|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | |  | | | | | |
|  | | |
|  | Институт информационных технологий (ИТ) |
|  | Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ) |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2** | |
| **по дисциплине** | |
| **«Структуры и алгоритмы обработки данных»**  **По теме: «Структура данных - стек»**  **Вариант 2** | |
|  | |
| Выполнил студент группы ИВБО-06-17 | Бикеев А. И. |
| Принял преподаватель | Скворцова Л.А. |

Москва 2019

**Оглавление**

[1. Задание 1. Разработать класс Стек в соответствии задачей варианта, с реализацией на одномерном динамическом массиве. 3](#_Toc26217752)

[1.1. Вариант задания 3](#_Toc26217753)

[1.2. Абстрактный тип данных (далее АТД) для варианта задания, включая список общих функций из задания. 3](#_Toc26217754)

[1.3. Реализация АТД 4](#_Toc26217755)

[Список модулей реализации АТД (или описать где расположена 4](#_Toc26217756)

[реализация АТД) 4](#_Toc26217757)

[1.4. Таблица тестов 4](#_Toc26217758)

[1.5. Текст исходного кода (листинг) программы 5](#_Toc26217759)

[2. Задание 2. Разработать класс Стек в соответствии задачей варианта, с реализацией на однонаправленном списке 8](#_Toc26217760)

[2.1. Вариант задания 8](#_Toc26217761)

[2.2. Реализация АТД 8](#_Toc26217762)

[Список модулей реализации АТД (или описать где расположена 9](#_Toc26217763)

[реализация АТД) 9](#_Toc26217764)

[2.3. Таблица тестов 9](#_Toc26217765)

[2.4. Текст исходного кода (листинг) программы 9](#_Toc26217766)

[3. Дополнительные задания 12](#_Toc26217767)

[3.1. Вариант задания 12](#_Toc26217768)

[3.2. Вариант задания 12](#_Toc26217769)

[4. Общие контрольные прогоны программы 12](#_Toc26217770)

[Выводы 13](#_Toc26217771)

1. Задание 1. Разработать класс Стек в соответствии задачей варианта, с реализацией на одномерном динамическом массиве.

1.1. Вариант задания

Вычислить значение арифметического бесскобочного выражения, введенного в виде строки. Операнды операций – это целые числа. Перед вычислением значения выражения следует проверить его на корректность записи операций и операндов.

Требования:

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

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

Абстрактный тип данных StackDynamic

Данные:

size– размер стека, количество операций в мат. выражении;

mass – стек.

Операции:

// Инициализирует стек

void init(string exp);

// Добавляет в конец стека новый элемент

void push(Expression elem);

// Удаляет последний элемент из стека

void pop();

// Возвращает последний элемент в стеке

Expression top();

// Возвращает значение математического выражения

double getResultExpression();

## 1.3. Реализация АТД

АТД реализуется на одномерном динамическом массиве.

class StackDynamic

{

private:

Expression\* mass;

int size;

public:

StackDynamic();

~StackDynamic();

/// <summary>

/// Инициализирует стек

/// </summary>

/// <param name="exp">Математическое выражение</param>

void init(string exp);

/// <summary>

/// Добавляет в конец стека новый элемент

/// </summary>

/// <param name="elem">Новое выражение</param>

void push(Expression elem);

/// <summary>

/// Удаляет последний элемент из стека

/// </summary>

void pop();

/// <summary>

/// Возвращает последний элемент в стеке

/// </summary>

Expression top();

/// <summary>

/// Возвращает значение математического выражения

/// </summary>

/// <returns></returns>

double getResultExpression();

};

Список модулей реализации АТД (или описать где расположена

реализация АТД)

Реализация АТД состоит из двух файлов StackDynamic.h и StackDynamic.cpp.

## 1.4. Таблица тестов

|  |  |  |
| --- | --- | --- |
| Номер теста | Исходные данные | Эталон результата |
| 1 | 11+2-2+2-2\*3+12/4\*3/9 | 8 |
| 2 | 11\*2-2+6\*8/12 | 24 |
| 3 | 11+2-2\*3+4/5 | 7.8 |

## 1.5. Текст исходного кода (листинг) программы

#include "StackDynamic.h"

StackDynamic::StackDynamic()

{

    this->mass = nullptr;

    this->size = 0;

}

StackDynamic::~StackDynamic()

{

}

void StackDynamic::init(string exp)

{

    this->mass = new Expression[1];

    this->size = 1;

    while(!this->mass) this->mass = new Expression[1];

    if (this->mass) {

        Expression obj(exp);

        this->mass[0] = obj;

        if (obj.isPriorityOperation() && isExpression(obj.getLastArg()))

        {

            Expression expNew(obj.getLastArg());

            Expression exp = Expression(

                obj.getFirstArg() +

                obj.getOperation() +

                Expression(obj.getLastArg()).getFirstArg()

            );

            expNew.setFirstArg(to\_string(exp.getResult()));

            this->mass[0] = obj = expNew;

        }

        //Если второй аргумент является мат. выражением - добавляем в стек

        while (isExpression(obj.getLastArg()))

