

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

**Отчет по выполнению практического задания номер шесть**

**Тема: двунаправленный список**

**Дисциплина: Структуры и алгоритмы обработки данных**

Выполнил студент Михайлюк Д. С.

Группы ИНБО-07-21

Москва 2021

**СОДЕРЖАНИЕ**

[1 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc100793787)

[2 Определение списка операций над списком, которые выявлены в процессе исследования задач дополнительного задания 4](#_Toc100793788)

[2.1 Проверка списка на пустоту 4](#_Toc100793789)

[2.2 Метод добавления узла в начало 5](#_Toc100793790)

[2.3 Метод добавления узла в конец 7](#_Toc100793791)

[2.4 Метод вывода списка на экран 9](#_Toc100793792)

[2.5 Метод поиска узлов по значению 11](#_Toc100793793)

[2.6 Метод удаления первого узла 12](#_Toc100793794)

[2.7 Метод удаления последнего узла 14](#_Toc100793795)

[2.8 Метод удаления узла по его значению 16](#_Toc100793796)

[2.9 Метод перемещения первых n-узлов в конец списка 19](#_Toc100793797)

[2.10 Метод удаления предпоследнего узла 21](#_Toc100793798)

[2.11 Метод обмена местами 2х узлов 23](#_Toc100793799)

[2.12 Метод обмена местами узлов с наибольшим и наименьшим значениями 25](#_Toc100793800)

[3 КОД ВСЕЙ ПРОГРАММЫ 27](#_Toc100793801)

[4 ВЫВОДЫ 32](#_Toc100793802)

[5 ОТВЕТЫ НА ВОПРОСЫ 33](#_Toc100793803)

# ПОСТАНОВКА ЗАДАЧИ

Разработать многомодульную программу, которая демонстрирует выполнение всех операций, определенных вариантом, над линейным двунаправленным динамическим списком.

Персональный вариант - №2:

Структура данных:

-Номер телефона (из 7 цифр)

-Время разговора (целое число)

-Номер телефона вызываемого абонента.

Методы персонального варианта:

-Добавить новый узел в список, упорядочивая узлы по первым четырем цифрам телефона в порядке возрастания

-Удалить последний узел с заданным значением телефона

-Подсчитать суммарное время разговора с заданного телефона.

# Определение списка операций над списком, которые выявлены в процессе исследования задач дополнительного задания

## структура узла двунаправленного списка в соответствии с вариантом

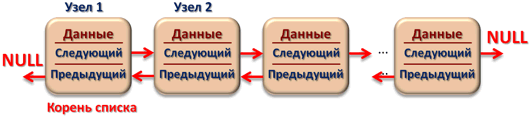


Рис. 1 – структура двунаправленного списка

В соответствии с индивидуальным вариантом №2 в графе “данные” каждого узла будут содержаться данные о номере телефона (long long поле), время разговора (int поле) и номер телефона вызываемого абонента (long long поле).

## Проверка списка на пустоту

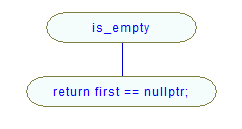


Рисунок 2 – Блок схема для метода

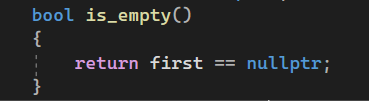


Рисунок 3 – Код метода

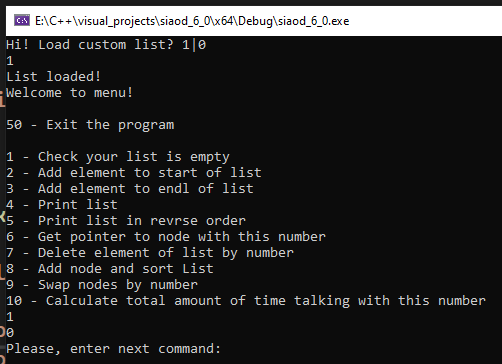


Рисунок 4 – Тест метода

Метод проверяет список на пустоту.

