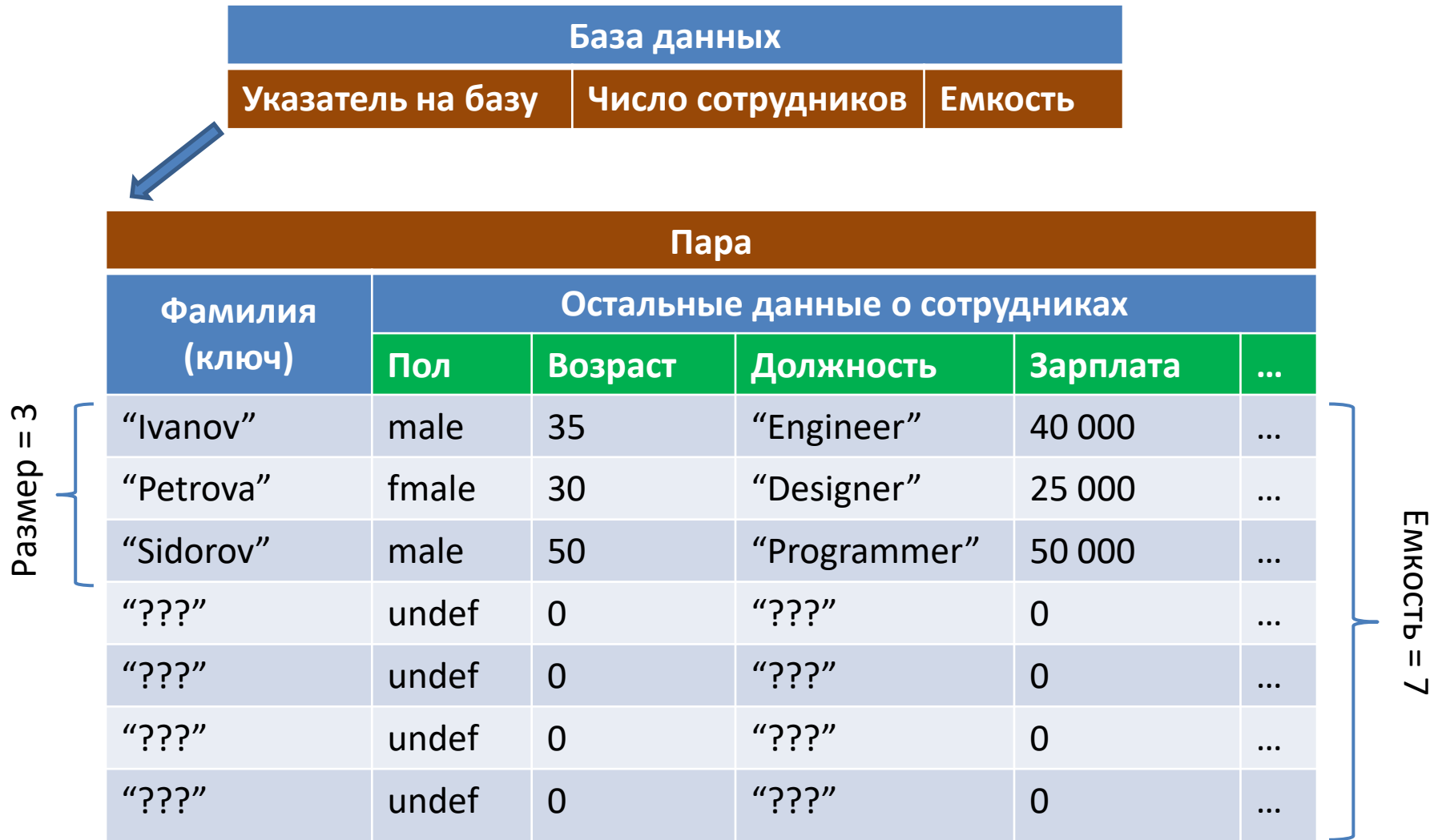


«База данных»

Реализация «базы данных»,
хранящей информацию о
сотрудниках, на основе
ассоциативного массива

Структура базы



Данные о сотрудниках

MyData				
Sex sex	size_t age	MyString job	float salary	...
s1	age1	job1	salary1	...
s2	age2	job2	salary2	...
s3	age3	job3	salary3	...
?	?	?	?	...
?	?	?	?	...
?	?	?	?	...

