**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ УЧРЕЖДЕНИЯ ОБРАЗОВАНИЯ “БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”**

**КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Лабораторная работа №14

По дисциплине “Основы алгоритмизации и программирования”

Тема: «**Динамические структуры: списки и деревья**»

**Выполнил:**

Студент группы ИИ-23

Макаревич Н.Р.

**Проверила:**

Гирель Т.Н.

**БРЕСТ 2023**

**Вариант 2**

**Цель работы:** изучить принципы модульного программирования; ознакомиться с основными возможностями межмодульного взаимодействия.

**Задания по варианту:**

2. Написать программу, которая вводит с клавиатуры список целых чисел, подсчитывает среднее арифметическое элементов списка, затем вводит с клавиатуры дерево поиска и считает количество элементов дерева.

8. Написать программу, обеспечивающую работу с однонаправленным нециклическим списком: добавление/удаление элементов в голову, просмотр списка, инверсию списка.

21. Написать программу, реализующую в виде списка представление многочлена ***P(x) = a0+a1x+...+aixi,***где *ai* –вещественные числа, *i* – целые положительные числа, причем, если *ai=0*, то соответствующий элемент-слагаемое должен отсутствовать в списке. Пользователь должен иметь возможность произвольно добавлять элементы-слагаемые через меню. Реализовать функцию вычисления значения многочлена при заданном x

**Решение задания :**

#include <iostream>

#include <cmath>

#include <cstdio>

using namespace std;

struct Cell {

int data;

Cell\* left, \* right;

};

struct SearchTree {

Cell\* root;

SearchTree() {

root = NULL;

}

void addCell(int val)

{

Cell\* parent = NULL;

Cell\* cur = root;

while (cur != NULL)

{

parent = cur;

if (val < cur->data)

cur = cur->left;

else if (val > cur->data)

cur = cur->right;

else

return;

}

Cell\* newNode = new Cell;

newNode->data = val;

newNode->left = newNode->right = NULL;

if (parent == NULL)

root = newNode;

else

if (val < parent->data)

parent->left = newNode;

else

if (val > parent->data)

parent->right = newNode;

}

void printTree(Cell\* cur, int space = 0)

{

if (!cur)

return;

enum { COUNT = 2 };

space += COUNT;

printTree(cur->right, space);

for (int i = COUNT; i < space; ++i)

cout << " ";

cout << cur->data << endl;

printTree(cur->left, space);

}

};

struct Node

{

int data;

Node\* next;

};

struct Ellement {

double coef;

double power;

Ellement\* next;

};

struct List

{

Node\* head;

Ellement\* firstEl;

List()

{

head = NULL;

firstEl = NULL;

}

void addNode(int val)

{

Node\* newNode = new Node;

newNode->data = val;

newNode->next = NULL;

if (head == NULL)

head = newNode;

else

{

Node\* curr = head;

while (curr->next != NULL)

curr = curr->next;

curr->next = newNode;

}

}

void deleteNode(int val)

{

Node\* cur = head;

if (head->data == val)

head = head->next;

else

while (cur)

{

if (cur->next && cur->next->data == val)

{

cur->next = cur->next->next;

break;

}

else

cur = cur->next;

}

}

int arithmeticMean() {

Node\* cur = head;

int sum = 0;

int i = 0;

while (cur) {

sum += cur->data;

i++;

cur = cur->next;

}

return sum / i;

}

void reverseList()

{

Node\* q = NULL;

Node\* r = NULL;

while (head) {

r = head->next;

head->next = q;

q = head;

head = r;

}

head = q;

}

void printList()

{

Node\* p = head;

while (p != NULL)

{

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

void deleteHead() {

head = head->next;

}

void addHead(int val) {

Node\* newNode = new Node();

newNode->data = val;

newNode->next = head;

head = newNode;