## Метод добавления узла в начало

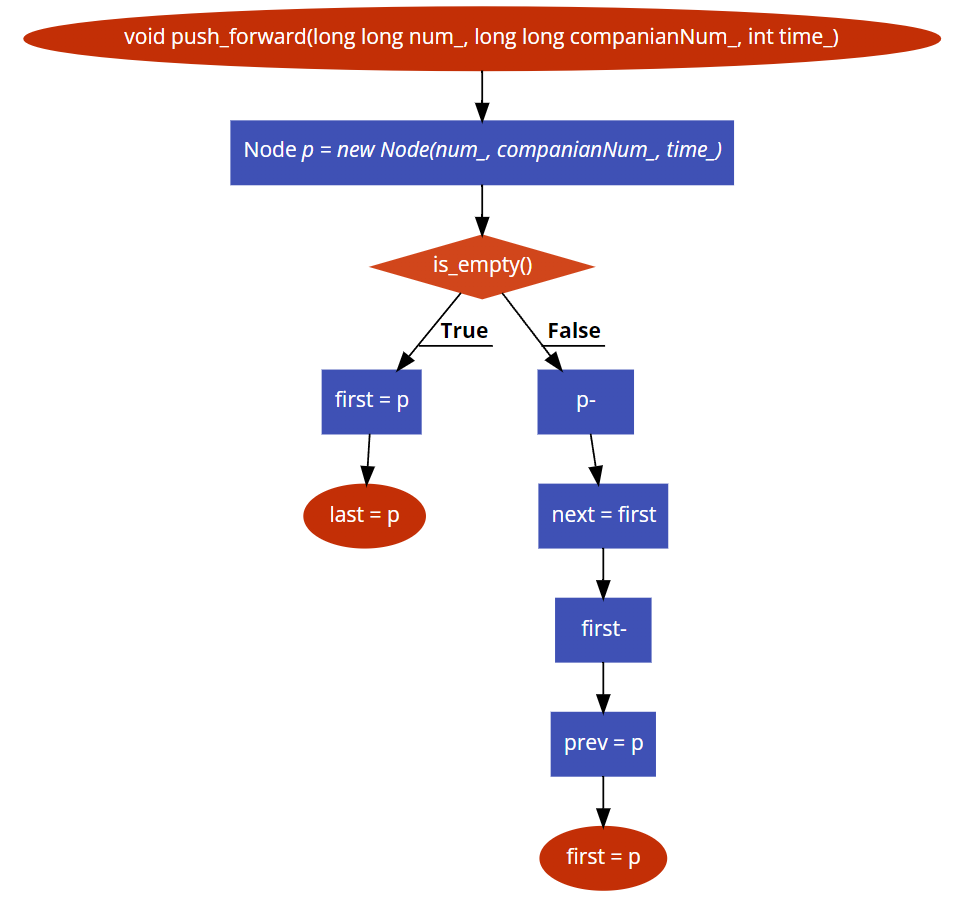


Рисунок 5 – Блок схема для метода

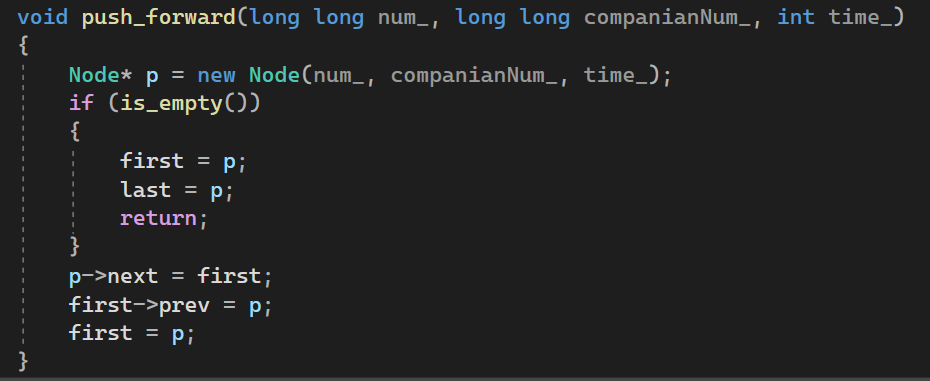


Рисунок 6 – Код метода

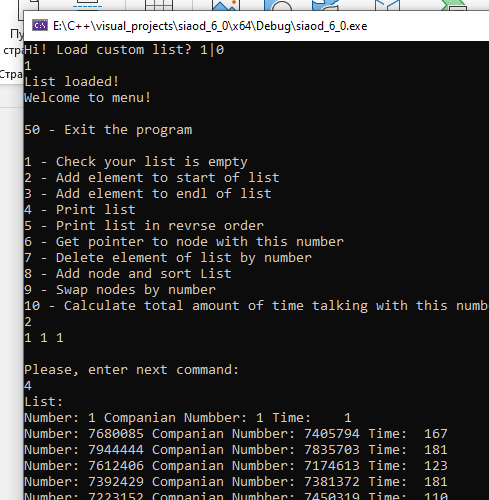


Рисунок 7 – Тест метода

Метод добавляет элемент в начало списка.

