|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

**Отчет по лабораторной работе**

|  |  |  |  |
| --- | --- | --- | --- |
| **Структуры и алгоритмы обработки данных** | | | |
| *( наименование дисциплины )* | | | |
| Направление подготовки | | | 09.03.01 Информатика и вычислительная техника |
|  | | |  |
| Тема | Линейные списки | | |
| Институт | | информационных технологий (ИТ) | |

Выполнил студент 3 курса

Группа ИВБО-06-17

Фамилия Имя Ушакова

Анастасия

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Оглавление

[Оглавление 2](#_Toc23895347)

[1. Задание 1 4](#_Toc23895348)

[1.1 Вариант задания 4](#_Toc23895349)

[1.2 Абстрактный тип данных (далее АТД) для варианта задания, включая список общих функций из задания. 4](#_Toc23895350)

[1.3 Реализация АТД 4](#_Toc23895352)

[1.4 Таблица тестов 7](#_Toc23895353)

[1.5 Код основной программы с комментариями 7](#_Toc23895354)

[1.6 Скриншоты выполнения тестов 12](#_Toc23895372)

[2. Задание 2 13](#_Toc23895373)

[2.1 Структура узла 13](#_Toc23895374)

[2.2 Структура класса 13](#_Toc23895375)

[2.3 Таблица тестов 14](#_Toc23895376)

[2.4 Код основной программы с комментариями 15](#_Toc23895377)

[2.5 Скриншоты выполнения тестов 23](#_Toc23895378)

[3. Задание 3 23](#_Toc23895379)

[3.1 Структура узла 23](#_Toc23895380)

[3.2 Таблица тестов 23](#_Toc23895381)

[3.3 Код основной программы с комментариями 24](#_Toc23895382)

[3.4 Скриншоты выполнения 24](#_Toc23895383)

[Заключение 25](#_Toc23895384)

# Задание 1

## Вариант задания

Вычислить значение арифметического выражения, введенного в виде строки, в котором последовательность выполнения операций определяется круглыми скобками. Операнды операций – это целые числа.

## Абстрактный тип данных (далее АТД) для варианта задания, включая список общих функций из задания.

Разработать класс **Стек в соответствии** задачей варианта, с реализацией на одномерном динамическом массиве.

Требования

1. Определение класса реализовать в отдельном заголовочном файле, реализация методов в файле cpp.
2. В классе определить: конструктор без параметров и деструктор, аа так же другие методы управления структурой, для решения задачи.
3. Разработать программу, тестирования класса.

## Реализация АТД

* Структура класса

template <class T>

class Stack {

T\* pstack; //массив

int istack; //кол-во элементов в стеке

int size; //максимальная вместимость стека

friend class Opz; //класс преобразования выражения в ОПЗ

//вставка элемента в стек

void push(T x){

if(istack < size) pstack[istack++] = x;

else {

std::cout<<"Переполнение стека!\n";

}

}

//удалить элемент с вершины стека

T pop(){

return (istack > 0)? pstack[--istack]: -1;

}

//вернуть верхний элемент

T top(){

return (istack > 0)? pstack[istack-1]: -1;

}

public:

Stack(): size(50) {

istack = 0;

pstack = new T[size];

}

~Stack() {if(pstack) delete[] pstack;}

};

* Структура узла
* Алгоритмы задач варианта, которые не включены в АТД на псевдокоде и использующие операции из АТД.

Обратная польская запись:

пока (не конец строки){

если(текущий символ является цифрой){

выходная строка += текущий символ;

продолжить;

}

Иначе выходная строка += ‘ ’;

если (тек.симв. это ‘(‘){

вставить его в стек;

}

если (тек.симв. это ‘\*‘,’+’,’/’ или ‘-’){

пока(приоритет тек.операции меньше или равен приоритету операции на вершине стека){

выходная строка += символ из вершина стека;

удалить символ из вершины стека;

}

если (приоритет тек.операции больше приоритета операции на вершине стека){

вставить в стек тек. символ;

}

}

если (тек.симв. это ‘)‘){

пока(на вершине стека не ‘(’){

тек.символ = символ с вершины стека;

удалить символ из вершины стека;

выходная строка += тек.символ;

}

Удалить символ из вершины стека;

}

пока (стек не пуст){

выходная строка += символ с вершины стека;

удалить символ с вершины стека;

}

}

Вычисление значения выражения:

Для(i от 0 до конца строки){

если(символ это ‘ ’) продолжить;

пока(символ это цифра и не конец строки){

число +=символ;

}

если(число существует){

вставить в стек преобразованное в int число;

}

удалить число;

если(символ это ‘ ’) продолжить;

}

взять два числа с конца стека;

если (символ это ‘+’){

сложить два числа;

}

если (символ это ‘-’){

найти разность двух чисел;

