ЛАБОРАТОРНАЯ РАБОТА № 5
ЛИНЕЙНЫЕ ОДНОСВЯЗНЫЕ СПИСКИ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение обучающимися навыков создания и обработки линейных динамических списков.

Задачами работы являются:

- получение практических навыков работы с линейными динамическими списками структур;
- тестирование работоспособности программы для различных исходных данных.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Список (англ. list) – это абстрактный тип данных , представляющий собой упорядоченный набор значений , в котором некоторое значение может встречаться более одного раза. Экземпляры значений, находящихся в списке, называются элементами списка.

Таким образом, списком называется упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения, исключения. Список, отражающий отношения соседства между элементами, называется линейным.

Длина списка равна числу элементов, содержащихся в списке, список нулевой длины называется пустым списком. Списки представляют собой способ организации структуры данных, при которой элементы некоторого типа образуют цепочку. Для связывания элементов в списке используют систему указателей. В минимальном случае, любой элемент линейного списка имеет один указатель, который указывает на следующий элемент в списке или является пустым указателем, что интерпретируется как конец списка.

Структура данных, элементами которой служат записи с одним и тем же форматом, связанные друг с другом с помощью указателей, хранящихся в самих элементах, называют связанным списком. В связанном списке элементы линейно упорядочены, но порядок определяется не номерами, как в списоке, а указателями, входящими в состав элементов списка. Каждый список имеет особый элемент, называемый указателем начала списка (головой списка), который обычно по содержанию отличен от остальных элементов.
В поле указателя последнего элемента списка находится специальный признак NULL, свидетельствующий о конце списка.

Линейные связные списки являются простейшими динамическими структурами данных. Из всего многообразия связанных списков можно выделить следующие основные:

- однонаправленные (односвязные) списки;
- двунаправленные (двусвязные) списки;
- циклические (кольцевые) списки.

В основном они отличаются видом взаимосвязи элементов и/или допустимыми операциями.

Однонаправленный (односвязный) список – это структура данных, представляющая собой последовательность элементов, в каждом из которых хранится значение и указатель на следующий элемент списка (рис. 1). В последнем элементеуказательна следующий элемент равен NULL.

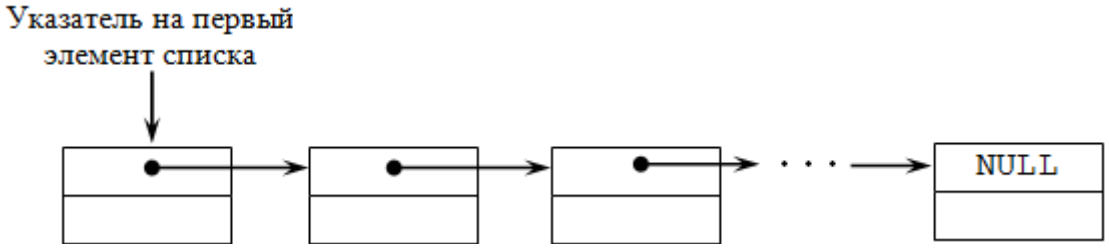

Рис. 1. Линейный однонаправленный список

Каждый элемент списка содержит ключ, который идентифицирует этот элемент. Ключ обычно бывает либо целым числом, либо строкой.

Основными операциями, осуществляемыми с однонаправленными списками, являются:

- создание списка;
- печать (просмотр) списка;
- вставка элемента в список;
- удаление элемента из списка;
- поиск элемента в списке
- проверка на отсутствие элементов в списке;
- удаление списка.

Особое внимание следует обратить на то, что при выполнении любых операций с линейным однонаправленным списком необходимо обеспечивать позиционирование какого-либо указателя на первый элемент. В противном случае часть списка или весь список будет недоступен.

Создание однонаправленного списка

Для того, чтобы создать список, нужно создать сначала первый элемент списка, а затем при помощи функции добавить к нему остальные элементы.

При относительно небольших размерах списка наиболее изящно и красиво использование рекурсивной функции. Добавление может выполняться как в начало, так и в конец однонаправленного списка.

Печать (просмотр) однонаправленного списка

Операция печати списка заключается в последовательном просмотре всех элементов списка и выводе их значений на экран. Для обработки списка организуется функция, в которой нужно переставлять указатель на следующий элемент списка до тех пор, пока указатель не станет равен NULL, то есть будет достигнут конец списка. Реализуется данная функция рекурсивно.