}

void addEllement(double coef, double power) {

Ellement\* newNode = new Ellement;

newNode->coef = coef;

newNode->power = power;

newNode->next = NULL;

if (firstEl == NULL)

firstEl = newNode;

else

{

Ellement\* curr = firstEl;

while (curr->next != NULL)

curr = curr->next;

curr->next = newNode;

}

}

double answerOfFunc(double x) {

Ellement\* cur = firstEl;

double sum = 0;

while (cur) {

sum = sum + (pow(x, cur->power) \* cur->coef);

cur = cur->next;

}

return sum;

}

};

double stringToFunc(string str, double x) {

List\* temp = new List();

double coef = 0, power = 0;

bool chC, chP;

chP = chC = false;

for (int i = 0; i < str.length(); i++) {

if (str[i] >= '0' && str[i] <= '9') {

if (str[i + 1] == 'x') {

coef = int(str[i] - '0');

chC = true;

if (str[i - 1] == '-')

coef \*= -1;

}

else if (str[i + 1] != 'x') {

power = int(str[i] - '0');

chP = true;

if (str[i - 1] == '-')

power \*= -1;

}

if (chP && chC) {

temp->addEllement(coef, power);

chP = chC = false;

}

}

}

return temp->answerOfFunc(x);

}

int main()

{

setlocale(LC\_ALL, "Russian");

int length, val, choise;

List\* list = new List();

while (true) {

cout << "1) Добавить эллементы в список" << endl;

cout << "2) Вывести список на консоль" << endl;

cout << "3) Удалить эллемент по значению" << endl;

cout << "4) Среднее арифметическое списка" << endl;

cout << "5) Создание дерева поиска" << endl;

cout << "6) Инвертировать список" << endl;

cout << "7) Удалить голову" << endl;

cout << "8) Дoбавить эллемент в голову" << endl;

cout << "9) Ввести и найти значение многочлена" << endl;

cout << "0) Выход" << endl;

cin >> choise;

switch (choise)

{

case 1: {

system("cls");

cout << "Введите кол-во эллементов: " << endl;

cin >> length;

cout << "Введите целые числа: " << endl;

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

int val;

cin >> val;

list->addNode(val);

}

break;

}

case 2: {

system("cls");

list->printList();

break;

}

case 3: {

system("cls");

cout << "Введите значение: " << endl;

cin >> val;

list->deleteNode(val);

break;

}

case 4: {

system("cls");

cout << "Среднее арифметическое: " << list->arithmeticMean() << endl;

break;

}

case 5: {

system("cls");

cout << "Введите кол-во эллементов дерева поиска: " << endl;

cin >> length;

cout << "Введите числа для дерева поиска(0 для завершения): " << endl;

SearchTree\* tree = new SearchTree();

int len = 0;

cin >> val;

while(val != 0){

tree->addCell(val);

cin >> val;

len++;

}

tree->printTree(tree->root, 0);

cout << "\n Всего эллементов: " << len;

delete tree;

break;

}

case 6: {

system("cls");

list->reverseList();

break;

}

case 7: {

system("cls");

list->deleteHead();

break;

}

case 8: {

system("cls");

cout << "Введите значение: " << endl;

cin >> val;

list->addHead(val);

break;

}

case 9: {

system("cls");

cout << "Введите многочлен: " << endl;

cout << "(Степень писать как ^,\nне использовать пробелы,\nесли коэфицент положительный то перед ним писать '+',\nдля завершения Enter)" << endl;

string str;

cin >> str;

cout << "Введите х: " << endl;

double x;

cin >> x;

cout << "Ответ: " << stringToFunc(str, x) << endl;

break;

}

case 0: {

delete list;

return 0;

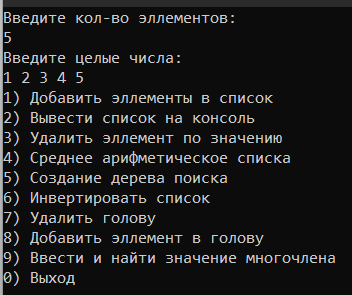
}

}

}

}

**Результат:**





**Вывод:**  изучил принципы динамических структур.