Класс MyData

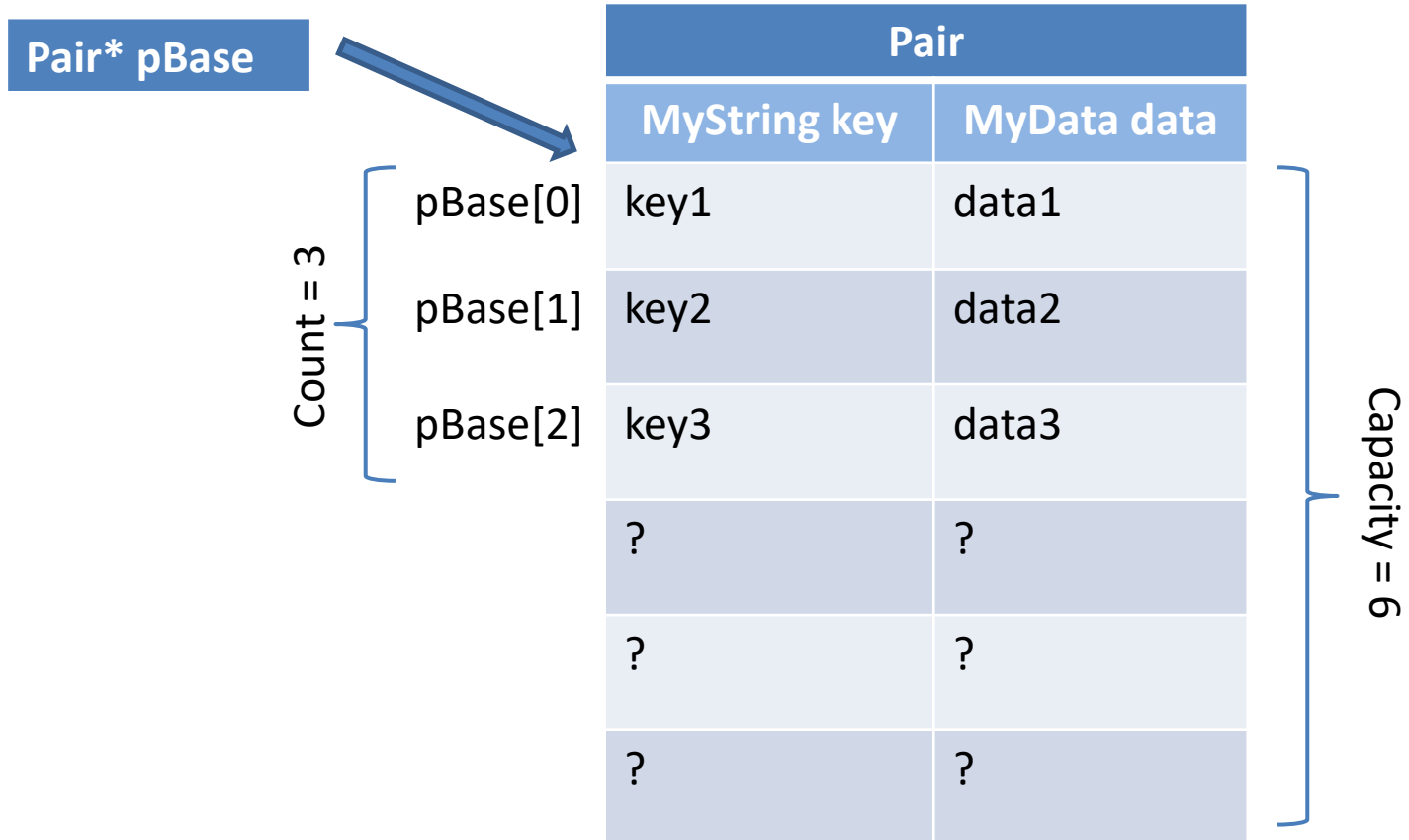
```
class MyData {
public:
    enum Sex { UNDEF, MALE, FEMALE };
private:
    Sex sex;
    size_t m_age;
    MyString m_job;
    float m_salary;
    ...
public:    //Подумайте, все ли перечисленные ниже методы надо реализовывать
    MyData();
    MyData(Sex s, size_t age, const char* job, float sal);
    ~MyData ();

    MyData(const MyData& d);
    MyData & operator=(const MyData& d);
    MyData(MyData&& d);
    MyData & operator=(MyData&& d);

    friend ostream& operator<<(ostream& os, const MyData& d);
    ...
    //или каких-либо методов не хватает
};
```