## Метод добавления узла в конец

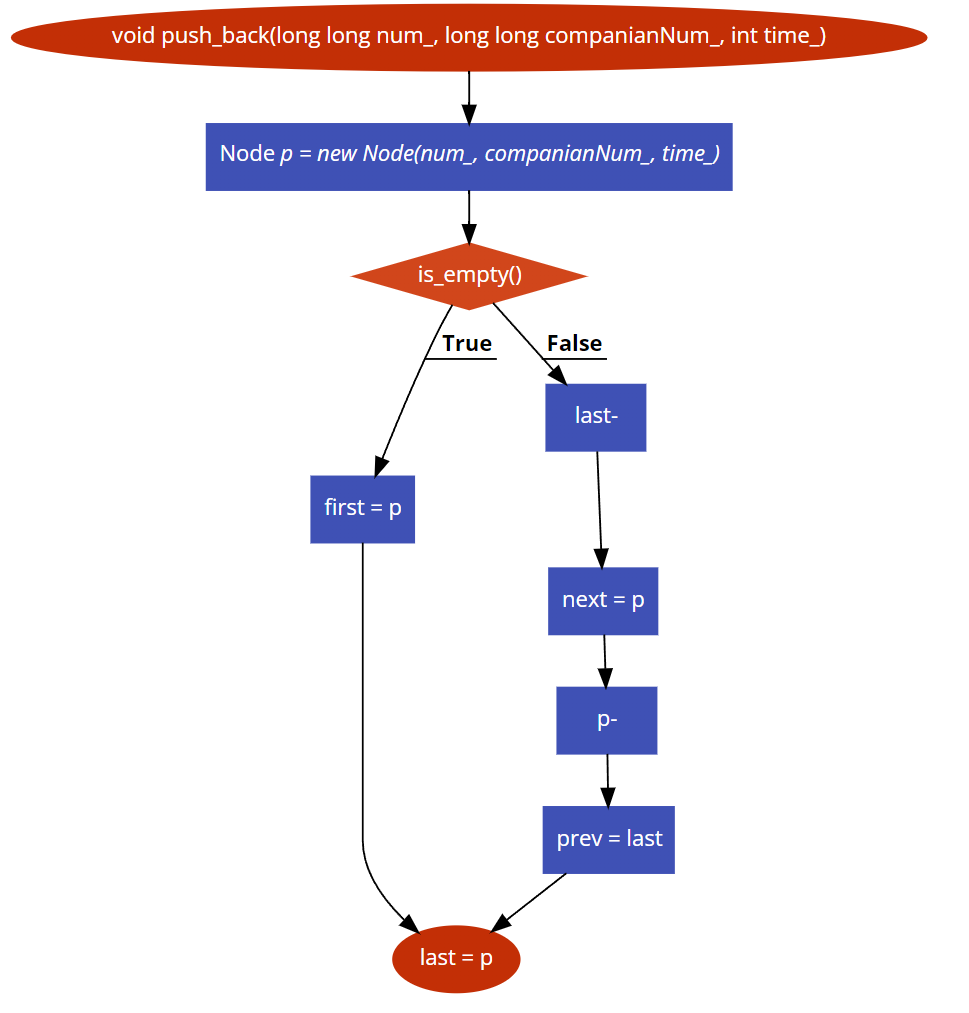


Рисунок 8 – Блок схема для метода

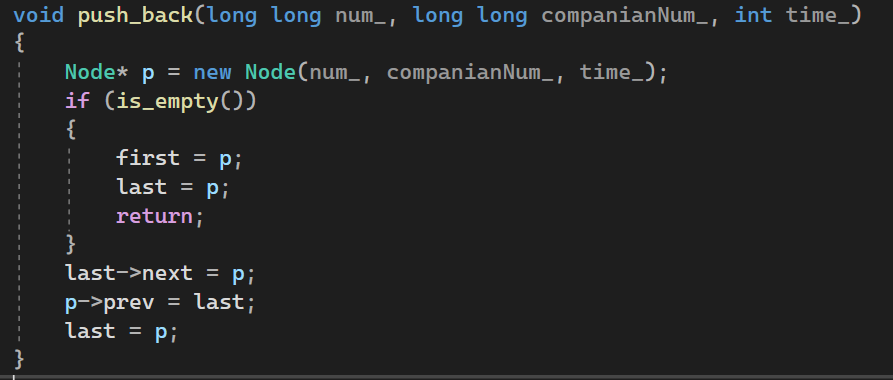


Рисунок 9 – Код метода

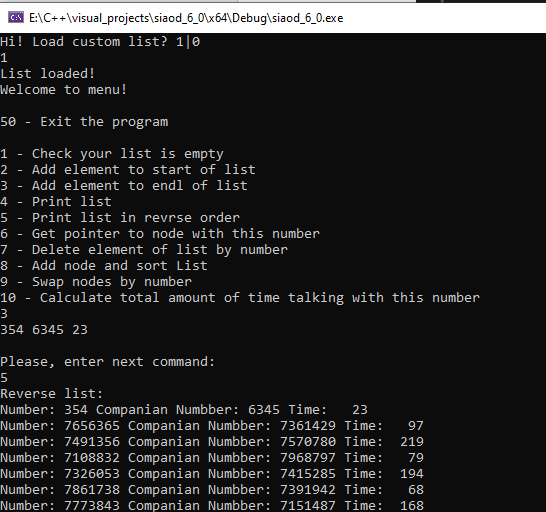


Рисунок 10 – Тест метода

Метод добавляет элемент в конец списка.

## Метод вывода списка на экран

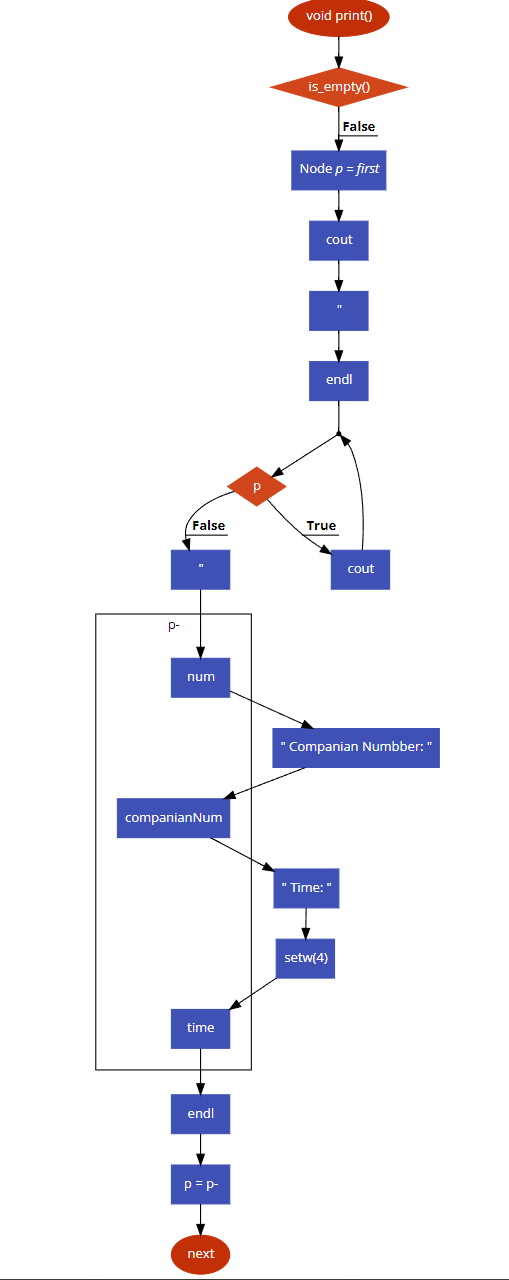


Рисунок 11 – Блок схема для метода

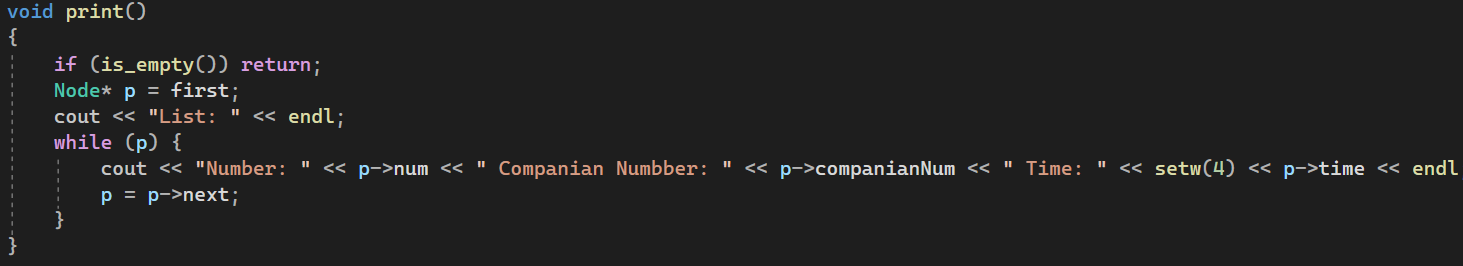


Рисунок 12 – Код метода

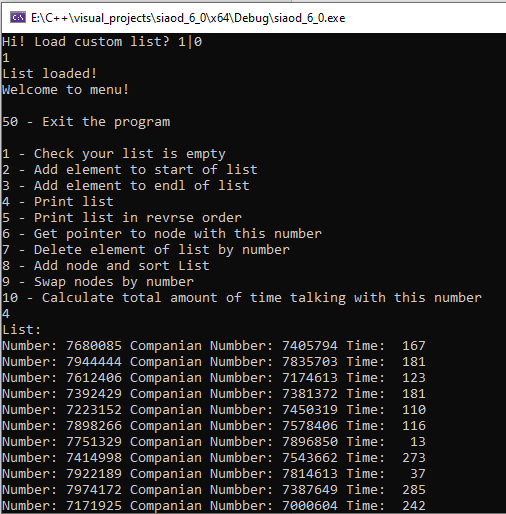


Рисунок 13 – Тест метода

Метод выводит поэлементно список на экран в прямом порядке.

