МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

КАФЕДРА МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Стек

Студент гр. 7383 _____ Ласковенко Е.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

Оглавление

Цель работы

Познакомиться со структурой и реализацией стека, получить навыки программирования функций, использующих стек, на языке программирования C++.

Задача

Определить, имеет ли заданная в файле F символьная строка следующую структуру:

a D b D c D d...,

где каждая строка a, b, c, d,..., в свою очередь, имеет вид x1 C x2. Здесь x1 есть строка, состоящая из символов A и B, а x2 - строка, обратная строке x1 (т. е. если x1 = ABABB, то x2 = BBABA). Таким образом, исходная строка состоит только из символов A, B, C и D. Исходная строка может читаться только последовательно (посимвольно) слева направо.

Реализация задачи

Описание функций, классов и переменных. Описание работы алгоритма

Функция **int main()**:

Является головной функцией, в которой происходит считывание данных из файла (название файла вводит пользователь в консоли). Если файла с введенным названием нет или же файл невозможно открыть на чтение, то в консоль выводится ошибка, а программа снова требует ввести название файла на открытие. Далее объявляются необходимые переменные, описанные ниже, происходит вывод на экран содержимого файла. Если файл пуст, то выводится соответствующее предупреждение, если файл не пустой, то происходит вызов функций анализатора строки. Все вызовы находятся в цикле, выходом из которого является условие ввода пользователем пустой строки в консоли. Также в функции реализована обработка исключений — проверка на валидность входных данных, проверка наличия файла с данными и прочее.

Объявляемые переменные:

- string filename — строковая переменная, хранящая имя файла, с которого необходимо считать данные.
- string temp_str — строковая переменная, хранящая входные данные.
- filebuf file — файловый буфер, используемый при считывании входных данных с файла.
- stringbuf str_buf — строковый буфер, используемый при считывании входных данных с консоли.
- istream is_str — входной поток, в который помещаются входные данные.
- char istr_null — символьная переменная, необходимая для проверки входных данных на валидность.

- auto size — переменная, хранящая размер (количество символов) файла с входными данными.

Функция **bool syn_analyzer(istream& is_str, long size):**

Функция, принимающая на вход входной поток с исходными данными и размер строки, содержащейся во входном потоке. Использует методы класса Stack. Обеспечивает проверку входных данных. Состоит из итеративного цикла. В цикле происходит считывание с потока символа, далее первым условием является проверка символа на соответствие символам из x1. Если это условие выполняется, то этот символ заносится в стек. Иначе проверяется символ на соответствие символу „C“. Если это условие выполняется, то происходит проверка на пустоту стека. Если стек пуст, то на экран выводится ошибка (срабатывает исключение). Если стек не пуст, то с помощью цикла происходит считывание символов с потока и проверка входных символов на соответствие символам из стека. В случае несоответствия x1 и перевернутой x2 выбрасывается исключение и обрабатывается. Если же входной символ равен символу «D», тогда происходит проверка флага, который обозначает, была ли структура x1 C x2 до этого символа. После завершения внешнего цикла происходит проверка на пустоту стека (стек должен быть пуст, иначе — исключение) и функция возвращает значение флага.

Объявляемые переменные:

- char ch — символьная переменная, текущий просматриваемый символ.
- bool flag — булевая переменная, являющаяся индикатором валидности строки.
- Stack<char> st(ARR_SIZE) — объект класса Stack, стек размера ARR_SIZE.

