

МИНИСТЕРСТВО 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

Плотников И.А.

Виноградов Б.С.

Приняли:

Деев М.В.

Юрова О. В.

Пенза 2025

Цель

Изучение динамических структур данных.

Лабораторное задание

Задание

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).

* На основе приведенного кода реализуйте структуру данных Очередь.

* На основе приведенного кода реализуйте структуру данных Стек.

Пояснительный текст к программе

Программа реализует три динамические структуры данных:

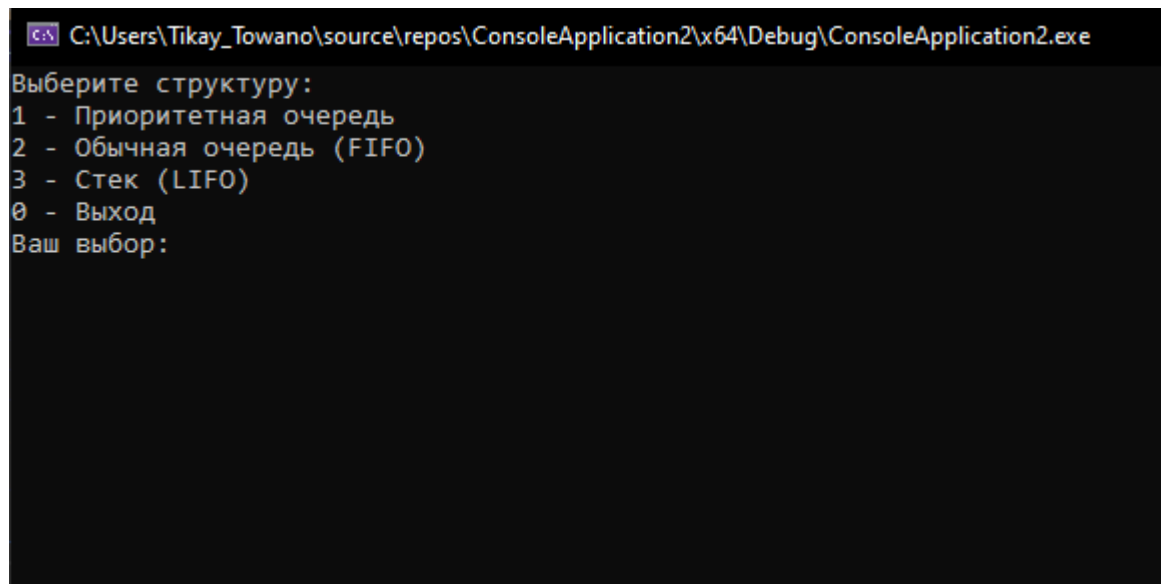
1. Приоритетная очередь
2. Обычная очередь (FIFO)
3. Стек (LIFO)

Основные возможности программы:

1. Добавление элементов.
2. Удаление элементов.
3. Просмотр содержимого.
4. Поиск.
5. Изменение приоритета.
6. Выход.

