

«Картотека»

Реализация «картотеки», хранящей
информацию о книгах, на основе
динамического массива

Задание

Пользователь должен иметь возможность по своему желанию выполнять разные действия с картотекой => нужно такую возможность ему предоставить: это может выглядеть как вывод "меню" (перечень возможных действий и соответствующих реакций пользователя), например:

- Распечатать содержимое картотеки (1)
- Ввести новую книгу (2)
- Удалить существующую(ие) (3)
- Записать текущее содержимое картотеки в файл (4)
- Считать из файла содержимое в картотеку (5)
- ...
- Выход из программы (...)

Реализуйте посредством функций разные возможности работы с картотекой

Структура BOOK

```
enum eCategory{ PROSE, POETRY , SCIENCE , UNDEF };  
struct BOOK  
{  
    char author[30];           //автор  
    char name[80];             //название  
    unsigned short year;       //год издания  
    float price;               //цена  
    eCategory category;        //категория  
    ...  
};
```

Примечание: название структуры и ее полей можете выбирать самостоятельно

BOOK

Прежде, чем приступить к работе над картотекой, надо подготовить функции, обеспечивающие работу с **отдельной книгой**:

- Заполнение книги данными, введенными с консоли
- Печать информации о книге – вывод на консоль.

Эти функции пригодятся, когда будем работать с картотекой (с несколькими книгами). Можно будет просто вызывать их для каждой книги из картотеки.

Картотека

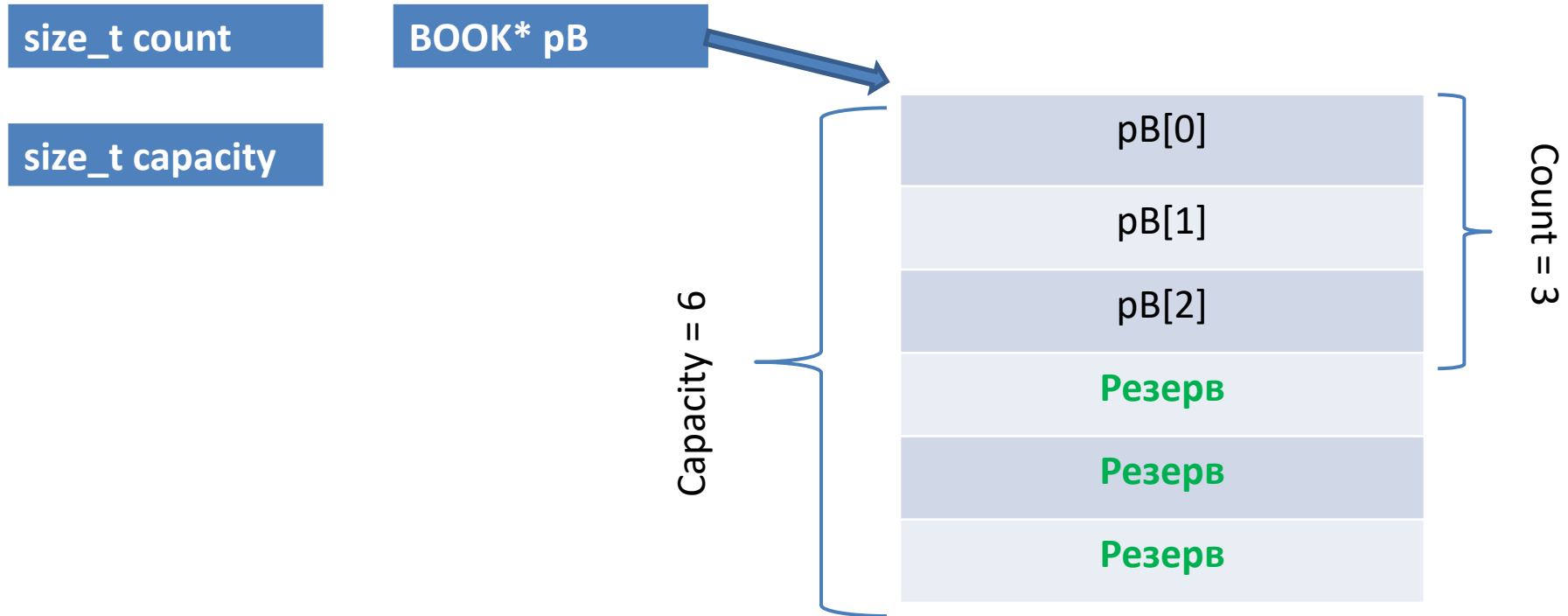
Размер картотеки может меняться как в сторону увеличения, так и в сторону уменьшения числа книг. Из этого следует несколько выводов:

1. требуется хранить актуальное **количество книг**.
2. требуется хранить **емкость** картотеки, которая может быть отлична (больше) от количества книг
3. в качестве хранилища книг должен выступать **динамический массив**.

