Министерство науки и высшего образования Российской Федерации

Чайковский филиал

федерального государственного автономного образовательного учреждения

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра Автоматизации, информационных и инженерных технологий

Направление: 09.03.01 Информатика и вычислительная техника

Направленность: Автоматизированные системы обработки информации и управления

**О Т Ч Е Т**

**по лабораторной работе №3**

«Динамические структуры данных»

по дисциплине

«Теория алгоритмов и структуры данных»

|  |  |  |
| --- | --- | --- |
|  |  | Выполнили: |
|  |  | студенты группы АСУ-22-1б ЧФ |
|  |  | *Базуев Д.О., Гимадеев Р.Р.* |
|  |  | *(Фамилия Имя Отчество)* |
|  |  |  |
|  |  | Проверил: |
|  |  | *Ст. преподаватель* |
|  |  | *(должность, ученая степень, ученое звание)* |
|  |  | *Сухих И.И.* |
|  |  | *(Фамилия Имя Отчество)* |
|  |  | *(дата, подпись)* |

Чайковский 2023

Цель работы:

1. Получить практические навыки работы с однонаправленными списками.
2. Получить практические навыки работы с двунаправленными списками.
3. Получить практические навыки работы с деревьями.

Программное обеспечение: Microsoft Visual Studio 2022.

Задание 1. Сформировать однонаправленный список. Тип информационного поля double. Удалить из списка все элементы с чётными номерами (2, 4, 6 и т.д.). Вывести полученный список.

Исходный код:

#include <iostream>

#include <time.h>

using namespace std;

struct Node {

double data;

Node\* next;

Node(double value) : data(value), next(nullptr)

{}

};

Node\* sozdanie(int size);

void ydalenie(Node\*& head);

void pechat(Node\* head);

int main()

{

setlocale(0, "");

int n, num, i;

cout << "Введите количество элементов в списке: ";

cin >> n;

Node\* spicok = sozdanie(n);

cout << "Исходный список: ";

pechat(spicok);

cout << "Удаление из списка элементов с чётными номерами:\n";

ydalenie(spicok);

pechat(spicok);

}

Node\* sozdanie(int size)

{

if (size <= 0)

{

return nullptr;

}

Node\* head = nullptr;

Node\* tail = nullptr;

cout << "Введите элементы списка:" << endl;

double value;

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

{

cout<<"Элемент " << i + 1 << ": ";

cin >> value;

Node\* newNode = new Node(value);

if (!head)

{

head = newNode;

tail = newNode;

}

else

{

tail->next = newNode;

tail = newNode;

}

}

return head;

}

void ydalenie(Node\*& head)

{

if (head == NULL) // Если список пустой, ничего не делаем

{

cout << "Список пуст\n";

return;

}

int i = 1;

Node\* prev = nullptr;

Node\* curr = head;

while (curr != NULL)

{

if (i % 2 == 0) // Если элемент с чётным номером

{

Node\* temp = curr;

if (prev)

{

prev->next = curr->next; // Пропускаем текущий элемент, обновляя ссылки

}

else

{

head = curr->next;

}

curr = curr->next;

delete temp;

}

else

{

prev = curr; // Переходим к следующему элементу

curr = curr->next;

}

i++;

}

}

void pechat(Node\* head)

{

Node\* temp = head;

while (temp != NULL)

{

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

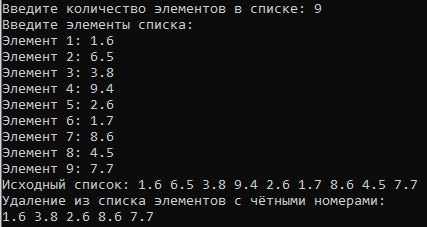


Рисунок 1 – Результат работы программы 1

Задание 2. Сформировать двунаправленный список. Тип информационного поля string. Добавить в список элементы с номерами 1, 3, 5 и т. д. Вывести полученный список.

