Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №14

за 2 семестр

По дисциплине: «ОАиП»

Тема: «Динамические структуры данных: списки и деревья»

Выполнила:

Студентка 1 курса

Группы ПО-3(1)

Гаврилюк Р.И.

Проверил:

Хацкевич М. В.

2019

Лабораторная работа №14

Динамические структуры данных: списки и деревья

Цель работы: приобретение навыков работы с динамической памятью и указателями на С/C++; изучение принципов работы с динамическими структурами данных: списками и деревьями.

Вариант 2.10

**Задание:**

Общие требования: в начале программы вывести задание; в процессе работы выводить подсказки пользователю (что ему нужно ввести, чтобы продолжить выполнение программы). Основные алгоритмы, ввод/вывод списков реализовать в виде функций с необходимыми параметрами. После работы программы вся динамически выделенная память должна быть освобождена. Взаимодействие с пользователем организовать в виде простого меню, обеспечивающего возможность переопределения исходных данных и завершение работы программы.

Задание по варианту: написать программу, обеспечивающую работу с однонаправленным нециклическим упорядоченным списком: добавление/удаление элементов, просмотр списка, поиск элемента в списке.

**Блок-схема:**

****

****

**Текст программы:**

#include <iostream>

#include <windows.h>

using namespace std;

struct list {

int data;

list \*next;

};

void menu(list\*head); //меню

void create\_list(list\*&head); //создание списка

void create\_node(list \*&head); //создание записи

void find(list \*head); //поиск

void deleting(list \*&head); //удаление элемента списка

void output(list \*head); //вывод на экран

void add\_to\_the\_end(list \*head); //добавление элемента в конец

int empty(list\*head); //проверка на пустоту списка

void delete\_list(list \*head); //удаление списка

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("color F0");

cout << "Написать программу, обеспечивающую работу с однонаправленным нециклическим упорядоченным списком:\n" << " - добавление/удаление элементов,\n - просмотр списка," << "\n - поиск элемента в списке.\n";

cin.ignore();

system("cls");

list \*head = NULL;

menu(head);

delete\_list(head);

return 0;

}

void menu(list \*head) {

int choise = 0;

do {

cout << "Выберите действие:\n";

cout << "1) Создание спика\n";

cout << "2) Добавление элементов\n";

cout << "3) Удаление элемента\n";

cout << "4) Просмотр списка\n";

cout << "5) Поиск элемента в списке\n";

cout << "0) Выход\n";

cin >> choise;

system("cls");

switch (choise)

{

case 1:

cout << "Создание спика\n";

create\_list(head);

break;

case 2:

cout << "Добавление элементов\n";

add\_to\_the\_end(head);

break;

case 3:

cout << "Удаление элемента\n";

deleting(head);

cin.ignore();

break;

case 4:

cout << "Просмотр списка\n";

cin.ignore();

output(head);

break;

case 5:

cout << "Поиск элемента в списке\n";

cin.ignore();

find(head);

break;

default: break;

}

cin.ignore();

system("cls");

} while (choise != 0);

}

void create\_list(list \*&head) {

head = NULL;

cout << "Введите количество элементов списка:\n";

int N;

cin >> N;

cout << "Введите список, состоящий из " << N << " элементов\n";

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

create\_node(head);

}

if (N > 0) {

output(head);

cin.ignore();

}

}

void find(list\*head) {

if (empty(head) == 1) {

list \*q = head;

cout << "Введите элемент, котoрый хотите найти:\n";

int find;

cin >> find;

bool is\_find = false;

int index = 0;//индекс искомого элемента

while (q) {

if (find == q->data) {

cout << "Элемент найден.\n" << "Его индекс: " << index << endl;;

is\_find = true;

}

index++;

q = q->next;

}

if (is\_find == false) {

cout << "Элемeнт не найден.\n";

}

cin.ignore();

}

}

void add\_to\_the\_end(list \*head) {

int choise;

do {

cout << "Введите новый элемент списка:\n";

create\_node(head);

cout << "Хотите добавить еще элемент? (1-да, 2-нет):\n";

cin >> choise;

if (choise != 1 && choise != 2) {

cout << "Вы ввели неправильный номер.\nВвод будет прекращен.\n";

cin.ignore();

cin.ignore();

}

} while (choise != 2);

output(head);

cin.ignore();

}

void deleting(list \*&head) {

if (empty(head) == 1) {

output(head);

int element;

cout << "\nВведите элемент списка, который хотите удалить.\n";

cin >> element;

list \*current = head; //указатель на текущий элемент

bool find = false;

while (current) {

if (element == current->data) {

find = true;

if (current != NULL) {

if (current == head) { //удаляем первый элемент

head = head->next;

delete current;

current = head;

}

else { //удаляем непервый элемент

list \*q;

q = head;

while (q->next != current) {

q = q->next;

}

q->next = current->next;

delete current;

current = q;

}

}

}

else current = current->next;

}

if (find == false) {

cout << "Данный элемент не найден.\n";

}

else {

output(head);

cin.ignore();

}

}

}

void output(list \*head) {

if (empty(head) == 1) {

cout << "Список:\n";

int i = 0;

while (head) {

cout << head->data << " ";

head = head->next;

i++;

}

}

}

void create\_node(list \*&head) {

list \*new\_node = new list;

cin >> new\_node->data;

new\_node->next = NULL;

list \*q = head;

if (q == NULL) {

new\_node->next = head;

head = new\_node;

}

else {

while (q->next) {

q = q->next;

}

new\_node->next = q->next;

q->next = new\_node;

}

}

int empty(list\*head) {

if (head == NULL) {

cout << "Список пуст.\n";

return 0;

}

else {

return 1;

}

}

void delete\_list(list \*head) {

if (empty(head) == 1) {

if (head) {

delete\_list(head->next);

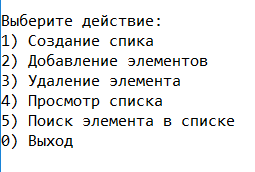
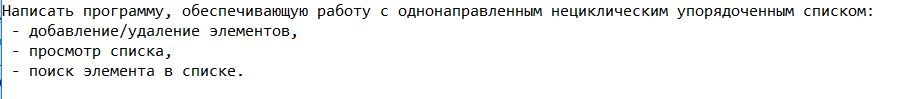
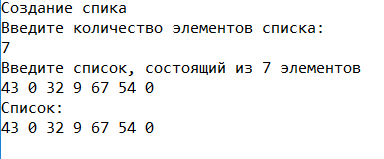
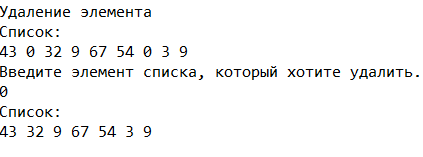
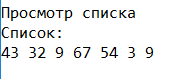
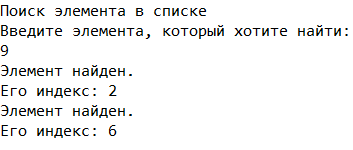
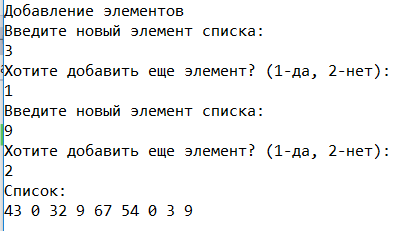
delete head;

}

}

}

**Результаты выполнения программы:**

Вывод: приобрела навыков работы с динамической памятью и указателями на С/C++; изучила принципов работы с динамическими структурами данных: списками и деревьями.