При этом возникает вопрос **какого типа данные** должен хранить этот массив?

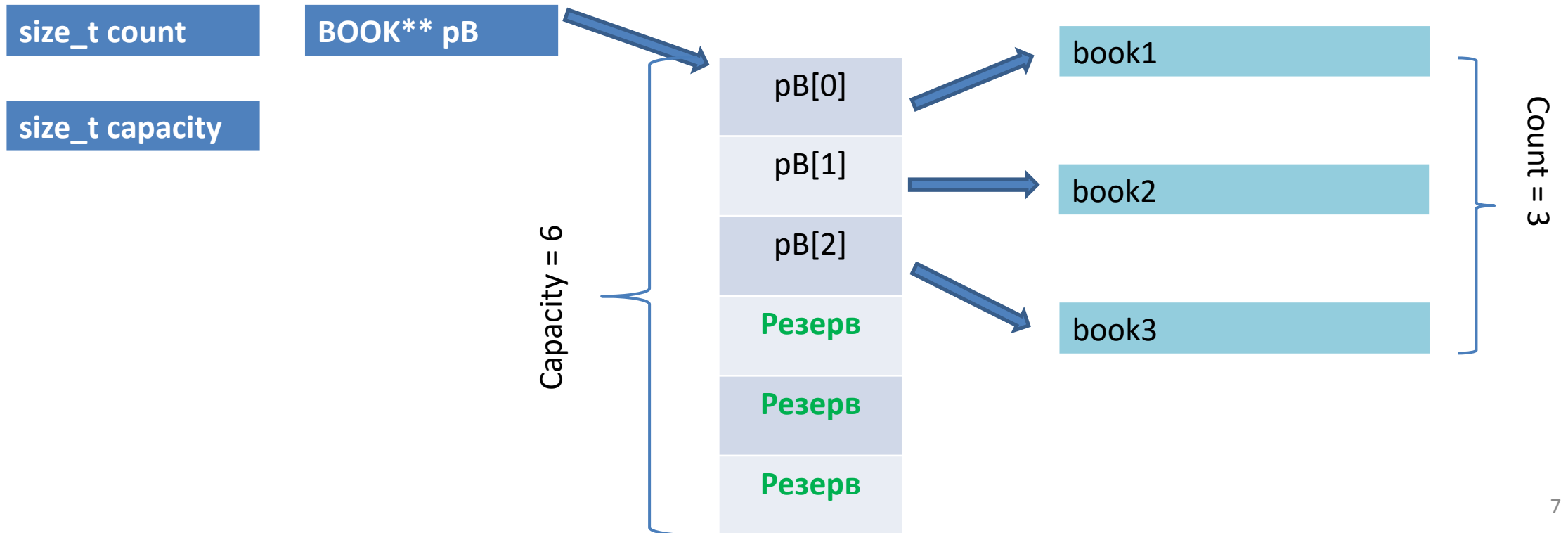
Первое, что приходит в голову:

```
size_t count=0;           //актуальное количество книг  
size_t capacity=0;        //емкость картотеки  
BOOK *pB=nullptr;      //динамический массив книг
```



Возможен и иной вариант:

```
size_t count=0;           //актуальное количество книг  
size_t capacity=0;        //емкость картотеки  
BOOK **pB=nullptr;      //динамический массив указателей на книги
```



Состав картотеки

Для логически связанных понятий удобно использовать **структуры**, поэтому для картотеки предлагается использовать **структуру**.