Вставка элемента в однонаправленный список

В динамические структуры легко добавлять элементы, так как для этого достаточно изменить значения адресных полей. Вставка первого и последующих элементов списка отличаются друг от друга. Поэтому в функции, реализующей данную операцию, сначала осуществляется проверка, на какое место вставляется элемент. Далее реализуется соответствующий алгоритм добавления (рис. 2).

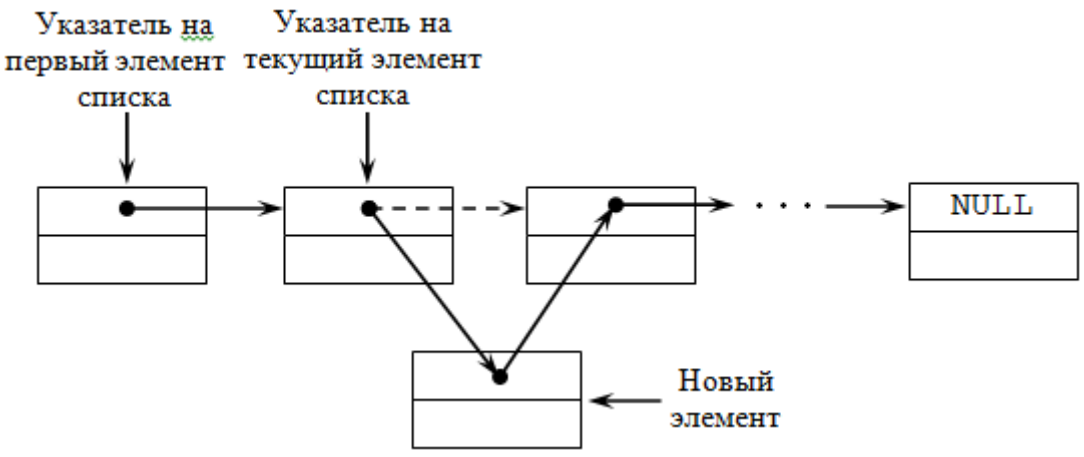

Рис. 2 – Вставка элемента в однонаправленный список

Удаление элемента из однонаправленного списка

Из динамических структур можно удалять элементы, так как для этого достаточно изменить значения адресных полей. Операция удаления элемента однонаправленного списка осуществляет удаление элемента, на который установлен указатель текущего элемента. После удаления указатель текущего элемента устанавливается на предшествующий элемент списка или на новое начало списка, если удаляется первый. При этом осуществляется проверка пустоты списка, т.е. наличие элементов в списке после удаления указанного.

Алгоритмы удаления первого и последующих элементов списка отличаются друг от друга. Поэтому в функции, реализующей данную операцию, осуществляется проверка, какой элемент удаляется. Далее реализуется соответствующий алгоритм удаления (рис. 3)

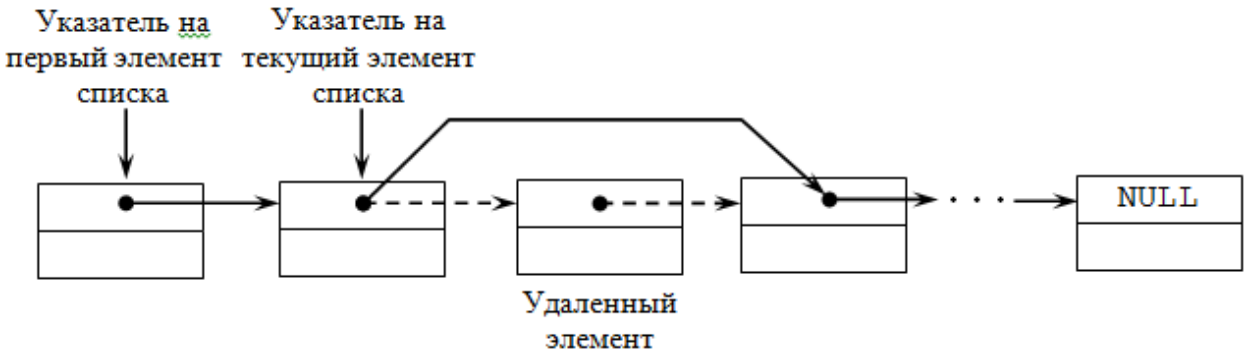

Рис. 3 – Удаление элемента из однонаправленного списка

Поиск элемента в списке

Операция поиска элемента в списке заключается в последовательном просмотре всех элементов списка до тех пор, пока текущий элемент не будет содержать заданное значение или пока не будет достигнут конец списка. В последнем случае фиксируется отсутствие искомого элемента в списке (функция принимает значение false). Выделяют поиск до первого найденного совпадения с ключом или поиск всех совпадений с заданным.

