МИНИСТЕРВСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №3

по дисциплине: "Логика и основы алгоритмизации в инженерных задачах"

на тему: "Динамические списки"

Выполнили студенты группы 24ВВВ3:

Пяткин Р. С.

Гусаров Е. Е.

Принял:

к.т.н., доцент, Юрова О. В.

к.т.н., Деев М. В.

Пенза 2025

Цель

Изучение динамических списков.

Лабораторное задание

Задание 1:

a. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).

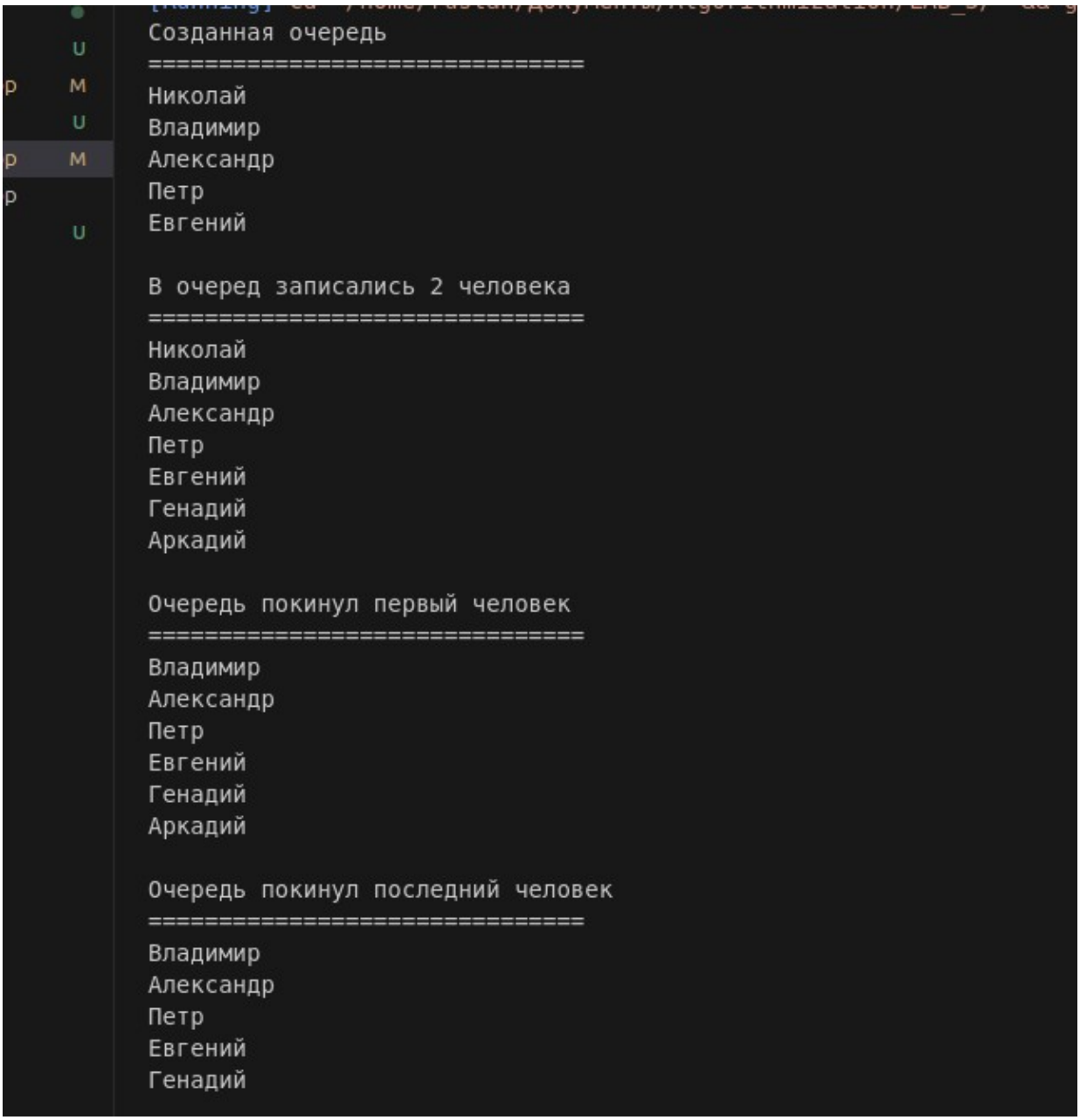
b. На основе приведенного кода реализуйте структуру данных Очередь.

