МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-2 «Прикладные информационные технологии»

А.А. МЕРСОВ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКОЙ РАБОТЫ № 3

Создание динамических списков.

по дисциплине: **«Языки программирования»**
(наименование дисциплины)

Москва – 2021

УДК
ББК

Печатается по решению редакционно-издательского совета «МИРЭА – Российский технологический университет»

Мерсов А.А.

Методические указания по выполнению практической работы № 3 по языкам программирования / А.А. Мерсов– М.: МИРЭА – Российский технологический университет, 2021.

Методические указания предназначены для выполнения практической работы по дисциплине «Языки программирования» и содержит перечень вариантов практической работы, а также краткое изложение теоретического материала в форме пояснений к заданию на работу. Для студентов, обучающихся по направлениям 09.03.02, 10.03.01, 10.05.02, 10.05.03, 10.05.04.

Материалы рассмотрены на заседании учебно-методической комиссии КБ-2 Протокол №1 от «28» августа 2021 г. и одобрены на заседании кафедры КБ-2.

зав. кафедрой КБ-2

к.т.н. _____ / О.В.Трубиенко /

УДК ББК

Содержание

Общие указания к выполнению практической работы 4
Цель практической работы 4
Основные сведения из языков программирования 5
Методический пример 6
Варианты заданий 13

Общие указания к выполнению практической работы

Практические работы выполняются с использованием персональных компьютеров. Указания по технике безопасности совпадают с требованиями, предъявляемыми к пользователю ЭВМ. Другие опасные факторы отсутствуют.

Цель практической работы

Цель работы: ознакомление с методами работы с динамическими структурами.

Практическая работа предполагает выполнение задания разработке и тестированию программного обеспечения.

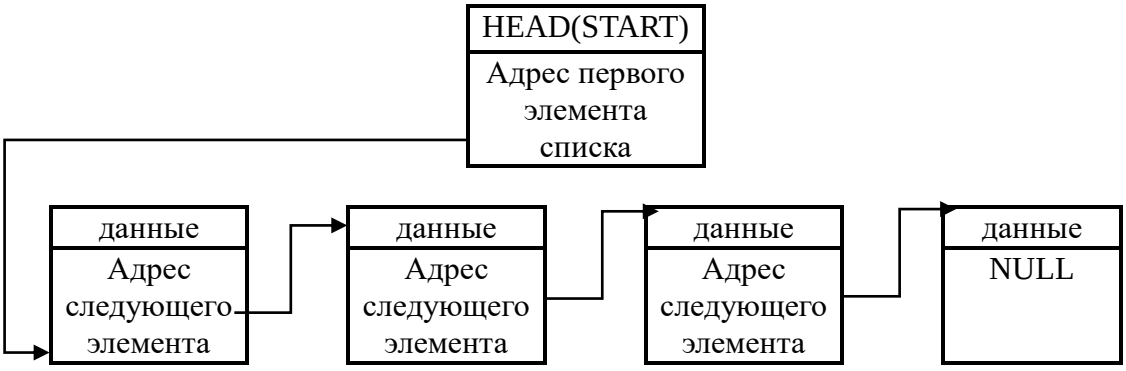
Связный список.

Структура данных, представляющая собой конечное множество упорядоченных элементов (узлов), связанных друг с другом посредством указателей, называется связным списком. Каждый элемент связного списка содержит поле с данными, а также указатель (ссылку) на следующий и/или предыдущий элемент. Эта структура позволяет эффективно выполнять операции добавления и удаления элементов для любой позиции в последовательности.

Причем это не потребует реорганизации структуры, которая бы потребовалась в массиве. Минусом связного списка, как и других структур типа «список», в сравнении его с массивом, является отсутствие возможности работать с данными в режиме произвольного доступа, т. е. список – структура последовательно доступа, в то время как массив – произвольного. Последний недостаток снижает эффективность ряда операций.

По типу связности выделяют односвязные, двусвязные, кольцевые и некоторые другие списки.

