|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | | 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 №3** | |
| **по дисциплине** | |
| **«Структуры и алгоритмы обработки данных»**  **По теме: «Рекурсивные функции»**  **Вариант 7(23)** | |
|  | |
| Выполнил студент группы ИВБО-06-17 | Ушакова А.С. |
| Принял преподаватель | Скворцова Л.А. |

Москва 2019

**Оглавление**

[1. Задание 1. Разработать функцию на рекурсии. 3](#_Toc24544460)

[1.1. Вариант задания 3](#_Toc24544461)

[1.2. Рекурентное соотношение 3](#_Toc24544462)

[1.3. Реализация рекурсивной функции 3](#_Toc24544463)

[1.4. Таблица тестов 3](#_Toc24544465)

[1.5. Текст исходного кода (листинг) программы 4](#_Toc24544466)

[2. Задание 2. Разработать функцию на рекурсии, используя структуру списка. 4](#_Toc24544468)

[2.1. Вариант задания 4](#_Toc24544469)

[2.2. Рекурентное соотношение 4](#_Toc24544471)

[2.3. Реализация рекурсивной функции 4](#_Toc24544472)

[2.4. Таблица тестов 4](#_Toc24544473)

[2.5. Текст исходного кода (листинг) программы 5](#_Toc24544474)

[3. Задание 3. Разработать программу тестирования рекурсивных функций. 5](#_Toc24544476)

[3.1. Вариант задания 5](#_Toc24544477)

[3.2. Контрольные прогоны программы 6](#_Toc24544479)

[Вывод 7](#_Toc24544482)

[Приложение 1 7](#_Toc24544484)

1. Задание 1. Разработать функцию на рекурсии.
   1. Вариант задания

Найти максимальный элемент в массиве из n элементов.

Требования:

* 1. Описать рекурсивную зависимость в виде табличной функции
  2. Определить глубину рекурсии.
  3. Рекуррентное соотношение

, где arr[] – массив, sz – размер массива;

* 1. Реализация рекурсивной функции

double maxEl(double arr[], int sz) {

cnt++;

if(!sz) return arr[0];

else{

return maxEl(arr+(arr[0]<arr[sz]),sz - 1);

}

}

* 1. Таблица тестов

|  |  |  |
| --- | --- | --- |
| **Номер теста** | **Входные данные** | **Эталон результата, глубина рекурсии** |
| **1** | sz: 2  arr = {4,-9} | 4  Глубина рекурсии: 2 |
| **2** | sz: 2  arr = {-100,-9} | -9  Глубина рекурсии: 2 |
| **3** | sz: 4  arr = {0,-87,2,1} | 2  Глубина рекурсии: 4 |

* 1. Текст исходного кода (листинг) программы

Листинг программы представлен в приложении 1.

1. Задание 2. Разработать функцию на рекурсии, используя структуру списка.
   1. Вариант задания

Создание очереди на однонаправленном списке.

* 1. Рекуррентное соотношение

, где n – данные с клавиатуры.

Реализация рекурсивной функции

Node Queue::createQ(){

Node top;

int data;

std::cout<<"Введите элементы очереди или 0, если хотите закончить ввод.\n";

std::cin>>data;

if (!data) return nullptr;

top= new tNode;

top->data=data;

if(first){

last = top;

}else first = last = top;

last-> next =createQ();

return top;

}

* 1. Таблица тестов

|  |  |  |
| --- | --- | --- |
| **Номер теста** | **Входные данные** | **Эталон результата** |
| **1** | 1 2 5 6 0 | 1 2 5 6 |
| **2** | 100 6 5 0 | 100 6 5 |
| **3** | 10 1 17 0 | 10 1 17 0 |

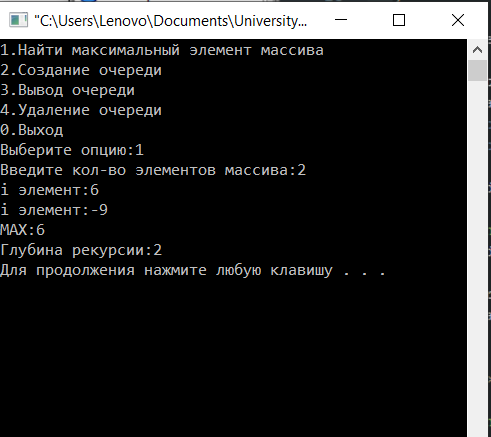
* 1. Текст исходного кода (листинг) программы