## Метод вывода списка на экран в обратном порядке

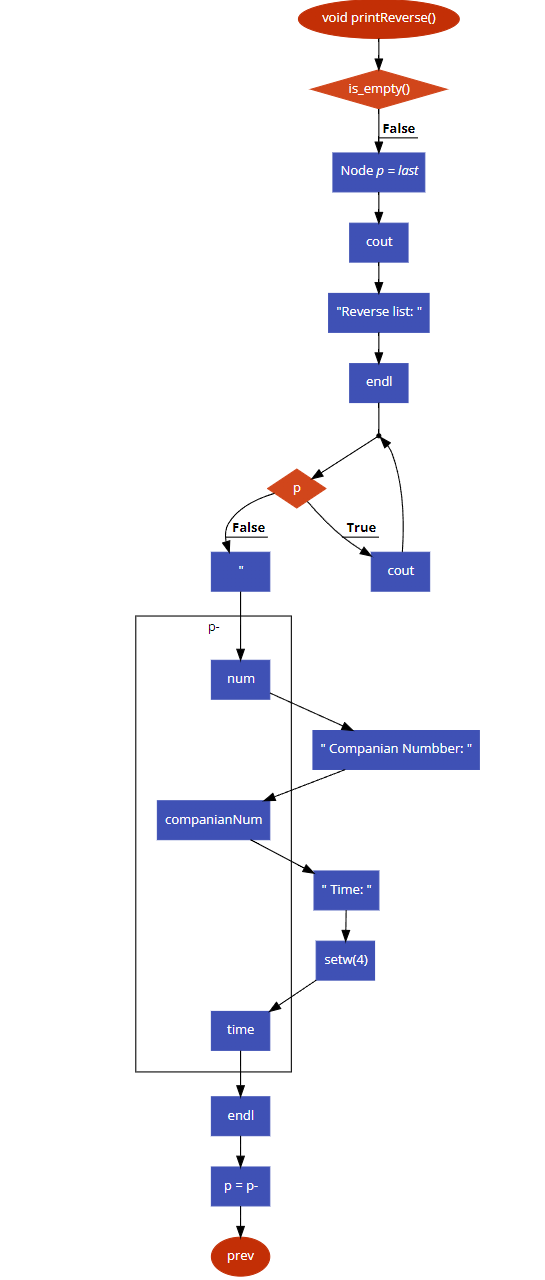


Рисунок 14 – Блок схема для метода

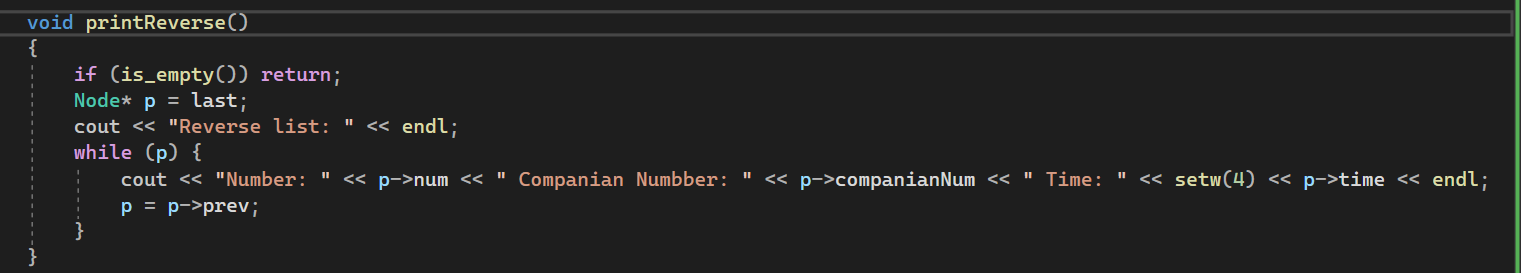


Рисунок 15 – Код метода

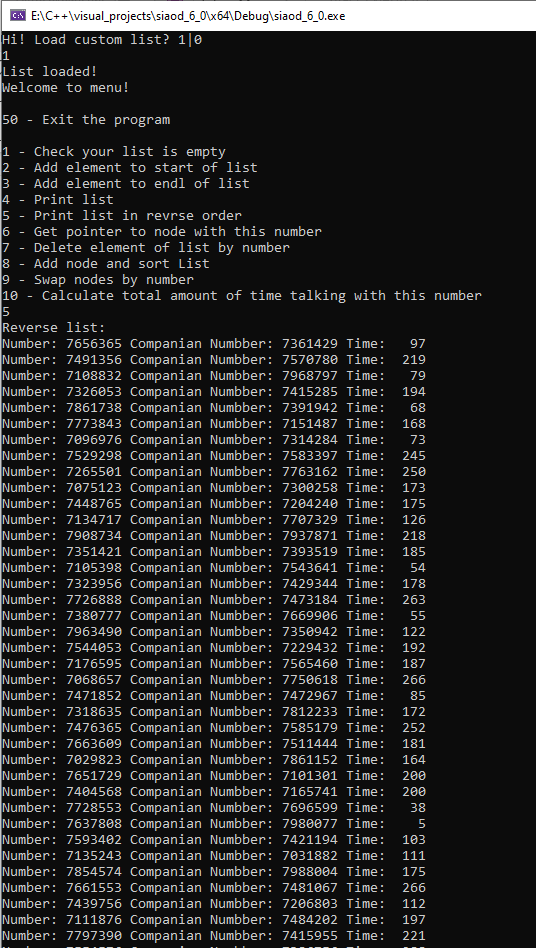


Рисунок 16 – Тест метода

Метод выводит поэлементно список на экран в прямом порядке.

## Метод поиска узлов по значению

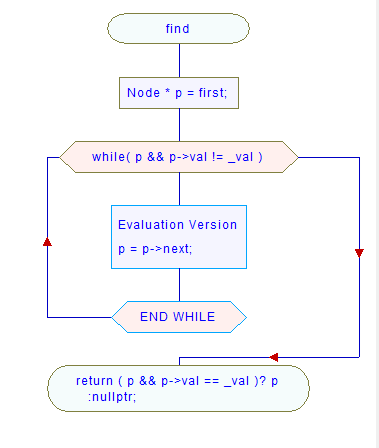


Рисунок 17 – Блок схема для метода

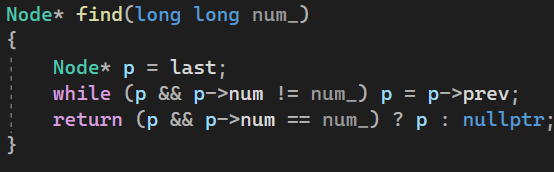


Рисунок 18 – Код метода

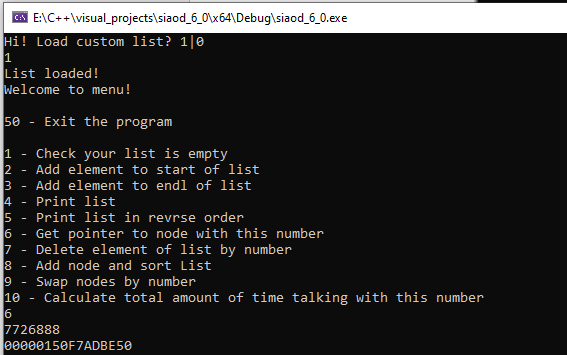


Рисунок 19 – Тест метода

Метод возвращает указатель на узел с введенным значением.

