

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования «Томский государственный университет систем  
управления и радиоэлектроники»

Факультет систем управления (ФСУ)

Кафедра автоматизированных систем управления (АСУ)

Стеки и очереди  
отчет по лабораторной работе №1 по дисциплине  
«Структуры и алгоритмы обработки данных в ЭВМ»

Обучающийся гр. 431-3

\_\_\_\_\_ Сергиевский Д.В.

«\_\_\_» \_\_\_\_\_ 202\_ г.

Проверил: доцент каф. АСУ, д.т.н.

\_\_\_\_\_ Горитов А. Н.

«\_\_\_» \_\_\_\_\_ 202\_ г.

Томск 202\_

## Содержание

Введение.....	3
1 Ход работы.....	4
1.1 Алгоритм решения.....	4
1.2 Реализация решения.....	5
Вывод.....	6
Приложение А.....	7

## Введение

В рамках данной лабораторной работы необходимо решить небольшую задачу с применением АТД Стек для закрепления теоретического материала.

Задание на лабораторную работу представлено на Рисунке 1.

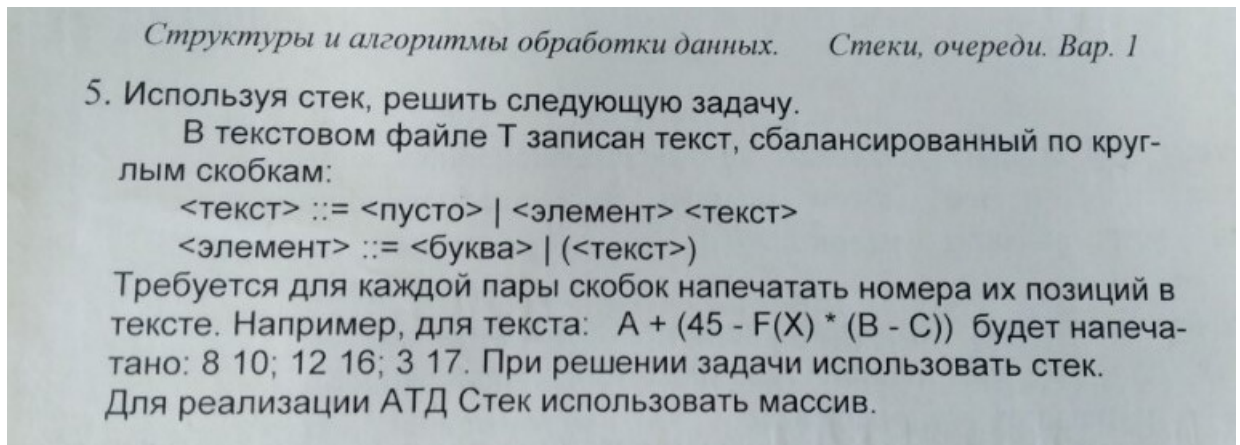


Рисунок 1 - задание

## **1 Ход работы**

### **1.1 Алгоритм решения**

Необходимые для вывода данные разнесены по содержимому и обладают рекурсивной вложенностью. В связи с чем требуются либо многократные проходы по входным данным, либо применение динамической структуры данных для хранения фрагментов информации. Согласно требованиям, в качестве данной структуры должна выступать АТД Стек, причём в статическом варианте реализации, что подразумевает достаточность размера хранилища.

Для решения данной задачи необходимо посимвольно считать файл, содержащий входные данные, подсчитывая количество считанных символов для определения положения текущего символа в тексте. В случае считывания открывающей скобки, индекс её позиции в тексте заносится в стек. В случае считывания закрывающей скобки, из стека извлекается индекс, соответствующий предыдущей открывающей скобки, то есть парной для текущей закрывающей скобки, после чего производится вывод текущего и извлеченного индексов, что соответствует информации о положении пары скобок в тексте.

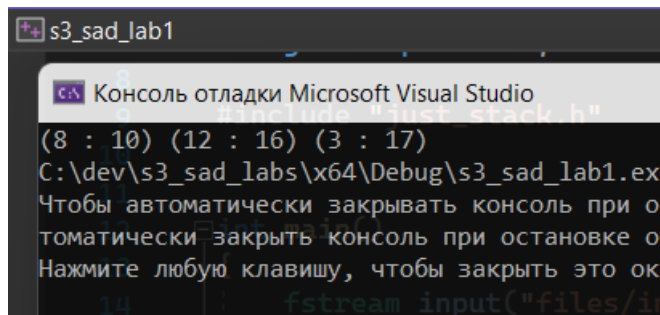
В случае неверных входных данных программа завершается с выводом пояснительного сообщения.

