**Министерство высшего образования и науки Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)**

**Электротехнический факультет**

**Дисциплина: основы алгоритмизации и программирования, 2 семестр**

**ОТЧЁТ**

**Тема: «Лабораторная работа №11»**

Выполнил

Студент РИС-22-2б

Третьяков Н.А.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

Содержание

Оглавление

[Логическое применение списков 3](#_Toc133950350)

[Разбор видов списков 4](#_Toc133950351)

[Блок схема 5](#_Toc133950352)

[Программный код 6](#_Toc133950353)

[Приложение 7](#_Toc133950354)

# Логическое применение списков

Список — это динамическая структура данных которая позволяет хранить информацию в памяти компьютера.

Списки похожи на динамические списки, однако они различаются тем что память, выделяемая для работы со списками не последовательна как у массивов.

# Разбор видов списков

Списки можно разбить на 4 вида, при чём два из них имеют особенность в коде, а 2 других особенность при логической реализации.

Первый самый простой однонаправленный список особенностей у данного списка в принципе нет, особенность структуры заключается только в том, что его узловые элементы хранят в себе данные и указатель на следующий элемент.

Второй вид двунаправленные списки они похожи на однонаправленные только в их узловых элементах хранятся два указателя на следующий элемент и на предыдущий

Третий вид списки типа очередь при работе с данным видом списков использую логику первый зашёл первый вышел, то есть любой добавляемый элемент добавляется в конец списка, в остальном можно настроить реализацию как в двунаправленных, так и однонаправленных списках.

Четвёртый вид стек данный тип обратен очереди и подчиняется логики первый пришёл первый вышел, реализовывать можно как в однонаправленных, так и двунаправленных списках

# Блок схема

Рассмотрим однонаправленный список так как другие три типа строятся на подобной основе, но только со своими особенностями.

В Приложении 1 указываются структуры

Приложение 2 основной код программы в котором настроен выбор действий

Приложение 3 функция Меню для того чтобы пользователь мог увидить что может делать программа и выбрать нужное действие

Приложение 4 реализация функции для добавления элемента в конец списка

Приложение 5 реализация функции для добавления элемента в начало списка

Приложение 6 функция для удаления элементов с конца списка

Приложение 7 функция для удаления элементов из начала списка

Приложение 8 функция для добавления нескольких элементов в список

Приложение 9 функция для удаления нескольких элементов из списка

Приложение 10 функция для вывода на экран списка

Приложение 11 функция для удаления списка целиком

Приложение 12 функция для проверки пусто список или нет

Данные приложение расположены в порядке возрастания слева на право в прикреплённом файле

# Программный код

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

struct Node

{

string a;

Node\* next = nullptr;

};

struct List

{

Node\* head = nullptr;

Node\* tail = nullptr;

size\_t size = 0;

};

void Menu();

bool Chek(List\* list);

void push\_buck(List\* list);

void del\_buck(List\* list);

void push\_front(List\* list);

void del\_front(List\* list);

void PushK(List\* list);

void DelK(List\* list);

void DelList(List\* list);

void CoutList(List\* list);

int main()

{

setlocale(0, "RUS");

SetConsoleCP(1251);

List\* list = new List;

int i;

bool F = 1;

while (F)

{

Menu();

cout << "Enter the action number" << endl;

cin >> i;

switch (i)

{

case 1:

{

system("cls");

push\_front(list);

break;

}

case 2:

{

system("cls");

push\_buck(list);

break;

}

case 3:

{

system("cls");

del\_front(list);

break;

}

case 4:

{

system("cls");

del\_buck(list);

break;

}

case 5:

system("cls");

PushK(list);

break;

case 6:

system("cls");

DelK(list);

break;

case 7:

{

system("cls");

CoutList(list);

break;

}

case 8:

F = 0;

default:

cout << "Select the action number that is in the menu" << endl;

}

}

delete list;

}

void Menu()

{

cout << "\t Menu" << endl;

cout << "1) Add an item to the top of the list" << endl;

cout << "2) Add an item to the end of the list" << endl;

cout << "3) Remove K items from the beginning of the list" << endl;

cout << "4) Remove K items from the end of the list" << endl;

cout << "5) Add to the elements from the entered number" << endl;

cout << "6) Delete K elements from the entered number" << endl;

cout << "7) Display the list on the screen" << endl;

cout << "8) Finish working with the list" << endl;