c. На основе приведенного кода реализуйте структуру данных Стек.

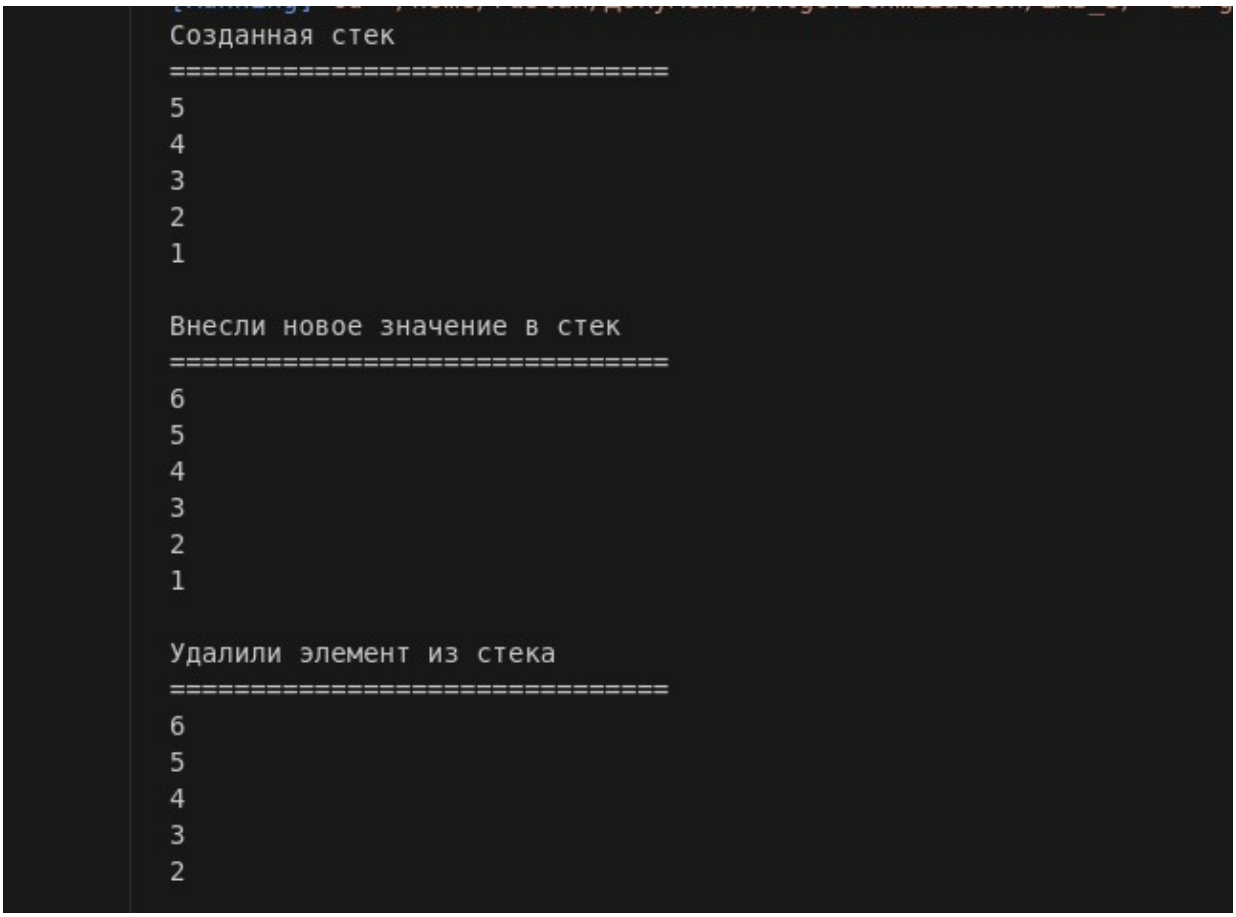
Результаты программ

```
ruslan@DexpAtlas:~/Документы/Algorithmization/LAB_3$ ./prog
=== ДОБАВЛЕНИЕ ЭЛЕМЕНТОВ С ПРИОРИТЕТАМИ ===
[0] Очень срочная (приоритет: 5)
[1] Срочная задача (приоритет: 4)
[2] Важная задача (приоритет: 3)
[3] Средняя задача (приоритет: 2)
[4] Обычная задача (приоритет: 1)
[5] Не очень важная (приоритет: 0)
==========================================
[1] Изменить список
[2] Завершить программу
[=] 1
[1] Добавить элемент
[2] Удалить элемент
[=] 1
введите данные (для завершение программы введите: 0)
[=] Новый
введите приоритет (для завершение программы введите: 0)
[=] 10
Элемент успешно добавлен!
[0] Новый (приоритет: 10)
[1] Очень срочная (приоритет: 5)
[2] Срочная задача (приоритет: 4)
[3] Важная задача (приоритет: 3)
[4] Средняя задача (приоритет: 2)
[5] Обычная задача (приоритет: 1)
[6] Не очень важная (приоритет: 0)
ruslan@DexpAtlas:~/Документы/Algorithmization/LAB_3$
```

