МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное общеобразовательное учреждение высшего образования

«Ижевский государственный технический университет имени М.Т. Калашникова»

Кафедра «Программное обеспечение»

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

на тему «Обработка линейных связных списков»

Выполнил: студент группы Б03-191-3 Р.А. Гумметов

Принял: д.т.н., профессор М.А. Сенилов

Ижевск 2019

1. Цель работы

Цель работы: изучить структуры данных типа «линейный список», научиться их программировать и использовать.

2. Задание

Разработать и описать на языке программирования структуры данных для линейных связных списков, а также реализовать базовый комплект операций над линейными списками; сформировать тесты (список элементов) для тестирования разработанных программ; представить (описать) результаты тестирования.

3. Постановка задачи

Задача состоит в том, чтобы разработать и описать на языке программирования структуры данных для представления элементов списка и самого списка, а также реализовать комплект наиболее распространенных (базовых) операций над списком.

Базовые операции над линейным связным списком:

- создание и формирование списка;

- удаление всего списка;

- поиск элемента в списке;

- проверка, есть ли в списке заданный элемент;

- вставка элемента в список (в том числе добавление элемента в начало списка, добавление элемента в конец списка);

- удаление элемента из списка (в том числе удаление элемента из начала списка, удаление элемента из конца списка);

- удаление всех элементов списка;

- просмотр элементов списка (печать значений элементов, хранящихся в списке).

4. Код программы

#include <cstdlib>

#include <iostream>

using namespace std;

template<typename T>

class List

{

public:

List();

~List();

void pop\_front();

void push\_back(T data);

void clear();

int GetSize() { return Size; }

T& operator[](const int index);

void push\_front(T data);

void insert(T data, int index);

void removeAt(int index);

void pop\_back();

private:

template<typename T>

class Node

{

public:

Node \* pNext;

T data;

Node(T data = T(), Node \*pNext = nullptr)

{

this->data = data;

this->pNext = pNext;

}

};

int Size;

Node<T> \*head;

};

template<typename T>

List<T>::List()

{

Size = 0;

head = nullptr;

}

template<typename T>

List<T>::~List()

{

clear();

}

template<typename T>

void List<T>::pop\_front()

{

Node<T> \*temp = head;

head = head->pNext;

delete temp;

Size--;

}

template<typename T>

void List<T>::push\_back(T data)

{

if (head == nullptr)

head = new Node<T>(data);

else

{

Node<T> \*current = this->head;

while (current->pNext != nullptr)

{

current = current->pNext;

}

current->pNext = new Node<T>(data);

}

Size++;

}

template<typename T>

void List<T>::clear()

{

while (Size)

{

pop\_front();

}

}

template<typename T>

T & List<T>::operator[](const int index)

{

int counter = 0;

Node<T> \*current = this->head;

while (current != nullptr)

{

if (counter == index)

return current->data;

current = current->pNext;

counter++;

}

}

template<typename T>

void List<T>::push\_front(T data)

{

head = new Node<T>(data, head);

Size++;

}

template<typename T>

void List<T>::insert(T data, int index)

{

if (index == 0)

push\_front(data);

else

{

Node<T> \*previous = this->head;

for (int i = 0; i < index - 1; i++)

{

previous = previous->pNext;

}

Node<T> \*newNode = new Node<T>(data, previous->pNext);

previous->pNext = newNode;

Size++;

}

}

template<typename T>

void List<T>::removeAt(int index)

{

if (index == 0)

pop\_front();

else

{

Node<T> \*previous = this->head;

for (int i = 0; i < index - 1; i++)

{

previous = previous->pNext;

}

Node<T> \*toDelete = previous->pNext;

previous->pNext = toDelete->pNext;

delete toDelete;

Size--;

}

}

template<typename T>

void List<T>::pop\_back()

{

removeAt(Size - 1);

}

int main()

{

setlocale(LC\_ALL, "ru");

cout << "1. Создание и формирование списка" << endl;

cout << "2. Вставка элемента в список по индексу" << endl;

cout << "3. Вставка элемента в начало списка" << endl;

cout << "4. Вставка элемента в конец списка" << endl;

cout << "5. Удаление элемента в списке по индексу" << endl;

cout << "6. Удаление элемента в начале списка" << endl;

cout << "7. Удаление элемента в конце списка" << endl;

cout << "8. Удаление всех элементов списка" << endl;

cout << "9. Поиск элемента в списке" << endl;

cout << "10. Просмотр всех элементов списка" << endl;

cout << "-1. Выход из программы" << endl;