}

void push\_buck(List\* list)

{

list->size++;

Node\* node = new Node;

cout << "Enter the data" << endl;

getline(cin>>ws, node->a);

if (Chek(list))

{

list->tail->next = node;

list->tail = node;

}

else

{

list->head = node;

list->tail = node;

}

}

void del\_buck(List\* list)

{

int i;

int n;

cout << "Enter how many elements you need to delete" << endl;

cin >> i;

if (i >= list->size)

{

cout << "In the list, the number of items is less than or equal to the number of items to be deleted, so the entire list is deleted." << endl;

DelList(list);

}

else

{

n = 1;

Node\* temp = list->head;

while (n < list->size - i)

{

n++;

temp = temp->next;

}

list->size = list->size - i;

list->tail = temp;

temp = temp->next;

while (i > 0)

{

Node\* temp1 = temp;

temp = temp->next;

delete temp1;

i--;

}

delete temp;

cout << "Elements removed" << endl;

}

}

void push\_front(List\* list)

{

list->size++;

Node\* node = new Node;

string str;

cout << "Enter the data" << endl;

getline(cin>>ws, str);

node->a = str;

if (Chek(list))

{

node->next = list -> head;

list->head = node;

}

else

{

list->head = node;

list->tail = node;

}

}

void del\_front(List\* list)

{

int i;

cout << "Enter the number of items to delete" << endl;

cin >> i;

if (i >= list->size)

{

cout << "In the list, the number of items is less than or equal to the number of items to be deleted, so the entire list is deleted." << endl;

DelList(list);

}

else

{

list->size = list->size - i;

while (i > 0)

{

Node\* temp;

temp = list->head;

list->head = list->head->next;

delete temp;

i--;

}

cout << "Elements removed" << endl;

}

}

void PushK(List\* list)

{

int i, n;

string str;

int k = 1;

cout << "Enter the element from which to start adding elements to start adding" << endl;

cin >> n;

cout << "Enter the number of items to be entered" << endl;

cin >> i;

if ((n > list->size) && (n < 1))

{

cout << "There is no such item in the list" << endl;

}

else

{

Node\* temp;

Node\* temp1;

temp = list->head;

while (k < n)

{

temp = temp->next;

k++;

}

temp1 = temp->next;

list->size = list->size + i;

while (i > 0)

{

Node\* node = new Node;

cout << "Enter the data" << endl;

getline(cin >> ws, str);

node->a = str;

temp->next = node;

temp = node;

i--;

}

temp->next = temp1;

}

}

void DelK(List\* list)

{

int i, n;

int k = 1;

cout << "Enter the element from which to start deleting elements start adding" << endl;

cin >> n;

cout << "Enter the number of items to be entered" << endl;

cin >> i;

if ((n > list->size) && (n < 1))

{

cout << "There is no such item in the list" << endl;

}

else

{

if (i >= list->size)

{

cout << "In the list, the number of items is less than or equal to the number of items to be deleted, so the entire list is deleted." << endl;

DelList(list);

}

else

{

Node\* temp;

Node\* temp1;

temp = list->head;

while (k < n-1)

{

temp = temp->next;

k++;

}

temp1 = temp->next;

list->size = list->size - i;

i--;

while (i > 0)

{

Node\* temp2;

temp2 = temp1;

temp1 = temp1->next;

i--;

delete temp2;

}

temp->next = temp1;

}

}

}

bool Chek(List\* list)

{

if (list->head == nullptr)

{

return 0;

}

else

{

return 1;

}

}

void CoutList(List\* list)

{

int i = 1;

int n = list->size;

Node\* temp = list->head;

if (Chek(list))

{

while (n > 0)

{

cout <<i << ") "<< temp->a << endl;

temp = temp->next;

i++;

n--;

}

}

}

void DelList(List\* list)

{

bool F = 1;

while (F)

{

Node\* temp;

if (list->head != NULL)

{

temp = list->head;

list->head = list->head->next;

delete temp;

}

else

{

F = 0;

cout << "The list has been deleted" << endl;

}

}

list->size = 0;

}

# Приложение

Файл 11.drawio