Исходный код:

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;

struct Node {

string data;

Node\* prev;

Node\* next;

};

class DoublyLinkedList {

private:

Node\* head;

Node\* tail;

public:

DoublyLinkedList() : head(NULL), tail(NULL) {}

void insertAtPosition(string value, int position)

{

Node\* newNode = new Node();

newNode->data = value;

if (head == NULL) {

head = newNode;

tail = newNode;

newNode->prev = NULL;

newNode->next = NULL;

return;

}

if (position == 1) {

newNode->prev = NULL;

newNode->next = head;

head->prev = newNode;

head = newNode;

}

else {

Node\* temp = head;

int currentPosition = 1;

while (currentPosition < position - 1 && temp != NULL) {

temp = temp->next;

currentPosition++;

}

if (temp == NULL) {

cout << "Неправильный номер\n";

delete newNode;

return;

}

newNode->prev = temp;

newNode->next = temp->next;

if (temp->next != NULL) {

temp->next->prev = newNode;

}

temp->next = newNode;

if (temp == tail) {

tail = newNode;

}

}

}

void pechat() {

if (head == NULL) {

cout << "Список пуст\n";

return;

}

Node\* temp = head;

cout << "Ваш список: ";

while (temp != NULL) {

cout << "\"" << temp->data << "\" ";

temp = temp->next;

}

cout << endl;

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

DoublyLinkedList List;

string value, mas[100];

int position, size, size2;

cout << "Введите количество строк: ";

cin >> size;

cin.ignore();

for (int i = 1; i < size + 1; i++)

{

cout << "Введите строку для добавления: ";

getline(cin, value);

List.insertAtPosition(value, i);

mas[i] = value;

}

List.pechat();

cout << "После добавления элементов: " << endl;

size2 = size + 1;

for (int i = 1; i < size + 1; i++)

{

if (i % 2 == 1)

{

List.insertAtPosition(mas[i], size2);

size2++;

}

}

List.pechat();

return 0;

}

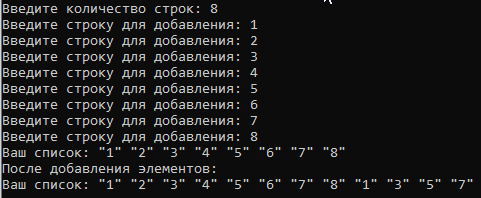


Рисунок 2 – Результат работы программы 2

Задание 3. Сформировать идеально сбалансированное бинарное дерево. Тип информационного поля int. Найти максимальный элемент в дереве. Вывести полученное дерево.

Исходный код:

#include <iostream>

#include <algorithm>

using namespace std;

// Структура для узла дерева

struct Node {

int data;

Node\* left;

Node\* right;

};

// Функция для создания нового узла дерева

Node\* Create\_Node(int data) {

Node\* newNode = new Node();

if (!newNode) {

cout << "Ошибка при выделении памяти!" << endl;

return NULL;

}

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

// Функция для построения идеально сбалансированного бинарного дерева

Node\* Build\_Balanced\_Tree(int arr[], int start, int end) {

// Проверка базового случая

if (start > end) {

return NULL;

}

// Находим середину массива и делаем её корнем нового узла дерева

int mid = (start + end) / 2;

Node\* root = Create\_Node(arr[mid]);

// Рекурсивно строим левое и правое поддерево

root->left = Build\_Balanced\_Tree(arr, start, mid - 1);

root->right = Build\_Balanced\_Tree(arr, mid + 1, end);

return root;

}

// Функция для печати дерева (обход в инфиксном порядке)

void Print\_Tree(Node\* root) {

if (root == NULL) {

return;

}

Print\_Tree(root->left);

cout << root->data << " ";

Print\_Tree(root->right);

}

int main()

{

setlocale(0, "");

int n;

int max = INT\_MIN;

cout << "Введите количество элементов: ";

cin >> n;

int\* arr = new int[n];

cout << "Введите элементы: " << endl;

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

cin >> arr[i];

if (max < arr[i])

{

max = arr[i];

}

}

sort(arr, arr + n);

Node\* root = Build\_Balanced\_Tree(arr, 0, n - 1);

cout << "Идеально сбалансированное бинарное дерево: ";

Print\_Tree(root);

cout << "\nМаксимальный элемент в дереве: " << max << "\n";

}

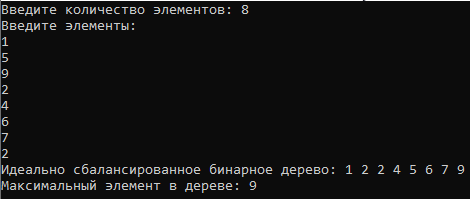


Рисунок 3 – Результат работы программы 3

Вывод: Получены практические навыки работы с однонаправленными, двунаправленными списками и деревьями.