

Контрольное тестирование

Структура окна

1
2
...

Текст вопроса

Рисунок

☐ а)

Ответ 1

☐ б)

Ответ 2

...

Примеры вопросов

Элемент списка имеет следующую структуру:

```
struct Item {  
    int k;  
    struct Item *next;  
};
```

Задан следующий массив структур:

```
struct Item a[5] = {{12, a + 3}, {25, NULL}, {38, a + 2}, {47, a + 4},  
{20, a + 1}};
```

Что получится в результате выполнения следующего фрагмента программы:

```
struct Item *ptr = a;  
int n = 3;  
while(n-- > 0)  
    ptr = ptr->next;  
printf("%d\n", ptr->k);
```

Примеры вопросов

Ответы:

1. 12
2. 25
3. 38
4. Ошибка времени выполнения
5. 47
6. 20

Примеры вопросов

Очередь, представленная вектором, задана следующим образом:

```
int Q[ ] = {10, 12, 23, 38, 43, 58};
```

Определены указатели на начало очереди **pb** и конец очереди **pe**:

```
int pb = 5, pe = 3;
```

Определены операции с очередью:

```
int put(int el); - запись в очередь значения el,
```

```
int get(int *pel); - чтение из очереди значения в область памяти по  
указателю pel.
```

Как изменятся состояние очереди, значения указателей на начало и конец очереди и переменной **el** после выполнения следующего фрагмента программы:

```
int el;
```

```
get(&el);
```

```
put(3);
```

```
put(13);
```

Примеры вопросов

Ответы:

1. $Q[] = \{3, 13, 23, 38, 43, 58\}$, $pb = 2$, $pe = 2$, $el = 12$
2. $Q[] = \{3, 12, 23, 38, 43, 58\}$, $pb = 2$, $pe = 1$, $el = 12$
3. $Q[] = \{3, 12, 23, 38, 43, 58\}$, $pb = 1$, $pe = 1$, $el = 13$
4. $Q[] = \{13, 12, 23, 38, 43, 58\}$, $pb = 1$, $pe = 2$, $el = 3$
5. $Q[] = \{3, 0, 23, 38, 43, 58\}$, $pb = 2$, $pe = 2$, $el = 12$
6. Правильного ответа нет

Примеры вопросов

Элемент очереди, представленной циклическим списком, имеет следующую структуру:

```
struct Item {  
    int val;  
    struct Item *next;  
};
```

Определен указатель очереди:

```
struct Item *ptr;
```

Определены следующие операции с очередью:

int **put**(struct Item **el**); - запись в очередь значения **el**,

int **get**(int ***pel**); - чтение из очереди значения в область памяти по указателю **pel**.

Текущее состояние очереди задано следующей последовательностью элементов:

Q1(12) ---> Q2(23) ---> Q3(38) ---> Q4(43) ---> Q5(58) ---> Q1,
ptr ---> Q3.

(здесь ---> означает указатель на элемент списка).

Примеры вопросов

Как изменится состояние очереди, значение указателя и переменной **el** после выполнения следующего фрагмента программы:

```
int el;  
struct Item N1 = {3, NULL}, N2 = {3, NULL};  
put(N1);  
put(N2);  
get(&el);
```


Примеры вопросов

Ответы:

1. Q1(12) ---> Q2(23) ---> N1(3) ---> N2(3) ---> Q4(43) ---> Q5(58) ---> Q1,
ptr ---> Q4, el = 38
2. Q1(12) ---> Q2(23) ---> Q3(38) ---> Q4(43) ---> N1(3) ---> N2(3) ---> Q1,
ptr ---> N2, el = 58
3. Q1(12) ---> Q2(23) ---> Q3(38) ---> N1(3) ---> N2(3) ---> Q5(58) ---> Q1,
ptr ---> N2, el = 43
4. Q1(12) ---> Q2(23) ---> Q3(38) ---> N1(3) ---> N2(3) ---> Q5(58) ---> Q1,
ptr ---> Q3, el = 43
5. Q2(23) ---> Q3(38) ---> Q4(43) ---> Q5(58) ---> N1(3) ---> N2(3) --> Q2,
ptr ---> Q3, el = 12
6. Правильного ответа нет

Примеры вопросов

Дана упорядоченная таблица, элемент которой имеет следующую структуру:

```
struct Item {  
    int key;  
    char *info;  
};
```

Таблица задана следующим образом:

```
struct Item table[SIZE]; // SIZE - некоторая ранее определенная константа,  
задающая размер вектора  
int n; // текущий размер таблицы
```

В таблице не могут находиться два или более элементов с одинаковыми ключами.

Для таблицы определены следующие операции:

- найти в таблице элемент, заданный ключом:

```
int find(int k);
```

- включить элемент в таблицу:

```
int insert(int k, char *info); - данная функция использует find();
```

- удалить из таблицы элемент, заданный ключом:

```
int del(int k); - данная функция использует find();
```

Примеры вопросов

В указанных функциях используются следующие глобальные переменные:

`int compare;` - используется в функции `find()` для определения количества просмотренных функцией элементов таблицы;

`int move;` - используется в функциях `insert()` и `del()` для определения количества перемещаемых в функциях элементов таблицы.