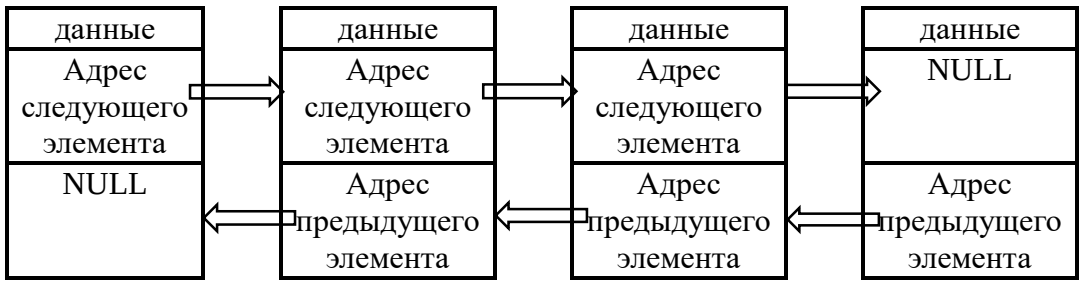
Каждый узел односвязного (однонаправленного связного) списка содержит указатель на следующий узел. Из одной точки можно попасть лишь в следующую точку, двигаясь тем самым в конец. Так получается своеобразный поток, текущий в одном направлении.



Односвязный список

На изображении каждый из блоков представляет элемент (узел) списка. Head – заголовочный элемент списка(самому списку не принадлежит), но содержит в себе адрес начального элемента списка. Признаком отсутствия указателя является поле, содержащее NULL.

Односвязный список не самый удобный тип связного списка, т. к. из одной точки можно попасть лишь в следующую точку, двигаясь тем самым в конец(либо в начало списка). Когда кроме указателя на следующий элемент есть указатель на предыдущий, то такой список называется двусвязным.



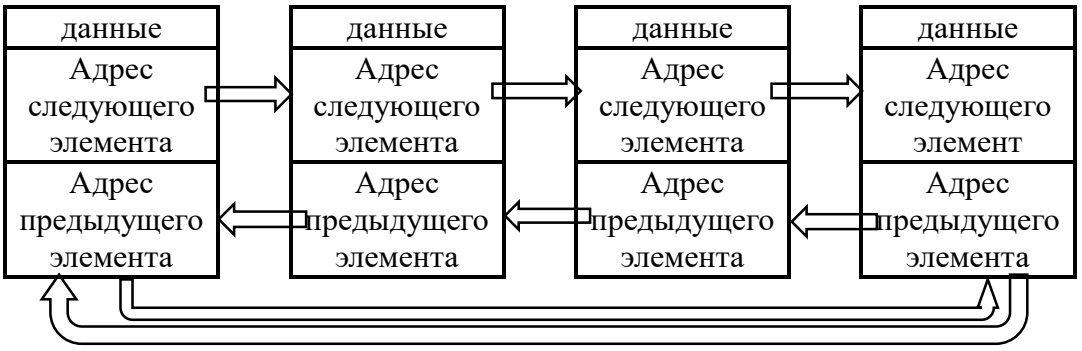
Двусвязный список

В двусвязном списке так же, как и в односвязном, присутствует элемент HEAD(START). Особенность двусвязного списка заключается в том, что каждый элемент имеет две ссылки: на следующий и на предыдущий элемент, позволяет двигаться как в его конец, так и в начало. Операции добавления и удаления здесь наиболее эффективны, чем в

односвязном списке, поскольку всегда известны адреса тех элементов списка, указатели которых направлены на изменяемый элемент, но добавление и удаление элемента в двусвязном списке, требует изменения большого количества ссылок, чем этого потребовал бы односвязный список.

Возможность двигаться как вперед, так и назад полезна для выполнения некоторых операций, но дополнительные указатели требуют задействования большего количества памяти, чем таковой необходимо в односвязном списке.

Еще один вид связного списка – кольцевой список. В кольцевом односвязном списке последний элемент ссылается на первый. В случае двусвязного кольцевого списка – плюс к этому первый ссылается на последний. Таким образом, получается зацикленная структура.



Кольцевой список

Методический пример

Рассмотрим основные операции над связными списками.

Программная реализация списка.

На примере двусвязного списка, разберем принцип работы этой структуры данных. При реализации списка удобно использовать структуры.

Приведем пример разработанной программы. Комментарии в тексте программы дают представление о ее реализации.