## Метод удаления узла по его значению

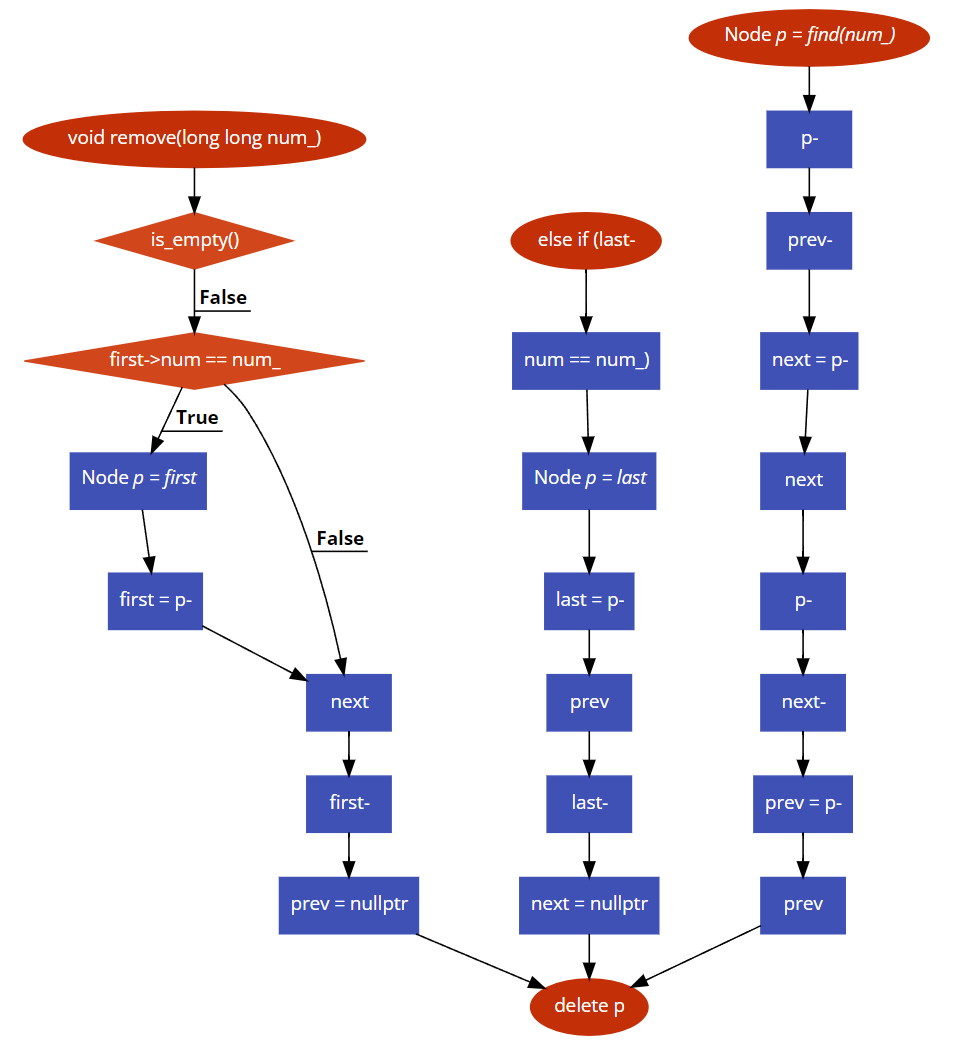


Рисунок 20 – Блок схема для метода

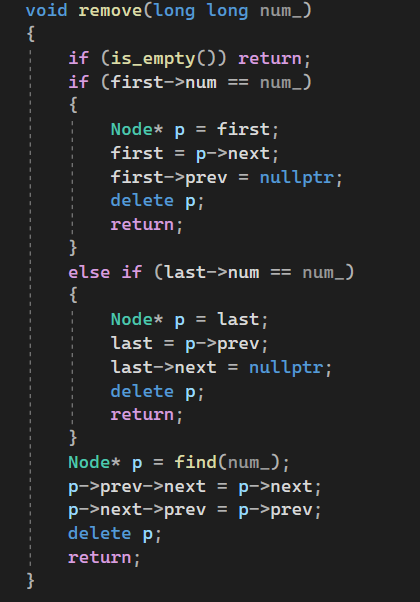


Рисунок 21 – Код метода

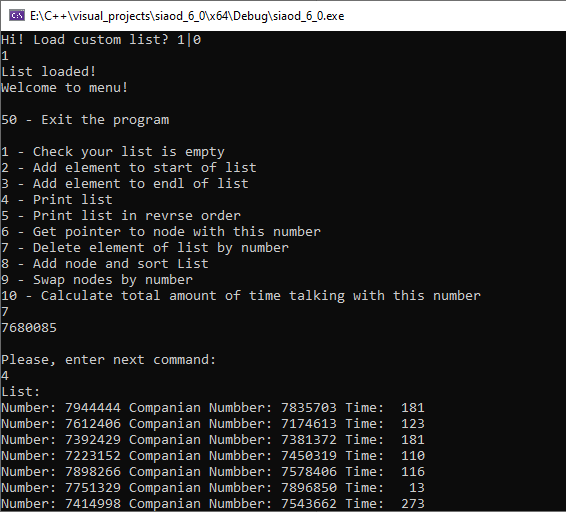


Рисунок 22 – Тест метода

Метод удаления узла по значению.