        {

            Expression secondExp(obj.getLastArg());

            if (secondExp.isPriorityOperation() && isExpression(secondExp.getLastArg()))

            {

                Expression expNew(secondExp.getLastArg());

                Expression exp = Expression(

                    secondExp.getFirstArg() +

                    secondExp.getOperation() +

                    Expression(secondExp.getLastArg()).getFirstArg()

                );

                expNew.setFirstArg(to\_string(exp.getResult()));

                obj = secondExp = expNew;

            }

            this->push(secondExp);

            obj = secondExp;

        }

    }

    else {

        throw  exception("Не удалось выделить память \n");

    }

}

void StackDynamic::push(Expression elem)

{

    this->mass = updateSize(this->mass, this->size, this->size + 1);

    this->size++;

    if (this->mass) {

        this->mass[size - 1] = elem;

    }

    else {

        throw  exception("Не удалось выделить память \n");

    }

}

void StackDynamic::pop()

{

    this->mass = updateSize(this->mass, this->size, this->size - 1);

    this->size--;

}

Expression StackDynamic::top()

{

    if (this->size != 0)

        return this->mass[this->size - 1];

    else

        return Expression();

}

double StackDynamic::getResultExpression()

{

    Expression hellper;

    //Пробегаемся с конца стека, удаляя посчитанные значения

    while (this->size != 1)

    {

        int indexLast= this->size - 1;//индекс последнего выражения

        Expression \*last = &this->mass[indexLast];

        Expression \*preLast = &this->mass[indexLast - 1];

        Expression\* prePreLast = &this->mass[indexLast - 2];

        //Если знак операции последнего элемента стека - имеет высокий приоритет

        if (last->isPriorityOperation() || (!preLast->isPriorityOperation() && preLast->getOperation() != '-'))

        {

            preLast->setLastArg(to\_string(last->getResult()));

        }

        else if (preLast->getOperation() == '-') {

            //Если оба числа отрицательны

            if (last->getOperation() == '-')

            {

                last->setOperation('+');

                preLast->setLastArg(to\_string(last->getResult()));

            }

            else {

                //changeNodes(\*last, \*preLast);

                last->setOperation('-');

                //Меняем местами аргументы в связи обнаруженным минусом

                changeArgs(\*last);

                double res = last->getResult();

                if (res < 0)

                {

                    preLast->setOperation('-');

                }

                else {

                    preLast->setOperation('+');

                }

                preLast->setLastArg(to\_string(abs(res)));

            }

        }

        else {

            changeNodes(\*last, \*preLast);

            changeArgs(\*last);

            if (prePreLast->getOperation() == '-')

            {

                preLast->setOperation('-');

                prePreLast->setOperation('+');

                preLast->setLastArg(to\_string(last->getResult()));

            }

            else {

                preLast->setLastArg(to\_string(last->getResult()));

                if (preLast->getOperation() == '-') {

                    changeArgs(\*preLast);

                }

            }

        }

        this->pop();

    }

    return this->mass[0].getResult();

}

2. Задание 2. Разработать класс Стек в соответствии задачей варианта, с реализацией на однонаправленном списке

2.1. Вариант задания

Требования

* 1. Структуру узла определить отдельным классом с набором необходимых методов для инициализации переменных.
  2. Определить класс Стек со структурой узла, реализованного согласно п.1. В классе определить: конструктор без параметров и деструктор для удаления списка, а также другие методы управления структурой, для решения задачи.
  3. Для тестирования созданного класса используйте программу, созданную в задании 1.

## 2.2. Реализация АТД

АТД реализуется на однонаправленном списке.

class LinkedStack

{

private:

LinkedStack\* next;

Expression value;

/// <summary>

/// Возвращает предпоследний элемент в стеке

/// </summary>

LinkedStack\* preTop();

/// <summary>

/// Возвращает предпоследний элемент в стеке

/// </summary>

LinkedStack\* prePreTop();

public:

LinkedStack(Expression elem);

LinkedStack(const Expression& elem);

~LinkedStack();

/// <summary>

/// Добавляет в конец стека новый элемент

/// </summary>

/// <param name="elem">Новое выражение</param>

void push(Expression elem);

/// <summary>

/// Удаляет последний элемент из стека

/// </summary>

void pop();

/// <summary>

/// Возвращает последний элемент в стеке

/// </summary>

LinkedStack\* top();

/// <summary>

/// Возвращает значение математического выражения

/// </summary>

/// <returns></returns>

double getResultExpression(LinkedStack\* head);

};

Список модулей реализации АТД (или описать где расположена

реализация АТД)

Реализация АТД расположена в трех файлах: LinkedStack.cpp и LinkedStack.h

## 2.3. Таблица тестов

Тесты аналогичны тестам из задания 1.