- Динамический массив книг (или динамический массив указателей на книгу)
- Актуальное количество книг
- Емкость картотеки

Варианты реализации структуры

Вариант 1 (простой, но неэффективный)

```
struct CARD_INDEX
{
    BOOK *pB; //динамический массив книг
    size_t count; //актуальное количество книг в
    картотеке
    size_t capacity; //емкость картотеки – сколько
    всего зарезервировано памяти для книг =>
    (capacity – count) == количество резервных
    книг, которые можно в дальнейшем
    использовать для добавления книг. Если
    происходит удаление книги, то удаляемая
    книжка должна стать резервной
};
```

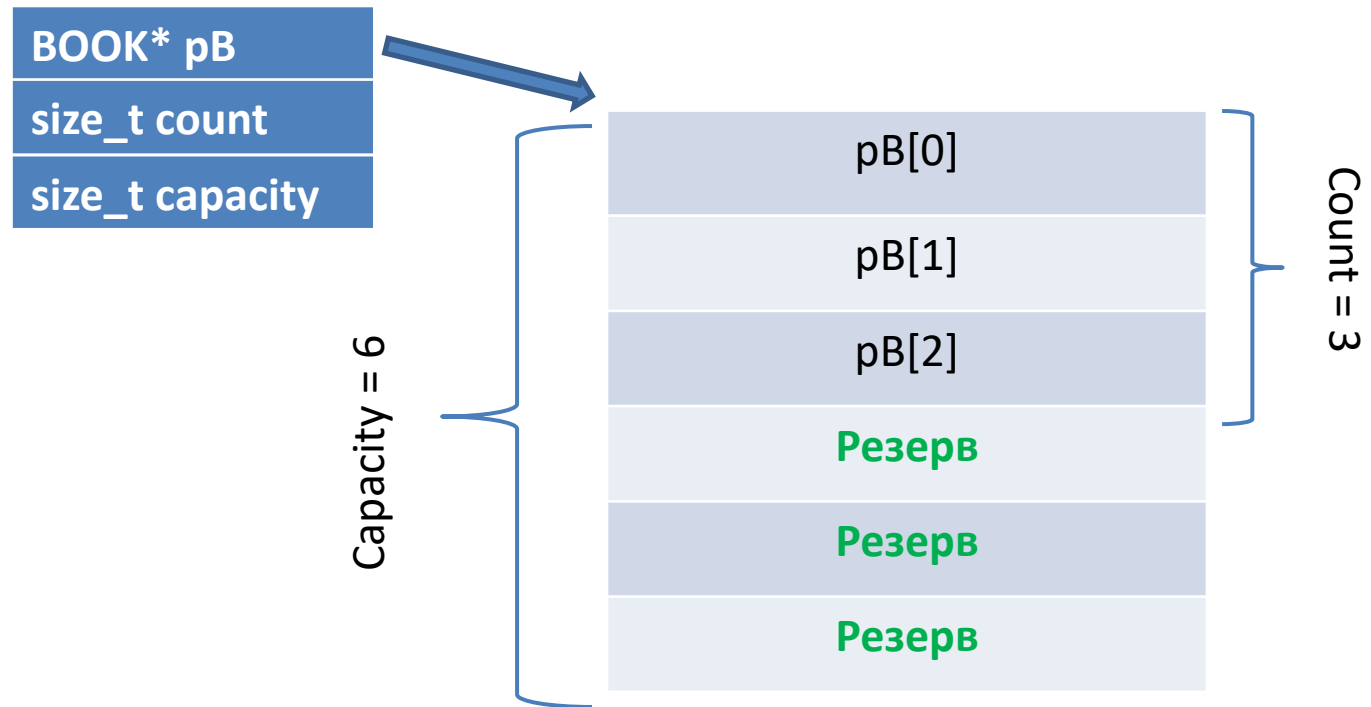
Вариант 2 (посложнее, но эффективнее)

```
struct CARD_INDEX
{
    BOOK **pB; //динамический массив
    указателей на книги
    size_t count; //актуальное количество книг
    в картотеке
    size_t capacity; //емкость картотеки –
    сколько всего зарезервировано памяти для
    указателей => (capacity – count) ==
    количество резервных указателей
};
```

Примечание: название структуры и ее полей можете выбирать самостоятельно

Картотека (вариант 1-простой)