}

если (символ это ‘\*’){

умножить два числа;

}

если (символ это ‘/’){

найти частное двух чисел;

}

Вставить результат в конец стека;

}

* Список модулей реализации АТД (или описать где расположена реализация АТД)

Для реализации использовались:

Класс Stack;

## Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| №  теста | Эталон результата | Результат программы | Результат анализа  теста (+ или -) |
|
| 1 |  |  | **+** |

## Код основной программы с комментариями

#include <windows.h>

#include <iostream>

#include "Opz.h"

#include "Stack.h"

#include "List.h"

#include "STLStack.h"

using namespace std;

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int var;

while (var){

system("cls");

//Задание 4

cout<<"Меню:\n1.Стек на основе списка\n2.Стек на основе массива\n3.Стек STL\n0.Выход\n";

cout<<"Выберите опцию:";

cin>>var;

switch(var){

//Задание 2

case 1:{

char buf1[50];

string strin;

cout << "Введите выражение" << endl;

cin >> buf1; //get the input string

if(buf1[0] == '0') break;

strin = buf1;

Ppn ppn;

ppn.convert(strin);

cout << "Обратная польска запись:" << ppn.getOutputStr()<< endl;

cout<<"Результат:"<<ppn.calculate(ppn.getOutputStr())<<endl;

system("pause");

break;

}

case 2:{

//Задание 1

char buf[50];

string str\_in;

cout << "Введите выражение" << endl;

cin >> buf; //get the input string

if(buf[0] == '0') break;

str\_in = buf;

Opz ppn;

ppn.convert(str\_in);

cout << "Обратная польска запись:" << ppn.getOutputStr()<< endl;

cout<<"Результат:"<<ppn.calculate(ppn.getOutputStr())<<endl;

system("pause");

break;

}

case 3:{

//Задание 3

char buf2[50];

string strin\_g;

cout << "Введите выражение" << endl;

cin >> buf2; //get the input string

if(buf2[0] == '0') break;

strin\_g = buf2;

STLStack ppn;

ppn.convert(strin\_g);

cout << "Обратная польска запись:" << ppn.getOutputStr()<< endl;

cout<<"Результат:"<<ppn.calculate(ppn.getOutputStr())<<endl;

system("pause");

break;

}

case 0:{

exit(0);

system("pause");

break;

}

default:{

cout<<"Выберите опцию из списка!\n";

system("pause");

break;

}

}

}

return 0;

}

Файл Stack.h

#ifndef STACK\_H

#define STACK\_H

#include <iostream>

#include "Opz.h"

template <class T>

class Stack {

T\* pstack; //массив

int istack; //кол-во элементов в стеке

int size; //максимальная вместимость стека

friend class Opz; //класс преобразования выражения в ОПЗ

//вставка элемента в стек

void push(T x){

if(istack < size) pstack[istack++] = x;

else {

std::cout<<"Переполнение стека!\n";

}

}

//удалить элемент с вершины стека

T pop(){

return (istack > 0)? pstack[--istack]: -1;

}

//вернуть верхний элемент

T top(){

return (istack > 0)? pstack[istack-1]: -1;

}

public:

Stack(): size(50) {

istack = 0;

pstack = new T[size];

}

~Stack() {if(pstack) delete[] pstack;}

};

#endif // STACK\_H

Файл Opz.h

## #ifndef OPZ\_H

## #define OPZ\_H

## #include <string>

## #include <stdlib.h>

## using namespace std;

## class Opz {

## char current; //текущий символ

## string inputStr, outputStr; //входная и выходная строки

## int indInpStr; //индекс символа входной строки

## char nextChar(); //следующий символ строки

## bool isDigit(char c);

## int priorOp(char c); //приоритет операций

## public:

## void convert(string); //преобразование в ОПЗ

## string getOutputStr(); //выходная строка

## int calculate(string str\_in);

## };

Файл Opz.cpp

#include "Opz.h"

#include "Stack.h"

using namespace std;

//if the character is a digit

bool Opz::isDigit(char c) {

return (c>='0' && c<='9');

}

//the priopity of the operation

int Opz::priorOp(char c) {

switch(c) {

case '(': return 1;

case '+': case '-': return 2;

case '\*': case '/': return 3;

default: return 0;

}

}

char Opz::nextChar() {

if(indInpStr < inputStr.length()) {

return current = inputStr[indInpStr++];

}

else return current = '\0';

}

//полученная строка

string Opz::getOutputStr(){

return outputStr;

}

//преобразование в ОПЗ

void Opz::convert(string str) {

int op = 0, np = 0; //флаги, показывающие наличие операторов и скобок

indInpStr= 0;

