Лекция 2

Табличные структуры данных.

1. Понятие таблицы. Виды таблиц.
2. Условия поиска в таблицах.
3. Линейные таблицы.
4. Логически связанные таблицы.

Литература

[Хусаинов] – Глава 6 «Табличные структуры»

Климачев Сергей Александрович

E-mail: klimachev@mail.osu.ru

1. Понятие таблицы. Виды таблиц.

Таблица (table) — набор элементов одинаковой организации, каждый из которых можно представить в виде двойки $\langle K, V \rangle$, где K — ключ, а V — тело (информационная часть) элемента.

Идентификатор	Фамилия	Имя	Отчество
232	Иванов	Иван	Иванович
453	Петров	Петр	Петрович
545	Сидоров	Савелий	Сергеевич

Ключ уникален для каждого элемента.

Ключ (key) – поле или набор полей, однозначно (уникально) идентифицирующих запись.

Примеры: табельный номер, ИНН, СНИЛС, заводской номер, серийный номер.

Представление таблицы. Классификация таблиц

```
struct Man
{
    int Id;
    string Surname;
    string Name;
};

struct Man table[15];
…
cin >> table[0].Surname;
…
```

Классификация таблиц

- по месту хранения
  - внутренние;
  - внешние (файлы);
- по отношениям связи между элементами
  - линейные;
  - нелинейные;
- по упорядоченности записей
  - упорядоченные;
  - неупорядоченные.

2. Условия поиска в таблицах.

Таблица (table) – набор элементов одинаковой организации, каждый из которых можно представить в виде двойки $\langle K, V \rangle$, где K – ключ, а V – тело (информационная часть) элемента.

3. Линейные таблицы.

- В линейной таблице элементы располагаются друг за другом.
- В оперативной памяти отображаются в массивы или линейные связанные списки.

5	11	23	7	1	9	21	14	97	45

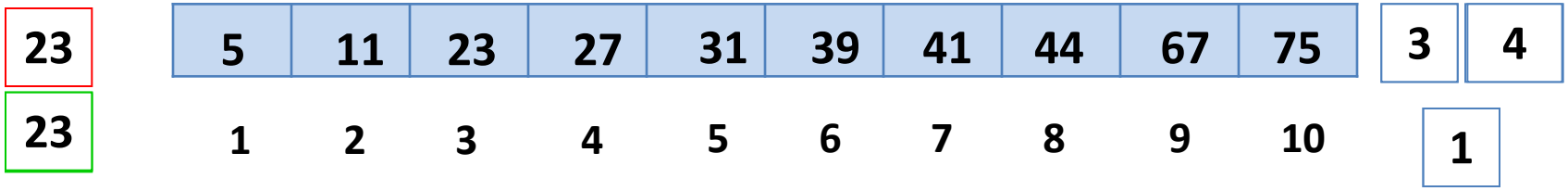
Таблица (table) – набор элементов одинаковой организации, каждый из которых можно представить в виде двойки $\langle K, V \rangle$, где K – ключ, а V – тело (информационная часть) элемента.

Таблица (table) – набор элементов одинаковой организации, каждый из которых можно представить в виде двойки $\langle K, V \rangle$, где K – ключ, а V – тело (информационная часть) элемента.

Таблица (table) – набор элементов одинаковой организации, каждый из которых можно представить в виде двойки $\langle K, V \rangle$, где K – ключ, а V – тело (информационная часть) элемента.

Поиск в упорядоченных таблицах

Двоичный (бинарный, логарифмический) поиск




```
int binarySearch(int *table, int n, int key)
{
    int left, right, index;
    left = 0;
    right = n-1;
    while(left<=right)
    {
        index = (left + right)/2;
        if(key > table[index])
            left = index + 1;
        else if(key < table[index])
            right = index - 1;
        else return index;
    }
    return -1;
}
```

Поиск в неупорядоченных таблицах

Последовательный поиск

Таблица (table) – набор элементов одинаковой организации, каждый из которых можно представить в виде двойки $\langle K, V \rangle$, где K – ключ, а V – тело (информационная часть) элемента.

Таблица (table) – набор элементов одинаковой организации, каждый из которых можно представить в виде двойки $\langle K, V \rangle$, где K – ключ, а V – тело (информационная часть) элемента.