внутреннее представление

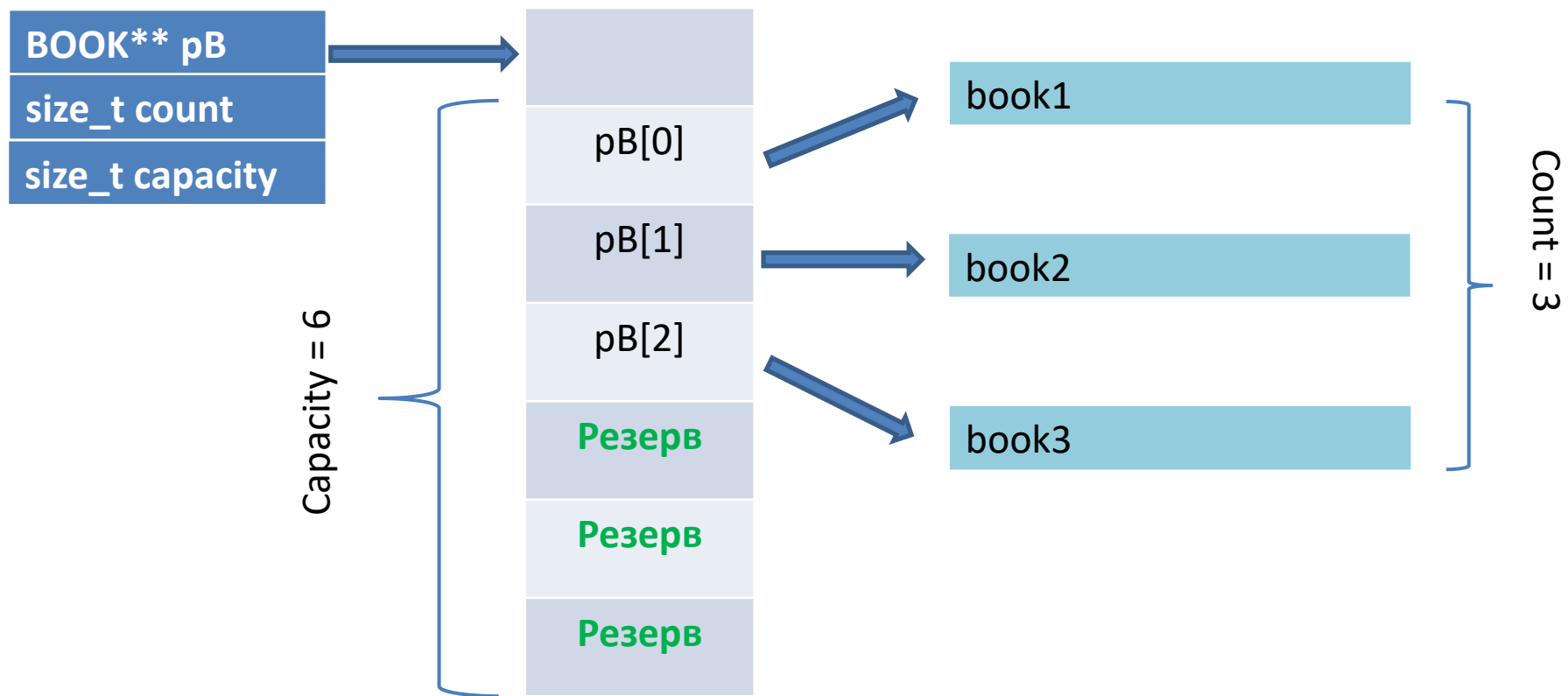


Структура картотеки (вариант 