Удаление однонаправленного списка

Операция удаления списка заключается в освобождении динамической памяти. Для данной операции организуется функция, в которой нужно переставлять указатель на следующий элемент списка до тех пор, пока указатель не станет равен NULL, то есть не будет достигнут конец списка. Реализуется рекурсивная функция.

Двунаправленные (двусвязные) списки

Для ускорения многих операций целесообразно применять переходы между элементами списка в обоих направлениях. Это реализуется с помощью двунаправленных списков, которые являются сложной динамической структурой.

Двунаправленный (двусвязный) список – это структура данных, состоящая из последовательности элементов, каждый из которых содержит информационную часть и два указателя на соседние элементы (рис.4). При этом два соседних элемента должны содержать взаимные ссылки друг на друга.

В таком списке каждый элемент (кроме первого и последнего) связан с предыдущим и следующим за ним элементами. Каждый элемент двунаправленного списка имеет два поля с указателями: одно поле содержит ссылку на следующий элемент, другое поле – ссылку на предыдущий элемент и третье поле – информационное. Наличие ссылок на следующее звено и на предыдущее позволяет двигаться по списку от каждого звена в любом направлении: от звена к концу списка или от звена к началу списка, поэтому такой список называют двунаправленным.

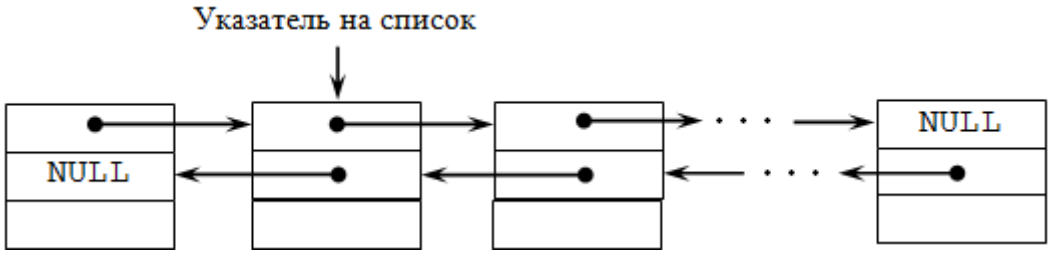

Рис. 4. Двунаправленный список

Основные операции, осуществляемые с двунаправленными списками, следующие:

- создание списка;
- печать (просмотр) списка;
- вставка элемента в список;
- удаление элемента из списка;
- поиск элемента в списке;
- проверка на отсутствие элементов в списке;
- удаление списка.

Особое внимание следует обратить на то, что в отличие от однонаправленного списка здесь нет необходимости обеспечивать позиционирование какого-либо указателя именно на первый элемент списка, так как благодаря двум указателям в элементах можно получить доступ к любому элементу списка из любого другого элемента, осуществляя переходы в прямом или обратном направлении. Однако по правилам хорошего тона программирования указатель желательно ставить на заголовок списка.

Списки в библиотеке STL

Библиотека стандартных шаблонов (STL – Standard Template Library) – набор согласованных обобщённых алгоритмов, контейнеров, средств доступа к их содержимому и различных вспомогательных функций в C++.

Контейнер list реализует структуру данных «двусвязный список» с возможностью добавления (и удаления) элементов и в начало, и в конец, и в середину списка за время O(1). Недостатком двусвязного списка является отсутствие возможности произвольного доступа, которое есть у вектора.

Некоторые методы класса list

- front() — первый элемент списка
- back() — последний элемент списка
- push_back(value) — добавляет элемент в конец списка
- push_front(value) — добавляет элемент в начало списка
- insert(it, value) — вставказначения value перед it
- insert(i, n, value) — вставка n копий элементов со значением value перед i
- pop_front() — удаляет первый элемент списка
- pop_back() — удаляет последний элемент (не возвращает значение!)
- erase(it) — удаляет элемент, на который указывает данный итератор
- erase(start, fin) — удалению подлежат все элементы между start и fin, но fin не удаляется.
- clear() — удаляет все элементы списка
- size() — выдает количество элементов списка
- empty() — возвращает истину, если список пуст

Пример. Задана структура «Студент» (фамилия; имя; возраст, курс). Информация хранится в бинарном файле. Реализовать следующие операции работы со списком:

- считать данные из файла в список,
- добавить данные,
- удалить данные,
- вывести на экран,
- сохранить в файл.