## 2.4. Текст исходного кода (листинг) программы

#include "LinkedStack.h"

LinkedStack\* LinkedStack::preTop()

{

    LinkedStack\* ptr = this;

    while (ptr->next != nullptr && ptr->next->next != nullptr)

    {

        ptr = ptr->next;

    }

    return ptr;

}

LinkedStack\* LinkedStack::prePreTop()

{

    LinkedStack\* ptr = this;

    while (ptr->next != nullptr && ptr->next->next != nullptr && ptr->next->next->next != nullptr)

    {

        ptr = ptr->next;

    }

    return ptr;

}

LinkedStack::LinkedStack(Expression elem)

{

    this->next = nullptr;

    this->value = elem;

    if (elem.isPriorityOperation() && isExpression(elem.getLastArg()))

    {

        Expression expNew(elem.getLastArg());

        Expression exp = Expression(

            elem.getFirstArg() +

            elem.getOperation() +

            Expression(elem.getLastArg()).getFirstArg()

        );

        expNew.setFirstArg(to\_string(exp.getResult()));

        this->value = elem = expNew;

    }

    if (elem.getOperation() != ' ') {

        //Если второй аргумент является мат. выражением - добавляем в стек

        if (isExpression(elem.getLastArg()))

        {

            elem = Expression(elem.getLastArg());

            this->push(elem);

        }

    }

    else {

        throw  exception("Не удалось выделить память \n");

    }

}

LinkedStack::~LinkedStack()

{

    if (this->next != nullptr)

        free(this->next);

}

void LinkedStack::push(Expression elem)

{

    LinkedStack\* stack = this->top();

    stack->next = new LinkedStack(elem);

}

void LinkedStack::pop()

{

    LinkedStack\* preLast = this->preTop();

    free(preLast->next);

    preLast->next = nullptr;

}

LinkedStack\* LinkedStack::top()

{

    LinkedStack\* ptr = this;

    while (ptr->next != nullptr)

    {

        ptr = ptr->next;

    }

    return ptr;

}

double LinkedStack::getResultExpression(LinkedStack\* head)

{

    Expression hellper;

    //Пробегаемся с конца стека, удаляя посчитанные значения

    while (this->top() != head)

    {

        Expression\* last = &this->top()->value;

        Expression\* preLast = &this->preTop()->value;

        Expression\* prePreLast = &this->prePreTop()->value;

        //Если знак операции последнего элемента стека - имеет высокий приоритет

        if (last->isPriorityOperation() || (!preLast->isPriorityOperation() && preLast->getOperation() != '-'))

        {

            preLast->setLastArg(to\_string(last->getResult()));

        }

        else if (preLast->getOperation() == '-') {

            //Если оба числа отрицательны

            if (last->getOperation() == '-')

            {

                last->setOperation('+');

                preLast->setLastArg(to\_string(last->getResult()));

            }

            else {

                //changeNodes(\*last, \*preLast);

                last->setOperation('-');

                //Меняем местами аргументы в связи обнаруженным минусом

                changeArgs(\*last);

                double res = last->getResult();

                if (res < 0)

                {

                    preLast->setOperation('-');

                }

                else {

                    preLast->setOperation('+');

                }

                preLast->setLastArg(to\_string(abs(res)));

            }

        }

        else {//Если операция / или \*

            changeNodes(\*last, \*preLast);

            changeArgs(\*last);

            if (prePreLast != preLast && prePreLast->getOperation() == '-')

            {

                preLast->setOperation('-');

                prePreLast->setOperation('+');

                preLast->setLastArg(to\_string(last->getResult()));

            }

            else {

                preLast->setLastArg(to\_string(last->getResult()));

                if (preLast->getOperation() == '-') {

                    changeArgs(\*preLast);

                }

            }

        }

        this->pop();

    }

    return this->value.getResult();

}

3. Дополнительные задания

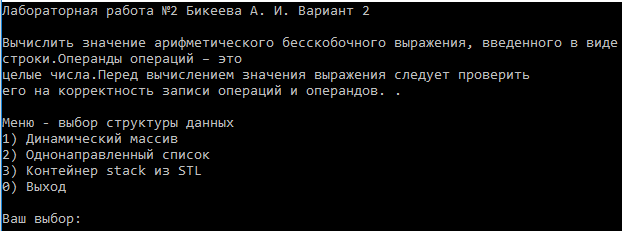
3.1. Вариант задания

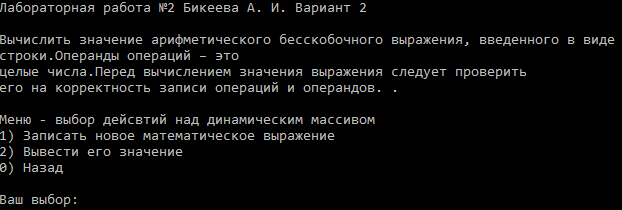
Разработайте программу тестирования созданных классов, управляемую консольным меню.

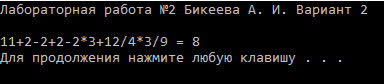
3.2. Вариант задания

Реализуйте задачу варианта, используя контейнер stack из STL. Добавьте тестирование в созданную программу.

4. Общие контрольные прогоны программы







Выводы

В результате выполнения лабораторной работы были получены навыки по созданию структуры данных стек и ее применению.

Стек был разработан на одномерном динамическом массиве, однонаправленном списке, а также с помощью контейнера stack из STL. Была разработана программа тестирования для проверки правильности вывода результата.