1) - уточнение

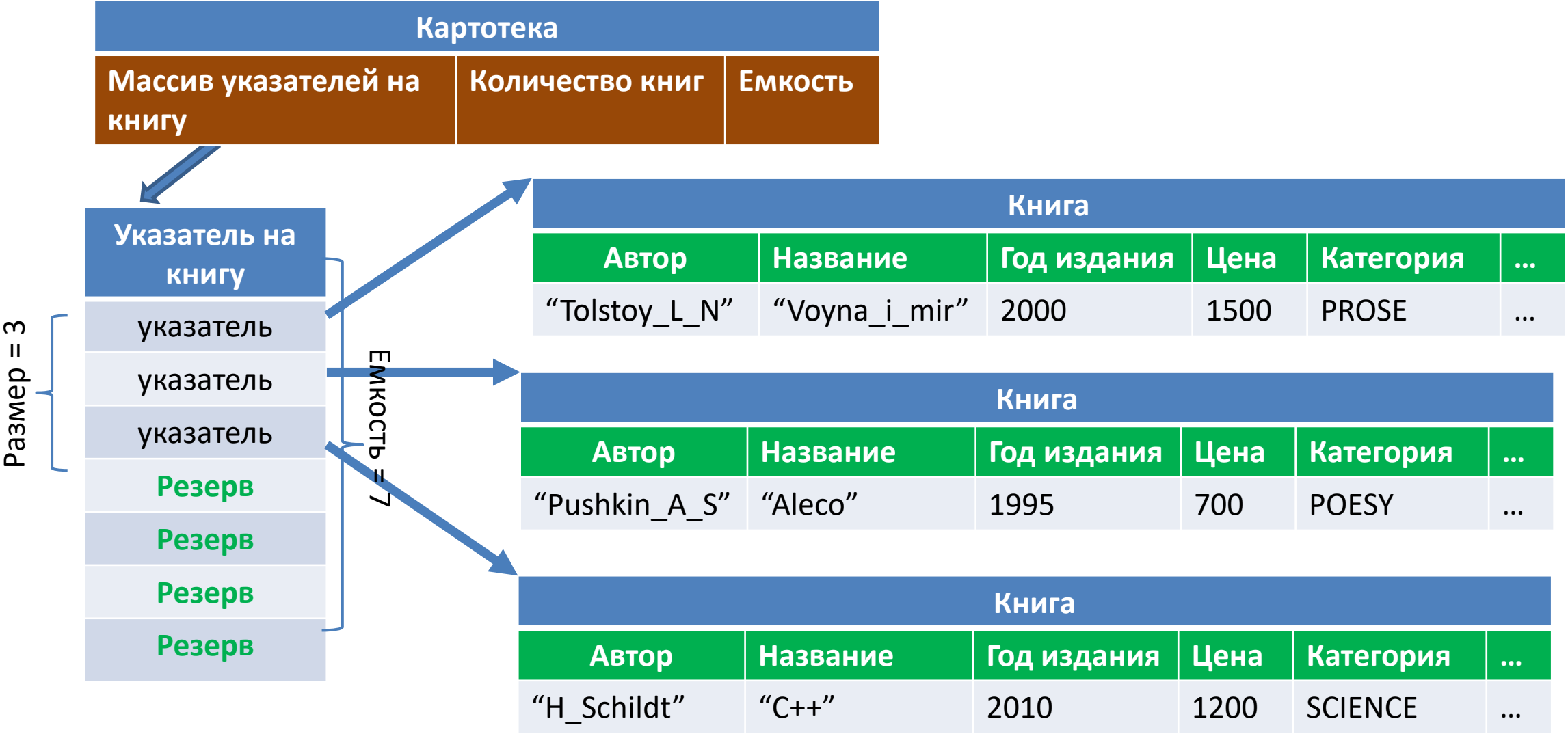


Картотека (вариант 2- предпочтительный)

