**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: Работа со стеком и очередями

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Федоров И.А |
| Преподаватель |  | Фирсов М.А |

Санкт-Петербург

2019

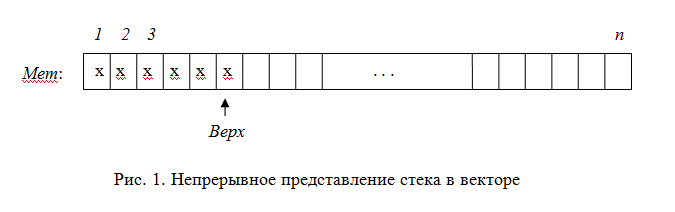
**Цель работы.**

Ознакомиться с абстрактными структурами данных, такими как стек и очередь, и способами работы с ними. Изучить методы работы со стеком на примере вычисления выражения в постфиксной записи.

**Основные теоретические положения.**

Стек (англ. *stack* – стопка) – это структура данных, в которой новый элемент всегда записывается в ее начало (вершину) и очередной читаемый элемент также всегда выбирается из ее начала. В стеках используется метод доступа к элементам *LIFO* ( *Last Input – First Output*, "последним пришел – первым вышел"). Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно сначала взять верхнюю. У стека  доступен один элемент (одна позиция). Этот элемент называется *вершиной стека*. Взять элемент можно только из вершины стека, добавить элемент можно только в вершину стека. Поскольку для стека, доступ к элементам осуществляется только через начало последовательности, то эта структура данных допускает эффективную *непрерывную реализацию на базе вектора*.

При *непрерывной реализации ограниченного стека* на базе вектора для представления стека используется одномерный массив (вектор) *Mem*: **array**[1..*n*] **of**α и переменная *Верх*: 1..*n* (рис. 1).



Для пустого стека *Верх* = 0, для целиком заполненного стека *Верх* = *n*. Вершина стека доступна как *Mem* [*Верх*], операция *Pop* реализуется как *Верх*:= *Верх* − 1, а операция *Push* (*p*, *s*) как **begin** *Верх*:= *Верх* + 1; *Mem* [*Верх*]: = *p* **end** при 0 ≤ *Верх*< *n*.

**Задание.**

Вариант 11(а)-векторно: вычислить как целое число значение выражения (без переменных), записанного в постфиксной форме в заданном текстовом файле *postfix*

***Инфиксная***  ***Постфиксная***

*a*−*b* *ab−*

*a*\**b*+*c* *ab*\**c*+

*a*\*(*b*+*c*) *abc*+\*

*a*+*b*^*c*^*d*\**e* *abc*^*d*^*e*\*+

**Реализация**

**class** stack{ **// класс стека**

**private**:

**int** data[SIZE];

**int** topElem; **// переменная, хранящая**

**public**: **// индекс верхнего элемента**

stack();

**bool** isEmpty(); **// основные функции для**

**int** push(**int** elem); **// работы со стеком**

**void** pop();

**int** top();

**int** pop2();

**bool** isOne();

};

1) Класс ***stack*** содержит в себе основные функции для работы со стеком: ***pop(); push(); top()***, а так же вспомогательные функции ***isOne()*** и ***isEmpty()***. Данные функции являются public-функциями, т.к. представляют собой интерфейс для работы со стеком. Private-членами класса ***stack*** являются

• ***topElem*** - целочисленная переменная, хранящая индекс верхнего (доступного) элемента стека.

• ***data[]*** - целочисленный массив, хранящий в себе элементы стека.

2) bool isEmpty( );

Функция *isEmpty* проверяет, является ли стек пустым и возвращает значение *true* в таком случае*.*

3) bool isOne( );

Функция *isOne* проверяет, состоит ли стек из одного элемента и возвращает *true* в этом случае.

4) int push( int elem);

Функция *push* добавляет элемент *elem* в вершину стека, увеличивая значение переменной *topElem* на единицу. Возвращает 0 в случае, если стек переполнен.

5) void pop( );

Функция *pop* извлекает элемент из вершины стека, уменьшая при этом значение *topElem* на единицу.

6) int top( );

Функция *top* возвращает элемент вершины стека, при этом не удаляя его из самого стека.

7) int pop2( );

Функции *pop2()* извлекает из вершины стека элемент и возвращает его значение. Действие это функции основана на выполнении подряд функций *top()* и *pop(),* уже реализованных ранее.