Программа на языке C++.

Файл student.h – заголовок структуры `Student`

```
#pragma once
/*структура студент*/

#include <fstream>

struct Student
{
	char NameLast[20]; // Фамилия
	char Name[20]; // Имя
	int Age; //Возраст
	int Course; // Курс
	//метод записи данных в файл
	//поток должен быть уже открыт
	bool write( std::ofstream *fout );
	//метод чтения данных из файла
	//поток должен быть уже открыт
	bool read(std::ifstream *fin );
};
```

Файл student.cpp – реализация методов структуры `Student`

```
#include "student.h"

using namespace std;

//методы структуры Student ------------------------
bool Student::write(ofstream* fout) {
	//Если файл не открыт
	if (!fout) return false;
	//записываем
	fout->write((char*)this, sizeof(Student));
	if (fout->good())
		return true;
	//else
	return false;
}

//метод чтения данных из файла
//поток должен быть уже открыт
bool Student::read(ifstream* fin) {
	//Если файл не открыт
	if (!fin) return false;
	if (fin->peek() == EOF) return false;
	fin->read((char*)this, sizeof(Student));
	if (fin->good())
		return true;
```

```
	//else
	return false;
}
```

Файл item.h - заголовок структуры Item

```
#pragma once
#include "student.h"

//Элемент односвязного списка
struct Item {
	Student data;
	Item* next;
	//конструктор
	Item(Student* st) {
		//копируем поля из параметра st в поле data
		strcpy_s(data.NameLast, st->NameLast);
		strcpy_s(data.Name, st->Name);
		data.Age = st->Age;
		data.Course = st->Course;

		next = NULL;
	}
};
```

Файл mylist.h – заголовок класса MyList

```
#pragma once

#include "student.h"
#include "item.h"

class MyList
{
	Item* Head; //Указатель на начало списка
public:
	MyList() :Head(NULL) {}; //Конструктор по умолчанию (Head = NULL)
	~MyList();//Деструктор
	void AddToTop(Student* st); // добавление записи в начало списка
	void AddToEnd(Student* st); // добавление записи в конец списка
	void Show(); // вывод списка на экран
	void Clear(); // очистка списка (удаление всех эл.)
	bool LoadToFile(char* file_name); // сохранение списка в файл
	bool ReadFromFile(char* file_name); // чтение из файла в тот же список
	bool isEmpty() {
		if (Head == NULL) return true;
		else return false;
	}
};
```

Файл mylist.cpp– реализация методов класса MyList

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <stdlib.h>
#include <iomanip>

#include "mylist.h"

using namespace std;
```

```
//Деструктор класса List
MyList::~MyList()
{
	Clear();
}

// очистка списка (удаление всех эл.)
void MyList::Clear()
{
	while (Head != NULL) //Пока по адресу есть хоть что-то
	{
		Item* temp = Head->next; //Сразу запом. указатель на адрес след. элемента структуры
		delete Head; //Освобождаем память по месту начала списка
		Head = temp; //Меняем адрес начала списка
	}
	Head = NULL;
}

/*функция добавления новой записи в начало списка*/
void MyList::AddToTop(Student* st)
{
	Item* temp = new Item(st); //Выделение памяти под новый элемент
	temp->next = Head; //Указываем, что адрес следующего элемента это старое начало списка
	Head = temp; //Смена адреса начала списка
}
/*функция добавления новой записи в конец списка*/
void MyList::AddToEnd(Student* st)
{
	Item* tmpNewItem = new Item(st); //Выделение памяти под новый элемент

	//если список пустой, то новый эл. это голова списка
	if (Head == NULL) {
		Head = tmpNewItem;
		return;
	}

	//если список не пустой, то переходим на последний эл. и добавляем в конец
	Item* p = Head;
	while (p->next != NULL) {
		p = p->next;
	}
	//теперь p указывает на последний эл-т
	p->next = tmpNewItem; //добавляем новый эл-т в конец
}