внутреннее представление



Структура картотеки (вариант 2) - уточнение



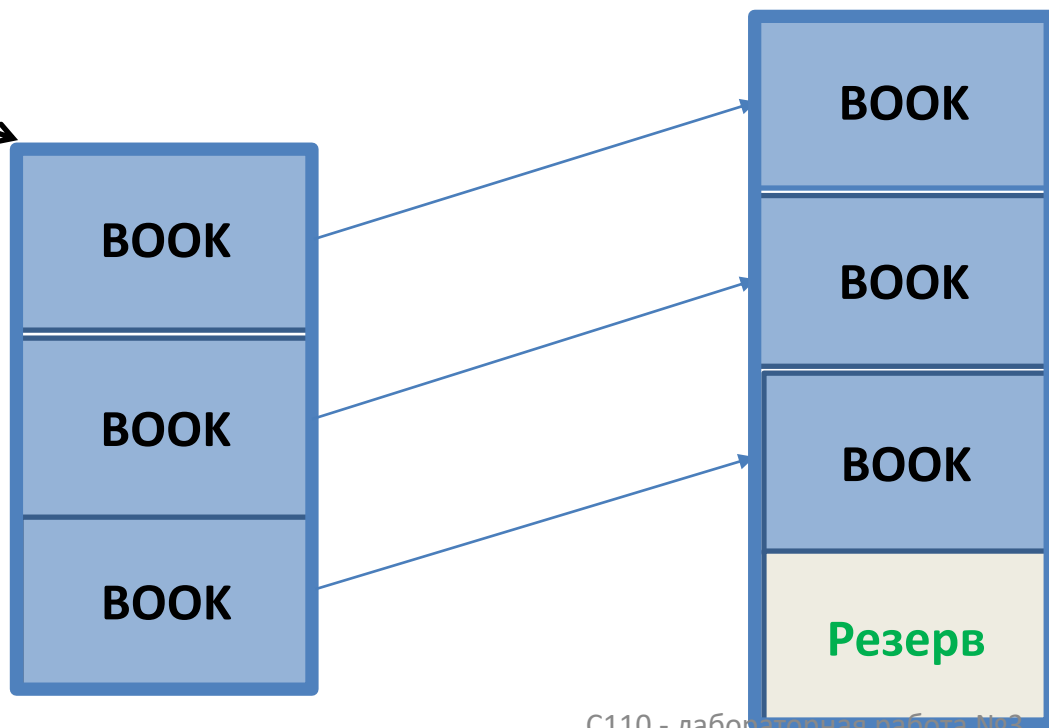
Добавление книг в картотеку

- Добавление книг происходит по мере необходимости.
- В том случае, если памяти в картотеке недостаточно, приходится расширять внутренний массив, т.е. перераспределять память, т.е. увеличивать **емкость** массива.

Добавление книги в картотеку (для варианта 1)

Если это динамический массив **книг**, т.е. **BOOK***, то при расширении, придется копировать книги, что не является эффективным.

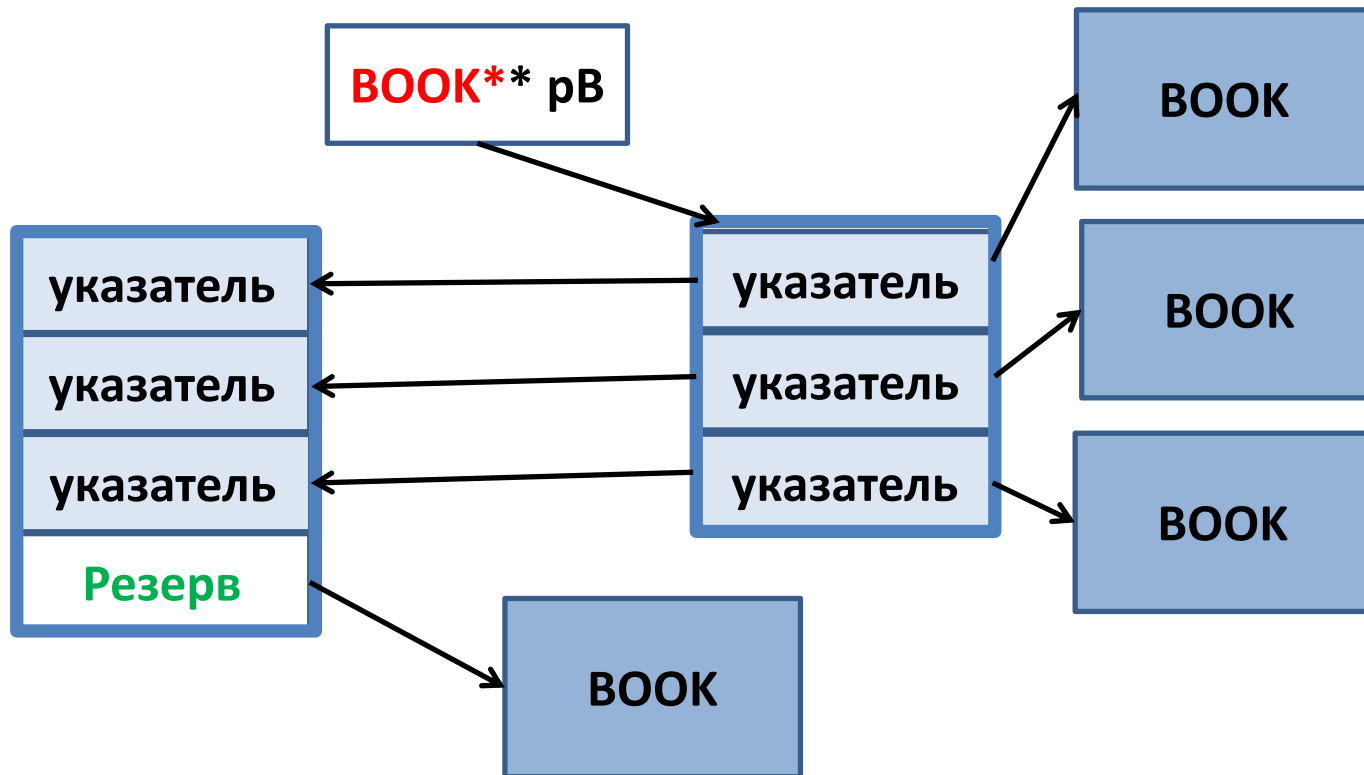
BOOK* pB



	было	стало
count	3	3
capacity	3	4
резерв	0	1

Добавление книги в картотеку (для варианта 2)

Если это динамический массив указателей на книгу, т.е. **BOOK****, то при расширении, придется копировать **НЕ** книги, а указатели что, несомненно, более эффективно.