1 Рис. — lab_3_1


Созданная очередь
==========
Николай
Владимир
Александр
Петр
Евгений
В очеред записались 2 человека
==========
Николай
Владимир
Александр
Петр
Евгений
Генадий
Аркадий
Очередь покинул первый человек
==========
Владимир
Александр
Петр
Евгений
Генадий
Аркадий
Очередь покинул последний человек
==========
Владимир
Александр
Петр
Евгений
Генадий

2 Рис. — lab_3_2



3 Рис. — lab_3_3

Листинг

Файл lab_3_1.cpp

```
#include "fstream"
#include "iostream"
#include <string>

using namespace std;

template<typename T>
class PriorityNode {
public:
PriorityNode* pNext;
T data;
int priority;
PriorityNode(T data = T(), int priority = 0, PriorityNode *pNext =
nullptr) {
this->data = data;
this->priority = priority;
this->pNext = pNext;
}
};

template<typename T>
```

```
class PriorityList {
private:
int Size;
PriorityNode<T> *head;
public:
PriorityList();
~PriorityList();
void push_preority(T data, int priority);
void removePriority(int index);
void delete_front();
void clear();
int GetSize() { return Size; }
void printList();
};

template<typename T>
PriorityList<T>::PriorityList() {
Size = 0;
head = nullptr;
}

template<typename T>
PriorityList<T>::~PriorityList() {
clear();
}

template<typename T>
void PriorityList<T>::clear() {
while(Size) {
delete_front();
}
}

template<typename T>
void PriorityList<T>::delete_front() {
if (head == nullptr) return;
PriorityNode<T> *temp = head;
head = head->pNext;
delete temp;
Size--;
}

template<typename T>
void PriorityList<T>::removePriority(int priority) {
if (head == nullptr) {
cout<<"Список пуст";
return;
}
while(head != nullptr && head->priority == priority){
delete_front();
}
```

```
if(head == nullptr){
return;
}
PriorityNode<T>* current = head;
while (current->pNext != nullptr)
{
if(current->pNext->priority == priority){
PriorityNode<T>* toDelete = current->pNext;
current->pNext = toDelete->pNext;
delete toDelete;
Size--;
}else{
current = current->pNext;
}
}
}

template<typename T>
void PriorityList<T>::push_preority(T data, int priority) {
PriorityNode<T>* newNode = new PriorityNode<T>(data, priority);
if (head == nullptr || priority > head->priority) {
newNode->pNext = head;
head = newNode;
} else {
PriorityNode<T>* current = head;
while (current->pNext != nullptr && current->pNext->priority >=
priority) {
current = current->pNext;
}
newNode->pNext = current->pNext;
current->pNext = newNode;
}
Size++;