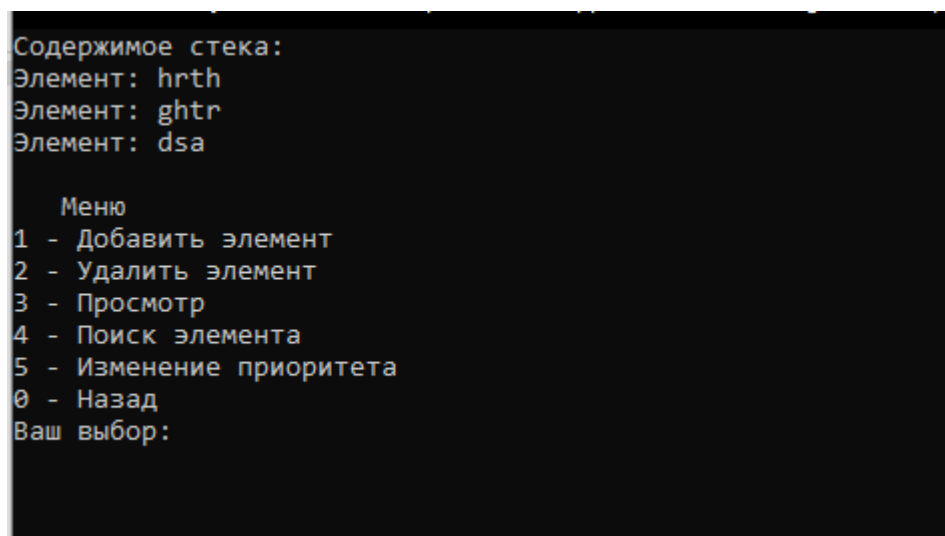
Результаты работы программы

1 Рис. - Результат работы **ex1_lab3.c**



```
C:\Users\Tikay_Towano\source\repos\ConsoleApplication2\x64\Debug\ConsoleApplication2.exe
Выберите структуру:
1 - Приоритетная очередь
2 - Обычная очередь (FIFO)
3 - Стек (LIFO)
0 - Выход
Ваш выбор:
```

2 Рис. – Просмотр содержимого стека.



```
Содержимое стека:
Элемент: hrth
Элемент: ghtr
Элемент: dsa

    Меню
1 - Добавить элемент
2 - Удалить элемент
3 - Просмотр
4 - Поиск элемента
5 - Изменение приоритета
0 - Назад
Ваш выбор:
```

3 Рис. – Поиск по неполному совпадению.

```
Содержимое стека:
Элемент: hrth
Элемент: ghtr
Элемент: dsa

    Меню
1 - Добавить элемент
2 - Удалить элемент
3 - Просмотр
4 - Поиск элемента
5 - Изменение приоритета
0 - Назад
Ваш выбор: 4
Введите имя для поиска: hr
Найден элемент: hrth | приоритет: 0

    Меню
1 - Добавить элемент
2 - Удалить элемент
3 - Просмотр
4 - Поиск элемента
5 - Изменение приоритета
0 - Назад
Ваш выбор:
```

4 Рис. – Удаление элемента.

```
    Меню
1 - Добавить элемент
2 - Удалить элемент
3 - Просмотр
4 - Поиск элемента
5 - Изменение приоритета
0 - Назад
Ваш выбор: 2
Извлечён элемент: hrth

    Меню
1 - Добавить элемент
2 - Удалить элемент
3 - Просмотр
4 - Поиск элемента
5 - Изменение приоритета
0 - Назад
Ваш выбор: █
```

Листинг

Файл ex1_lab3.c

```
#include <iostream>
#include <string>
#include <limits>

void clearScreen();

struct Node {
    std::string inf;
    int priority;
    Node* next;
};

Node* head = nullptr;
Node* last = nullptr;

// создание структуры
Node* getStructSimple();
Node* getStructPriority();

// поиск
Node* findElement(const std::string& searchStr);

// приоритетная очередь
void pushPriority();
void popPriority();
void reviewPriority();
void switchPriority();

// обычная очередь
void enqueue();
void dequeue();
void reviewQueue();

// стек
void pushStack();
void popStack();
void reviewStack();

// очистка
void clearList();

int main() {
    int mainChoice, subChoice;
    std::string name;

    while (true) {
        clearScreen();
        // выбор структуры
        while (true) {
            std::cout << "Выберите структуру:\n";
            std::cout << "1 - Приоритетная очередь\n";
            std::cout << "2 - Обычная очередь (FIFO)\n";
            std::cout << "3 - Стек (LIFO)\n";
```

```

        std::cout << "0 - Выход\n";
        std::cout << "Ваш выбор: ";
        if (!(std::cin >> mainChoice)) {
            std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
            std::cout << "Неверный ввод, введите число\n";
            continue;
        }
        break;
    }

    if (mainChoice == 0) break;

    clearList();

    // меню выбора
    while (true) {
        std::cout << "\n    Меню    \n";
        std::cout << "1 - Добавить элемент\n2 - Удалить элемент\n3
- Просмотр\n4 - Поиск элемента\n5 - Изменение приоритета\n0 -
Назад\n";

        std::cout << "Ваш выбор: ";
        if (!(std::cin >> subChoice)) {
            std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
            std::cout << "Неверный ввод, введите число\n";
            continue;
        }

        if (subChoice == 0) break;

        switch (mainChoice) {
            case 1:
                switch (subChoice) {
                    case 1:
                        pushPriority();
                        break;
                    case 2:
                        popPriority();
                        break;
                    case 3:
                        reviewPriority();
                        break;
                    case 4:
                        std::cout << "Введите имя для поиска: ";
                        std::cin >> name;
                        findElement(name);
                        break;
                    case 5:
                        switchPriority();
                        break;
                    default:
                        std::cout << "Неверный выбор\n";
                }
                break;
        }
    }
}
break;