Stack<char> opStack; //стек

inputStr = str;

outputStr.erase();

if((!isDigit(inputStr[0])) && inputStr[0]!='(')

cout<<"Ошибка: неправильно записано выражение!\n";

while (nextChar() != '\0') {

if(isDigit(current)) {

outputStr += current;

op = 0;

continue;

}

else outputStr += ' ';

switch (current) {

case '(':

opStack.push(current);

++np;

op = 0;

break;

case '\*': case '/': case '+': case '-':

if(indInpStr == inputStr.length())

cout<<"Ошибка: оператор не может стоять в конце!\n";

if(!op) {

op = 1;

while(priorOp(current) <= priorOp(opStack.top())) {

outputStr += opStack.pop();

}

if(priorOp(current) > priorOp(opStack.top())){

opStack.push(current);

}

break;

}

else cout<<"Ошибка приоритета операции!\n";

case ')':

if(op)

cout<<"Ошибка: неверно поставлена скобка!\n";

else

while((current= opStack.pop())!= '(' && np>0) {

outputStr += current;

}

np--;

break;

default: {

cout<<"Error: неопознанный символ\n";

}

}

}

while(opStack.top() != -1)

outputStr += opStack.pop();

if(np)

cout<<"Error: неравное кол-во скобок!\n";

}

int Opz::calculate(string str) {

Stack<int> valStack; //стек

int num1, num2, result =0;

string num;

for(int i = 0; i<str.length(); ++i) {

if(outputStr[i] == ' ') continue;

while(isDigit(outputStr[i]) && i<str.length()) {

// valStack.push(outputStr[i] - '0');

num += outputStr[i++];

}

if(!num.empty()){

valStack.push(atoi(num.c\_str()));

num.erase();

if(i>=outputStr.length())

break;

if(outputStr[i] == ' ') continue;

}

num2 = valStack.pop();

num1 = valStack.pop();

switch(outputStr[i]) {

case '+': result = num1 + num2; break;

case '-': result = num1 - num2; break;

case '\*': result = num1 \* num2; break;

case '/': result = num1 / num2; break;

default: cout<<"Ошибка !\n";

}

valStack.push(result);

}

return valStack.pop();

}

#endif // OPZ\_H

## 

## Скриншоты выполнения тестов

# Задание 2

## Структура узла

## Структура класса

template<typename T>

class List

{

friend class Ppn;

public:

List();

~List();

void push(T data);

int getSize() {return Size;}

void removeAt(int index);

T top() {return (tail)? tail->data : -1;}

T& operator[](const int index);

void pop();

private:

template<typename U>

class Node{

public:

Node \*next;

U data;

Node(U data=U(), Node \*next = 0){

this->data = data;

this->next = next;

}

};

int Size;

Node<T> \*head;

Node<T> \*tail;

};

## Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| №  теста | Эталон результата | Результат программы | Результат анализа  теста (+ или -) |
|
| 1 |  |  | **+** |

## Код основной программы с комментариями

Файл List.h

#ifndef LIST\_H

#define LIST\_H

#include <iostream>

#include "Ppn.h"

template<typename T>

class List

{

friend class Ppn;

public:

List();

~List();

void push(T data);

int getSize() {return Size;}

void removeAt(int index);

T top() {return (tail)? tail->data : -1;}

T& operator[](const int index);

void pop();

private:

template<typename U>

class Node{

public:

Node \*next;

U data;

Node(U data=U(), Node \*next = 0){

this->data = data;

this->next = next;

}

};

int Size;

Node<T> \*head;

Node<T> \*tail;

};

template<typename T>

List<T>::List()

{

Size = 0;

head = 0;

}

template<typename T>

List<T>::~List()

{

Node<T>\* cur = head;

while(cur){

cur = head->next;

delete head;

head = cur;

}

}

template<typename T>

void List<T>::push(T data){

Node<T> \*cur = new Node<T>(data);

if(head == 0){

head = cur;

}else{

tail->next = cur;

}

tail = cur;

Size++;

}

template<typename T>

T& List<T>::operator[](const int index){

int cnt = 0;

Node<T> \*cur = this->head;

while (cur){

if(cnt == index){

return cur->data;

}

cur = cur->next;

cnt++;

}

}

template<typename T>

void List<T>::removeAt(int index){

if (index == 0){

Node<T> \*temp = head;

head = head->next;

tail=head;

delete temp;

Size--;

}

else

{

Node<T> \*previous = this->head;

for(int i = 0; i<index - 1; i++){

previous = previous->next;

}

Node<T> \*toDel = previous->next;

tail = previous;

previous->next = toDel->next;

delete toDel;

toDel=nullptr;

Size--;

}

}

template<typename T>

void List<T>::pop(){

removeAt(Size-1);