Класс **Stack:**

Класс стека. Состоит из следующих полей: массива шаблонного типа arr, размера массива, индекс последнего элемента (верхушка стека). Для класса были реализованы следующие методы:

- Stack<BASE>::Stack(long size) - конструктор класса. Принимает на вход размер массива, выделяет память под массив указанного размера, инициализирует поле размера входным аргументом, инициализирует поле индекса нулевым значением.
- BASE Stack<BASE>::top() — метод, возвращающий верхний элемент стека, но не удаляющий его.
- BASE Stack<BASE>::pop() - метод, возвращающий верхний элемент стека и удаляющий его из стека.
- Void Stack<BASE>::resize(long new_size) - метод, который увеличивает размер стека. Принимает на вход новый размер стека, выделяет память под массив нового размера, инициализирует его значениями старого массива, инициализирует поле размера новым значением, вызывает деструктор для старого массива, а новым инициализирует поле массива.
- Void Stack<BASE>::push(BASE el) — метод, который записывает элемент в начало стека. Проверяет размер массива и при необходимости вызывает метод resize, увеличивая размер массива в 2 раза. Инициализирует верхушку стека новым значением, прибавляет к индексу единицу.
- Void Stack<BASE>::clear() — метод, инициализирующий массив и индекс последнего элемента нулевыми значениями.
- Bool Stack<BASE>::is_empty() — метод, возвращающий значение TRUE, если стек пуст. Иначе возвращает FALSE.
- Stack<BASE>::~Stack<BASE>() — деструктор класса.

Тестирование программы

Процесс тестирования

Программа собрана в операционной системе Linux Mint 19, с использованием компилятора G++. В других ОС и компиляторах тестирование не проводилось.

Результаты тестирования

Тестовые случаи представлены в Приложении А.

По результатам тестирования было показано, что поставленная задача была выполнена.

Выводы

В ходе лабораторной работы были получены навыки работы со стеком. В работе был реализован шаблонный класс, который представляет собой стек на базе вектора. Стек удобно использовать, если необходимо хранить данные, упорядоченные в одностороннем порядке.

ПРИЛОЖЕНИЕ А.

Тестовые случаи

Входные данные	Вывод	Верно?
A B B B C B B B A A D A B C B A D A C A D B C B	File contains: A B B B C B B B A A D A B C B A D A C A D B C B x2 is not reversed x1!	Да
A B B B C B B B A D A B C B A D A C A D B C B	File contains: A B B B C B B B A D A B C B A D A C A D B C B It is an expression	Да
A B B B C B B B A	File contains: A B B B C B B B A It is an expression	Да
A B B B C B B B A D	File contains: A B B B C B B B A D It isn't an expression!	Да
A B D C B A	File contains: A B D C B A D is before a,b,c,d-elements?	Да
E A B D C B A	File contains: E A B D C B A Wrong character in string! (Not A, B, C or D)	Да

ПРИЛОЖЕНИЕ Б.

Код программы

1. Код **main.cpp:**

```
#include <istream>
#include <fstream>
#include <sstream>
#include <string>
#include <exception>
#include "stack.h"
#include "analyzer.h"
#include "myexception.h"

int
main()
{
    string filename;
    string temp_str;
    filebuf file;
    stringbuf str_buf;
    istream is_str(&str_buf);

    while(true)
    {
        try
        {
             cout << "Enter a name of the data-file or press ENTER to exit:"
<< endl;
            getline(cin, filename);
            if(filename.empty())
                break;
            if(!file.open(filename, ios::in))
            {
               file.close();
               throw new client_err("Incorrect filename.");
            }

            auto size = file.in_avail();
            char temp_ch;

            cout << "File contains: ";
            for(auto c_size = 0; c_size<size; c_size++)
            {
                temp_ch=file.sbumpc();
                cout << temp_ch;
                if(temp_ch != ' ' && temp_ch != '\n')
                    temp_str.append(1, temp_ch);
            }
            cout << endl;
            file.close();
            if(!temp_str.size())
                throw new client_err("Input file is empty.");
            str_buf.str(temp_str);
            temp_str.clear();
            if(syn_analyzer(is_str, str_buf.in_avail()))
                cout << "It is an expression" << endl;
            else
                throw new client_err("It isn't an expression!");
        }
        catch (exception* ex)
```

```
        {
            cout << ex->what() << endl;
        }

    }
    return 0;
}
```

