Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ *ИУК "Информатика и управление"*

КАФЕДРА *ИУК5 "Системы обработки информации"*

ЛАБОРАТОРНАЯ РАБОТА №3

«Списки, стеки, очереди.»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

Выполнил: студент гр.ИУК5-41Б _____ (____Шиндин А.О.___)
(Подпись) (Ф.И.О.)

Проверил: _____ (____ Вершинин В.Е__)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2023

Цель: получение практических навыков при работе со связными списками, стеками, очередями.

Задачи:

• В соответствии с вариантом, выданным преподавателем, решить задачи из 1й, 2й и 3й части
• Не использовать стандартные коллекции, структуры реализовывать самостоятельно
• В любой из 3х задач использовать стандартные коллекции, сравнить время выполнения при использовании собственных структур и стандартных коллекций

Задание:

Вариант 13:

1) Поменять порядок узлов в списке на обратный (только путем изменения указателей на первый элемент и в узлах)

2) Дан односвязный список, содержащий целые числа. Написать функцию, которая за один проход по списку распечатывает эти числа таким образом, что сначала распечатаются все положительные числа обратном порядке, затем – все отрицательные в обратном порядке. Элементы, равные нулю, печататься не должны. Для решения задачи использовать стеки

3) Дан односвязный список. Написать функцию, определяющую, образуют ли его элементы симметричную последовательность. Для решения задачи использовать стек и очередь. Функция будет возвращать значение true, если список является симметричным, false – в противном случае. По определению пустой список является симметричным. Поэтому, если список пуст, то возвращаем значение true. Для проверки симметричности списка нужно проверить на равенство все пары элементов, равноотстоящих от середины списка. Каждая пара содержит один элемент из первой половины списка и один – из второй. Элементы первой половины списка последовательно заносятся в очередь, второй половины в стек. Если количество элементов списка будет нечетным, то серединный элемент будет симметричен сам себе и не будет помещен ни в очередь, ни в стек.

Ход работы:

