Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет (НИУ)»

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ОТЧЕТ

о выполнении дополнительного практического задания №1

по дисциплине

«Структуры и алгоритмы обработки данных»

Вариант 5

Проверил:

ст. преподаватель кафедры СП

Петрова Л.Н.

Выполнил:

Студент группы КЭз-391

Галиулин Р.Р.

Челябинск
2025

Содержание

1. Описание задачи

Дополнительное задание 1: «ДВУСВЯЗНЫЙ СПИСОК»

В созданном списке определить максимальное значение и удалить его.

Входные данные

- Количество элементов списка, целое число больше нуля
- Данные элементов списка, целые числа

Все данные вводятся с помощью стандартного потока ввода

Выходные данные

- Строка, представляющая список чисел, введенных пользователем. Максимальное значение будет выделено квадратными скобками
- Строка, представляющая список после удаления максимального элемента

Все данные выводятся с помощью стандартного потока вывода

2. Листинг программы

Язык программирования: C++ 14. Среда разработки: Ubuntu 24.10, gcc 14.2.0, nvim

Листинг 1: Дополнительное задание 1: «ДВУСВЯЗНЫЙ СПИСОК»

```
// Галиулин РР.. КЭз-391
// Структуры и алгоритмы обработки данных
// Дополнительное практическое занятие №1

// В созданном списке определить максимальное значение и удалить его.

#include <iostream>
#include <ctime>
#include <cstdlib>
#include <limits>

// Структура для представления узла двусвязного списка
struct Node {
    int data;
    Node* prev; // Указатель на предыдущий узел
    Node* next; // Указатель на следующий узел
};

// Функция для поиска максимального элемента в списке
Node* FindMax(Node* head) {
    // Если список пуст, возвращаем nullptr
    if (head == nullptr) return nullptr;

    // Инициализируем максимальный узел и его значение начальным узлом
    Node* max_node = head;
    int max_value = head->data;

    // Проходим по списку, начиная со следующего узла
    Node* temp = head->next;
    while (temp != nullptr) {
        // Если текущий элемент больше максимального, обновляем max_node и max_value
        if (temp->data > max_value) {
            max_value = temp->data;
            max_node = temp;
        }
        temp = temp->next;
    }
    // Возвращаем указатель на узел с максимальным значением
    return max_node;
}

// Функция для удаления узла из списка
void DeleteNode(Node** list_head_ptr, Node* del) {
    // Если список пуст или узел для удаления не существует
    if (*list_head_ptr == nullptr || del == nullptr)
        return;

    // Если узел головной, то меняем указатель на голову
    if (*list_head_ptr == del)
        *list_head_ptr = del->next;

    // Если есть следующий узел, то изменяем его указатель на предыдущий
    if (del->next != nullptr)
        del->next->prev = del->prev;

```

```
56      // Если есть предыдущий узел, то изменяем его указатель на следующий
57      if (del->prev != nullptr)
58          del->prev->next = del->next;
59
60      // Чистим
61      delete del;
62  }
63
64  // Функция для удаления максимального элемента из списка
65  void DeleteMax(Node** list_head_ptr) {
66      // Если список пуст
67      if (*list_head_ptr == nullptr)
68          return;
69
70      // Находим максимальный узел
71      Node* max_node = FindMax(*list_head_ptr);
72      // и удаляем
73      DeleteNode(list_head_ptr, max_node);
74  }
75
76  // Функция для добавления нового узла в конец списка
77  void Append(Node** list_head_ptr, int new_data) {
78      Node* new_node = new Node();
79      Node* last = *list_head_ptr; // временный указатель для перемещения по списку
80
81      // Записываем данные в новый узел, устанавливаем next = nullptr
82      new_node->data = new_data;
83      new_node->next = nullptr;
84
85      // Если список пуст, новый узел становится головным
86      if (*list_head_ptr == nullptr) {
87          new_node->prev = nullptr;
88          *list_head_ptr = new_node;
89          return;
90      }
91      // Находим последний узел
92      while (last->next != nullptr)
93          last = last->next;
94
95      // Привязываем новый узел к последнему узлу
96      last->next = new_node;
97      new_node->prev = last;
98  }
99
100 // Функция для вывода двусвязного списка в консоль.
101 // Параметр highlight_max определяет, нужно ли выделять максимальный эл.
102 void PrintList(Node* node, bool highlight_max) {
103     // Если список пуст
104     if (node == nullptr) return;
105
106     // Определяем max_node и его значение, для выделения максимума
107     Node* max_node = nullptr;
108     int max_value = 0;
109     if (highlight_max) {
110         max_node = FindMax(node);
111         max_value = (max_node != nullptr) ? max_node->data : 0;
112     }
113
114     // Проходим по списку и печатаем значения
115     while (node != nullptr) {
```

```
116         // максимум выводим в скобках
117         if (highlight_max && node->data == max_value) {
118             std::cout << "[" << node->data << "] ";
119         } else {
120             std::cout << node->data << " ";
121         }
122         node = node->next; // Переходим к следующему узлу
123     }
124     std::cout << std::endl;
125 }
126 
127 // Функция для получения корректного целого числа
128 int get_int_from_user(const std::string& prompt) {
129     int input_value;
130     while (true) {
131         std::cout << prompt;
132         std::cin >> input_value;
133 
134         if (std::cin.fail()) {
135             std::cout << "Некорректный ввод. Введите целое число.\n";
136             std::cin.clear();
137             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n
    ');
138         } else {
139              std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\
    n');
140             break; // если ввод корректен
141         }
142     }
143     return input_value;
144 }
145 
146 int main() {
147     Node* head = nullptr; // Инициализируем голову списка nullptr
148     int num_elements;
149 
150     // Получаем количество элементов
151     num_elements = get_int_from_user("Введите количество элементов в списке: ");
152 
153     // Заполняем список данными
154     for (int i = 0; i < num_elements; ++i) {
155         int data = get_int_from_user("Введите целое число для элемента " + std::
    to_string(i + 1) + ": ");
156         Append(&head, data);
157     }
158 
159     std::cout << "Исходный список: ";
160     PrintList(head, true);
161 
162     // Удаляем максимальный элемент
163     DeleteMax(&head);
164 
165     std::cout << "Список после удаления максимального элемента: ";
166     PrintList(head, false);
167 
168     return 0;
169 }
```

3. Контрольные тесты

3.1 Задание №1: Двух связанный список

Исходные данные	**Результат**
10 43 45 86 11 94 29 75 99 41 12	Исходный список: 43 45 86 11 94 29 75 [99] 41 12 Список после удаления максимального элемента: 43 45 86 11 94 29 75 41 12
5 5 5 9 2 10	Исходный список: 5 5 9 2 [10] Список после удаления максимального элемента: 5 5 9 2

Таблица 1: Таблица с результатами контрольных тестов Задания №1