```

```

        case 2:
            switch (subChoice) {
                case 1:
                    enqueue();
                    break;
                case 2:
                    dequeue();
                    break;
                case 3:
                    reviewQueue();
                    break;
                case 4:
                    std::cout << "Введите имя для поиска: ";
                    std::cin >> name;
                    findElement(name);
                    break;
                default:
                    std::cout << "Неверный выбор\n";
            }
            break;

        case 3:
            switch (subChoice) {
                case 1:
                    pushStack();
                    break;
                case 2:
                    popStack();
                    break;
                case 3:
                    reviewStack();
                    break;
                case 4:
                    std::cout << "Введите имя для поиска: ";
                    std::cin >> name;
                    findElement(name);
                    break;
                default:
                    std::cout << "Неверный выбор\n";
            }
            break;

        default:
            std::cout << "Неверный выбор\n";
    }
}

clearList();
return 0;
}

void clearScreen() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

```

```

}

Node* getStructSimple() {
    std::string name;
    while (true) {
        std::cout << "Введите название: ";
        std::cin >> name;
        if (!name.empty()) break;
        std::cout << "Имя не может быть пустым. Попробуйте снова\n";
    }
    Node* p = new Node{name, 0, nullptr};
    return p;
}

Node* getStructPriority() {
    std::string name;
    int pr;
    while (true) {
        std::cout << "Введите название: ";
        std::cin >> name;
        if (!name.empty()) break;
        std::cout << "Имя не может быть пустым. Попробуйте снова\n";
    }

    while (true) {
        std::cout << "Введите приоритет (целое число): ";
        if (!(std::cin >> pr)) {
            std::cin.clear();
        }

        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        std::cout << "Неверный ввод, введите число\n";
        continue;
    }

    if (pr < 1) {
        std::cout << "Ошибка: Число < 1, повторите ввод" << "\n";
        continue;
    }

    break;
}

Node* p = new Node{name, pr, nullptr};
return p;
}

Node* findElement(const std::string& searchStr) {
    Node* cur = head;
    bool found = false;

    if (!cur) {
        std::cout << "Структура пуста\n";
        return nullptr;
    }

    while (cur) {
        if (cur->inf.find(searchStr) != std::string::npos) {
            std::cout << "Найден элемент: " << cur->inf
                << " | приоритет: " << cur->priority << "\n";
        }
    }
}

```



```

        found = true;
    }
    cur = cur->next;
}

if (!found) {
    std::cout << "Элемент с именем, содержащим '" << searchStr <<
"'" не найден\n";
}

return nullptr;
}

void pushPriority() {
    Node* p = getStructPriority();
    if (!head) {
        head = last = p;
        return;
    }
    if (p->priority < head->priority) {
        p->next = head;
        head = p;
        return;
    }
    Node* cur = head;
    while (cur->next && cur->next->priority <= p->priority) {
        cur = cur->next;
    }
    p->next = cur->next;
    cur->next = p;
    if (!p->next) last = p;
}

void popPriority() {
    if (!head) {
        std::cout << "Очередь пуста\n";
        return;
    }
    Node* temp = head;
    std::cout << "Извлечён элемент: " << temp->inf << " (приоритет "
<< temp->priority << ")\n";
    head = head->next;
    delete temp;
}

void reviewPriority() {
    clearScreen();
    if (!head) {
        std::cout << "Очередь пуста\n";
        return;
    }
    Node* cur = head;
    std::cout << "Содержимое приоритетной очереди:\n";
    while (cur) {
        std::cout << "Элемент: " << cur->inf << " | приоритет: " <<
cur->priority << "\n";
        cur = cur->next;
    }
}