## Метод обмена местами 2х узлов по их значению

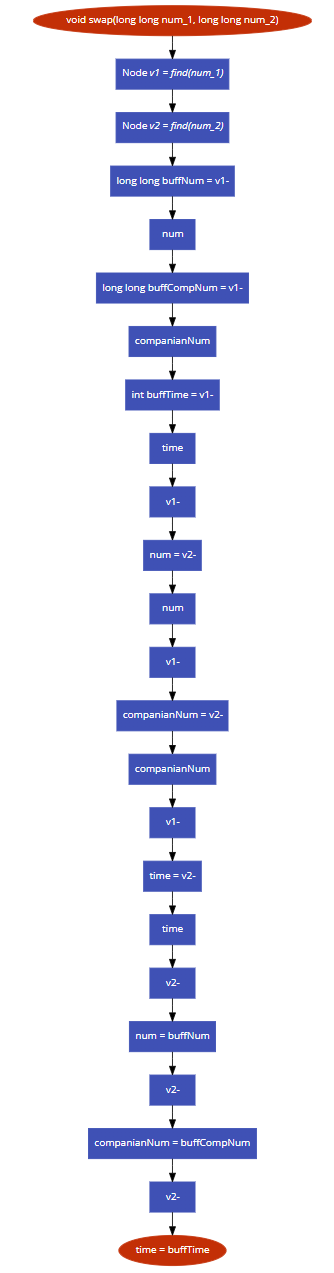


Рисунок 23 – Блок схема для метода

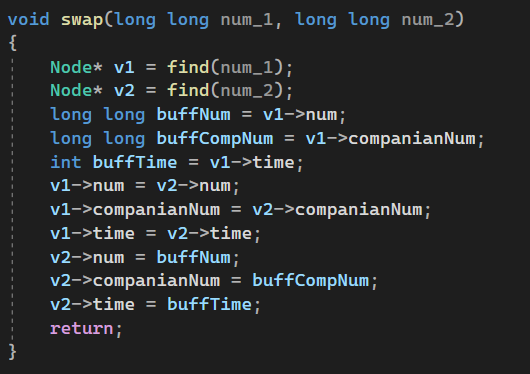


Рисунок 24 – Код метода

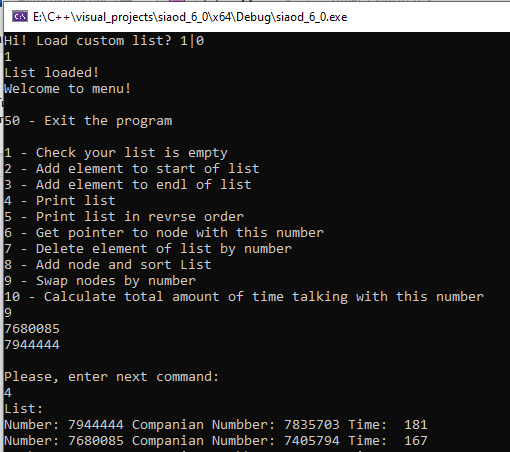


Рисунок 25 – Тест метода

Метод меняет местами 2 узла по значению.