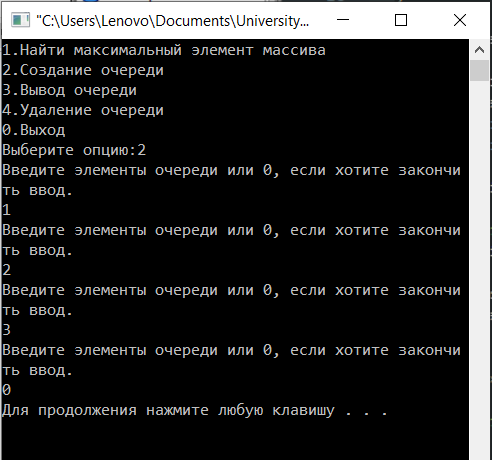
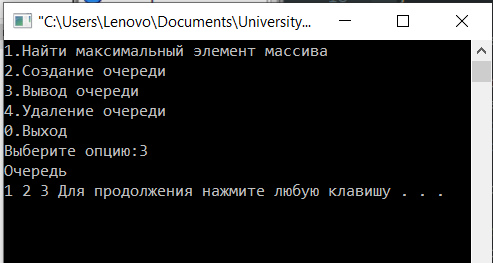
Листинг программы представлен в приложении 1.

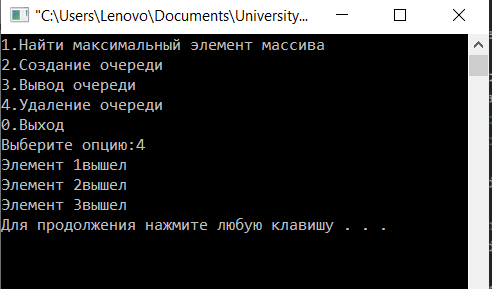
1. Задание 3. Разработать программу тестирования рекурсивных функций.
   1. Вариант задания

Разработать программу тестирования разработанных рекурсивных функций, которая управляется консольным меню.

* 1. Контрольные прогоны программы





Вывод

В результате выполнения лабораторной работы были получены навыки по созданию и работе с рекурсивными функциями.

Приложение 1

Queue.h

#ifndef QUEUE\_H

#define QUEUE\_H

#include <iostream>

#include <stdlib.h>

struct tNode{

int data;

tNode\* next;

};

typedef tNode \*Node;

class Queue

{

tNode \*last;

tNode \*first;

public:

Queue():first(nullptr),last(nullptr){};

~Queue();

Node getLast();

Node getFirst();

Node createQ(); //создание очереди

void outQ(Node a); //вывод очереди

void erase(); //удаление элемента из очереди

};

#endif // QUEUE\_H

Queue.cpp

#include "Queue.h"

Queue::~Queue()

{

erase();

}

Node Queue::getLast(){

return this->last;

}

Node Queue::getFirst(){

return this->first;

}

Node Queue::createQ(){

Node top;

int data;

std::cout<<"Введите элементы очереди или 0, если хотите закончить ввод.\n";

std::cin>>data;

if (!data) return nullptr;

top= new tNode;

top->data=data;

if(first){

last = top;

}else first = last = top;

last-> next =createQ();

return top;

}

void Queue::outQ(Node a){

Node cur = a;

if (cur){

std::cout<<cur->data<<" ";

outQ(cur->next);

}

return;

}

void Queue::erase(){

Node temp = first;

if(temp){

temp = first->next;

std::cout<<"Элемент "<<first->data<<"вышел\n";

delete first;

first = temp;

erase();

}

return;

}

main.cpp

#include <iostream>

#include <Queue.h>

#include <windows.h>

static int cnt = 0;

using namespace std;

double maxEl(double arr[], int sz);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Queue Q;

int var;

while (var){

system("cls");

cout<<"1.Найти максимальный элемент массива\n2.Создание очереди\n3.Вывод очереди\n4.Удаление очереди\n0.Выход\n";

cout<<"Выберите опцию:";

cin>>var;

switch(var){

case 1:{

int size;

cout<<"Введите кол-во элементов массива:";

cin>>size;

double arr[size];

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

cout<<"i элемент:";

cin>>arr[i];

}

cout<<"MAX:"<<maxEl(arr,size - 1)<<endl;

cout<<"Глубина рекурсии:"<<cnt<<endl;

cnt = 0;

system("pause");

break;

}

case 2:{

Q.createQ();

system("pause");

break;

}

case 3:{

cout<<"Очередь\n";

Q.outQ(Q.getFirst());

system("pause");

break;

}

case 4:{

Q.erase();

system("pause");

break;

}

case 0:{

exit(0);

system("pause");

break;

}

default:{

cout<<"Выберите номер из списка!\n";

system("pause");

break;

}

}

}

return 0;

}

//Поиск максимума в массиве

double maxEl(double arr[], int sz) {

cnt++;

if(!sz) return arr[0];

else{

return maxEl(arr+(arr[0]<arr[sz]),sz - 1);

}

}