// SPISOK.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

```
//
// реализация программы по построению двунаправленного
// кольцевого списка
// выполняются следующие действия
// ввод элемента
// удаление элемента
// замена элемента
// добавление элемента
// печать списка
// поиск элемента
// печать элемента
//получение адреса элемента списка по его id

#include "pch.h"
#include <iostream>
#include <windows.h>
#include <stdio.h>
using namespace std;

int fmenu(char **);          // объявление функции меню
```

```
// создаем структуру списка
struct Spis
{
	Spis *previous_item; //место для хранения адреса предыдущего элемента
	int d;		// значение элемента
	int id;		// идентификатор элемента списка создается автом начиная с 0
	Spis *next_item;	// место для хранения адреса последующего элемента
};
// окончание создания структуры списка


//-----------------------------------------------------
// создаем массив меню

char m1[] = { "1.ввод элемента                  :\n" };
char m2[] = { "2.удаление элемента по его id     :\n" };
char m3[] = { "3.замена элемента по его значению  :\n" };
char m4[] = { "4.добавление элемента            :\n" };
char m5[] = { "5.печать списка                  :\n" };
char m6[] = { "6.поиск элемента по его значению   :\n" };
char m7[] = { "7.печать элемента по его id       :\n" };
char m8[] = { "8.печать элемента по его значению  :\n" };
char m9[] = { "9.выход                          :\n" };
char *menu[] = {m1,m2,m3,m4,m5,m6,m7,m8,m9};
//--------------------печать---------------------------------
Spis *pStart,	// пер-ная. для хра-ния адреса начала списка
*pEnd,		// пер-ная. для хра-ния адреса последнего эл-та
*pCurrent,	// пер-ная. для хра-ния адреса текущего эл-та списка
*pTemp,		// пер-ная. для хра-ния промежуточного адреса эл-та списка
*pPrev;		// пер-ная. для хра-ния предыдущего адреса эл-та списка
int count0 = -1;

void input_item();		// функция ввода элемента списка
void del_item(int);		// функция удаления элемента списка по его id
void change_item(int);		// функция замены элемента списка по его значению
void add_item(int,int);		// функация добавления элемента в список 1 аргумент
				// перед каким добавляем 2-ой что добавляем
void print_spis();		// печать всего списка
int find_item(int);		// поиск элемента списка по его значению возвращает его id
Spis* find_item(int,int fl);	// поиск элемента списка по его значению возвращает
адрес
void print_item_id(int);	// печать элемента списка по его id
void print_item_value(int);	// печать элемента списка по его значению
void exit_programm();		// выход из программы
char z;
int main()
{

	SetConsoleCP(1251);	// подключаем русский язык
	SetConsoleOutputCP(1251);// на вывод и ввод
		// счетчик элементов списка -1 элем-тов нет
	Spis *pStart,		// пер-ная. для хра-ния адреса начала списка
		*pEnd,		// пер-ная. для хра-ния адреса последнего эл-та
```

```
                        *pCurrent,        // пер-ная. для хра-ния адреса текущего эл-та списка
                        *pTemp,           // пер-ная. для хра-ния промежуточного адреса эл-та
списка
                        *pPrev;           // пер-ная. для хра-ния предыдущего адреса эл-та списка
                for(;1;)
                switch (fmenu(menu))
                {
                case 1:input_item(); break;//вызов функции ввода элемента;
                case 2:int id; cout << "введи id для удаления:"; cin >> id;
                        del_item(id); break;//вызов функции удаления элемента по его id;
                case 3:int value1; cout << "введи value для поиска и дальнейшей замены:";
                        cin >> value1; change_item(value1); break;//вызов функции замены
элемента по его номеру в списке;
                case 4:int value3,value2; cout << "введи value для поиска:";
                         cin >> value3;
                              cout <<endl<< "введи value для добавления:";
                              cin >> value2;
                              add_item(value3,value2); break;//вызов функции добавление
элемента на место элемента
                                        // с определенным значением;
                        case 5:print_spis(); break;//вызов функции печать списка;
                        case 6:int value,id_find; cout << "введи value для поиска:"; cin >> value;
                                id_find=find_item(value); cout << id_find<<endl; break;
                                        //вызов функции поиск элемента по его значению;
                                        // возвращает id найденного элемента
                        case 7:int id0; cout << "введи id для поиска:"; cin >> id0;
                                print_item_id(id0); break;//вызов функции печать элемента по
его номеру;
                        case 8:int value0; cout << "введи value для поиска:"; cin >> value0;
                                print_item_value(value0); break;// печать элемента по его
значению
                        case 9:exit_programm();//вызов функции выход;
                        default: {cout << "неверен пункт меню, повторите"; system("pause");
break;
                        }
                }
        }
        void flush_stdin()
        {
                cin.clear();
                while (cin.get() != '\n');
        }
        int fmenu(char *x[])
        {
                int q;
                for (int i = 0; i < 9; i++)
                        cout << x[i] ;  // вывод пунктов меню
                cin >> q;

                return q;               // возвращается пункт меню
        }
        void input_item()
        {
```

```
		pCurrent = new Spis;
		cout << endl << "введите значение элемента:";
		cin >> pCurrent->d;
		if (count0 == -1) //не создавали ни одного элемента списка
		{
			count0 += 1;
			pCurrent->id = count0;
			pStart = pCurrent; // запомнили адрес начала списка
			pEnd = pCurrent;   // запомнили адрес последнего элемента списка
			pCurrent->previous_item = pCurrent;//    для 0-го элемента адрес предыдущего эл-та
			pCurrent->next_item = pCurrent;    // совпадает с адресом последующего эл-та
			pTemp = pCurrent;                  // запомнили тек. адрес, потребуется при вводе
                                                                                //
следующего эл-та
		}
		else
		{
			count0 += 1;
			pCurrent->id = count0;
			pEnd = pCurrent;       // запомнили адрес последнего элемента списка
