ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | М. А. Мурашова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| Линейные и циклические списки |
| по курсу: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

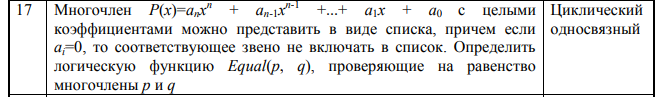
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

* 1. **Цель работы**

Целью работы является изучение структур данных «линейный список» и «циклический список», а также получение практических навыков их реализации.

* 1. **Задание** на лабораторную работу Реализовать структуры данных «линейный список» и «циклический список» в соответствии с заданным вариантом. Дополнительно программа должна осуществлять следующие операции: 1) Добавление/удаление элемента в список (с клавиатуры); 2) Вывод исходного и результирующего списков на экран; 3) Если списки являются многочленами, в выводе должна быть отражена степень каждого элемента.

**Вариант** 

**Ход выполнения**

**Листинг**

#include <iostream>

#include <string>

template<typename T>

class List

{

public:

List();

~List();

void push(T data);

int get\_size() { return size; }

int value(int index);

void remove(int index);

private:

template<typename T>

class Node

{

public:

Node\* pNext;

T data;

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

{

this->data = data;

this->pNext = pNext;

}

};

Node<int> \*head;

int size;

};

template<typename T>

List<T>::List()

{

size = 0;

head = nullptr;

}

template<typename T>

List<T>::~List()

{

}

template<typename T>

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

{

if (head == nullptr)

{

head = new Node<T>(data);

head->pNext = head;

}

else

{

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

while (current->pNext!=head)

{

current = current->pNext;

}

Node<T> \*tempHead = current->pNext;

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

current->pNext->pNext = tempHead;

}

size++;

}

int\* add(int value, int arr [], int size)

{

size++;

int\* res = new int[size];

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

{

res[i] = arr[i];

}

res[size-1] = value;

return res;

}

template<typename T>

int List<T>::value(int index)

{

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

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

{

current = current->pNext;

}

return current->data;

}

template<typename T>

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

{

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

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

{

current = current->pNext;

}

Node<T>\* rightP = current->pNext;

delete current;

size--;

current = this->head;

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

{

current = current->pNext;

}

current->pNext = rightP;

}

template<typename T>

bool equal(List<T> left, List<T> right) {

if (left.get\_size() != right.get\_size()) return 0;

for (int i = 0; i < left.get\_size(); i++)

{

if (left.value(i) != right.value(i)) return 0;

}

return 1;

}

int main()

{

List <int> lst;

int\* A\_a = new int [1];

int\* X\_a = new int[1];

X\_a[0] = 0;

A\_a[0] = 0;

int c = 1;

while (1)

{

int decide;

std::cout << "1 to push\n2 to remove\n";

std::cin >> decide;

if (decide ==1)

{

int A, X;

std::cout << "Enter A, X: ";

std::cin >> A;

std::cin >> X;

lst.push(A\*pow(X, lst.get\_size()));

A\_a=add(A, A\_a, c);

X\_a=add(X, X\_a, c);

c++;

}

else if(decide ==2)

{

std::cout << "Enter index: ";

int index;

std::cin >> index;

if (lst.get\_size() == 0) continue;

lst.remove(index);

}

else

{

break;

}

}

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

if (A\_a[i] == 0) continue;

std::cout << A\_a[i] << "\*" << X\_a[i] << "^" << i;

if (i != c - 1) { std::cout << "+"; }

}

std::cout << std::endl;

for (int i = 0; i < lst.get\_size(); i++)

{

std::cout << lst.value(i);

if (i!=lst.get\_size()-1)

{

std::cout << "+";

}

}

std::cout << std::endl;

List<int> lllst;

int\* a2 = new int[1];

a2[0] = 0;

int\* x2 = new int[1];

x2[0] = 0;

c = 1;

while (1)

{

int decide;

std::cout << "1 to push\n2 to remove\n";

std::cin >> decide;

if (decide == 1)

{

double A, X;

std::cout << "Enter A, X: ";

std::cin >> A;

std::cin >> X;

lllst.push(A \* pow(X, lllst.get\_size()));

a2 = add(A, a2, c);

x2 = add(A, x2, c);

c++;

}

else if (decide == 2)

{

std::cout << "Enter index: ";

int index;

std::cin >> index;

if (lst.get\_size() == 0) continue;

lllst.remove(index);

}

else

{

break;

}

}

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

if (a2[i] == 0) { continue; }

std::cout << a2[i] << "\*" << x2[i] << "^" << i;

if (i != c - 1) { std::cout << "+"; }

}

std::cout << std::endl;

for (int i = 0; i < lllst.get\_size(); i++)

{

std::cout << lllst.value(i);

if (i != lllst.get\_size() - 1)

{

std::cout << "+";

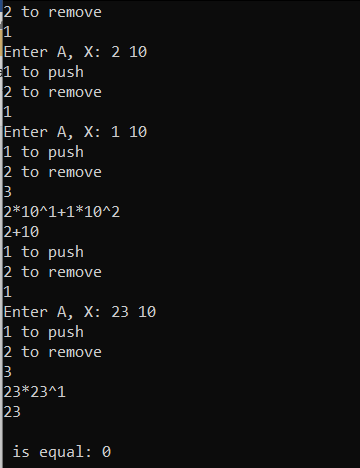
}

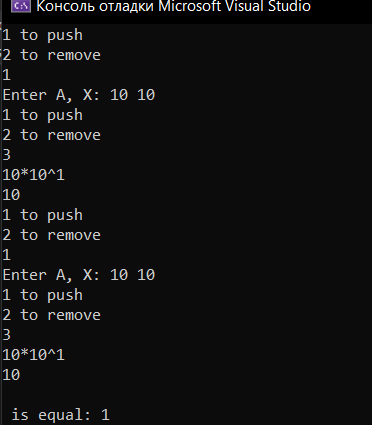
}

std::cout << "\n\n is equal: " << equal(lst, lllst) << "\n";

}

**Результат работы программы**





**Вывод**

Я изучил и закрепил методы работы с линейными списками, научился создавать односвязные циклические списки, которые можно будет применять для решения разных задач.