## Метод подсчета суммы времени которое разговаривал абонент с указанным номером

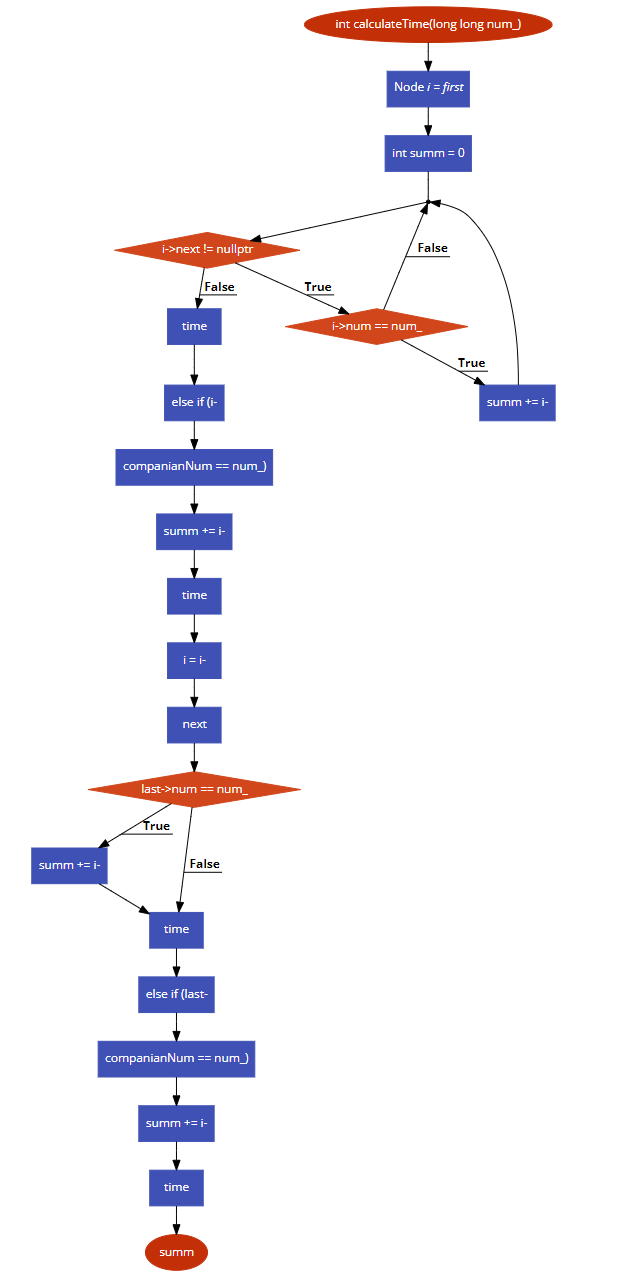


Рисунок 26 – Блок схема для метода

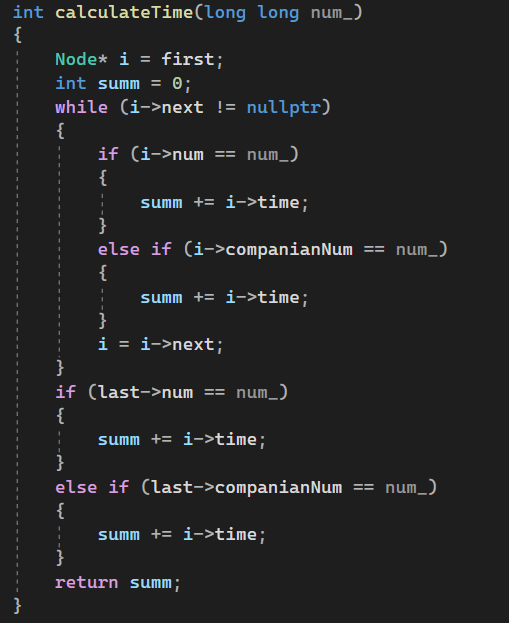


Рисунок 27 – Код метода

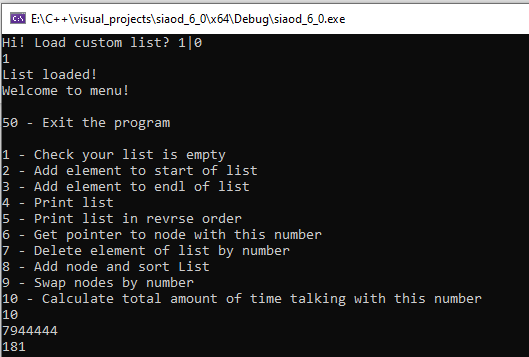


Рисунок 28 – Тест метода

Метод по указанному значению ищет сумму времени разговоров где звонили от лица этого номера и этому номеру.

## Метод обмена местами 2х узлов

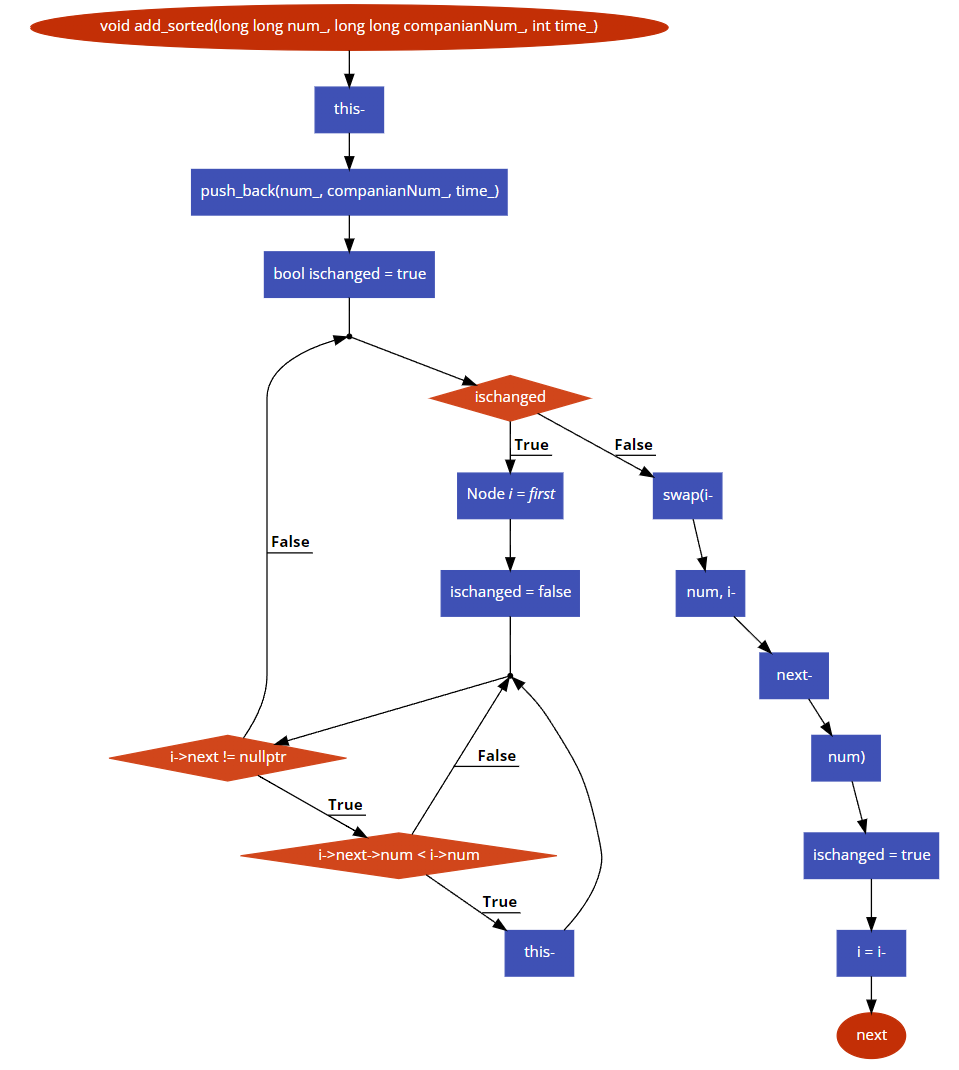


Рисунок 29 – Блок схема для метода

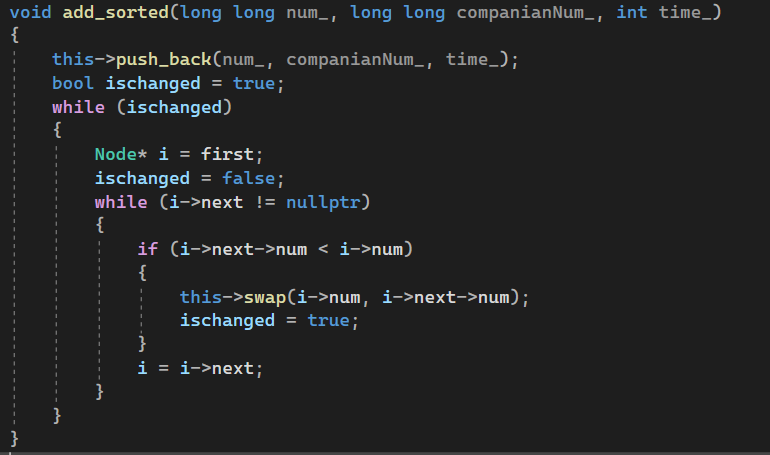


Рисунок 30 – Код метода

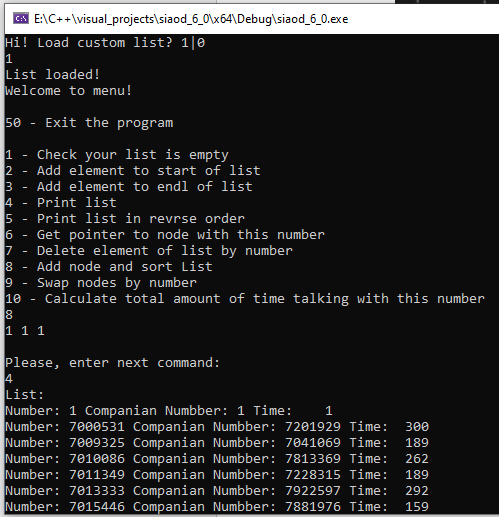


Рисунок 30 – Тест метода

Метод добавляет новый узел в список, упорядочивая узлы по первым четырем цифрам телефона в порядке возрастания.

