Министерство образования РФ

Пермский государственный технический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 11

ПО ИНФОРМАТИКЕ ЗА I СЕМЕСТР

Вариант 9

Выполнил студент:

Главатских Максим Николаевич

Группа РИС-20-1бз

Шифр 20-ЭТФ-659

Кафедра ИТАС:

Полякова Ольга Андреевна

ПЕРМЬ, 2020

**Лабораторная работа № 11**

**"Информационные динамические структуры"**

**Вариант №9**

**Цель:**

Знакомство с динамическими информационными структурами на примере одно- и двунаправленных списков.

**Задача:**

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

Для каждого вариант разработать следующие функции:

1. Создание списка.

2. Добавление элемента в список (в соответствии со своим вариантом).

3. Удаление элемента из списка (в соответствии со сво-им вариантом).

4. Печать списка.

5. Запись списка в файл.

6. Уничтожение списка.

7. Восстановление списка из файла.

**Вариант задания:**

Записи в линейном списке содержат ключевое поле типа int. Сформировать двунаправленный список. Удалить из него К элементов перед элементом с заданным номером, добавить К элементов в конец списка.

**Текст программы**

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

struct List {

string data;

int key;

struct List\* next;

};

void print\_list(List\* list) {

List\* cur = list;

while (cur) {

cout << cur->key << " " << cur->data << endl;

cur = cur->next;

}

}

// Функция необходимая для формирования линейного списка по заданию

void append(List\*\* list, int key, string data) {

List\* cur = \*list;

if (cur == NULL) {

cur = new List;

cur->next = NULL;

cur->key = key;

cur->data = data;

\*list = cur;

}

else {

while (cur->next) {

cur = cur->next;

}

cur->next = new List;

cur->next->next = NULL;

cur->next->key = key;

cur->next->data = data;

}

}

// Наследуемся от линейного списка -> можем добавлять и печатать через функции для линейного списка

struct BidirectionalList : public List {

List\* prev;

};

void create\_bidirectional\_list(BidirectionalList\*\* b\_list, List\* list) {

List\* cur = list;

if (\*b\_list == NULL)

\* b\_list = new BidirectionalList;

List \* b\_cur = \*b\_list;

List \* prev\_addr = NULL;

while (cur) {

b\_cur->key = cur->key;

b\_cur->data = cur->data;

b\_cur->next = cur->next;

((BidirectionalList\*)b\_cur)->prev = prev\_addr;

prev\_addr = cur;

b\_cur = b\_cur->next;

cur = cur->next;

}

}

void add\_k\_el\_to\_tail(BidirectionalList\*\* b\_list, int k, int key, string data) {

List\* cur = \*b\_list;

while (cur) {

if (cur->next == NULL) {

for (int i = 0; i < k; ++i) {

cur->next = new List;

cur->next->next = NULL;

cur->next->key = key;

cur->next->data = data;

((BidirectionalList\*)cur->next)->prev = cur;

cur = cur->next;

}

return;

}

cur = cur->next;

}

}

void delete\_k\_el\_before(BidirectionalList\*\* b\_list, int idx, int k) {

List\* cur = \*b\_list;

int i = 0;

while (cur) {

if (i == idx - 1) {

for (int j = 0; j < k; ++j) {

if (cur == NULL)

return;

List \* time\_cur = cur;

cur = ((BidirectionalList\*)cur)->prev;

if (cur != NULL)

cur->next = time\_cur->next;

if (time\_cur->next != NULL)

((BidirectionalList\*)time\_cur->next)->prev = cur;

delete time\_cur;

}

return;

}

++i;

cur = cur->next;

}

}

int main() {

List\* list = NULL;

append(&list, 1, "aaaa");

append(&list, 2, "bbbb");

print\_list(list);

cout << endl;

BidirectionalList\* b\_list = NULL;

create\_bidirectional\_list(&b\_list, list);

add\_k\_el\_to\_tail(&b\_list, 4, 10, "new data");

print\_list(b\_list);

cout << endl;

delete\_k\_el\_before(&b\_list, 4, 2);

print\_list(b\_list);

cout << endl;

return 0;

}