/*вывод списка на экран*/
void MyList::Show()
{
	Item* temp = Head; //Объявляем указатель, он указывает на начало
	system("cls"); // очисткаэкрана
	cout << setw(20) << "NameLast" << setw(20) << "Name" <<
		setw(10) << "Age" << setw(10) << "Course\n"; // выводзаголовка
	cout << "_____________________________________"
		<< endl << endl;
	while (temp != NULL) //Пока по адресу на начало есть данные
	{
		//Выводим все элементы структуры
		cout << setw(20) << temp->data.NameLast; //Вывод имени
		cout << setw(20) << temp->data.Name; //Вывод фамилии
		cout << setw(10) << temp->data.Age; //Вывод возраста
		cout << setw(10) << temp->data.Course << endl; //Вывод курса

		//Переходим на следующий адрес из списка
		temp = temp->next;
	}
```

```
	cout << "_____" << endl << endl;
}

// чтение из файла в тот же список
// старые элеметы удаляются
bool MyList::ReadFromFile(char * file_name)
{
	//если имя файла отсутствует
	if (file_name == NULL) return false;

	ifstream fin;
	fin.open(file_name, ios::binary | ios::in);
	//Если файл не открыт
	if (!fin) return false;

	// когда файл открыт ------------------
	Clear();  // удаляем старые элементы

	//в цикле читаем из файла и добавляем в конец списка
	Student tmpStud;
	bool isOk;
	while (!fin.eof())  {
		if (fin.peek() != EOF) {
			isOk = tmpStud.read(&fin);
			if (!isOk) {
				fin.close(); //Закрываем файл
				return false;
			}
			AddToEnd( &tmpStud );
		}
	}

	fin.close(); //Закрываем файл
	return true;
}


bool MyList::LoadToFile(char* file_name)
{
	//если имя файла отсутствует
	if (file_name == NULL) return false;

	ofstream fout(file_name, ios::binary | ios::out);
	//Если файл не открыт
	if (!fout) return false;

	Item* temp = Head; //Объявляем указатель, он указывает на начало списка
	while (temp != NULL) //Пока по адресу на начало есть данные
	{
		temp->data.write( &fout );
		temp = temp->next; //переходим на следующий адрес из списка
		if (!fout.good()) {
			fout.close();
			return false; //если ошибка записи
		}
	}
	fout.close();
	return true;
}
```

Файл lab2_4_1.cpp – файл с главной функцией и меню

```
#include <iostream>
#include <fstream>
#include <string.h>
```

```cpp
#include <stdlib.h>
#include <iomanip>
#include <ctime>

#include "student.h"
#include "item.h"
#include "mylist.h"

using namespace std;

//--------------------прототипы функций ------------------------
//реализация пункта меню "1: Read from the file"
void ReadListFromFile(MyList* lst);
//реализация пункта меню "3: Add student"
void AddStudToList(MyList* lst);
//реализация пункта меню "4: Save to the file"
void LoadListToFile(MyList* lst);


int main()
{
	cout.setf(ios::left);  // выравнивание по левому краю
	bool flag = true; // флаг продолжения работы
	int choice;       // выбор пользователя
	Student student;  // переменная типа Студент
	MyList spisok; // переменная типа Список
	while (flag)
	{
		system("cls"); // очисткаэкрана
		cout << "        MENU" << endl;
		cout << "_____________________" << endl;
		cout << "1: Read from the file" << endl;
		cout << "2: Show list" << endl;
		cout << "3: Add student" << endl;
		cout << "4: Save to the file" << endl;
		cout << "5: Delete last the record" << endl;
		cout << "6: Search" << endl;
		cout << "7: Sort" << endl;
		cout << "8: Exit" << endl;
		cout << "_____________________" << endl << endl;
		cout << "Make your choice (1-8): ";
		cin >> choice;
		switch (choice)
		{
		case 1: ReadListFromFile( &spisok); system("PAUSE"); break;
		case 2: spisok.Show(); system("PAUSE"); break;
		case 3: AddStudToList( &spisok ); break;
		case 4: LoadListToFile( &spisok ); break;
		case 5:
		case 6:
		case 7: cout << "Not implemented :(  " << endl; break;
		case 8: flag = false; break;
		default: cout << "You are wrong. " << endl;
		}
	}
	system("PAUSE");
	return 0;
}