Таблица (table) – набор элементов одинаковой организации, каждый из которых можно представить в виде двойки $\langle K, V \rangle$, где K – ключ, а V – тело (информационная часть) элемента.

```
int search(int *table, int n, int key)
{
    int i = 0;
    while(i<n && table[i]!=key)
        i++;

    return (i==n)?-1:i;
}
```

14	5	11	23	7	1	9	21	34	97	45	14	45

Поиск в неупорядоченных таблицах

Использование заграждающего элемента

```
int search(int *table, int n, int key)
{
   int i = 0;
   int r = table[n-1];
   table[n-1] = key;

   while(table[i]!=key)
      i++;

   table[n-1] = r;

   return (i==n-1 && r!=key)?-1:i;
}
```

Работа с линейными таблицами

Основные операции с данными

- Создание (create)
- Чтение (read)
- Модификация (update)
- Удаление (delete)

```
struct Man {
    int Id;
    string Surname;
    string Name;
    string Patronymic;
};
…
Man* table;
…
```

Работа с линейными таблицами

```
void serialize(fstream & stream, Man & man)
{
    stream.write((char *)&man.Id, sizeof(man.Id));

    size_t dataSize = man.Surname.size();
    stream.write((char *)&dataSize, sizeof(dataSize));
    stream.write(&man.Surname[0], dataSize);

    dataSize = man.Name.size();
    stream.write((char *)&dataSize, sizeof(dataSize));
    stream.write(&man.Name[0], dataSize);

    dataSize = man.Patronymic.size();
    stream.write((char *)&dataSize, sizeof(dataSize));
    stream.write(&man.Patronymic[0], dataSize);
}
```

Работа с линейными таблицами

```
void deserialize(std::fstream & stream, Man & man)
{
    stream.read((char *)&man.Id, sizeof(man.Id));

    size_t dataSize = 0;
    stream.read((char *)&dataSize, sizeof(dataSize));
    man.Surname.resize(dataSize);
    stream.read(&man.Surname[0], dataSize);

    dataSize = 0;
    stream.read((char *)&dataSize, sizeof(dataSize));
    man.Name.resize(dataSize);
    stream.read(&man.Name[0], dataSize);

    dataSize = 0;
    stream.read((char *)&dataSize, sizeof(dataSize));
    man.Patronymic.resize(dataSize);
    stream.read(&man.Patronymic[0], dataSize);
}
```

Работа с линейными таблицами

```
int add(Man & man) {
    int code = 0;
    int size = 0;

    fstream file("table.dat", ios::in | ios::out | ios::ate
                              | ios::binary);
    if (file.is_open()) {
        file.seekg(0, ios::beg);
        file.read((char*)&size, sizeof(size));
        size++;
        if (file.eof() || file.fail())
           file.clear();
        file.seekg(0, ios::beg);
        file.write((char*)&size, sizeof(size));
        file.seekg(0, ios::end);
        serialize(file, man);
        if (file.fail())
           code = -1;
        file.close();
    }
    else
        code = -1;
    return code;
}
```

Работа с линейными таблицами

```
Man* loadTable(int & size) {
    Man* table = 0;

    fstream file("table.dat", ios::in | ios::out | ios::ate
                              | ios::binary);
    if (file.is_open()) {
        file.seekg(0, ios::beg);
        file.read((char*)&size, sizeof(size));

        if (size > 0)
        table = new Man[size];

        for(int i=0; i<size && !file.eof(); i++)
           deserialize(file, table[i]);

        file.close();
    }

    return table;
}
```

Работа с линейными таблицами

```
bool writeTable(Man* table, int size) {
    bool result = true;
    fstream file("table.dat", ios::out | ios::binary);
    if (file.is_open() && size > 0) {
        file.write((char*)&size, sizeof(size));
        for (int i = 0; i < size; i++)
            serialize(file, table[i]);

        file.close();
        if (file.bad())
            result = false;
    }
    return result;
}
```

Работа с линейными таблицами

```
bool deleteFromTable(Man* table, int size, int index) {
    bool result = true;
    fstream file("table.dat", ios::out | ios::binary);
    if (file.is_open() && size > 0) {
        file.write((char*)&size, sizeof(size));
        for (int i = 0; i < size; i++)
            if (i != index)
                serialize(file, table[i]);

    file.close();
    if (file.bad())
        result = false;
    }
    return result;
}
```

4. Логически связанные таблицы.

Студенты

Идентификатор	Фамилия	Имя	Отчество
232	Иванов	Иван	Иванович
545	Петров	Петр	Петрович

Дисциплины

Идентификатор	Название
1	Структуры и алгоритмы обработки данных

Успеваемость

Идентификатор	Идентификатор студента	Идентификатор дисциплины	Оценка
1	232	1	4
2	545	1	5

Контрольные вопросы.

- Дайте определение таблицы.
- Что называется ключом таблицы? Приведите примеры.
- Назовите основные условия поиска в таблицах.
- Чем отличается поиск в упорядоченных и неупорядоченных таблицах?
- Каков наихудший случай поиска в неупорядоченной таблице?
- В чем сущность бинарного поиска? Какова его сложность?
- Сколько логически связанных таблиц необходимо для хранения адреса?

  Пример: г. Оренбург, проспект Победы, д. 13, корпус 3, ауд. 3306.