

МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**

Отчет по выполнению практического задания № 5

**Тема:**

«Однонаправленный динамический список»

Дисциплина: «Алгоритмы внешних сортировок»

Выполнил студент: Моисенко М. О.

Фамилия И.О.

Группа: ИКБО-00-22

Номер группы

Москва – 2023

СОДЕРЖАНИЕ

[1 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc132854649)

[1.1 Условие задания 3](#_Toc132854650)

[2 РЕШЕНИЕ ЗАДАНИЯ 3](#_Toc132854651)

[2.1 Реализация односвязного списка и его основных функций 3](#_Toc132854652)

[2.2 Разработка и реализация функций индивидуального варианта 7](#_Toc132854653)

[2.3 Тестирование работы алгоритма 8](#_Toc132854654)

[3 ВЫВОД 9](#_Toc132854655)

[4 СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 9](#_Toc132854656)

# ПОСТАНОВКА ЗАДАЧИ

Цель:получить знания и практические навыки управления динамическим однонаправленным списком

## Условие задания

Разработать реализацию односвязного списка средствами языка C++ с информационной частью узла списка согласно индивидуальному варианту. Реализовать три функции для работы над односвязным списком согласно индивидуальному варианту.

Индивидуальный вариант под номеров 8.

Тип информационной части узла – float.

Функции:

1. Разработать функцию, которая переносит первые k узлов в конец списка.

2. Разработать функцию, которая переставляет местами узлы с

максимальным и минимальным значениями.

3. Разработать функцию, которая удаляет предпоследний узел

списка.

# РЕШЕНИЕ ЗАДАНИЯ

## Реализация односвязного списка и его основных функций



Рисунок 1 – Схема узла однонаправленного списка

Листинг 1 – Реализация узла однонаправленного списка

struct Usel
{
 float kost;
 Usel\* next;
 Usel(float a): kost(a), next(nullptr) {}
};



void add(float val)
{
 Usel\* node = new Usel(val);
 if (is\_empty())
 {
 first = node;
 last = node;
 return;
 }
 node->next = first;
 first = node;
};

void print()
{
 if (is\_empty())
 {
 cout << "Nothing is here, absolutely nothing" << endl; return;
 }
 Usel\* node = first;
 cout << "Here is your list: ";
 while (node != nullptr)
 {
 cout << node->kost << " ";
 node = node->next;
 }
 cout << endl;
}



void remove(float val)
{
 cout << "Removing process is started..." << endl;
 if (is\_empty())
 {
 return;
 };
 if (first->kost == val)
 {
 Usel\* node = first;
 first = node->next;
 delete node;
 return;
 };
 if (first->kost == val)
 {
 Usel\* node = first;

 while (node->next != last)
 node = node->next;

 node->next = nullptr;
 delete last;
 last = node;
 return;
 };
 Usel\* thes = first;
 Usel\* that = first->next;

 while (that && (that->kost != val))
 {
 thes = thes->next;
 that = that->next;
 }
 if (!that)
 {
 cout << "Nothing is here" << endl;
 return;
 }

 thes->next = that->next;
 delete that;
 cout << "Process is done, now money" << endl;
};

Listik manuCreate(int len)
{
 float val;
 Listik klen;
 cout << "Manual entry" << endl;
 for (int i = 0; i < len; i++)
 {
 cout << "Enter " << len - i << " value:";
 cin >> val;
 klen.add(val);
 }
 klen.print();
 cout << "Okay, it's done" << endl;
 return klen;
}

Listik autoCreate(int len)
{
 float val;
 Listik klen;
 cout << "Auto entry" << endl;
 srand(time(NULL));
 for (int i = 0; i < len; i++)
 {
 klen.add(rand());
 }
 klen.print();
 cout << "Okay, it's done" << endl;
 return klen;
}