//реализация пункта меню "1: Read from the file"
void ReadListFromFile( MyList * lst) {
	if (lst == NULL) {
		cout << "Wrong list!!" << endl;
		return;
	}
```

```
	char file_name[30]; // если имя файла задается пользователем
	//ifstream fin("C:\\Users\\Stud\\Desktop\\temp\\student.dat", ios::binary | ios::in);
	cout << "Input file name: ";
	cin >> file_name;
	if (! lst->ReadFromFile(file_name) )
		cout << "Wrong File\n";
	else
		cout << "Data is received\n";

}

//реализация пункта меню "3: Add student"
void AddStudToList(MyList* lst) {
	if (lst == NULL) {
		cout << "Wrong list!!" << endl;
		return;
	}

	Student  st;
	system("cls"); // Очистка экрана
	cin.ignore(); //Игнорируем символ
	cout << "NameLast:  "; cin.getline(st.NameLast, 20); //Ввод имени
	cout << "Name:  "; cin.getline(st.Name, 20); //Ввод фамилии
	cout << "Age: "; cin >> st.Age;  //Ввод возраста
	cin.ignore(); //Игнорируем символ
	cout << "Course: "; cin >> st.Course; //Ввод курса
	cin.ignore(); //Игнорируем символ

	lst->AddToTop(&st);
}

//реализация пункта меню "4: Save to the file"
void LoadListToFile(MyList* lst) {
	if (lst == NULL) {
		cout << "Wrong list!!" << endl;
		return;
	}

