setlocale(LC_ALL, "Russian");

      cout << "Введите 1 для ввода с консоли и 2 для ввода с файла\n";
      char c = cin.get();
      if (c == '1') {
            cin.get();
            Bracket();
      }
      else if (c == '2') {

            FILE* F = std::fopen("test.txt", "r");
            if (!F) cout << "Ошибка при открытии файла\n";
            else {
                  Bracket(F);
            }
      }
      else cout << "Некорректный ввод\n";
      return 0;
}

bool Square(char c, FILE* F) {
      cout << "Проверка символа " << c << "\n";
      if (c == '+') return true;
      else if (c == '[') {
            (!F) ? c = cin.get() : c = std::getc(F);
            if (Round(c,F)) { //проверка первого знака скобки. если да -
переходим к проверке второго
                  (!F) ? c = cin.get() : c = std::getc(F);
                  if (Figure(c,F)) { // проверка вторго знака скобки. если да -
переходим к проверке закрывающего символа
                        (!F) ? c = cin.get() : c = std::getc(F);
                        cout << "Проверка символа " << c << "\n";
                        if (c == ']') //проверяем наличие закрывающего символа.
если он есть - возвращаем значение true
                              return true;

                        else {
                              cout << "Закрывающий символ отсутствует или
неверен\n";
                              return false; //в случае если символа нет, выводим
соответствующее сообщение
                        }
                  }
                  else {
                        cout << "Второй знак скобки неверен или отсутствует \n";
```

```
                        return false; // вывод сообщения о неуспешной проверки
второго символа скобки
                  }
            }
            else {
                  cout << "Первый знак неверен\n";
                  return false; // вывод сообщения о неуспешной проверки первого
символа скобки
            }
      }
      else return false;

      return false;
}
bool Round(char c, FILE* F) { //функция проверяет является ли скобка круглой
      cout << "Проверка символа " << c << "\n";
      if (c == '-') return true;

      else if (c == '(') {
            (!F) ? c = cin.get() : c = std::getc(F);
            if (Figure(c,F)) { //проверка, является ли первый символ фигурной
скобкой
                  (!F) ? c = cin.get() : c = std::getc(F);
                  if (Square(c,F)) { //проверка, является ли второй символ
квадратной скобкой
                        (!F) ? c = cin.get() : c = std::getc(F);
                        cout << "Проверка символа " << c << "\n";
                        if (c == ')') return true; //проверка наличия
закрывающего символа
                        else {
                              cout << "Закрывающий символ пропущен или
некорректен \n";
                              return false; //если его нет, выводится сообщение
об этом и функция завершает работу, возвратив при этом false
                        }
                  }
                  else {
                        cout << "Второй знак скобки неверен или отсутствует
\n";// сообщение о некорректности второго символа скобки
                        return false;
                  }
            }
            else {
                  cout << "Первый знак некорректен\n";//сообщение о
некорректности первого символа скобки
                  return false;
            }
      }
      return false;
}



bool Figure(char c, FILE* F) {
      cout << "Проверка символа " << c << "\n";
      if (c == '0') return true; //проверка, является ли
      else if (c == '{') {
            (!F) ? c = cin.get() : c = std::getc(F);
            if (Square(c,F)) { //проверка первого знака скобки. если да -
переходим к проверке второго
                  (!F) ? c = cin.get() : c = std::getc(F);
                  if (Round(c,F)) {// проверка вторго знака скобки. если да -
переходим к проверке закрывающего символа
```

```
                        (!F) ? c = cin.get() : c = std::getc(F);
                        cout << "Проверка символа " << c << "\n";
                        if (c == '}') return true;
                        else {
                              cout << "Закрывающий символ пропущен или
некорректен \n";
                              return false;
                        }
                  }
                  else {
                        cout << "Второй символ скобки неверен \n"; //вывод
сообщения об ошибке
                        return false;
                  }
            }
            else {
                  cout << "Первый символ некорректен\n";//вывод информации об
инкорректности первого символа скобки
                  return false;
            }
      }
      return false;
}


void Bracket(FILE* F) {

      char c;
      char* s = new char[20];
      (!F) ?c = cin.get():c = std::getc(F);
      //if (F) {
            //cout << s << endl;
      //}
      bool isBracket;

      switch (c) { //считывание первого символа и определение дальнейших
действий
      case '+':

            (!F) ? isBracket = Square(c) : isBracket = Square(c, F); //в случае,
если первый символ +,- или 0, то проверяется нет ли еще символов
            break;

      case '-':

            (!F) ? isBracket = Round(c) : isBracket = Round(c, F); //если есть,
то эти символы считаются лишние и выводится сообщение об ошибке
            break;

      case '0':

            (!F) ? isBracket = Figure(c) : isBracket = Figure(c, F);
            break;

      case '(':
            (!F) ? isBracket = Round(c) : isBracket = Round(c, F);//проверка
первого знака, является ли он началом скобки
            break;

      case '[':
            (!F) ? isBracket = Square(c) : isBracket = Square(c, F);//в
соответствии с каждым символом вызывается определенная функция проверки скобки
            break;
```

```
	case '{':

		(!F) ? isBracket = Figure(c) : isBracket = Figure(c, F);
		break;

	default:
		isBracket = false;
		break;

	}
	if (isBracket && !F) {
		if (cin.get() != '\n') {
			cout << "Это не скобки\n";//проверка на наличие лишних
символов в консоли
		}
		else cout << "Это скобки\n";
	}
	else if (isBracket && F) { //проверка на наличие лишних символов в файле
		char c = std::getc(F);
		if (c == '\n' || c == EOF) {

			cout << "Это скобки\n\n";
			Bracket(F);
		}
		else {
			cout << "Присутствуют лишние символы\n";
			//cout << "Это не скобки\n\n";
			char c = std::getc(F);
			if (c != EOF)cout << "Это не скобки\n\n";
			while (1) {
				if (c == '\n' || c == EOF) {
					if (c != EOF) Bracket(F);
					break;
				}
				c = std::getc(F);
			}

		}

	}
	else if (!isBracket && F) { //проверка является ли строка в файле
последней, если нет - проверяется следующая строка
		char c = std::getc(F);
		if (c != EOF)cout << "Это не скобки\n\n";
		while (1) {
			if (c == '\n' || c == EOF) {
				if (c != EOF) Bracket(F);
				break;
			}
			c = std::getc(F);
		}

	}
	else cout << "Это не скобки\n";
}
```