Указать, что будет выведено в поток при выполнении следующего фрагмента программы для таблицы, приведенной на рисунке:

```
int k1 = 0, k2 = 0;
```

```
insert(45, "str1");
```

```
k1 += compare, k2 += move;
```

```
del(42);
```

```
k1 += compare, k2 += move;
```

```
printf("n = %d, compare = %d, move = %d\n", n, k1, k2);
```

Примеры вопросов

Ответы:

- а) $n = 6$, compare = 4, move = 6
- б) $n = 7$, compare = 6, move = 4
- в) $n = 7$, compare = 6, move = 2
- г) $n = 6$, compare = 4, move = 3
- д) $n = 7$, compare = 4, move = 3
- е) правильного ответа нет

Примеры вопросов

Дана просматриваемая таблица, представленная списком. Элемент таблицы имеет следующую структуру:

```
struct Item {  
    int key, rel;  
    char *info;  
    struct Item *next;  
};
```

Таблица задана следующим образом:

```
struct Item *table;
```

В таблице могут находиться элементы с одинаковыми ключами; для таких элементов автоматически формируется номер версии (если элемент с некоторым ключом появляется первый раз, его номер версии равен 1; для каждого следующего появления данного ключа его последний номер версии увеличивается на 1).

Для таблицы определены следующие операции:

- найти в таблице последнюю версию элемента, заданного ключом:

```
int findLastRel(int key);
```

Примеры вопросов

- включить элемент в таблицу:

`int insert(int k, char *info);` - в данной функции используется функция `findLastRel();`

- удалить из таблицы указанную версию элемента, заданного ключом:

`int del(int key, int rel);`

В указанных функциях используется глобальная переменная

`int used;` - для определения количества просмотренных функцией элементов таблицы.

Указать, что будет выведено в поток при выполнении следующего фрагмента программы для таблицы, приведенной на рисунке:

```
int k = 0;
```

```
insert(45, "str1");
```

```
k += used;
```

```
del(12, 2);
```

```
k += used;
```

```
printf("in insert: %d, in del: %d\n", k);
```

Примеры вопросов

Ответы:

а) used = 8

б) used = 6

в) used = 7

г) used = 5

д) used = 4

е) правильного ответа нет

Примеры вопросов

Дана хэш таблица, использующая перемешивание сцеплением. Элемент таблицы имеет следующую структуру:

```
struct Item {  
    int key;  
    char *info;  
    struct Item *next;  
};
```

Таблица задана следующим образом:

```
struct Item *table[SIZE]; // SIZE - некоторая ранее определенная константа,  
задающая размер вектора
```

В таблице не могут находиться два или более элементов с одинаковыми ключами.

Определена следующая хэш-функция: `int Hash(int k) {return k % SIZE; }`

Для таблицы определены следующие операции:

- найти в таблице элемент, заданный ключом:

```
struct Item *find(int k);
```

- включить элемент в таблицу:

```
int insert(int k, char *info); - данная функция использует find();
```


Примеры вопросов

- удалить из таблицы элемент, заданный ключом:

`int del(int k);` - поиск удаляемого элемента выполняется в самой функции.

В указанных функциях используются глобальные переменные:

`int used;` - используется для определения количества просмотренных функцией элементов таблицы (и из основной области, и из области переполнения).

`int n;` - используется для определения количества занятых элементов в основной области хэш-таблицы.

Указать, что будет выведено в поток при выполнении следующего фрагмента программы для таблицы, приведенной на рисунке:

```
int k = 0;
```

```
insert(24);
```

```
k += used;
```

```
del(20);
```

```
k += used;
```

```
printf("n = %d, used = %d\n", n, k);
```

Примеры вопросов

Ответы:

а) $n = 4$, $used = 6$

б) $n = 3$, $used = 6$

в) $n = 4$, $used = 7$

г) $n = 5$, $used = 1$

д) $n = 4$, $used = 8$

е) правильного ответа нет

Примеры вопросов

Дана хэш таблица, использующая перемешивание сложением. Элемент таблицы имеет следующую структуру:

```
struct Item {  
    int busy;  
    int key;  
    char *info;  
};
```

Таблица задана следующим образом:

```
struct Item table[SIZE]; // SIZE - некоторая ранее определенная константа,  
задающая размер вектора  
int h = 1; // шаг перемешивания
```

В таблице не могут находиться два или более элементов с одинаковыми ключами.

Определена следующая хэш-функция: `int Hash(int k) {return k % SIZE; }`

Примеры вопросов

Для таблицы определены следующие операции:

- найти в таблице элемент, заданный ключом:

```
int find(int k);
```

- включить элемент в таблицу:

```
int insert(int k, char *info);
```

 - в функции используется функция find();

- удалить из таблицы элемент, заданный ключом:

```
int del(int k);
```

 - в функции используется функция find().

В указанных функциях используются следующие глобальные переменные:

```
int used;
```

 - используется для определения количества просмотренных функцией элементов таблицы;

```
char *errmsg;
```

 - используется для формирования диагностического сообщения.

Указать, что будет выведено в поток при выполнении следующего фрагмента программы для таблицы, приведенной на рисунке:

```
insert(34, "str1");
```

```
printf("%s: used = %d\n", errmsg, used);
```


Примеры вопросов

Ответы:

а) дублирование ключей: `used = 2`

б) `ok: used = 2`

в) `ok: used = 3`

г) `ok: used = 6`

д) `ok: used = 5`

е) правильного ответа нет