}

template<typename T>
void PriorityList<T>::printList() {
PriorityNode<T>* current = head;
int index = 0;
while(current != nullptr) {
cout << "[" << index << "] " << current->data << " (приоритет: " <<
current->priority << ")" << endl;
current = current->pNext;
index++;
}
}

int main() {
string request_one;
string request_two;
PriorityList<string> lst;
```

```
lst.push_preority("Не очень важная", 0);
lst.push_preority("Обычная задача", 1);
lst.push_preority("Средняя задача", 2);
lst.push_preority("Важная задача", 3);
lst.push_preority("Срочная задача", 4);
lst.push_preority("Очень срочная", 5);
cout << "=== ДОБАВЛЕНИЕ ЭЛЕМЕНТОВ С ПРИОРИТЕТАМИ ===" << endl;
lst.printList();
cout << "========================================" << endl;
cout<<"[1] Изменить список"<<endl<<"[2] Завершить
программу"<<endl<<"[=] ";
cin>>request_one;
if(request_one == "1" || request_one == "Изменить список" ||
request_one == "изменить список"){
cout<<"[1] Добавить элемент"<<endl<<"[2] Удалить элемент"<<endl<<"[=]
";
cin>>request_two;
if(request_two == "1" || request_two == "Добавить элемент" ||
request_two == "добавить элемент"){
string request_push;
string request_push_priority;
cout<<"введите данные (для завершение программы введите:
0)"<<endl<<"[=] ";
cin>>request_push;
if(request_push == "0" || request_push == "Завершить" || request_push
== "завершить"){
return 0;
}
cout<<"введите приоритет (для завершение программы введите:
0)"<<endl<<"[=] ";
cin>>request_push_priority;
if(request_push_priority == "0" || request_push_priority ==
"Завершить" || request_push_priority == "завершить"){
return 0;
}
lst.push_preority(request_push, stoi(request_push_priority));
cout<<"Элемент успешно добавлен!"<<endl;
lst.printList();
}else if(request_two == "2" || request_two == "Удалить элемент" ||
request_two == "удалить элемент"){
string request_push;
cout<<"введите преоритет элемента для его удаления (для завершение
программы введите: 0)"<<endl<<"[=] ";
cin>>request_push;
if(request_push == "0" || request_push == "Завершить" || request_push
== "завершить"){
return 0;
}else if(stoi(request_push) > lst.GetSize()){
cout<<"Ошибка!"<<endl;
return 0;
}
```

```
lst.removePriority(stoi(request_push));
cout<<"Элемент успешно удален!"<<endl;
lst.printList();
}

}else if(request_one == "2" || request_one == "Завершить программу" ||
request_one == "завершить программу"){
return 0;
}
return 0;
}
```

