МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Линейные структуры данных: стек, очередь

Студентка гр. 7383 _____ Иолшина В.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2018

Содержание

Цель работы.

Познакомиться с линейными структурами данных и их реализацией на языке программирования C++.

Формулировка задачи: В заданном текстовом файле F записана формула вида

< формула > ::= < цифра > | M (< формула >, < формула >) |

m (< формула >, < формула >)

< цифра > ::= 0 | 1 | … | 9

где M обозначает функцию max, а m – функцию min. Вычислить (как целое число) значение данной формулы. Например, M (5, m(6, 8)) = 6.

Реализация задачи.

В данной работе стек реализован с помощью класса. Описание класса представлено в файле Stack.h в приложении А. Класс Stack содержит следующие функции:

- base top() возвращает значение головного элемента стека
- void pop() осуществляет удаление элемента из стека, не возвращая значение этого элемента
- base pop2() осуществляет удаление элемента из стека, возвращая значение этого элемента
- void push(const base &x) осуществляет добавление нового элемента в стек
- bool isNull() проверяет, является ли стек пустым
- bool destroy() освобождает динамическую память, выделенную под стек

В функции main выводится приглашение выбрать способ ввода входных данных или закончить работу. В случае выбора ввода данных из файла, программа считывает строку из файла text.txt и записывает её в поток ввода, после этого закрывает файл. В случае выбора ввода информации с консоли, функция main считывает строку и записывает её в этот же поток ввода. Если файл пустой, то main выводит ошибку и начинает выполнение программы

заново. Если все было выбрано верно, происходит вызов функции read_expr.

Функция read_expr работает как синтаксический анализатор, который во время проверки, является ли введенная формула корректной, при встрече закрывающей скобки вызывает функцию calc. До встречи закрывающей скобки все считанные цифры и эквивалентные операциям min и max значения помещаются в стек operation. Функция calc достает из стека последние три значения и производит вычисления по формуле, после чего возвращаясь в функцию read_expr. Далее функция read_expr либо вызывает себя рекурсивно, либо завершает работу, записывая результат выражения в стек.

Тестирование.

Процесс тестирования.

Программа собрана в операционной системе Ubuntu 16.04.2 LTS", с использованием компилятора g++ (Ubuntu 5.4.0-6ubuntu1~16.04.5). В других ОС и компиляторах тестирование не проводилось.

Результаты тестирования

Тестовые случаи представлены в Приложении А.

Во время тестирования была обнаружена ошибка: программа выводила неправильный результат, если в формуле сравнивались одинаковые цифры. Чтобы устранить ошибку, в функцию calc была добавлена проверка на совпадение сравниваемых цифр.

Выводы.

В ходе выполнения лабораторной работы были получены навыки работы с такой линейной структурой данных, как стек: реализация стека при помощи линейного однонаправленного списка и написание функций для работы со стеком на языке C++.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ

Stack.h :

```
#pragma once
namespace stack
{
	typedef int base;
	class Stack
	{
		private:
		struct node;
		node *topOfStack;
		public:
		Stack()
		{
			topOfStack = NULL;
		};
		base top();
		void pop();
		base pop2();
		void push(const base &x);
		bool isNull();
		void destroy();
	};
}
```

Stack.cpp :

```
#include <fstream>
#include <iostream>
#include <cstdlib>
#include "stack.h"

using namespace std;
namespace stack
{
	struct Stack::node
	{
		base *hd;
		node *tl;
```

```
		node() //constructor
		{
			hd = NULL;
			tl = NULL;
		}
	};

base Stack::top()
{// PreCondition: not null
	if (topOfStack == NULL)
	{
		cerr << "Error: top(null) \n";
		exit(1);
	}
	else
		return *topOfStack->hd;
}

void Stack::pop()
{// PreCondition: not null
	if (topOfStack == NULL)
	{
		cerr << "Error: pop(null) \n";
		exit(1);
	}
	else
	{
		node *oldTop = topOfStack;
		topOfStack = topOfStack->tl;
		delete oldTop->hd;
		delete oldTop;
	}
}

base Stack::pop2()
{// PreCondition: not null
	if (topOfStack == NULL)
	{
		cerr << "Error: pop2(null) \n";
```

```
                exit(1);
        }
        else
        {
                node *oldTop = topOfStack;
                base r = *topOfStack->hd;
                topOfStack = topOfStack->tl;
                delete oldTop->hd;
                delete oldTop;
                return r;
        }
}

void Stack::push(const base &x)
{
        node *p;
        p = topOfStack;
        topOfStack = new node;
        if (topOfStack != NULL)
        {
                topOfStack->hd = new base;
                *topOfStack->hd = x;
// cout << "push -> " << x << endl; // Demo
                topOfStack->tl = p;
        }
        else
        {
                cerr << "Memory not enough\n";
                exit(1);
        }
}

bool Stack::isNull()
{
        return (topOfStack == NULL);
}

void Stack::destroy()
{
```

```
        while (topOfStack != NULL)
        {
                pop();
        }
}

} // end of namespace stack
```