```

```

}

void enqueue() {
    Node* p = getStructSimple();
    if (!head) {
        head = last = p;
    } else {
        last->next = p;
        last = p;
    }
}

void dequeue() {
    if (!head) {
        std::cout << "Очередь пуста\n";
        return;
    }
    Node* temp = head;
    std::cout << "Извлечён элемент: " << temp->inf << "\n";
    head = head->next;
    delete temp;
}

void reviewQueue() {
    clearScreen();
    if (!head) {
        std::cout << "Очередь пуста\n";
        return;
    }
    Node* cur = head;
    std::cout << "Содержимое обычной очереди:\n";
    while (cur) {
        std::cout << "Элемент: " << cur->inf << "\n";
        cur = cur->next;
    }
}

void pushStack() {
    Node* p = getStructSimple();
    if (!head) {
        head = last = p;
    } else {
        p->next = head;
        head = p;
    }
}

void popStack() {
    if (!head) {
        std::cout << "Стек пуст\n";
        return;
    }
    Node* temp = head;
    std::cout << "Извлечён элемент: " << temp->inf << "\n";
    head = head->next;
    delete temp;
}

```

```

void reviewStack() {
    clearScreen();
    if (!head) {
        std::cout << "Стек пуст\n";
        return;
    }
    Node* cur = head;
    std::cout << "Содержимое стека:\n";
    while (cur) {
        std::cout << "Элемент: " << cur->inf << "\n";
        cur = cur->next;
    }
}

void clearList() {
    Node* cur = head;
    while (cur) {
        Node* temp = cur;
        cur = cur->next;
        delete temp;
    }
    head = last = nullptr;
}

void switchPriority() {
    clearScreen();

    if (!head) {
        std::cout << "Очередь пуста\n";
        return;
    }

    std::string name;
    int newPriority;

    std::cout << "Введите имя элемента, чей приоритет нужно изменить:
";
    std::cin >> name;

    Node* prev = nullptr;
    Node* cur = head;

    while (cur && cur->inf != name) {
        prev = cur;
        cur = cur->next;
    }

    if (!cur) {
        std::cout << "Элемент с именем '" << name << "' не найден\n";
        return;
    }

    std::cout << "Текущий приоритет элемента '" << cur->inf
        << "' = " << cur->priority << "\n";

    while (true) {
        std::cout << "Введите новый приоритет: ";
        if (!(std::cin >> newPriority)) {

```

```

        std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        std::cout << "Ошибка ввода. Введите целое число\n";
        continue;
    }
    if (newPriority < 1) {
        std::cout << "Приоритет должен быть >= 1\n";
        continue;
    }
    break;
}

if (prev) {
    prev->next = cur->next;
}
else {
    head = cur->next;
}

if (cur == last) {
    last = prev;
}

cur->priority = newPriority;
cur->next = nullptr;

if (!head) {
    head = last = cur;
}
else if (cur->priority < head->priority) {
    cur->next = head;
    head = cur;
}
else {
    Node* iter = head;
    while (iter->next && iter->next->priority <= cur->priority) {
        iter = iter->next;
    }
    cur->next = iter->next;
    iter->next = cur;
    if (!cur->next)
        last = cur;
}

std::cout << "Приоритет элемента '" << name << "' успешно изменён
на '" << newPriority << "\n";
}

```

Вывод:

В ходе выполнения лабораторной работы были разработаны программы для выполнения заданий Лабораторной работы №2. В процессе выполнения работы был изучены динамические структуры данных.