|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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. Реализация пакета подпрограмм для работы с однонаправленным связным списком(поиск узла по значению, удаление, печать списка)

**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 | Up\_List L | false |
| Проверка первого элемента в непустом списке | Up\_List L | true |

**Function:**

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

Выполняет разность двух списков и создает результирующий список, возвращает его.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Разность списков A и B: ai != bj, i = 1, …, n, j = 1, …, m | Up\_List A,B | List C = A |
| Разность списков A и B: ai == bj, i = 1, …, n, j = 1, …, m | Up\_List A,B | List C = {0}; |

**Function:**

Node\* searchVal(int \_val)

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Поиск элемента в пустом списке | List A, int \_val | List C = A |
| Поиск элемента в непустом списке | List A, int \_val | List C = {0}; |

**procedure :**

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

Вставляет элемент в конец однонаправленного списка.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Вставка числа в пустой список | Int Value = a, up\_list L | List L = {a} |
| Вставка числа непустой список | Int Value = c, up\_list L = {a,b} | List L = {a, c, b}, |

**procedure :**

**void print();**

Выводит список в консоль

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Печать заполненного списка | Up\_List L = {a} | a |
| Печать пустого списка | Up\_List L = {} | - |

**procedure :**

**void deleteList();**

Удаляет все элементы списка, указателю на первый элемент списка присваивает нулевой.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Удаление списка из n элементов | Up\_List L = {ai | i = 1, …, n} | L = nullptr; |

**3 Текст программы**

#include <iostream>

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

struct Node

{

int val;

Node\* next;

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

};

struct up\_list

{

Node\* first;

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

bool is\_empty()

{

return first == nullptr;

}

void push(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;

}

}

}

}

Node\* searchVal(int \_val)

{

Node\* arrow = first;

while (arrow && (arrow->val != \_val))

return arrow;

if (!arrow)

return 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)

{

if (!is\_empty())

{

Node\* left = first;

Node\* arrow = left->next;

Node\* memory;

while (first && (first->val == \_val))

{

memory = first;

first = first->next;

delete(memory);

left = first;

if (left)

arrow = left->next;

}

while (arrow)

{

if (arrow->val == \_val)

{

memory = arrow;

left->next = arrow->next;

delete(memory);

if (left)

arrow = left->next;

}

else

{

left = left->next;

if (left)

arrow = left->next;

}

}

}

}

void deleteList()

{

Node\* arrow = first->next;

Node\* memory = arrow;

while (arrow)

{

memory = arrow;

arrow = arrow->next;

delete(memory);

}

first = nullptr;

}

};

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

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(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(input\_val);

}

Result = substraction(L, L1);

Result.print();

Result.deleteList();

L.deleteList();

L1.deleteList();

return 0;

}

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

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

{

Node\* arrow2 = list2.first;

up\_list result = list1;

while (arrow2)

{

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

arrow2 = arrow2->next;

}

return result;

}