}

#endif // LIST\_H

Файл Ppn.h

#ifndef PPN\_H

#define PPN\_H

#include <string>

#include <stdlib.h>

using namespace std;

class Ppn {

char current; //текущий символ

string inputStr, outputStr; //входная и выходная строки

int indInpStr; //индекс символа входной строки

char nextChar(); //следующий символ строки

bool isDigit(char c);

int priorOp(char c); //приоритет операций

public:

void convert(string); //преобразование в ОПЗ

string getOutputStr(); //выходная строка

int calculate(string str\_in);

};

#endif // PPN\_H

Файл Ppn.cpp

#include "Ppn.h"

#include "List.h"

using namespace std;

//if the character is a digit

bool Ppn::isDigit(char c) {

return (c>='0' && c<='9');

}

//the priopity of the operation

int Ppn::priorOp(char c) {

switch(c) {

case '(': return 1;

case '+': case '-': return 2;

case '\*': case '/': return 3;

default: return 0;

}

}

char Ppn::nextChar() {

if(indInpStr < inputStr.length()) {

return current = inputStr[indInpStr++];

}

else return current = '\0';

}

//полученная строка

string Ppn::getOutputStr(){

return outputStr;

}

//преобразование в ОПЗ

void Ppn::convert(string str) {

int op = 0, np = 0; //флаги, показывающие наличие операторов и скобок

indInpStr= 0;

List <char> opStack; //стек

inputStr = str;

outputStr.erase();

if((!isDigit(inputStr[0])) && inputStr[0]!='(')

cout<<"Ошибка: неправильно записано выражение!\n";

while (nextChar() != '\0') {

if(isDigit(current)) {

outputStr += current;

op = 0;

continue;

}

else outputStr += ' ';

switch (current) {

case '(':

opStack.push(current);

++np;

op = 0;

break;

case '\*': case '/': case '+': case '-':

if(indInpStr == inputStr.length())

cout<<"Ошибка: оператор не может стоять в конце!\n";

if(!op) {

op = 1;

while(priorOp(current) <= priorOp(opStack.top())) {

outputStr += opStack.top();

if(opStack.top() != -1)

opStack.pop();

}

if(priorOp(current) > priorOp(opStack.top())){

opStack.push(current);

}

break;

}

else cout<<"Ошибка приоритета операции!\n";

case ')':

if(op)

cout<<"Ошибка: неверно поставлена скобка!\n";

else

while(opStack.top() != '(' && np>0) {

current = opStack.top();

opStack.pop();

outputStr += current;

}

np--;

opStack.pop();

break;

default: {

cout<<"Error: неопознанный символ\n";

}

}

}

while(opStack.top() != -1){

outputStr += opStack.top();

opStack.pop();

}

if(np)

cout<<"Error: неравное кол-во скобок!\n";

}

int Ppn::calculate(string str) {

List<int> valStack; //стек

int num1, num2, result =0;

string num;

for(int i = 0; i<str.length(); ++i) {

if(outputStr[i] == ' ') continue;

while(isDigit(outputStr[i]) && i<str.length()) {

num += outputStr[i++];

}

if(!num.empty()){

valStack.push(atoi(num.c\_str()));

num.erase();

if(i>=outputStr.length())

break;

if(outputStr[i] == ' ') continue;

}

num2 = valStack.top();

valStack.pop();

num1 = valStack.top();

valStack.pop();

switch(outputStr[i]) {

case '+': result = num1 + num2; break;

case '-': result = num1 - num2; break;

case '\*': result = num1 \* num2; break;

case '/': result = num1 / num2; break;

default: cout<<"Ошибка !\n";

}

valStack.push(result);

}

return valStack.top();

}

## Скриншоты выполнения тестов

# Задание 3

## Структура узла

template<

class T,

class Container = std::deque<T>

> class stack;

## Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| №  теста | Эталон результата | Результат программы | Результат анализа  теста (+ или -) |
|
| 1 |  |  | **+** |

## Код основной программы с комментариями

Файл STLStack.h

#ifndef STLSTACK\_H

#define STLSTACK\_H

#include <string>

#include <stdlib.h>

using namespace std;

class STLStack {

char current; //текущий символ

string inputStr, outputStr; //входная и выходная строки

int indInpStr; //индекс символа входной строки

char nextChar(); //следующий символ строки

bool isDigit(char c);

int priorOp(char c); //приоритет операций

public:

void convert(string); //преобразование в ОПЗ

string getOutputStr(); //выходная строка

int calculate(string str\_in);

};

#endif // STLSTACK\_H

## Скриншоты выполнения

# Заключение

В данной лабораторной работе мы научились создавать стек посредством реализации на элементарных структурах и классах с использованием ООП на языке C++.

Для выполнения активно использовались указатели на объекты линейного списка с помощью которых, реализовывалась связь между элементами списка.

Полученные навыки и знания дают возможность выполнения последующих лабораторных работ и заданий, а так же создание собственных стеков.