	было	стало
count	3	3
capacity	3	4
резерв	0	1

Добавление книги в картотеку

```
void addBook(CARD_INDEX *pCard) //void addBook(CARD_INDEX &pCard) {  
    if (pCard->count == pCard->cap) //емкость исчерпана, перераспределяем память  
    {  
        //1) увеличиваем емкость картотеки  
        //2) выделяем память под новый массив  
        //3) переписываем из старого в новый  
        //4) удаляем старый  
        //5) «перекидываем» указатели  
    }  
    // емкости достаточно  
    // создаем книгу и добавляем ее картотеку  
}  
}
```

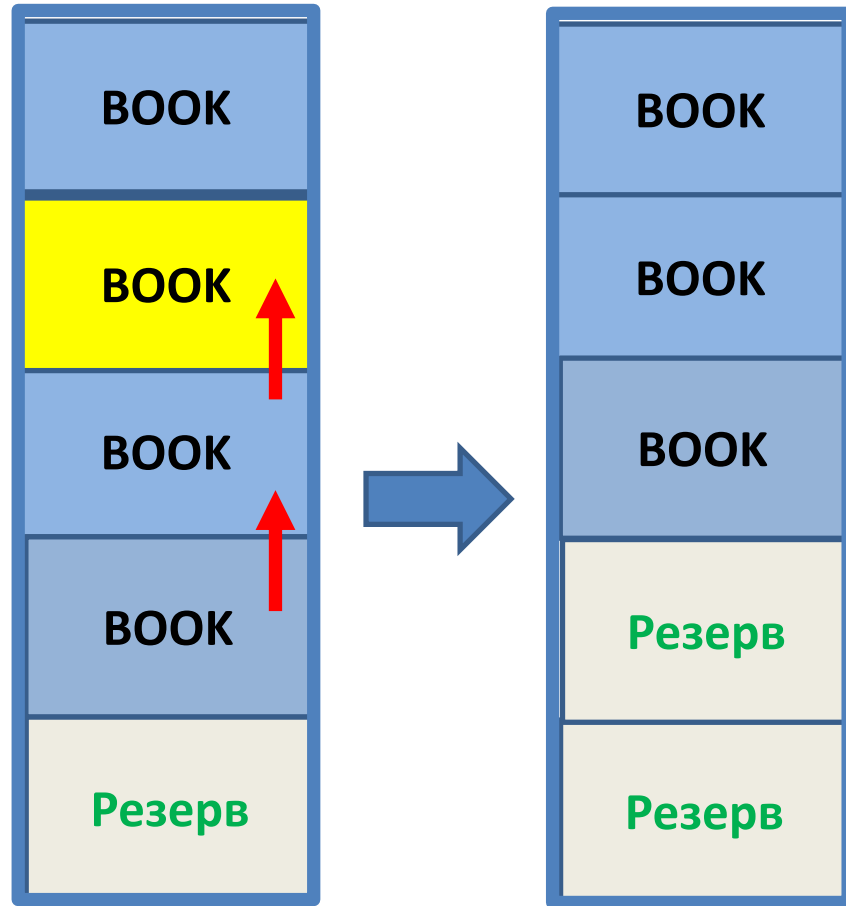
Удаление книги из картотеки

В том случае, когда требуется удалить книгу из картотеки, перераспределения памяти производить **не** следует, так как надо стараться избегать фрагментации heap.