Программа написанная на языке c++:

```
#pragma region MAIN

#include <iostream>
#include <fstream>
#include <memory>
```

```
#include <ctime>
#include <thread>
#include <mutex>
#include <iomanip>
#include <stdexcept>
#include <stack>
#include <queue>

#pragma region LinkedList

struct Node {
int data;
std::shared_ptr<Node> next;
};

class LinkedList {
private:
std::shared_ptr<Node> head;
std::ofstream logfile;
std::mutex mtx;

public:

LinkedList() : head(nullptr) {
logfile.open("log.txt", std::ios::out | std::ios::app);
if (!logfile.is_open()) {
std::cerr << "Failed to open the log file\n";
} else {
// Определение времени начала сессии
std::time_t startSessionTime = std::time(nullptr);
std::string startTimeStamp = std::asctime(std::localtime(&startSessionTime));
startTimeStamp.pop_back();
logfile << "=== SRT session - (" + startTimeStamp + ") ===\n";
}
log("Program started", true);
}
```

```
~LinkedList() {
log("Program ended", true);
std::time_t startSessionTime = std::time(nullptr);
std::string startTimeStamp = std::asctime(std::localtime(&startSessionTime));
startTimeStamp.pop_back();
logfile << "=== END session - (" + startTimeStamp + ") ===\n";
logfile.close();
clear();
}

void prepend(int value) {
std::thread t([&] {
std::lock_guard<std::mutex> lock(mtx);
std::shared_ptr<Node> newNode = std::make_shared<Node>();
newNode->data = value;
newNode->next = head;
head = newNode;
log("Added to the beginning", true);
});
t.detach();
}

void append(int value) {
std::thread t([&] {
std::lock_guard<std::mutex> lock(mtx);
std::shared_ptr<Node> newNode = std::make_shared<Node>();

newNode->data = value;
newNode->next = nullptr;

if (!head) {
head = newNode;
} else {
std::shared_ptr<Node> temp = head;

while (temp->next) {
temp = temp->next;
```

```
}

temp->next = newNode;
}
log("Added to the end", true);
});
t.detach();
}

void removeFirst() {
std::thread t([&] {
std::lock_guard<std::mutex> lock(mtx);
if (head) {
head = head->next;
log("Removed from the beginning", true);
} else {
log("Rm from stp failed: List is empty", false);
}
});
t.detach();
}

void removeLast() {
std::thread t([&] {
std::lock_guard<std::mutex> lock(mtx);
if (head) {
if (!head->next) {
head = nullptr;
} else {
std::shared_ptr<Node> temp = head;
while (temp->next->next) {
temp = temp->next;
}
temp->next = nullptr;
}
log("Removed from the end", true);
} else {
log("Rm from end failed: List is empty", false);
```

```
}
});
t.detach();
}

void printPosNegReverse() {
try {
std::stack<int> positiveStack;
std::stack<int> negativeStack;

std::shared_ptr<Node> temp = head;
while (temp) {
if (temp->data > 0) {
positiveStack.push(temp->data);
} else if (temp->data < 0) {
negativeStack.push(temp->data);
}
temp = temp->next;
}

std::cout << "Positive numbers in reverse order: ";
while (!positiveStack.empty()) {
std::cout << positiveStack.top() << " ";
positiveStack.pop();
}

std::cout << "\nNegative numbers in reverse order: ";
while (!negativeStack.empty()) {
std::cout << negativeStack.top() << " ";
negativeStack.pop();
}
std::cout << std::endl;
log("Print list with stack", true);

} catch (const std::exception& e) {
log("Print list with stack", false);
std::cerr << "Exception caught: " << e.what() << std::endl;
```

```
}
}

void clear() {
std::thread t([&] {
std::lock_guard<std::mutex> lock(mtx);
head = nullptr;
log("List cleared", true);
});
t.detach();
}

void reverse() {
std::thread t([&] {
std::lock_guard<std::mutex> lock(mtx);
std::shared_ptr<Node> prev = nullptr;
std::shared_ptr<Node> current = head;
std::shared_ptr<Node> next = nullptr;

while (current) {
next = current->next;
current->next = prev;
prev = current;
current = next;
}

head = prev;
log("List reversed", true);
});
t.detach();
}

void display() {
std::thread t([&] {
std::lock_guard<std::mutex> lock(mtx);
std::shared_ptr<Node> temp = head;
while (temp) {
```

```
std::cout << temp->data << " ";
temp = temp->next;
}
std::cout << std::endl;
log("Displayed list", true);
});
t.detach();
}

bool isSymmetric() {
try {
std::stack<int> stack;
std::queue<int> queue;

std::shared_ptr<Node> temp = head;
while (temp) {
stack.push(temp->data);
queue.push(temp->data);
temp = temp->next;
}

while (!stack.empty() && !queue.empty()) {
if (stack.top() != queue.front()) {
log("check a symmetric", true);
return false;
}
stack.pop();
queue.pop();
}

log("Check a symmetric", true);
return stack.empty() && queue.empty();

} catch (const std::exception& e) {
log("Check a symmetric", false);
std::cerr << "Exception caught: " << e.what() << std::endl;
return false;
```

```
}
}

void log(const std::string& command, bool success) {
std::thread::id threadId = std::this_thread::get_id();
std::time_t currentTime = std::time(nullptr);
std::string timestamp = std::asctime(std::localtime(&currentTime));
timestamp.pop_back();

// Определяем фиксированную ширину для каждого столбца
const int timestampWidth = 25;
const int threadIdWidth = 20;
const int commandWidth = 35;
const int statusWidth = 20;

// Форматируем строки для каждого столбца
std::stringstream ss;
ss << "[" << std::left << std::setw(timestampWidth) << timestamp << "] ";
ss << "Thread ID: " << std::setw(threadIdWidth) << threadId << " | ";
ss << "Command: " << std::setw(commandWidth) << command << " | ";
ss << "Status: " << std::setw(statusWidth) << (success ? "Success" : "Failure") <<
std::endl;

logfile << ss.str();
}

};

#pragma endregion
#pragma endregion

int main() {
LinkedList list;
int choice;
int value;
```

```
do {
std::cout << "\nOperations:\n";
std::cout << "1. Add to beginning\n";
std::cout << "2. Add to end\n";
std::cout << "3. Remove from beginning\n";
std::cout << "4. Remove from end\n";
std::cout << "5. Clear list\n";
std::cout << "6. Reverse list\n";
std::cout << "7. Display list\n";
std::cout << "8. Function to print with stack\n";
std::cout << "9. Function to check a symmetric\n";
std::cout << "0. Exit\n";
std::cout << "Enter your choice: ";
std::cin >> choice;


try {
std::thread t;
switch (choice) {
case 1:
std::cout << "Enter value to add to the beginning: ";
std::cin >> value;
if (std::cin.fail()) {
throw std::invalid_argument("Invalid input");
}
t = std::thread([&] { list.prepend(value); });
break;
case 2:
std::cout << "Enter value to add to the end: ";
std::cin >> value;
if (std::cin.fail()) {
throw std::invalid_argument("Invalid input");
}
t = std::thread([&] { list.append(value); });
break;
case 3:
t = std::thread([&] { list.removeFirst(); });
break;
case 4:
t = std::thread([&] { list.removeLast(); });
```

```
break;
case 5:
t = std::thread([&] { list.clear(); });
break;
case 6:
t = std::thread([&] { list.reverse(); });
break;
case 7:
t = std::thread([&] { list.display(); });
break;
case 8:
t = std::thread([&] { list.printPosNegReverse(); });
break;
case 9:
t = std::thread([&] { std::cout << (list.isSymmetric() ? "is symmetric" : "is not
symmetric"); });
break;
case 0:
std::cout << "Exiting\n";
break;

default:
std::cout << "Invalid choice. Please try again.\n";
}
if (t.joinable()) {
t.join();
}
} catch (const std::invalid_argument& e) {
std::cerr << "Error: " << e.what() << ". Please enter a valid number.\n";
list.log("Invalid input", false);
std::cin.clear();
std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}
} while (choice != 0);

return 0;
}
```

