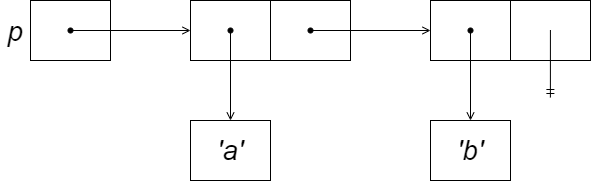
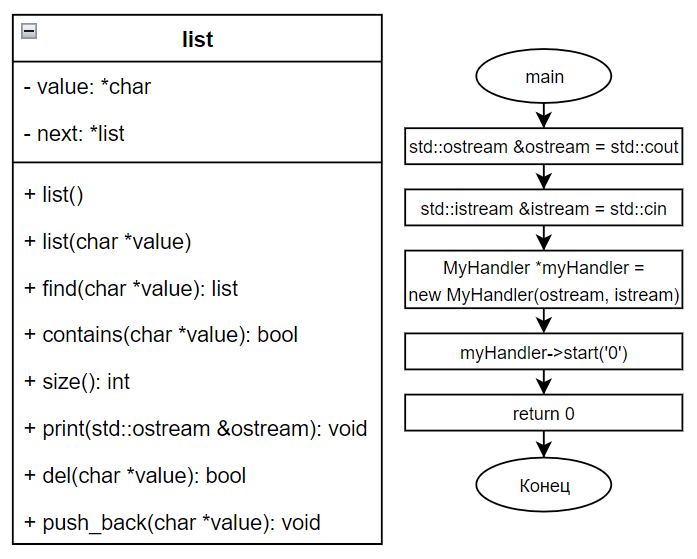
|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 1 | | |
| по дисциплине «Основы программирования» | | |
| **Структуры данных: список** | | |
|  | | |
|  | Бригада | ФИО |
| №2 | Веселый Денис Андреевич |
|  | Ворончук Илья Игоревич |
| Группа | ПМИ-32 |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

**1) Цель работы**

Получить практические навыки в организации и обработки списков на языке СИ.

**2) Условие задачи**

****Для указанного ниже списка, опишите на Си соответствующий тип список, определите переменную p (и, если надо, вспомогательные переменные) и напишите последовательность операторов для создания этого списка.

**3) Стандартная схема программы**

**4) Код программы**

**linked\_list.h**

#ifndef LINKED\_LIST\_H

#define LINKED\_LIST\_H

#include <ostream>

struct list

{

char \*value;

list \*next;

list();

list(char \*value);

// производит поиск по списку по значению и возвращает ссылку на элемент, содержащий переданное значение

list \*find(char \*value);

// проверяет наличие в списке элементы с переданным значением

bool contains(char \*value);

// возвращает размер списка

int size();

// выводит в переданный поток вывода текущий список

void print(std::ostream &ostream);

// удаляет элемент списка по заданному значению

bool del(char \*value);

// добавляет заданное значение в список

void push\_back(char \*value);

};

#endif

**linked\_list.cpp**

#include <ostream>

#include "proto/linked\_list.h"

// определяет значения по умолчанию для полей структуры

list::list()

{

value = NULL;

next = NULL;

}

// инициализирует поле \*value переданным значением, остальные поля определяются значением по умолчанию

list::list(char \*value)

{

this->value = value;

next = NULL;

}

list \*list::find(char \*value)

{

list \*l = this;

while (l->next != NULL && \*l->next->value != \*value) // пока следующий элемент не равен NULL и значение в этом элементе не совпадает переданным значением

l = l->next; // следующий шаг – спуск ниже по списку

return l->next;

}

bool list::contains(char \*value)

{

list \*l = this;

while (l->next != NULL) // пока следующий элемент не равен NULL

{

if (\*l->next->value == \*value) // если значение в этом элементе совпадает переданным значением

{

return true;

}

l = l->next; // следующий шаг – спуск ниже по списку

}

return false;

}

int list::size()

{

int n = 1;

list \*l = this;

while (l->next != NULL) // пока следующий элемент не равен NULL

{

l = l->next; // следующий шаг – спуск ниже по списку

n++;

}

return n;

}

void list::print(std::ostream &ostream)

{

if (size() == 1) // если размер списка равен 1

{

ostream << "List contains only root" << std::endl;

return;

}

list \*cur = this;

while (cur->next != NULL) // пока следующий элемент не равен NULL

{

if (cur->next->value != NULL) // если значение в текущем элементе не равно NULL

ostream << \*cur->next->value << " "; // вывод найденного значения в поток вывода

cur = cur->next; // следующий шаг – спуск ниже по списку

}

ostream << std::endl;

}

bool list::del(char \*value)

{

list \*l = this;

while (l->next != NULL && \*l->next->value != \*value) // пока следующий элемент не равен NULL и значение в этом элементе не совпадает переданным значением

l = l->next; // следующий шаг – спуск ниже по списку

if (l->next) // если текущий элемент не равен NULL

{

list \*p = l->next; // инициализация переменной-буфера

l->next = p->next; // размещение следующего за буферным элемента в текущий элемент

delete p; // освобождение памяти

p = nullptr;

return 1;

}

return 0;

}

void list::push\_back(char \*value)

{

list \*p = new list(value);

if (next == NULL) // если следующий элемент в текущем экземпляре равен NULL

next = p;

else

{

list \*cur = next;

while (cur->next != NULL) // пока следующий элемент не равен NULL

cur = cur->next; // следующий шаг – спуск ниже по списку

// найден последний элемент в списке

cur->next = p; // присвоение последнему элементу следующий элемент

}

}

**main.cpp**

#include <stdio.h>

#include <iostream>

// подключение модулей

#include "commander/Command.cpp"

#include "commander/Executor.cpp"

#include "commander/Handler.cpp"

#include "commands.cpp" // подключение команд, вводимых в терминал

#include "linked\_list.cpp" // подключение списка

#include "myhandler.cpp" // подключение модуля, связующего работу команд и списка

int main()

{

std::ostream &ostream = std::cout;

std::istream &istream = std::cin;

MyHandler \*myHandler = new MyHandler(ostream, istream);

myHandler->start('0');

return 0;

}

**5) Тестирование программы**

|  |  |  |
| --- | --- | --- |
| № | Тестирование | Комментарий |
| 1 | p a p b p c p f p k z | Добавление элементов a, b, c, f, k в список, вывод размерности списка. |
| 2 | p a p b p c d b t | Добавление элементов a, b, c в список, удаление элемента b и вывод их в консоль. |
| 3 | p q p n p k p h c n c i | Добавление элементов q, n, k, h. Проверка наличия элемента c в списке – найден. Проверка наличия элемента i в списке – не найден. |

**6) Результаты отладки**

Результаты работы программы показали, что программа работает успешно.