Міністерство освіти і науки України

Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Звіт

з лабораторної роботи №8 з дисципліни

«Програмування - 2. Структури даних та алгоритми»

**«ДОСЛІДЖЕННЯ ПОБУДОВИ ТА ВИКОРИСТАННЯ СТРУКТУР ДАНИХ»**

Виконав студент

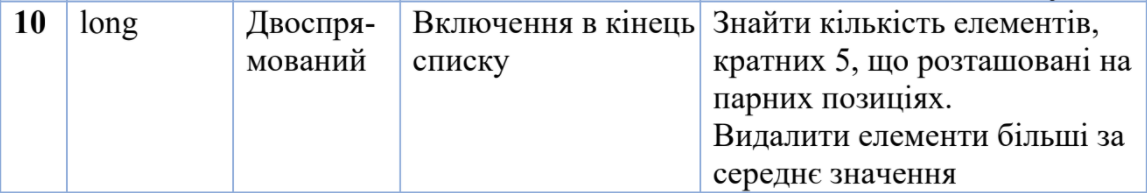
**ІС-01 Кубай Дмитро**

Перевірив викладач

**Проскура С. Л.**

**Київ – 2021**

1. **Завдання:**



1. **Код програми:**

**C++**

**Source.cpp**

#include "head.h"

Node\* MakeNode(long value);

void AddLast(Node\* head, long value);

int FindPairFive(Node\* head);

Node\* DeleteBiggerElements(Node\* head);

void PrintNodes(Node\* head);

int main()

{

Node\* head = MakeNode(11);

AddLast(head, 10);

AddLast(head, 1);

AddLast(head, 20);

AddLast(head, 1);

AddLast(head, 1);

AddLast(head, 11);

PrintNodes(head);

int count = FindPairFive(head);

cout << "Number of pair elements, that multiplies five: " << count << endl;

head = DeleteBiggerElements(head);

PrintNodes(head);

}

Node\* MakeNode(long value)

{

Node\* NewNode = new Node();

NewNode->value = value;

NewNode->previous = nullptr;

NewNode->next = nullptr;

NewNode->count = 1;

return NewNode;

}

void AddLast(Node\* head, long value)

{

Node \* q = head;

while (q->next != nullptr)

{

q = q->next;

//cout << q->value << endl;

}

Node \*NewNode = new Node();

NewNode->next = nullptr;

NewNode->previous = q;

NewNode->value = value;

NewNode->count = q->count + 1;

q->next = NewNode;

}

int FindPairFive(Node\* head)

{

int counter = 0;

Node\* q = head;

if (q->count % 2 == 0 && q->value % 5 == 0) counter++;

while (q->next != nullptr)

{

q = q->next;

if (q->count % 2 == 0 && q->value % 5 == 0) counter++;

}

return counter;

}

Node\* DeleteBiggerElements(Node\* head)

{

Node\* q = head;

long avereage;

int counter = 0;

avereage = q-> value;

counter++;

while (q->next != nullptr)

{

q = q->next;

avereage += q->value;

counter++;

}

avereage /= counter;

cout << "Avereage: " << avereage << endl;

q = head;

while (head->value > avereage)

{

head = q->next;

delete q;

q = head;

q->count--;

q->previous = nullptr;

while (q->next != nullptr)

{

q = q->next;

q->count--;

}

q = head;

}

while (q->next != nullptr)

{

q = q->next;

if (q->value > avereage)

{

if (q->next != nullptr)

{

Node\* left = q->previous;

Node\* right = q->next;

left->next = right;

right->previous = left;

Node\* p = q;

q = right;

q->count--;

while (q->next != nullptr)

{

q = q->next;

q->count--;

}

q = right;

delete p;

}

else

{

Node\* left = q->previous;

left->next = nullptr;

delete q;

q = left;

}

}

}

return head;

}

void PrintNodes(Node\* head)

{

Node\* q = head;

cout << "N" << q->count << ": " << q->value << endl;

while (q->next != nullptr)

{

q = q->next;

cout << "N" << q->count << ": " << q->value << endl;

}

}

**head.h**

#pragma once

#include <iostream>

using namespace std;

struct Node

{

public:

Node\* previous;

Node\* next;

long value;

int count;

};

**C#**

**Program.cs**

using System;

using NodesClass;

namespace Lab08sharpV1.\_0

{

class Program

{

static void Main(string[] args)

{

MyNodes list = new MyNodes(11);

list.AddLast(10);

list.AddLast(1);

list.AddLast(20);

list.AddLast(1);

list.AddLast(1);

list.AddLast(11);

PrintNodes(list);

int count = list.FindPairFive();

Console.WriteLine($"Number of pair elements, that multiplies five: {count}");

list.DeleteBiggerElements();

PrintNodes(list);

}

public static void PrintNodes(MyNodes list)

{

long[] arr = list.GetNodes();

for (int i = 0; i < arr.Length; i++)

{

Console.WriteLine($"N{i + 1}: {arr[i]}");

}

}

}

}

**MyNodes.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace NodesClass

{

public class MyNodes

{

private Node[] list;

private int length;

public MyNodes(long value)

{

list = new Node[1] { new Node(value)} ;

length = 1;

}

public void AddLast(long value)

{

length++;

Node[] NewArr = new Node[length];

for (int i = 0; i < length - 1; i ++)

{

NewArr[i] = list[i];

}

NewArr[length - 1] = new Node(value);

list = NewArr;

}

public long[] GetNodes()

{

long[] NewArr = new long[length];

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

{

NewArr[i] = list[i].value;

}

return NewArr;

}

public int FindPairFive()

{

int counter = 0;

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

{

if (list[i].value % 5 == 0 && list[i].value % 2 == 0)

counter++;

}

return counter;

}

public void DeleteBiggerElements()

{

long avereage = 0;

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

{

avereage += list[i].value;

}

avereage /= length;

int count = 1;

while (count != 0)

{

count = 0;

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

{

if (list[i].value > avereage)

{

Node[] NewArr = new Node[length - 1];

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

NewArr[j] = list[j];

for (int j = i; j < length - 1; j++)

NewArr[j] = list[j + 1];

list = NewArr;

length--;

}

}

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

if (list[i].value > avereage) count++;

}

}

}

}

**Node.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace NodesClass

{

internal struct Node

{

public long value;

internal Node(long value1)

{

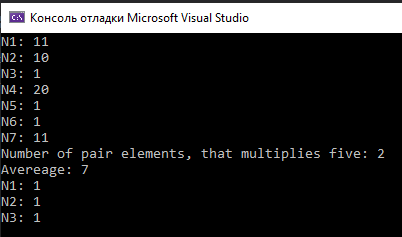
value = value1;

}

}

}

1. **Виконання програми**

****

1. **Висновок**

У даній роботі було завдання дослідити побудову та застосування структура данних у мовах програмування С++ та С# і написати програму, яка реалізує:

Зв’язаний лінійний список – це структура, всі елементи якої зв’язані

один з іншим за допомогою посилань (рис.3). На перший елемент списку

посилається окрема змінна, яка містить його адресу. Кожний елемент списку

містить покажчик з адресою наступного елементу. Останній елемент списку

містить посилання на null.

Моя програма створює зв’язний список і функції, які можуть знайти кількість елементів, кратних 5, що розташовані на парних позиціях, видалити елементи більші за середнє значення.