	char file_name[30];
	system("cls");
	cout << "Input file name: ";
	cin >> file_name;
	cout << "Begin writing...\n";
	if ( lst->LoadToFile( file_name ) ) //Если открытие файла прошло успешно
	{
		cout << "File created\n";
	}
	else cout << "File is not create\n";
	system("pause");
}
```

Остальные функции реализовать самостоятельно. Осуществить проверку пустоты списка.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получение варианта задания на лабораторную работу.
2. Разработка алгоритма решения задачи.
3. Составление программы на языке C++.
4. Отладка программы.
5. Тестирование программы.
6. Составление отчета о проделанной работе.

4. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

1. Оформить титульный лист с указанием темы работы.
2. Сформулировать цель и задачи работы.
3. Привести формулировку задания.
4. Записать текст программы.
5. Привести результаты тестирования программы для списка, содержащего не менее 15 элементов.
6. **Ответить на все контрольные вопросы**
7. Сформулировать вывод по проделанной работе.

5. ВАРИАНТЫ ЗАДАНИЙ

Задание 1. Реализовать метод класса MyList для удаления последнего элемента списка.

Задание 2. Осуществить поиск элемента списка MyList по фамилии студента; по возрасту. Информация для поиска запрашивается у пользователя.

Задание 3. В каждом варианте задана некоторая структура. Информация должна хранится в бинарном файле. Реализовать следующие функции обработки информации:

- добавить данные,
- удалить данные,
- вывести на экран,
- считать данные из файла в список,
- сохранить в файл,
- индивидуальное задание.

Внимание!! Пользоваться стандартным библиотечным классом списка нельзя! Использовать свой!

1. Задан список структур «Автосервис» (регистрационный номер автомобиля; фамилия владельца; пробег; стоимость ремонта; фамилия мастера, выполнившего ремонт). Вывести на экран общую сумму ремонта по каждому мастеру.
2. Задан список структур «Сотрудник» (фамилия сотрудника; должность; возраст; стаж; заработная плата). Вывести на экран среднюю заработную плату сотрудников каждой должности.
3. Задан список структур «Режим дня» (наименования занятия в детском саду; время начала занятия; время окончания занятия). Вывести на экран суммарное количество каждого занятия детей.
4. Задан список структур «Абонент» (фамилия; домашний адрес; номер телефона; тариф; баланс). Вывести на экран список тарифов с указанием количества абонентов.
5. Задан список структур «Склад» (вид, наименование товара; количество; стоимость; процент торговой надбавки). Вывести на экран суммарное количество товаров каждого вида.
6. Задан список структур «Ученик» (фамилия; класс; оценка по математике; оценка по русскому языку; оценка по литературе). Вывести на экран перечень классов с указанием среднего балла по классу.
7. Задан список структур «Афиша» (название фильма; номер зала; количество мест; стоимость билета; время сеанса). Вывести на экран информацию о максимально возможной выручке за каждый фильм.
8. Задан список структур «Покупатель» (фамилия покупателя; имя; номер карты; наименование товара; стоимость покупки). Вывести на экран сведения о покупателях и суммарной стоимости покупок каждого покупателя.
9. Задан список структур «Игрушка» (название игрушки; цена; количество; возрастные ограничения). Вывести на экран наименования и суммарное количество игрушек каждой возрастной группы согласно возрастным ограничениям (сначала 0+, потом 3+ и т.д.).
10. Задан список структур «Владелец» (фамилия владельца квартиры; номер дома; номер квартиры; площадь квартиры; задолженность по оплате коммунальных услуг). Вывести на экран сумму задолженности по оплате коммунальных услуг по каждому дому.

11. Задан список структур «Рейс» (пункт отправления; время отправления; пункт прибытия; время прибытия). Вывести на экран список возможных маршрутов из каждого пункта отправления места с указанием времени в пути.
12. Задан список структур «Читатель» (фамилия читателя; номер читательского билета; наименование книги; дата возврата). Вывести на экран фамилии и номера читательских билетов читателей с указанием количества взятых книг.
13. Задан список структур «Тур» (наименование тура; страна; продолжительность; стоимость; количество свободных путевок). Вывести на экран список стран с указанием суммарного количества путевок в наличии.
14. Задан список структур «Преподаватель» (фамилия; имя; отчество; разряд; год рождения). Вывести на экран количество преподавателей каждого разряда.
15. Задан список структур «Специальность» (название университета, шифр специальности подготовки; количество бюджетных мест; проходной балл за предыдущий год; количество заявлений). Вывести на экран суммарное количество заявлений по каждой специальности.
16. Задан список структур «Спортсмен» (фамилия спортсмена; порядковый номер; результаты трех попыток). Вывести на экран список суммарных результатов с указанием количества спортсменов, показавших данный результат.
17. Задан список структур «Вклад» (фамилия клиента банка; номер лицевого счета; сумма на счете; дата операции). Вывести на экран статистику операций по счетам по каждому месяцу с указанием количества клиентов, проводившим операции в данном месяце.
18. Задан список структур «Сотрудник» (фамилия; имя; квалификация; возраст; стаж работы). Вывести на экран данные о количестве сотрудников каждой квалификации.
19. Задан список структур «Государство» (материк; название; столица; численность населения; площадь). Вывести на экран список материков с указанием плотности населения.
20. Задан список структур «Кредит» (фамилия заемщика; сумма кредита; процентная ставка; срок кредитования). Вывести на экран количество заемщиков, взявших кредит с каждой процентной ставкой.

21. Задан список структур «Процесс» (наименование процесса; время начала; время окончания). Вывести на экран общую продолжительность выполнения каждого процесса.
22. Задан список структур «Картина» (автор; название картины; стиль; размеры; цена). Вывести на экран сведения о картинах, каждого стиля с указанием их количества.
23. Задан список структур «Телефон» (производитель; оперативная память; емкость аккумулятора; разрешение камеры; стоимость). Вывести на экран количество телефонов каждого производителя.
24. Задан список структур «Пациент» (фамилия пациента; номер страхового полиса; домашний адрес; диагноз). Вывести на экран перечень диагнозов с указанием количества пациентов с данным диагнозом.
25. Задан список структур «Тендер» (номер конкурса, наименование организации-участника тендера; количество сотрудников; собственный капитал; заявленная стоимость работ). Вывести на экран перечень организаций-участников с указанием количества заявок каждой организации.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение бинарного файла
2. В каком режиме производится обработка двоичных файлов?
3. Приведите алгоритм записи информации в бинарный файл. В чем специфика использования метода write?
4. Перечислите методы класса **ofstream** для обработки бинарного файла
5. Приведите алгоритм чтения информации из бинарного файла. В чем специфика использования метода read?
6. Перечислите методы класса **ifstream** для обработки бинарного файла
7. Как можно осуществить комбинацию режимов открытия файла?
8. Как осуществляется запись и чтение структур в бинарный файл?
9. Что такое динамическая структура данных?
10. Приведите классификацию динамических структур данных.
11. Дайте определение структуры данных “список”.
12. Какие виды списков существуют?
13. Опишите алгоритм добавления элемента в список.
14. Опишите алгоритм удаления элемента из списка.