Результат:

```
0. Exit
Enter your choice: 7
2 1 0 -554 222

Operations:
1. Add to beginning
2. Add to end
3. Remove from beginning
4. Remove from end
5. Clear list
6. Reverse list
7. Display list
8. Function to print with stack
9. Function to check a symmetric
0. Exit
Enter your choice: 9
is not symmetric
Operations:
1. Add to beginning
2. Add to end
3. Remove from beginning
4. Remove from end
5. Clear list
6. Reverse list
7. Display list
8. Function to print with stack
9. Function to check a symmetric
0. Exit
Enter your choice:
```

Логирование:

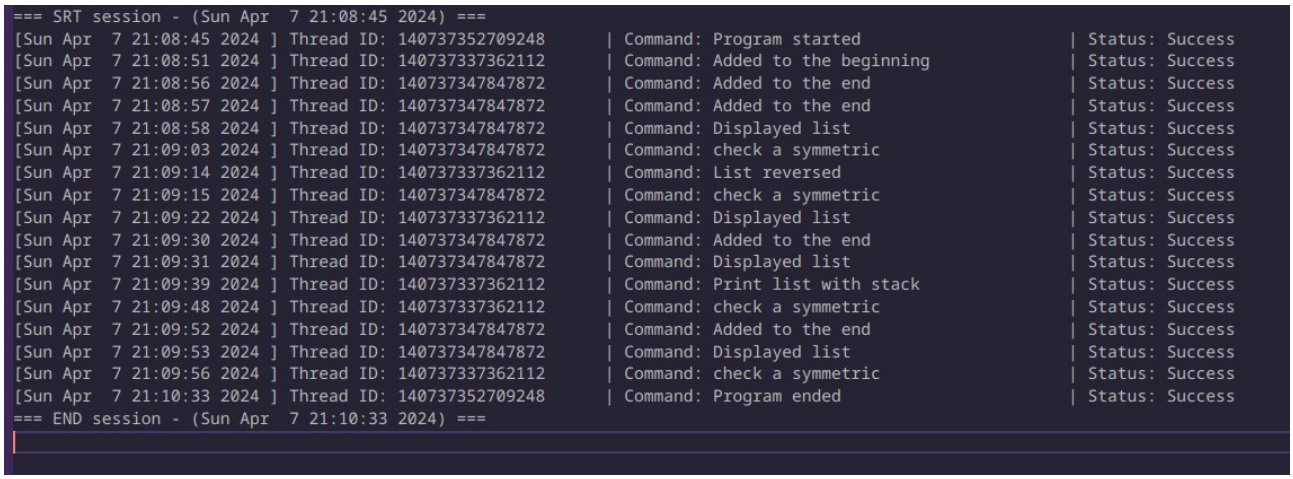
=== SRT session - (Mon Mar 11 15:07:39 2024) ===
[Mon Mar 11 15:07:39 2024] Thread ID: 140737352709248 | Command: Program started | Status: Success
[Mon Mar 11 15:07:42 2024] Thread ID: 140737337362112 | Command: Added to the beginning | Status: Success
[Mon Mar 11 15:07:45 2024] Thread ID: 140737337362112 | Command: Added to the beginning | Status: Success
[Mon Mar 11 15:07:49 2024] Thread ID: 140737347847872 | Command: check a symmetrict | Status: Success
[Mon Mar 11 15:07:56 2024] Thread ID: 140737337362112 | Command: Added to the beginning | Status: Success
[Mon Mar 11 15:07:59 2024] Thread ID: 140737337362112 | Command: check a symmetrict | Status: Failure

