МИНЕСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., канд. техн. наук |  |  |  | В. А. Галанина |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| Работа с динамическими структурами данных в С/С++. Списки |
| по курсу: ИНФОРМАТИКА |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | М023 |  |  |  | Д.А.Трегуб |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

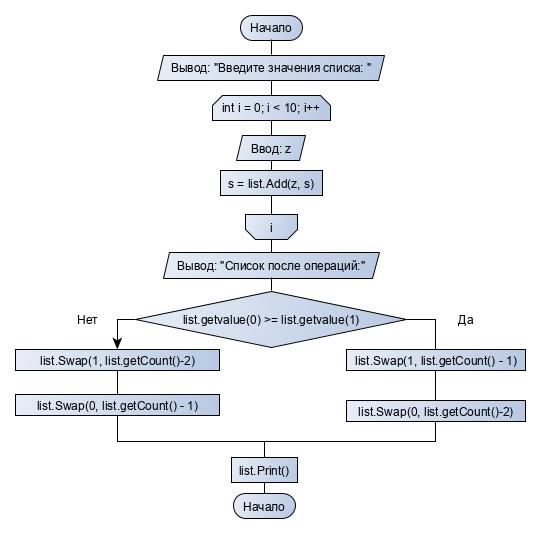
**1)Цель работы:** изучение способов создания и принципов использования односвязных(двусвязных) линейных списков; изучение стандартных средств языка C/C++ для работы с динамической памятью; совершенствование навыков модульного программирования на языке C/C++ при решении задач обработки линейных списков;

**2)Задание**: используя технологию объектно-ориентированного программирования разработать класс Список. С использованием методов этого класса написать программу обработки односвязных линейных списков с числом элементов в списке не менее десяти в соответствии с индивидуальным заданием.

Вариант №19:

Перенести в конец списка два его первых элемента таким образом, чтобы последним в списке оказался тот из них, значение которого меньше значения второго.

**3)Блок-схема:**



**4)Код:**

Файл List.h

#pragma once

template <typename Q> class List;

template <typename T> class Node;

using namespace std;

template < typename T > class Node

{

T field;

class Node\* ptr;

friend class List<T>;

};

template < typename Q > class List

{

private:

Node<Q>\* head; // Корень списка

int count = 0; // Количество узлов списка

//Переход к следующему узлу

Node<Q>\* Next(Node<Q>\* node)

{

if (isEmpty()) return NULL;

return node->ptr;

}

//Переход к предудыщему узлу

Node<Q>\* Prev(Node<Q>\* node)

{

if (isEmpty()) return NULL;

if (node == head) return NULL;

Node<Q>\* p = head;

while (p->ptr != node)

p = p->ptr;

return p;

}

//получения порядкового номера узла

int getNumber(Node<Q>\* number) {

Node<Q>\* p = head;

int countNumber = 0;

while (p != number) {

p = Next(p);

countNumber++;

}

return countNumber++;;

}

Node<Q>\* getNode(int node) {

Node<Q>\* p = head;

for (int i = 0; i != node; i++) {

p = Next(p);

}

return p;

}

public:

List() { head = NULL; } //изначально список пуст

int getCount() { return count; } //получение длины списка

bool isEmpty() { return head == NULL; } //проверка, пуст ли список

void setValue(Node<Q>\* p, Q val) { p->field = val; } //установка значения узла

Q getFirst() { return head->field; } //получение первого узла

//получение занчения узла за заданным номером

Q getvalue(int node) {

Node<Q>\* p = head;

for (int i = 0; i != node; i++) {

p = Next(p);

}

return p->field;

}

//получение значения последнего узла

Q getLast()

{

Node<Q>\* p = head;

int q = getNumber(p);

while ((Next(p) != NULL))

p = Next(p);

return p->field;

}

//отчистка списка

void Clear()

{

class Node<Q>\* p = head;

if (p == NULL) return;

do {

Node<Q>\* d = p;

p = Next(p);

delete d;

} while (p != NULL);

count = 0;

head = NULL;

}

//Добавление узла в список. Без второго аргумента всегда будет добавляться в начало. Иначе будет добавляться после заданного.

int Add(Q num, int q = 0)

{

Node<Q>\* node = getNode(q);

node = Prev(node);

Node<Q>\* elem = new Node<Q>();

elem->field = num;

count++;

if (node == NULL) // Добавление нового корня

{

if (head == NULL) {

elem->ptr = NULL;

head = elem;

}

else {

elem->ptr = head;

head = elem;

}

return q++;

}

elem->ptr = node->ptr; // Добавление узла после текущего

node->ptr = elem;

return q++;

}

//Удаление узла из списка

void Delete(int p)

{

Node<Q> node = getNode(p);

if (node == NULL) { return NULL; } // В списке нет узлов

count--;

if (node == head) // Удаление корневого узла

{

head = node->ptr;

delete node;

return head;

}

Node\* prev = Prev(node); // Удаление промежуточного узла

prev->ptr = node->ptr;

delete node;

}

//обмен узлов местами

void Swap(int p, int q)

{

Node<Q>\* node2 = getNode(p);

Node<Q>\* node1 = getNode(q);

if (node1 == NULL || node2 == NULL) return; // не допускаем обмен с несуществующим узлом

if (node1 == node2) return; // если один узел указан дважды, менять ничего не надо

Node<Q>\* prev1 = Prev(node1);

Node<Q>\* prev2 = Prev(node2);

Node<Q>\* next1 = Next(node1);

Node<Q>\* next2 = Next(node2);

if (next1 == node2) // обмен соседних узлов

{

if (prev1 != NULL)

prev1->ptr = node2;

else

head = node2;

node2->ptr = node1;

node1->ptr = next2;

return;

}

if (prev1 != NULL) // обмен разных узлов

prev1->ptr = node2;

else

head = node2;

if (prev2 != NULL)

prev2->ptr = node1;

else

head = node1;

node2->ptr = next1;

node1->ptr = next2;

}

//Вывод

void Print()

{

if (isEmpty()) { cout << "Список пуст" << endl; return; }

Node<Q>\* p = head;

do {

cout << getvalue(getNumber(p)) << " ";

p = Next(p);

} while (p != NULL);

cout << endl;

}

};

Файл Main.cpp:

#include "iostream"

#include "List.h"

using namespace std;

int main() {

system("chcp 1251");

system("cls");

List<int> list;

// Создаем список, помещаем элементы в конец

int z = 0;

int n;

int s = 0;

cout << "Сколько элементов ввести? " << endl;

cin >> n;

cout << "Введите значения списка: " << endl;

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

{

cin >> z;

list.Add(z, s);

s++;

}

list.Print();

cout << "Список после операций: " << endl;

if (list.getvalue(0) >= list.getvalue(1)) {

list.Swap(1, list.getCount() - 1);

list.Swap(0, list.getCount()-2);

}

else {

list.Swap(1, list.getCount()-2);

list.Swap(0, list.getCount() - 1);

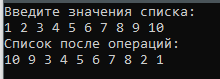
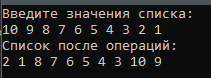
}

list.Print();

return 0;

}}

**5)Пример работы программы:**

**** ****

**6)Вывод:** В процессе выполнения лабораторной работы я изучил принцип построения и использования таких динамических структур как списки, а также улучшил навыки в объектно-ориентированного программирование на языке С++.