При удалении книги из картотеки необходимо:

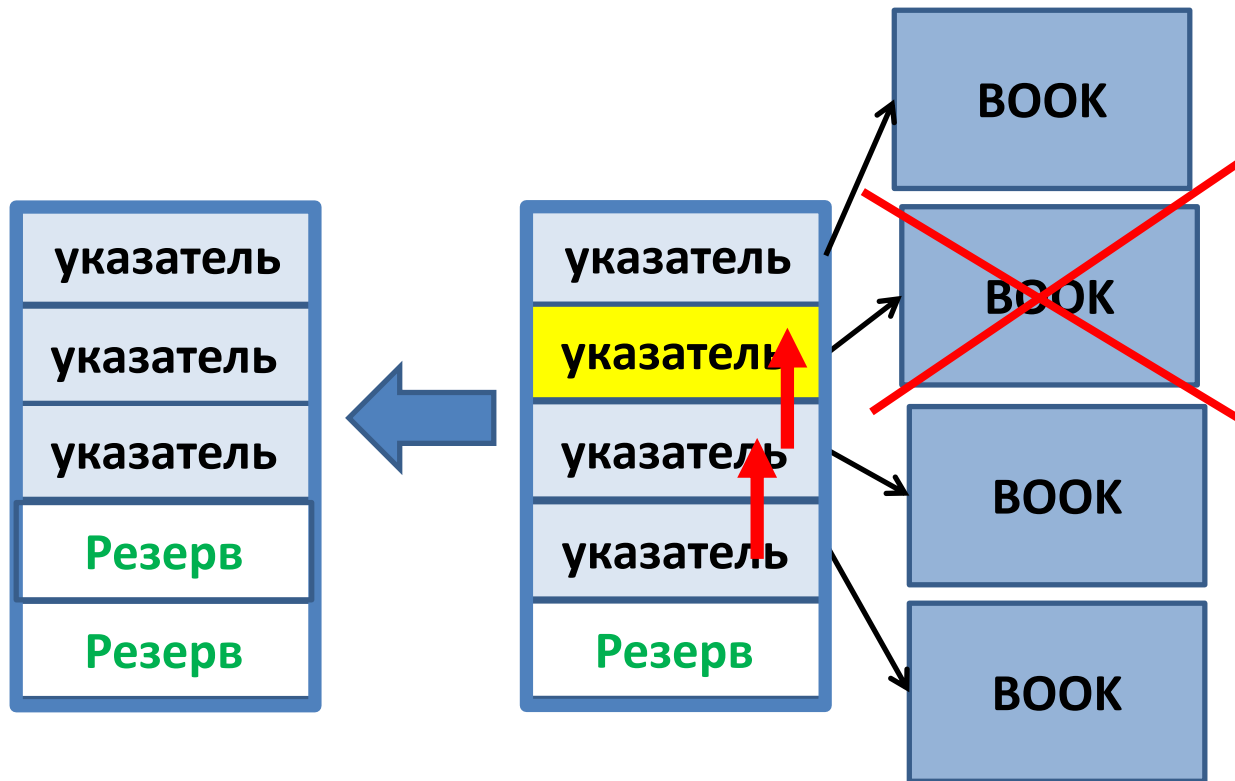
- "сомкнуть ряды",
- актуальное значение числа книг уменьшить на 1.

Удаление книги из картотеки(для варианта 1)



	было	стало
count	4	3
capacity	5	5
резерв	1	2

Удаление книги из картотеки(для варианта 2)



	было	стало
count	4	3
capacity	5	5
резерв	1	2

С чего начинать:

В функции main:

1. создайте объект: `CARD_INDEX card`;
2. задайте его полям значения, например, с помощью списка инициализации
3. Подготовьте функции для работы с картотекой
4. Подготовьте меню, с помощью которого будете вызывать нужные функции.
5. **НЕ** забудьте освободить память после окончания работы!

Маленькие советы:

- 1) Для лучшей структуризации программы рекомендуется помещать логически связанные понятия обособленно от других, а именно:
 - для работы со структурой BOOK предоставить файлы “book.h” и “book.cpp”
 - для работы со структурой CARD_INDEX предоставить файлы “card_index.h” и “card_index.cpp”

Маленькие советы:

2) Для того, чтобы выводить категорию не в виде числа, а в виде строки, можно воспользоваться вспомогательным массивом строк:

```
enum eCategory{ PROSE, POESY , SCIENCE , UNDEF };  
const char* strCategory[]={enum eCategory{ "PROSE", "POESY" ,  
"SCIENCE" , "UNDEF" } };  
printf("category=%s", strCategory[PROSE]); // "PROSE"
```

Маленькие советы:

3) Для вывода меню можно создать отдельную функцию, которая будет

- перечислять возможные действия с картотекой,
- вводить код команды
- возвращать код команды

Эту функцию можно вызывать в main:

```
switch( cmd=menu())  
{...}
```