[Mon Mar 11 15:08:03 2024] Thread ID: 140737347847872 | Command: List cleared | Status: Success
[Mon Mar 11 15:08:05 2024] Thread ID: 140737337362112 | Command: Print list wiwh stack | Status: Success
[Mon Mar 11 15:08:13 2024] Thread ID: 140737352709248 | Command: Program ended | Status: Success
=== END session - (Mon Mar 11 15:08:13 2024) ===
=== SRT session - (Mon Mar 11 15:10:18 2024) ===
[Mon Mar 11 15:10:18 2024] Thread ID: 140737352709248 | Command: Program started | Status: Success
[Mon Mar 11 15:10:20 2024] Thread ID: 140737352709248 | Command: Program ended | Status: Success
=== END session - (Mon Mar 11 15:10:20 2024) ===
=== SRT session - (Mon Mar 11 15:43:46 2024) ===
[Mon Mar 11 15:43:46 2024] Thread ID: 140737352709248 | Command: Program started | Status: Success
[Mon Mar 11 15:45:17 2024] Thread ID: 140737337362112 | Command: Added to the beginning | Status: Success
[Mon Mar 11 15:45:23 2024] Thread ID: 140737347847872 | Command: Added to the end | Status: Success
[Mon Mar 11 15:45:25 2024] Thread ID: 140737347847872 | Command: List cleared | Status: Success
[Mon Mar 11 15:45:27 2024] Thread ID: 140737352709248 | Command: Program ended | Status: Success
=== END session - (Mon Mar 11 15:45:27 2024) ===
=== SRT session - (Mon Mar 11 17:33:52 2024) ===
[Mon Mar 11 17:33:52 2024] Thread ID: 140737352709248 | Command: Program started | Status: Success
[Mon Mar 11 17:33:56 2024] Thread ID: 140737352709248 | Command: Program ended | Status: Success
=== END session - (Mon Mar 11 17:33:56 2024) ===
=== SRT session - (Mon Mar 11 17:36:10 2024) ===
[Mon Mar 11 17:36:10 2024] Thread ID: 140737352709248 | Command: Program started | Status: Success
[Mon Mar 11 17:36:15 2024] Thread ID: 140737337362112 | Command: Added to the beginning | Status: Success
[Mon Mar 11 17:36:16 2024] Thread ID: 140737337362112 | Command: List cleared | Status: Success
[Mon Mar 11 17:36:17 2024] Thread ID: 140737337362112 | Command: Rm from end failed: List is empty | Status: Failure
[Mon Mar 11 17:36:18 2024] Thread ID: 140737347847872 | Command: Rm from stp failed: List is empty | Status: Failure
[Mon Mar 11 17:36:21 2024] Thread ID: 140737347847872 | Command: Print list with stack | Status: Success