Файл lab_3_2.cpp

```
#include "fstream"
#include "iostream"

using namespace std;

template<typename T>
class List{
public:
List();
~List();
void pop_back();
void clear();
void pop_front();
void push_back(T data);
void printList();
int GetSize(){ return Size; }
private:
template<typename U>
class Node{
public:
Node* pNext;
U data;
Node(T data = T(), Node *pNext = nullptr){
this->data = data;
this->pNext = pNext;
}
};
int Size;
Node<T> *head;
};

template<typename T>
List<T>::List(){
Size = 0;
head = nullptr;
}

template<typename T>
```

```
List<T>::~List(){
clear();
}
template <typename T>
void List<T>::pop_back(){
if (head == nullptr) return;
if (head->pNext == nullptr) {
delete head;
head = nullptr;
} else {
Node<T> *current = head;
while (current->pNext->pNext != nullptr) {
current = current->pNext;
}
delete current->pNext;
current->pNext = nullptr;
}
Size--;
}
template <typename T>
void List<T>::pop_front(){
Node<T> *temp = head;
head = head->pNext;
delete temp;
Size--;
}
template<typename T>
void List<T>::clear(){
while(Size){
pop_front();
}
}

template <typename T>
void List<T>::push_back(T data){
if(head == nullptr){
head = new Node<T>(data);
}
else{
Node<T>* current = this->head;
while (current->pNext != nullptr)
{
current = current->pNext;
}
current->pNext = new Node<T>(data);
}
Size++;
}

template<typename T>
void List<T>::printList() {
```

```
Node<T>* current = head;
while(current != nullptr) {
cout<<current->data<<endl;
current = current->pNext;
}
}

int main(){
List<string> lst;
lst.push_back("Николай");
lst.push_back("Владимир");
lst.push_back("Александр");
lst.push_back("Петр");
lst.push_back("Евгений");

cout<<"Созданная очередь"<<endl;
cout<<"=============================="<<endl;
lst.printList();
cout<<endl;
cout<<"В очеред записались 2 человека"<<endl;
cout<<"=============================="<<endl;
lst.push_back("Генадий");
lst.push_back("Аркадий");
lst.printList();
cout<<endl;
cout<<"Очередь покинул первый человек"<<endl;
cout<<"=============================="<<endl;
lst.pop_front();
lst.printList();
cout<<endl;
cout<<"Очередь покинул последний человек"<<endl;
cout<<"=============================="<<endl;
lst.pop_back();
lst.printList();
}
```

Файл lab_3_3.cpp

```
#include "fstream"
#include "iostream"

using namespace std;

template<typename T>
class List{
public:
List();
~List();
void pop_back();
void push_front(T data);
void clear();
void printList();
```

```
int GetSize(){ return Size; }
private:
template<typename U>
class Node{
public:
Node* pNext;
U data;
Node(T data = T(), Node *pNext = nullptr){
this->data = data;
this->pNext = pNext;
}
};
int Size;
Node<T> *head;
};

template<typename T>
List<T>::List(){
Size = 0;
head = nullptr;
}

template<typename T>
List<T>::~List(){
clear();
}
template <typename T>
void List<T>::pop_back(){
if(head == nullptr) return;
if(head->pNext == nullptr){
delete head;
head == nullptr;
} else {
Node<T> *current = head;
while (current->pNext->pNext != nullptr) {
current = current->pNext;
}
delete current->pNext;
current->pNext = nullptr;
}
Size--;
}
template <typename T>
void List<T>::push_front(T data){
head = new Node<T>(data, head);
Size++;
}
template<typename T>
void List<T>::clear(){#include "fstream"
#include "iostream"
```

```
using namespace std;

template<typename T>
while(Size){
pop_back();
}
}
template<typename T>
void List<T>::printList() {
Node<T>* current = head;
while(current != nullptr) {
cout<<current->data<<endl;
current = current->pNext;
}
}
int main(){
List<int> lst;
lst.push_front(1);
lst.push_front(2);
lst.push_front(3);
lst.push_front(4);
lst.push_front(5);

cout<<"Созданная стек"<<endl;
cout<<"=============================="<<endl;
lst.printList();
cout<<endl;

cout<<"Внесли новое значение в стек"<<endl;
cout<<"=============================="<<endl;
lst.push_front(6);
lst.printList();
cout<<endl;
cout<<"Удалили элемент из стека"<<endl;
cout<<"=============================="<<endl;
lst.pop_back();
lst.printList();
}
```

Вывод:

В ходе выполнения лабораторной работы были разработаны программы для выполнения заданий Лабораторной работы №3. В процессе выполнения работы были использованы знания о динамических списках.