			pCurrent->previous_item = pTemp;// запомнили в текущем адрес предыдущего эл-та
			pCurrent->next_item = pTemp->next_item;// в поле след. текущего переписали
                                                // след. из предыщего
			//cin >> pCurrent->d;
			pStart->previous_item = pEnd;
			pTemp->next_item = pCurrent;
			pTemp = pCurrent;          // запомнили тек. адрес, потребуется при вводе
                                                                                //
следующего эл-та
		}
	}
	void print_spis()
	{
		if (count0 < 0) { cout << "элементов в списке нет" << endl;
		system("pause"); return;
                        }
		pCurrent = pStart;
		cout << pStart << endl;
		for (int i = 0; i <= count0; i++)
		{
			cout << pCurrent->previous_item << " : " << pCurrent->id << " : " <<
			pCurrent->d << " : " << pCurrent->next_item << endl;
			pCurrent = pCurrent->next_item;
	   }
		cout << pEnd << endl;
	}
	void print_item_id(int n)
	{
```

```
	int flag = 0;
	if (count0 < 0) {
		cout << "элементов в списке нет" << endl;
		system("pause"); return;
	}
	if (n > count0) {
		cout << "переданный номер больше махсимального в списке" << endl;
		system("pause"); return;
	}
	pCurrent = pStart;
	for (int i = 0; i <= count0; i++)
	{
		if (pCurrent->id == n) // совпадают ли id
		{
		flag = 1;
		cout << pCurrent->previous_item << " : " << pCurrent->id << " : " <<
				pCurrent->d << " : " << pCurrent->next_item << endl;
		return;
		}
		else
		  pCurrent = pCurrent->next_item;   // получение адреса след. элемента

	}
	if (!flag) cout << "элемента с id=" << n << " нет в списке" << endl;
}
void print_item_value(int value)
{
	int d;
	int flag = 0;
	pCurrent = pStart;
	for (int i = 0; i <= count0; i++)
	{
		if (pCurrent->d == value)
		{
			flag = 1;
			cout << pCurrent->previous_item << " : " << pCurrent->id << " : "
<<
				pCurrent->d << " : " << pCurrent->next_item << endl;
			cout << "искать далее?(1/0)"; cin >> d;
			if (d==0)return;
			pCurrent = pCurrent->next_item;
		}
		else
			pCurrent = pCurrent->next_item;
	}
	if (!flag) cout << "элемента с зачением=" << value << " нет в списке" << endl;

}
void exit_programm() { exit(1); }
void del_item(int n) {
	int flag = 0;
	if (count0 < 0) {
		cout << "элементов в списке нет" << endl;
```

```
                system("pause"); return;
        }
        if (n > count0) {
                cout << "переданный номер больше махсимального в списке" << endl;
                system("pause"); return;
        }
        pCurrent = pStart;
        for (int i = 0; i <= count0; i++)
        {
                if (pCurrent->id == n)
                {
                        flag = 1;
                        // берем в найденном элементе адрес предыдущего и в поле
следующего
                        // предыдущего элемента записываем значение следующего из
найденного
                        pCurrent->previous_item->next_item = pCurrent->next_item;

                        //берем в найденном элементе адрес последующего и в поле
предыдущего
                        // следущего элемента записываем значение предыдущего из
найденного
                        pCurrent->next_item->previous_item = pCurrent->previous_item;
                        delete pCurrent;
                        count0 -= 1;
                        return;
                }
                else
                        pCurrent = pCurrent->next_item;
        }
        if (!flag) cout << "элемента с id=" << n << " нет в списке" << endl;

}
int find_item(int value)
{
        int d;
        int flag = 0;
        pCurrent = pStart;
        for (int i = 0; i <= count0; i++)
        {
                if (pCurrent->d == value) // совпадают ли значения
                {
                        flag = 1;
                        cout << pCurrent->previous_item << " : " << pCurrent->id << " : "
<<
                                pCurrent->d << " : " << pCurrent->next_item << endl;
                        cout << "искать далее?(1/0)"; cin >> d;
                        if (d == 0) return pCurrent->id;//возвращаем id найденного
элемента
                        pCurrent = pCurrent->next_item; flag = 0;
                }
                else
                        pCurrent = pCurrent->next_item;
```

```
		}
		if (!flag) {
			cout << "элемента с значением=" << value << " нет в списке" << endl;
			return -1;
		}
	}
	Spis* find_item(int value, int fl)
	{
		int d;
		int flag = 0;
		pCurrent = pStart;
		for (int i = 0; i <= count0; i++)
		{
			if (pCurrent->d == value)
			{
				flag = 1;
				cout << pCurrent->previous_item << " : " << pCurrent->id << " : "
<<
					pCurrent->d << " : " << pCurrent->next_item << endl;
				cout << "искать далее?(1/0)"; cin >> d;
				if (d == 0) return pCurrent;//возвращаем адрес найденного
элемента
				pCurrent = pCurrent->next_item; flag = 0;
			}
			else
				pCurrent = pCurrent->next_item;
		}
		if (!flag) {
			cout << "элемента с зачением=" << value << " нет в списке" << endl;
			return nullptr;
		}

	}
	void change_item(int value1)
	{
		Spis* x;
		int z;		// переменная для замены
		x = find_item(value1, 1);
			if (x == nullptr) return;
			cout << "введи значение для замены:"; cin >> z;
			cout << endl;
			x->d = z;
	}
	void add_item(int value3, int value2)
	{
		Spis *pAdd,*pFind;
		pFind = find_item(value3, 1);//находим адрес искомого элемента
		if (pFind == nullptr) return;//если искомого нет то вываливаемся
		pAdd = new Spis;  // создаем новый элемент списка
		count0 += 1;// увеличиваем счетчик элементов и получаем новый id
		pAdd->d = value2;
		pAdd->id = count0;
		// заполнение полей
```

```
            // берем у найденного элемента(pFind) из поля предыдущий элемент(previous_item)
            // его адрес, и по этому адресу вытаскиваем из поле следующего элемента(previous_item)
            // адрес и заносим его в  поле (next)следующего добавляемого элемента по адресу pAdd
            pAdd->next_item = pFind->previous_item->next_item;
            // меняем на адрес pAdd поле next_item в
            //элементе pFind->previous_item
            pFind->previous_item->next_item = pAdd;
            // берем у найденного элемента(pFind) значение поля previous_item
            // и прописываем его в поле previous_item добавляемого элемента(pAdd)
            pAdd->previous_item = pFind->previous_item;
            // заменяем поле previous_item элемента pFind на значение pAdd
            pFind->previous_item = pAdd;
        }
```