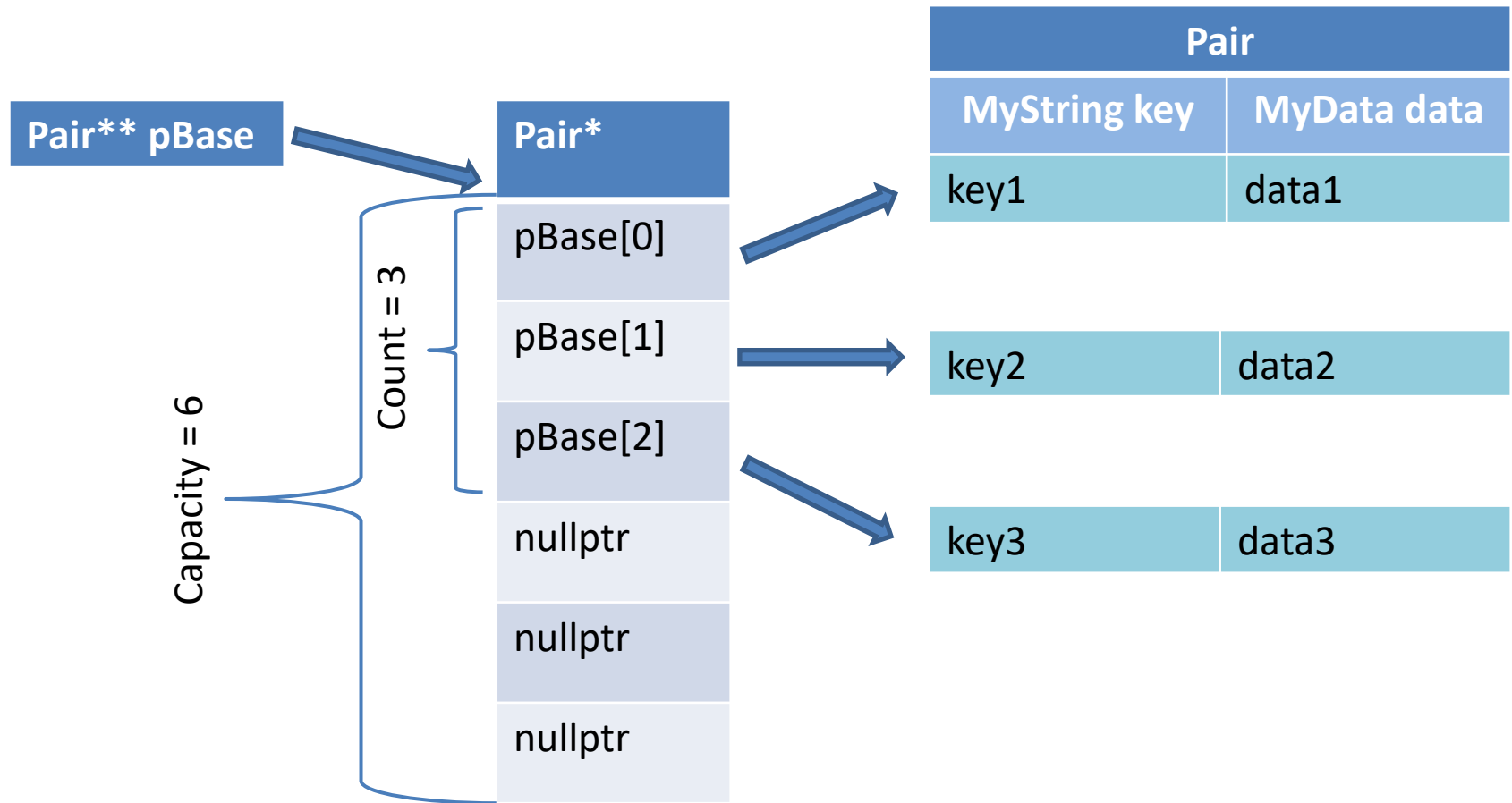
База данных

(внутреннее представление - вариант 1)



База данных

(внутреннее представление - вариант 2)



Класс Pair

```
class Pair {  
    MyString key;    //ключ - фамилия  
    MyData data;    //данные о сотруднике  
  
    //Подумайте, все ли перечисленные ниже методы надо реализовывать  
    Pair() { }  
    Pair(const char *k, const MyData& d);  
    ~Pair();  
  
    Pair(const Pair& other);  
    Pair& operator=(const Pair& other);  
    Pair(Pair&& other);  
    Pair& operator=(Pair&& other);  
  
    bool operator==(const char *k) const;  
  
    friend class Base;  
    friend ostream& operator<<(ostream& os, const Pair& pair);  
    ...  
    //или каких-либо методов не хватает  
};
```

Класс Base

```
class Base {
    Pair *pBase; // Pair **pBase;    //указатель на базу данных
    size_t count;    //количество элементов в базе
    size_t capacity;    //емкость базы
public:
    //Подумайте, все ли перечисленные ниже методы надо реализовывать
    Base();
    ~Base();

    Base(const Base& bd);
    Base& operator=(const Base& bd); //оптимизированный!!!
    Base(Base&& bd);
    Base& operator=(Base&& bd);

    MyData& operator[](const char * key);

    int deletePair(const char* key);

    friend ostream& operator<<(ostream& os, const Base &bd);
    ...