[Mon Mar 11 17:36:22 2024] Thread ID: 140737352709248 | Command: Program ended | Status: Success
=== END session - (Mon Mar 11 17:36:22 2024) ===
=== SRT session - (Sat Mar 23 22:12:50 2024) ===
[Sat Mar 23 22:12:50 2024] Thread ID: 140737352709248 | Command: Program started | Status: Success
[Sat Mar 23 22:13:00 2024] Thread ID: 140737337362112 | Command: Added to the beginning | Status: Success
[Sat Mar 23 22:13:03 2024] Thread ID: 140737337362112 | Command: Added to the end | Status: Success
[Sat Mar 23 22:13:05 2024] Thread ID: 140737347847872 | Command: Added to the end | Status: Success
[Sat Mar 23 22:13:06 2024] Thread ID: 140737347847872 | Command: Displayed list | Status: Success
[Sat Mar 23 22:13:10 2024] Thread ID: 140737347847872 | Command: List reversed | Status: Success
[Sat Mar 23 22:13:11 2024] Thread ID: 140737337362112 | Command: Displayed list | Status: Success
[Sat Mar 23 22:13:16 2024] Thread ID: 140737337362112 | Command: Added to the beginning | Status: Success
[Sat Mar 23 22:13:19 2024] Thread ID: 140737337362112 | Command: Added to the end | Status: Success
[Sat Mar 23 22:13:20 2024] Thread ID: 140737337362112 | Command: Displayed list | Status: Success
[Sat Mar 23 22:13:23 2024] Thread ID: 140737337362112 | Command: List reversed | Status: Success
[Sat Mar 23 22:13:25 2024] Thread ID: 140737347847872 | Command: Displayed list | Status: Success
[Sat Mar 23 22:13:37 2024] Thread ID: 140737337362112 | Command: Displayed list | Status: Success
[Sat Mar 23 22:13:51 2024] Thread ID: 140737347847872 | Command: Displayed list | Status: Success
[Sat Mar 23 22:15:57 2024] Thread ID: 140737337362112 | Command: Print list with stack | Status: Success
[Sat Mar 23 22:16:13 2024] Thread ID: 140737337362112 | Command: check a symmetric | Status: Success
[Sat Mar 23 22:16:22 2024] Thread ID: 140737347847872 | Command: List cleared | Status: Success
[Sat Mar 23 22:16:24 2024] Thread ID: 140737347847872 | Command: Added to the beginning | Status: Success
[Sat Mar 23 22:16:25 2024] Thread ID: 140737337362112 | Command: Added to the end | Status: Success
[Sat Mar 23 22:16:27 2024] Thread ID: 140737337362112 | Command: Added to the end | Status: Success

[Sat Mar 23 22:16:28 2024] Thread ID: 140737337362112 | Command: Added to the end | Status: Success
[Sat Mar 23 22:16:29 2024] Thread ID: 140737337362112 | Command: Displayed list | Status: Success
[Sat Mar 23 22:16:32 2024] Thread ID: 140737347847872 | Command: Check a symmetric | Status: Success
[Sat Mar 23 22:16:43 2024] Thread ID: 140737337362112 | Command: List cleared | Status: Success
[Sat Mar 23 22:16:46 2024] Thread ID: 140737352709248 | Command: Program ended | Status: Success
=== END session - (Sat Mar 23 22:16:46 2024) ===
=== SRT session - (Sun Apr 7 21:08:45 2024) ===
[Sun Apr 7 21:08:45 2024] Thread ID: 140737352709248 | Command: Program started | Status: Success
[Sun Apr 7 21:08:51 2024] Thread ID: 140737337362112 | Command: Added to the beginning | Status: Success
[Sun Apr 7 21:08:56 2024] Thread ID: 140737347847872 | Command: Added to the end | Status: Success
[Sun Apr 7 21:08:57 2024] Thread ID: 140737347847872 | Command: Added to the end | Status: Success
[Sun Apr 7 21:08:58 2024] Thread ID: 140737347847872 | Command: Displayed list | Status: Success
[Sun Apr 7 21:09:03 2024] Thread ID: 140737347847872 | Command: check a symmetric | Status: Success
[Sun Apr 7 21:09:14 2024] Thread ID: 140737337362112 | Command: List reversed | Status: Success
[Sun Apr 7 21:09:15 2024] Thread ID: 140737347847872 | Command: check a symmetric | Status: Success
[Sun Apr 7 21:09:22 2024] Thread ID: 140737337362112 | Command: Displayed list | Status: Success
[Sun Apr 7 21:09:30 2024] Thread ID: 140737347847872 | Command: Added to the end | Status: Success
[Sun Apr 7 21:09:31 2024] Thread ID: 140737347847872 | Command: Displayed list | Status: Success
[Sun Apr 7 21:09:39 2024] Thread ID: 140737337362112 | Command: Print list with stack | Status: Success
[Sun Apr 7 21:09:48 2024] Thread ID: 140737337362112 | Command: check a symmetric | Status: Success
[Sun Apr 7 21:09:52 2024] Thread ID: 140737347847872 | Command: Added to the end | Status: Success
[Sun Apr 7 21:09:53 2024] Thread ID: 140737347847872 | Command: Displayed list | Status: Success
[Sun Apr 7 21:09:56 2024] Thread ID: 140737337362112 | Command: check a symmetric | Status: Success

[Sun Apr 7 21:10:33 2024] Thread ID: 140737352709248 | Command: Program ended | Status: Success
=== END session - (Sun Apr 7 21:10:33 2024) ===



Вывод: в результате выполнения лабораторной работы были приобретены практические навыки работы со связными списками, стеками, очередями.