Варианты заданий

№	Тип списка	Элемент списка	Действия Создать список из соответствующих элементов
0	однонаправленный	Строка из 10 символов	Найти строку символов по строке, введенной с клавиатуре ,и удалить ее.
1	двунаправленный	вещественное	Найти вещественное число по числу, введенному с клавиатуры, и удалить его.
2	Однонаправленный кольцевой	целое	Найти целое число по числу, введенному с клавиатуры и удалить его.
3	Двунаправленный кольцевой	символ	Найти символ число по символу, введенному с клавиатуры, и удалить его.
4	однонаправленный	Строка из 10 символов	Найти строку сииволов по строке, введенной с клавиатуре ,и после нее добавить новую строку.
5	двунаправленный	вещественное	Найти вещественное число по числу, введенному с клавиатуры, и после него добавить новое число.
6	Однонаправленный кольцевой	целое	Найти целое число по числу, введенному с клавиатуры и после него добавить новое число.
7	Двунаправленный кольцевой	символ	Найти символ число по символу, введенному с клавиатуры, и после него добавить новый символ.
8	однонаправленный	Строка из 10 символов	Найти строку сииволов по строке, введенной с клавиатуре ,и заменитьее ее новой строкой.
9	двунаправленный	вещественное	Найти вещественное число по числу, введенному с клавиатуры, и заменить его новым числом.
10	Однонаправленный кольцевой	целое	Найти целое число по числу, введенному с клавиатуры и заменить его новым числом.
11	Двунаправленный кольцевой	символ	Найти символ число по символу, введенному с клавиатуры, и заменить его новым символом.