7) int postfixCalc(stack\* obj, ifstream& stream);

Функции *postfixCalc()* реализует обработку выражения в постфиксной форме, записанного в файл. Если при считывании функция обнаруживает число, то она заносит его в стек. Если же встречен один из символов арифметического действия ( +, -, ^, \*), то функция извлекает из стека два верхних элемента, производит над ними соответствующую операцию и заносит результат в стек (при этом для некоторых операций стоит иметь в виду, что они лежат в стеке в неправильном порядке). В случае если в стеке не достаточно элементов для операции (стек пуст или в нем один элемент), функция оповещает об ошибке и завершает работу программы. Пример действия алгоритма можно увидеть на рис. 2 :

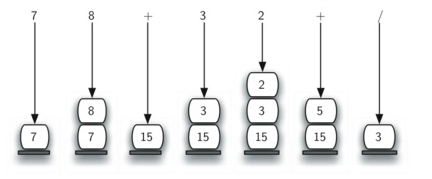


рис. 2

9) Основная функция программы (main) создает объекты класса stack, осуществляет ввод данных из консоли , сопровождая его соответствующем диалогом. Пользователь выбирает один из способов ввода списка (из файла или из консоли). После обработки пользователь может продолжить работу с программой или завершить её.

**Выводы.**

Ознакомился с абстрактной структурой данных, такой как стек, а так же со способами работы с ним. Реализовал программу по вычислению выражения, записанного в постфиксной записи с помощью стека. Ознакомился с одним из способов реализации стека, а именно - векторной реализации ограниченного стека.

Протокол

Таблица 1 - тестирование

|  |  |  |
| --- | --- | --- |
| № теста | Входные данные | Выходные данные |
| 1. | 5 6 + 7 9 \* - | -52 |
| 2. | 5 13 7 - \* | 30 |
| 3. | 115 | 115 |
| 4. | 2 5 \* 3 4 \* + | 22 |
| 5. | 100 0 ^ 89 \* 100 - | -11 |
| 6. | 10 2 3 4 5 6 7 8 9 10 11 12 13 14 15 + \* + + \* + + + + + + \* + + | 12 129 |
| 7. | -100 89 34 + - | -223 |
| 8. | 10 3 ^ 1000 - | 0 |

Таблица 2 - из файла.

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Содержимое файла | Выходные данные | Комментарий |
| 1. | "пустой файл" | "Стек пуст" | Так как файл пуст, то в стек ничего не будет занесено, следовательно при выводе результаты будет выведена строк "стек пуст". |
| 2. | 123 37 2 3 \* \* + | 345 | - |

Действия программа при некорректном вводе (см. рис 3-4):

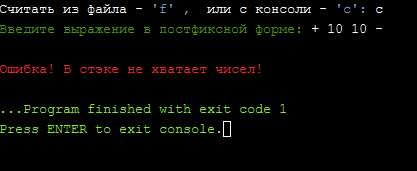


рис.3 (неверный ввод)

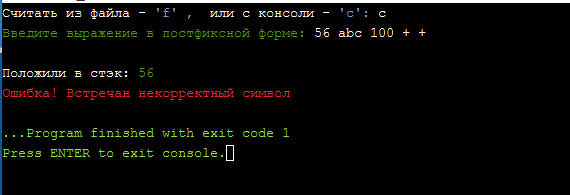


рис. 4 (неверный ввод)

Пример полной работы (см. рис 5):

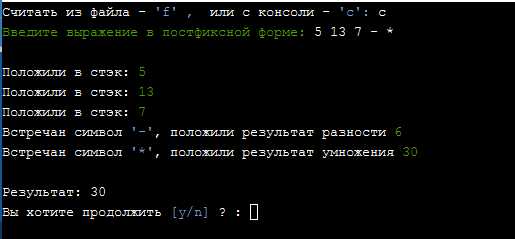


рис. 5

Приложение (код программы).