## Разработка и реализация функций индивидуального варианта

void peres(int k, Listik &l)
{
 int i = 1;
 Usel\* node = l.first;

 bool f;
 while (i < k)
 {
 f = true;
 if (node == nullptr)
 {
 cout << "ERROR" << endl;
 return;
 }
 node = node->next;
 i++;
 }
 if (f)
 {
 l.last->next = l.first;
 l.first = node->next;
 node->next = nullptr;
 }
}

void soft\_detention(Listik &l)
{
 Usel\* node = l.first;
 if (l.is\_empty() || (node->next == nullptr))
 {
 cout << "ERROR: Nothing is here" << endl;
 return;
 }
 Usel\* n\_node = node->next;
 if (n\_node->next == nullptr)
 {
 l.first = n\_node;
 delete node;
 return;
 }
 while (n\_node->next != l.last)
 {
 node = n\_node;
 n\_node = n\_node->next;
 }
 node->next=l.last;
 delete n\_node;
}

void big\_small(Listik &l)
{
 Usel\* node = l.first;
 if (l.is\_empty())
 {
 cout << "ERROR: Nothing is here" << endl;
 return;
 }

 Usel\* prebig = l.first;
 Usel\* presmall = l.first;
 Usel\* vrem1;
 Usel\* vrem2;
 float min = node->kost;
 float max = node->kost;

 while (node->next != nullptr)
 {
 if (node->next->kost > max)
 {
 max = node->next->kost;
 prebig = node;
 }
 if (node->next->kost < min)
 {
 min = node->next->kost;
 presmall = node;
 }
 node = node->next;
 }

 if (max == min)
 {return;}

 if (max == l.first->kost)
 {
 swap(l.first, presmall->next);
 swap(l.first->next, presmall->next->next);
 }
 else if (min == l.first->kost)
 {
 swap(l.first, prebig->next);
 swap(l.first->next, prebig->next->next);
 }
 else
 {
 swap(presmall->next, prebig->next);
 swap(presmall->next->next, prebig->next->next);
 }
}



Рисунок 1 – База данных после работы алгоритма

## Тестирование работы алгоритма

Результаты тестирования функций программы при различных входных данных представлены в таблице 1.

Таблица 1 - Результаты тестирования функций и методов программы

|  |  |  |  |
| --- | --- | --- | --- |
| Функция/метод | Входные данные | Ожидаемый результат | Полученныйрезультат |
| bool is\_empty() | 1 | false | false |
| [] | true | true |
| void add(float value) | 3 | 3 3 | 3, 3, 3 | 3, 3, 3 |
| 0 |  | 0 | 0 |
| void print() | 1, 1, 1 | 1, 1, 1 | 1, 1, 1 |
| [] | Nothing is here, absolutely nothing | Nothing is here, absolutely nothing |
| void remove(float value) | 6 | 2, 6, 3 | 2, 6 | 2, 6 |
| 7 | 2, 6, 3 | Nothing is here | Nothing is here! |
| Listik manuCreate(int len) | 3 | 1, 2, 3 | 1, 2, 3 | 1, 2, 3 |
| 0, [] | [] | [] |
| Listik autoCreate(int len) | 3 | a, b, c | 1376 30948 30475 |
| 0 | [] | [] |
| void peres(int k, Listik &l) | 2 | 1, 2, 3 | 3, 1, 2 | 3, 1, 2 |
| 3 | 1, 1 | ERROR | ERROR |
| void soft\_detention(Listik &l) | 4, 3, 6 | 4, 6 | 4, 6 |
| [] | ERROR: Nothing is here | ERROR: Nothing is here |
| void big\_small(Listik &l) | 11, 1, 4, 1 | 1, 11, 4, 1 | 1, 11, 4, 1 |
| 16, 17 | 17, 16 | [5, 1, 5, 1, 2] |

По результатам тестирования можно сделать вывод, что все функции и методы являются рабочими.

# ВЫВОД

В ходе выполнения работы были получены знания и практические навыки управления динамическим однонаправленным списком. Были реализованы все необходимые методы для работы с однонаправленным списком, а также функции согласно индивидуальному варианту.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Структуры данных и проектирование программ : Пер. с англ. / Р. Круз. — М.: БИНОМ. Лаборатория знаний, 2017. — 766 с.

2. Полный справочник по C++ : Пер. с англ. / Г. Шилдт. — М.: ООО "И.Д.Вильямс", 2016. — 796 с.: ил. — Предм. указ.: с. 787-796