# КОД ВСЕЙ ПРОГРАММЫ

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

using namespace std;

struct Node

{

long long num;

long long companianNum;

int time;

Node\* prev;

Node\* next;

Node(long long num\_, long long companianNum\_, int time\_) : num(num\_), companianNum(companianNum\_), time(time\_), next(nullptr), prev(nullptr) {}

};

struct list

{

Node\* first;

Node\* last;

list() : first(nullptr), last(nullptr) {}

bool is\_empty()

{

return first == nullptr;

}

void push\_forward(long long num\_, long long companianNum\_, int time\_)

{

Node\* p = new Node(num\_, companianNum\_, time\_);

if (is\_empty())

{

first = p;

last = p;

return;

}

p->next = first;

first->prev = p;

first = p;

}

void push\_back(long long num\_, long long companianNum\_, int time\_)

{

Node\* p = new Node(num\_, companianNum\_, time\_);

if (is\_empty())

{

first = p;

last = p;

return;

}

last->next = p;

p->prev = last;

last = p;

}

void print()

{

if (is\_empty()) return;

Node\* p = first;

cout << "List: " << endl;

while (p) {

cout << "Number: " << p->num << " Companian Numbber: " << p->companianNum << " Time: " << setw(4) << p->time << endl;

p = p->next;

}

}

void printReverse()

{

if (is\_empty()) return;

Node\* p = last;

cout << "Reverse list: " << endl;

while (p) {

cout << "Number: " << p->num << " Companian Numbber: " << p->companianNum << " Time: " << setw(4) << p->time << endl;

p = p->prev;

}

}

Node\* find(long long num\_)

{

Node\* p = last;

while (p && p->num != num\_) p = p->prev;

return (p && p->num == num\_) ? p : nullptr;

}

void remove(long long num\_)

{

if (is\_empty()) return;

if (first->num == num\_)

{

Node\* p = first;

first = p->next;

first->prev = nullptr;

delete p;

return;

}

else if (last->num == num\_)

{

Node\* p = last;

last = p->prev;

last->next = nullptr;

delete p;

return;

}

Node\* p = find(num\_);

p->prev->next = p->next;

p->next->prev = p->prev;

delete p;

return;

}

void swap(long long num\_1, long long num\_2)

{

Node\* v1 = find(num\_1);

Node\* v2 = find(num\_2);

long long buffNum = v1->num;

long long buffCompNum = v1->companianNum;

int buffTime = v1->time;

v1->num = v2->num;

v1->companianNum = v2->companianNum;

v1->time = v2->time;

v2->num = buffNum;

v2->companianNum = buffCompNum;

v2->time = buffTime;

return;

}

void add\_sorted(long long num\_, long long companianNum\_, int time\_)

{

this->push\_back(num\_, companianNum\_, time\_);

bool ischanged = true;

while (ischanged)

{

Node\* i = first;

ischanged = false;

while (i->next != nullptr)

{

if (i->next->num < i->num)

{

this->swap(i->num, i->next->num);

ischanged = true;

}

i = i->next;

}

}

}

int calculateTime(long long num\_)

{

Node\* i = first;

int summ = 0;

while (i->next != nullptr)

{

if (i->num == num\_)

{

summ += i->time;

}

else if (i->companianNum == num\_)

{

summ += i->time;

}

i = i->next;

}

if (last->num == num\_)

{

summ += i->time;

}

else if (last->companianNum == num\_)

{

summ += i->time;

}

return summ;

}

};

int main()

{

list l;

cout << "Hi! Load custom list? 1|0\n";

int af, i;

long long a, b;

cin >> af;

if (af)

{

ifstream fe("List.txt");

string st;

while (fe >> a >> b >> i)

{

l.push\_back(a, b, i);

}

cout << "List loaded! \n";

}

cout << "Welcome to menu!\n \n50 - Exit the program\n\n1 - Check your list is empty \n2 - Add element to start of list \n3 - Add element to endl of list\n4 - Print list\n5 - Print list in revrse order\n";

cout << "6 - Get pointer to node with this number\n7 - Delete element of list by number\n8 - Add node and sort List\n9 - Swap nodes by number\n10 - Calculate total amount of time talking with this number\n";

while (af != 50)

{

cin >> af;

switch (af)

{

case 0:

{

break;

}

case 1:

{

cout << l.is\_empty();

cout << "\nPlease, enter next command:\n";

break;

}

case 2:

{

cin >> a >> b >> i;

l.push\_forward(a, b, i);

cout << "\nPlease, enter next command:\n";

break;

}

case 3:

{

cin >> a >> b >> i;

l.push\_back(a, b, i);

cout << "\nPlease, enter next command:\n";

break;

}

case 4:

{

l.print();

cout << "\nPlease, enter next command:\n";

break;

}

case 5:

{

l.printReverse();

cout << "\nPlease, enter next command:\n";

break;

}

case 6:

{

cin >> a;

cout << l.find(a);

cout << "\nPlease, enter next command:\n";

break;

}

case 7:

{

cin >> a;

l.remove(a);

cout << "\nPlease, enter next command:\n";

break;

}

case 8:

{

cin >> a >> b >> i;

l.add\_sorted(a, b, i);

cout << "\nPlease, enter next command:\n";

break;

}

case 9:

{

cin >> a >> b;

l.swap(a, b);

cout << "\nPlease, enter next command:\n";

break;

}

case 10:

{

cin >> a;

cout << l.calculateTime(a);

cout << "\nPlease, enter next command:\n";

break;

}

default:

break;

}

}

}

# Сложность алгоритма первой дополнительной операции

Асимптотическая сложность алгоритма сортировки пузырьком, используемый в операции - f(n) = O(n^2). Кроме того, затраты по памяти программы носит линейный характер: O(n) = n.

График 1 – сложность первой дополнительной операции

# ВЫВОДЫ

В ходе выполнения практического задания были получены навыки по реализации двунаправленного линейного списка. Выяснилось, как именно работает данная линейная структура хранения многоэлементных последовательностей. Кроме того, были разработаны вспомогательные функции, увеличивающие функционал данной структуры.