List<int> lst;

int command = 0;

while (command != -1)

{

cout << "Введите команду" << endl;

cin >> command;

switch (command)

{

case 1:

{

cout << "Введите количество элементов в списке" << endl;

int n;

cin >> n;

if (n == 1)

cout << "Введите 1 элемент" << endl;

else if (n > 1 && n < 5)

cout << "Введите " << n << " элемента" << endl;

else

cout << "введите " << n << " элементов" << endl;

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

{

int num;

cin >> num;

lst.push\_back(num);

}

break;

}

case 2:

{

cout << "Введите добавляемый элемент" << endl;

int num;

cin >> num;

cout << "Введите индекс добавляемого элемента" << endl;

int ind;

cin >> ind;

lst.insert(num, ind);

break;

}

case 3:

{

cout << "Введите добавляемый элемент" << endl;

int num;

cin >> num;

lst.push\_front(num);

break;

}

case 4:

{

cout << "Введите добавляемый элемент" << endl;

int num;

cin >> num;

lst.push\_back(num);

break;

}

case 5:

{

cout << "Введите индекс удаляемого элемента" << endl;

int ind;

cin >> ind;

lst.removeAt(ind);

break;

}

case 6:

{

lst.pop\_front();

break;

}

case 7:

{

lst.pop\_back();

break;

}

case 8:

{

lst.clear();

cout << "Список очищен" << endl;

break;

}

case 9:

{

cout << "Введите искомый элемент" << endl;

int num;

cin >> num;

int ind = -1;

for (int i = 0; i < lst.GetSize(); i++)

{

if (lst[i] == num)

{

ind = i;

break;

}

}

if (ind == -1)

cout << "Искомого элемента нет в списке" << endl;

else

cout << "Искомый элемент находится под индексом " << ind << endl;

break;

}

case 10:

{

if (lst.GetSize() == 0)

cout << "Список пуст" << endl;

else

{

for (int i = 0; i < lst.GetSize(); i++)

{

cout << lst[i] << " ";

}

cout << endl;

}

break;

}

default:

break;

cout << "Введите команду" << endl;

}

}

}

5. Результаты работы программы

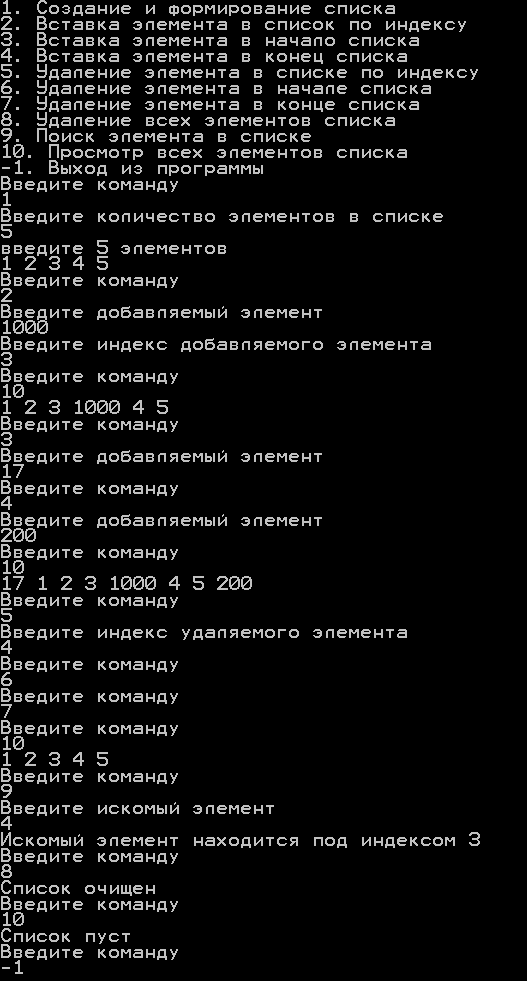


Рис. 1

6. Вывод

В ходе лабораторной работы разработаны и описаны на структуры данных для линейных связных списков, а также реализован базовый комплект операций над линейными списками.