2. Код **stack.h:**

```
#ifndef STACK_H
#define STACK_H

template<class BASE>
class Stack
{
public:
    Stack(long);
    BASE top();
    BASE pop();
    void resize(long);
    void push(BASE);
    void clear();
    bool is_empty();
    ~Stack();
private:
    BASE* arr;
    int arr_size;
    int end_index;
};

template<class BASE>
Stack<BASE>::Stack(long size)
{
    arr = new BASE[size];
    arr_size = size;
    end_index = 0;
}

template<class BASE>
BASE
Stack<BASE>::top()
{
    return arr[end_index-1];
}

template<class BASE>
BASE
Stack<BASE>::pop()
{
    end_index--;
    return arr[end_index];
}

template<class BASE>
void
Stack<BASE>::resize(long new_size)
{
    auto new_arr = new BASE[new_size];
    for(int i=0; i<arr_size; i++)
        new_arr[i] = arr[i];
    arr_size = new_size;
```

```
    delete arr;
    arr = new_arr;
}

template<class BASE>
void
Stack<BASE>::push(BASE el)
{
    if(end_index == arr_size-1)
        resize(arr_size*2);

    arr[end_index] = el;
    end_index++;
}

template<class BASE>
void
Stack<BASE>::clear()
{
    for(int i=0; i<end_index; i++)
        arr[end_index] = 0;
    end_index = 0;
}

template<class BASE>
bool
Stack<BASE>::is_empty()
{
    return end_index == 0;
}

template<class BASE>
Stack<BASE>::~Stack<BASE>()
{
    clear();
    delete[] arr;
}

#endif // STACK_H
```

3. Код **analyzer.h:**

```
#include <iostream>
#include "stack.h"
#include "myexception.h"

#define ARR_SIZE 10

using namespace std;

#ifndef ANALYZER_H
#define ANALYZER_H

bool
syn_analyzer(istream& is_str, long size)
{
    char ch;
    bool flag = false;
    Stack<char> st(ARR_SIZE);

    for(int i=0; i<size; i++)
    {
```

```
        is_str >> ch;
        if(ch=='A' || ch=='B')
            st.push(ch);
        else if(ch=='C')
        {
            if(st.is_empty())
                throw new analyzer_err("Where is x1? (Stack is empty)");
            while(!st.is_empty())
            {
                if(is_str >> ch)
                {
                    if(st.pop()==ch)
                        flag = true;
                    else
                        throw new analyzer_err("x2 is not reversed x1!");
                    i++;
                }
                else
                    throw new analyzer_err("x2 is wrong! (x1 is greater)");
            }
        }
        else if(ch=='D')
        {
            if(!flag)
                throw new analyzer_err("D is before a,b,c,d-elements?");
            flag = false;
        }
        else
              throw new analyzer_err("Wrong character in string! (Not A, B, C
or D)");
    }
    if(!st.is_empty())
        throw new analyzer_err("There is no C in a,b,c,d-element!");
    return flag;
}

#endif // ANALYZER_H
```

4. Код **myexception.h:**

```
#include <exception>
#include <iostream>


using namespace std;
#ifndef MYEXCEPTION_H
#define MYEXCEPTION_H
class analyzer_err : public exception
{
public:
    analyzer_err(const char* exp) noexcept
    {
        msg = exp;
    }
    virtual const char* what() const noexcept
    {
```

```
        return msg;
    }
    virtual ~analyzer_err(){}
private:
    const char* msg;
};

class client_err : public exception
{
public:
    client_err(const char* exp) noexcept
    {
        msg = exp;
    }
    virtual const char* what() const noexcept
    {
        return msg;
    }
    virtual ~client_err(){}
private:
    const char* msg;
};

#endif // MYEXCEPTION_H
```

5. Код **Makefile:**

```
all: main.cpp analyzer.h stack.h
      g++ main.cpp -o start

clean:
      rm start
```