    //или каких-либо методов не хватает
};
```


Оператор индексирования

```
MyData& Base::operator[](const char *key) {  
  
    //ищем сотрудника в базе  
    for (size_t i = 0; i < count; i++) {  
        if (pBase[i] == key) // if(*pBase[i] == key)  
            return ???;  
    }  
  
    //если сотрудник не найден, добавляем (всегда!!!)  
    if (count >= capacity) {  
        //перераспределяем память  
        ...  
    }  
  
    //добавляем сотрудника  
    ...  
    count++;  
    return ???;  
}
```

Работа с базой

```
Base myBase;           //создаем пустую базу
```

```
//добавляем сотрудников в базу
```

```
myBase["Ivanov"] = MyData(MALE, 35, <остальные данные>);  
myBase["Petrova"] = MyData(FEMALE, 30, <остальные данные>);  
myBase["Sidorov"] = MyData(MALE, 50, <остальные данные>);
```

```
std::cout << myBase["Ivanov"];           //выводим информацию о сотруднике
```

```
myBase.deletePair("Petrova");           //исключаем сотрудника
```

```
std::cout << myBase;                     //выводим информацию обо всех сотрудниках
```

```
Base db = myBase;                       // конструктор копирования
```

```
myBase = db;                            // оператор присваивания
```

```
Base db2 = move(db);                    //вспоминаем про семантику перемещения
```

```
db = move(db2);
```