#include <iostream>

#include <fstream>

#include <cstdio>

#include <cstring>

#include <cmath>

**using** **namespace** std;

#define SIZE 500

**class** stack{

**private**:

**int** array[SIZE];

**int** topElem;

**int** size;

**public**:

stack();

**bool** isEmpty();

**int** push(**int** a);

**void** pop();

**int** top();

**int** pop2();

**bool** isOne();

};

stack::stack(){

topElem = -1;

}

**bool** stack::isEmpty(){

**return** topElem == -1;

}

**int** stack::push(**int** a){

**if** (topElem == SIZE-1){

cout << "Стэк заполнен!";

**return** 0;

}

topElem++;

size++;

array[topElem] = a;

**return** 1;

}

**void** stack::pop(){

**if** (topElem == -1){

cout << "Стэк пуст!"; exit(1);

//return NULL;

}

topElem--;

size--;

}

**int** stack::top(){

**if** (topElem == -1){

cout << "Стэк пуст"; exit(1);

//return;

}

**return** array[topElem];

}

**int** stack::pop2(){

**if** (topElem == -1){

cout << "Стэк пуст"; exit(1);

}

**int** a = top();

pop();

**return** a;

}

**bool** stack::isOne(){

**if** (topElem == 0)

**return** **true**;

**else**

**return** **false**;

}

**int** isDigit(**char**\* str){

**if** (((str[0] == '-' || str[0] == '+') && strlen(str)>1) || isdigit(str[0])){

**for** (**int** i = 1; i < strlen(str) ; i++){

**if** (isdigit(str[i]) == 0) **return** 0;

}

**return** 1;

}

**else**

**return** 0;

}

**int** postfix(stack\* obj, ifstream& stream){

**int** res;

**char** buf[30];

**while**(stream>>buf) {

**if** (isDigit(buf)){

res = atoi(buf);

obj->push(res);

cout << "\nПоложили в стэк: \033[0;32m" << atoi(buf) << "\033[0m";

}

**if** (strlen(buf) == 1 && buf[0] == '+'){

**if** (obj->isOne() || obj->isEmpty()){

cout << "\nОшибка! В стэке не хватает чисел!"; exit(1);

}

cout <<"\nВстречан символ \033[1;34m \'+\'\033[0m, ";

res = obj->pop2() + obj->pop2();

obj->push(res);

cout << "положили результат суммы \033[0;32m"<< obj->top();

}

**if** (strlen(buf) == 1 && buf[0] == '\*'){

**if** (obj->isOne() || obj->isEmpty()){

cout << "\nОшибка! В стэке не хватает чисел!"; exit(1);

}

cout <<"\nВстречан символ \033[1;34m \'\*\'\033[0m, ";

res = obj->pop2() \* obj->pop2();

obj->push(res);

cout << "положили результат умножения \033[0;32m"<< obj->top() << "\033[0m";

}

**if** (strlen(buf) == 1 && buf[0] == '-') {

**if** (obj->isOne() || obj->isEmpty()){

cout << "\nОшибка! В стэке не хватает чисел!"; exit(1);

}

cout <<"\nВстречан символ \033[1;34m\'-\'\033[0m, ";

**int** a = obj->pop2();

res = obj->pop2() - a;

obj->push(res);

cout << "положили результат разности \033[0;32m"<< obj->top() << "\033[0m";

}

**if** (strlen(buf) == 1 && buf[0] == '^') {

**if** (obj->isOne() || obj->isEmpty()){

cout << "\nОшибка! В стэке не хватает чисел!"; exit(1);

}

cout <<"\nВстречан символ \033[1;34m \'^\'\033[0m, ";

**int** a = obj->pop2();

res = pow(obj->pop2(), a);

obj->push(res);

cout << "положили результат возведения в степень \033[0;32m"<< obj->top() << "\033[0m";

}

**if** (strlen(buf) == 1 && buf[0] == '/') {

**if** (obj->isOne() || obj->isEmpty()){

cout << "\nОшибка! В стэке не хватает чисел!"; exit(1);

}

cout <<"\nВ тречан символ \033[1;34m\'/\'\033[0m, ";

**int** a = obj->pop2();

res = obj->pop2()/a;

obj->push(res);

cout << "положили результат возведения в целочисленного деления \033[0;32m"<< obj->top() << "\033[0m";

}

**if** (!isDigit(buf) && strlen(buf) > 1)

cout << "\nОшибка! Встречан некорректный символ";

}

res = obj->pop2();

**return** res;

}

**void** fixed(){

cin.clear();

**while** (cin.get() != '\n');

}

**void** toLower(string &str){

**int** i = 0;

**while** (str[i]){

str[i] = tolower(str[i]);

i++;

}

}

**int** main()

{

string str;

stack\* obj;