Main.cpp :

```
#include <iostream>
#include <fstream>
#include "stack.h"
#include <sstream>
#include <ctype.h>

using namespace std;
using namespace stack;
void calc(stringstream &xstream, Stack *op);

bool read_expr(stringstream &xstream, Stack *op)
{
 bool res = false;
 char c;

 do
 xstream >> c;
 while (c == ' ');

 if (isdigit(c))
 { // если является цифрой
  op->push(c - '0'); //записываем в стек digit
  res = true;
 }
 else if ((c == 'M') || (c == 'm'))
 { //является обозначением операции min или max
  if (c == 'M') //является max
   op->push(-1); //записываем в стек operation - 1
  else if (c == 'm') //является min
   op->push(-2); //записываем в стек operation - 2
```

```
do xstream >> c; while (c == ' ');

if (c != '(')
{//если очередной символ НЕ окрывающая скобка - выход
  cout << "\n! - Error1\n";
  exit(1);
}
res = read_expr(xstream, op); //рекурсивный вызов
if (!res)
{//если предыдущее выражение НЕ формула - выход
  cout << "\n! - Error2\n";
  exit(1);
}

do xstream >> c; while (c == ' ');

if (c != ',')
{//если очередной символ НЕ запятая - выход
  cout << "\n! - Error3\n";
  exit(1);
}

res = read_expr(xstream, op);//рекурсивный вызов
if (!res)
{//предыдущее выражение НЕ формула - выход
  cout << "\n! - Error4\n";
  exit(1);
}

do xstream >> c; while (c == ' ');

if (c == ')' && !(op->isNull())) //если очередной символ - закрывающая скобка и стек не пуст, вычисление
{
  res = true;
  calc(xstream, op); //вычисление
}
if (c != ')')
{  //если очередной символ НЕ закрывающая скобка - выход
  cout << "! - Error5\n";
```

```
      exit(1);
    }
  }
  else
  {
    cout << "\n! - Error6\n";
    exit(1);
  }

return res;
}

void calc(stringstream &xstream, Stack *op)
{
  base a = op->pop2();
  base b = op->pop2();
  int fun = op->pop2();
  cout << fun << endl;
  base res;
  if (a == b)
  {
    cout << "Цифры равны\n";
    res = a;
  }
  else
  {
    if (fun == -1)//max
    {
      (a > b) ? res = a : res = b;
      cout << "M( " << a << " , " << b << " ) = " << res << endl;
    }
    else if (fun == -2)//min
    {
      (a < b) ? res = a : res = b;
      cout << "m( " << a << " , " << b << " ) = " << res << endl;
    }
  }
  op->push(res);
  cout << endl;
```

```
}

int main()
{
  stringstream xstream;
  bool b = 1;
  Stack operation; //стек для операций M(max) или m(min) из формулы
  //Stack digit; //стек для цифр из формулы
  char x;
  char str[100];
  int c = 0;
  while(c != 3)
  {
    cout << "Введите 1, если хотите ввести выражение с клавиатуры.\n"
          "Введите 2, если еспользовать выражение из файла test.txt.\n"
          "Введите 3, если хотите закончить работу." << endl;
    cin >> c;
    switch(c)
    {
      case 1:
        {
          cout << "Введите выражение: \n";
          cin.get();
          cin.getline(str, 1000);
          xstream << str;
          read_expr(xstream, &operation);
          break;
        }
      case 2:
        {
          ifstream outfile;
          outfile.open("test.txt");
          if (!outfile)
          {
            cout << "Входной файл не открыт!\n";
            b = 0;
            break;
          }
          outfile.read(str, 1000);
```

```
        outfile.close();
        xstream << str;
        read_expr (xstream, &operation);
        break;
       }
     case 3:
       {
        b=0;
        break;
       }
     default:
       {
        cout << "Введите верное число\n";
        break;
       }
   }
   if(b)
   {
    base result = operation.top();
    cout << "Результат работы программы: \n";
    cout << result;
    cout << endl;
   }
}
operation.destroy();
return 0;
}
```

ПРИЛОЖЕНИЕ Б.

ТЕСТОВЫЕ СЛУЧАИ

Ввод	Вывод	Верно
1 M(7, 9)	Результат работы программы 9	Да
1 m (7, 0)	Результат работы программы 0	Да
M(6, m(8, 9	Error	Да
(8, 4)	Error	Да
1	Результат работы программы 1	Да
M(6, M(4, 2))	Результат работы программы 6	Да
m(M(3, 5), 8)	Результат работы программы 5	Да
M(m (m(0, 5), M(4, 9)), m(6, 3)))	Результат работы программы 3	Да