## 1.2 Реализация решения

Для решения данной задачи требуется написать небольшую программу, реализующую вышеописанный алгоритм, а также АТД Стек на основе статического массива в качестве вспомогательного компонента.

Листинг программы приведен в Приложении А. Листинг реализации АТД Стек приведен в Приложении Б.

Результат вывода программы приведен на Рисунке 1.1. В качестве входных данных был использован пример входных данных из текста задания.



```
s3_sad_lab1
Консоль отладки Microsoft Visual Studio
(8 : 10) (12 : 16) (3 : 17)
C:\dev\s3_sad_labs\x64\Debug\s3_sad_lab1.exe
Чтобы автоматически закрывать консоль при о
томатически закрыть консоль при остановке от
Нажмите любую клавишу, чтобы закрыть это окн
14 fstream input("files/in
```

Рисунок 1.1 - пример вывода

## **Вывод**

В результате данной лабораторной работы были подкреплены теоретические знания по теме «АТД Стек» реализацией соответствующей структуры и её применением для решения небольшой задачи.

## Приложение А

Файл s3\_sad\_lab1.cpp. Точка входа в программу и решение задачи.

```
// s3_sad_lab1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается
// выполнение программы.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

#include "just_stack.h"

void do_task(string filename)
{
    fstream input(filename, fstream::in);

    if (!input.is_open())
    {
        throw filename + " didn't open";
    }

    just_stack s(20);

    s.pop();
```

```

char c;

for (size_t i = 1; input >> c; ++i)
{
    switch (c)
    {
        case '(':
            if (s.is_empty())
                throw "unpaired ')' in position " + to_string(i);

            s.push(i);

            break;

        case ')':
            cout << "(" << s.pop() << " : " << i << ") ";

            break;

    }
}

input.close();

if (!s.is_empty())
{
    string str = "unpaired '(' in positions: ";

    do
    {
        str += to_string(s.pop()) + " ";
    }
}

```



```

        } while (!s.is_empty());

        throw str;

    }
}

int main()
{
    try
    {
        do_task("files/input.txt");
    }
    catch (const char* message)
    {
        cout << "error: " << message;

        getchar();
    }
}

// Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"

// Отладка программы: F5 или меню "Отладка" > "Запустить отладку"

// Советы по началу работы

// 1. В окне обозревателя решений можно добавлять файлы и управлять ими.

```

- // 2. В окне Team Explorer можно подключиться к системе управления версиями.
- // 3. В окне "Выходные данные" можно просматривать выходные данные сборки и другие сообщения.
- // 4. В окне "Список ошибок" можно просматривать ошибки.
- // 5. Последовательно выберите пункты меню "Проект" > "Добавить новый элемент", чтобы создать файлы кода, или "Проект" > "Добавить существующий элемент", чтобы добавить в проект существующие файлы кода.
- // 6. Чтобы снова открыть этот проект позже, выберите пункты меню "Файл" > "Открыть" > "Проект" и выберите SLN-файл.

## Приложение Б

Файл just\_stack.h. Содержит объявление стека.

```
#pragma once
```

```
using val_type = unsigned int;
```

```
class just_stack
```

```
{
```

```
protected:
```

```
    val_type* buffer;
```

```
    size_t size;
```

```
    size_t length;
```

```
public:
```

```
    just_stack(size_t size) : size(size), length(0)
```

```
    {
```

```
        buffer = new val_type[size];
```

```
        *buffer = {};
```

```
    }
```

```
    ~just_stack()
```

```
    {
```

```
        delete[] buffer;
```

```
}
```

```
void push(const val_type& val);
```

```
val_type pop();
```

```
val_type top() const;
```

```
bool is_empty() const;
```

```
bool is_full() const;
```

```
void clear();
```

```
};
```

```
#include "just_stack.h"

#include <iostream>

using namespace std;

void just_stack::push(const val_type& val)
{
    if (is_full())
        throw "push() when stack is full";

    buffer[length] = val;
    ++length;
}

val_type just_stack::pop()
{
    if (is_empty())
        throw "pop() when stack is empty";

    --length;
    return buffer[length];
}
```

```
val_type just_stack::top() const
{
    if (is_empty())
        throw "top() when stack is empty";
    return buffer[length-1];
}
```

```
bool just_stack::is_empty() const
{
    return length == 0;
}
```

```
bool just_stack::is_full() const
{
    return length == size;
}
```

```
void just_stack::clear()
{
    length = 0;
}
```