**int** res = 0;

**int** a;

obj = **new** stack;

**do**{

**delete** obj;

obj = **new** stack;

**char**\* buf = **new** **char** [256];

**int** res = 0;

cout << "Считать из файла - \033[1;34m\'f\' , \033[0m или с консоли - \033[1;34m\'c\': \033[0m";

cin >> str; toLower(str);

**while** (str!=file && str!=con)

{

cout << "\033[1;31mОшибка ввода, попробуйте снова: \033[0m" ;

cin >> str;

}

**if** (str == file)

{

string fileName;

cout << "\033[0;32m Введите имя файла:\033[0m ";

cin >> fileName;

ifstream in(fileName.c\_str(),ios::in | ios::binary);

**while**(!in)

{

cout <<"\033[1;31mНе удается открыть файл. Введите корректное имя: \033[0m";

cin >> fileName;

in.open(fileName.c\_str(),ios::in | ios::binary);

}

res = postfix(obj, in);

}

**else** {

cout << "\033[0;32m Введите выражение в постфиксной форме:\033[0m ";

fixed();

**while**(cin.peek() != '\n') {

cin >> buf;

**if** (isDigit(buf)){

a = atoi(buf);

obj->push(a);

cout << "\n Положили в стэк: \033[0;32m" << atoi(buf);

}

**if** (strlen(buf) == 1 && buf[0] == '+'){

**if** (obj->isOne() || obj->isEmpty()){

cout << "\n\033[1;31m Ошибка! В стэке не хватает чисел!\033[0m"; exit(1);

}

cout <<"\n Встречан символ \033[1;34m\'+\'\033[0m, ";

res = obj->pop2() + obj->pop2();

obj->push(res);

cout << "положили результат суммы \033[0;32m "<< obj->top();

}

**if** (strlen(buf) == 1 && buf[0] == '\*'){

**if** (obj->isOne() || obj->isEmpty()){

cout << "\n\033[1;31m Ошибка! В стэке не хватает чисел!"; exit(1);

}

cout <<"\n Встречан символ \033[1;34m \'\*\'\033[0m, ";

res = obj->pop2() \* obj->pop2();

obj->push(res);

cout << "положили результат умножения \033[0;32m "<< obj->top() << "\033[0m";

}

**if** (strlen(buf) == 1 && buf[0] == '-') {

**if** (obj->isOne() || obj->isEmpty()){

cout << "\n\033[1;31m Ошибка! В стэке не хватает чисел!"; exit(1);

}

cout <<"\n Встречан символ \033[1;34m \'-\'\033[0m, ";

**int** a = obj->pop2();

res = obj->pop2() - a;

obj->push(res);

cout << "положили результат разности \033[0;32m "<< obj->top() << "\033[0m";

}

**if** (strlen(buf) == 1 && buf[0] == '^') {

**if** (obj->isOne() || obj->isEmpty()){

cout << "\n\033[1;31mОшибка! стэке не хватает чисел!"; exit(1);

}

cout <<"\n встречан символ \033[1;34m\'^\'\033[0m, ";

**int** a = obj->pop2();

res = pow(obj->pop2(), a);

obj->push(res);

cout << "положили результат возведения в степень \033[0;32m "<< obj->top() << "\033[0m";

}

**if** (strlen(buf) == 1 && buf[0] == '/') {

**if** (obj->isOne() || obj->isEmpty()){

cout << "\nОшибка! стэке не хватает чисел!"; exit(1);

}

cout <<"\n'стречан символ \, ";

**int** a = obj->pop2();

res = obj->pop2()/a;

obj->push(res);

cout << "положили результат возведения в целочисленного деления \033[0;32m"<< obj->top() << "\033[0m";

}

**if** (!isDigit(buf) && strlen(buf) > 1)

{

cerr << "\n\033[1;31mОшибка! стречан некорректный символ"; exit(1);

exit(1);

}

}

res = obj->pop2();

} // else

cout <<"\n\nРезультат: " << res;

cout << "\n" << "В хотите продолжить \033[1;34m[y/n]\033[0m ? : ";

cin >> str;

toLower(str);

**while** (str!=stop && str!=again){

cout << "\n\033[1;31mОшибка, введите [y/n] :\033[0m";

cin >> str; toLower(str);

}

cout << "\n"; step = 1;

**delete** [] buf;

}**while**(str != "n");

**return** 0;

}