|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | | | | | |
| Федеральное государственное автономное образовательное учреждение  высшего образования | | | | | | | | | | | | |
| **Дальневосточный федеральный университет** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **Департамент программной инженерии и искусственного интеллекта** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **О Т Ч Е Т** | | | | | | | | | | | | |
| по лабораторной работе №1.1  дисциплина «Фундаментальные структуры данных и алгоритмы» | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | |  |  | Студент гр. Б9122-09.03.04прогин | | | |
|  |  |  | | А.О. Лопатин | |
|  | | | | | | |  |  | (подпись) | |  | |
|  | | | | | | | | | | | | |
|  | | | | | | |  |  | Руководитель | | |  |
|  | | | | | | |  |  | ст. преподаватель | | | |
|  | | | |  |  | |  |  |  |  | О.А. Крестникова | |
|  | | | |  |  | |  |  | (подпись) |  | (И.О. Фамилия) | |
|  |  |  |  | | |  |  |  |  | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| г. Владивосток | | | | | | | | | | | | |
| 2023 | | | | | | | | | | | | |

**1 Неформальная постановка задачи**

Реализация однонаправленного списка, упорядоченного по возрастанию по вхождении; удаление всех вхождений заданного; реализация разности двух списков.

**2 Описание типа + спецификация подпрограмм + тесты**

**Function:**

**bool is\_empty();**

Проверяет есть ли первый элемент в списке и возвращает ***false***, если указатель на первый элемент равен nullptr, иначе true. Входные данные: список, значение. Выходные данные: true/false.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Проверка первого элемента в пустом списке | List L | false |
| Проверка первого элемента в непустом списке | List L | true |

**procedure :**

**void push\_back(int \_val);**

Вставляет элемент в конец однонаправленного списка.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Вставка числа | Value = 1, list L | List L = {1} |
| Вставка числа | Value = 2, list L = {1,3} | List L = {1, 2, 3} |

**procedure :**

**void print();**

Выводит список в консоль

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Печать заполненного списка | List L = {1,2,3,4,5} | 1,2,3,4,5 |
| Печать пустого списка | List L = {} | “List is empty” |

**procedure :**

**void sort();**

Сортирует элементы списка по возрастанию

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Выполняем вставку и сортируем по возрастанию | List L = {4, 5, 2, 1} | 1, 2, 4, 5 |

**3 Текст программы**

#include <iostream>

//----------------------------------------------

struct Node

{

int val;

Node\* next;

Node(int \_val) : val(\_val), next(nullptr) {}

};

struct up\_list

{

Node\* first;

Node\* last;

up\_list() : first(nullptr) {}

void list\_ini()

{

}

bool is\_empty()

{

return first == nullptr;

}

void push\_back(int val)

{

Node\* ptr = new Node(val);

if (is\_empty())

{

first = ptr;

}//endif

else

{

Node\* arrow = first;

if (ptr->val <= arrow->val)

{

ptr->next = first;

first = ptr;

}//endif

else

{

Node\* left = first;

arrow = left->next;

while (arrow)

{

if (arrow->val >= ptr->val)

{

left->next = ptr;

ptr->next = arrow;

break;

}

left = left->next;

arrow = arrow->next;

}

if ((arrow == nullptr) && (left->val < ptr->val))

{

arrow = ptr;

left->next = arrow;

arrow->next = nullptr;

}

}

}

}

void print()

{

int i = 1;

Node\* arrow = first;

while (arrow)

{

std::cout << i << ". " << arrow->val << '\n';

++i;

arrow = arrow->next;

}

}

void delete\_elements(int val)

{

Node\* arrow = first;

Node\* left = first;

while (arrow)

{

if ((arrow->val == val) && (arrow == first))

{

first = first->next;

}

else

if (arrow->val == val)

{

left->next = arrow->next;

arrow = arrow -> next;

}

else

{

left = arrow;

arrow = arrow->next;

}

}

}

};

//-----------------------------------------------

up\_list substraction(up\_list list1, up\_list list2);

//------------------

int main()

{

up\_list L;

up\_list L1, Result;

int count;

int input\_val;

int del\_val;

std::cout << "Input amount of numbers in list:";

std::cin >> count;

std::cout << '\n';

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

{

std::cout << "Input number: ";

std::cin >> input\_val;

L.push\_back(input\_val);

}

L.print();

std::cout << '\n';

std::cout << "Input value to delete in list:";

std::cin >> del\_val;

L.delete\_elements(del\_val);

L.print();

std::cout << '\n';

std::cout << "Input amount of numbers in second list to substruct from 1st one:";

std::cin >> count;

std::cout << '\n';

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

{

std::cout << "Input number: ";

std::cin >> input\_val;

L1.push\_back(input\_val);

}

Result = substraction(L, L1);

Result.print();

return 0;

}

//------------------

up\_list substraction(up\_list list1, up\_list list2)

{

Node\* arrow2 = list2.first;

while (arrow2)

{

list1.delete\_elements(arrow2->val);

arrow2 = arrow2->next;

}

return list1;

}