Лабораторная работа №6
Тема: Очередь и кольцо.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Очередью FIFO (First - In - First- Out - "первым пришел - первым исключается") называется такой последовательный список переменной длины, в котором включение элементов выполняется только с одной стороны списка (эту сторону часто называют концом или хвостом очереди), а исключение - с другой стороны (называемой началом или головой очереди).

Основные операции над очередью - включение, исключение, определение размера, очистка, неразрушающее чтение. Существуют различные реализации очереди, предоставляющие эти операции. Рассмотрим две реализации: на основе массива и односвязного списка.

При представлении очереди **массивом** в дополнение к нему необходимы параметры-указатели: на начало очереди (на первый элемент в очереди) и на ее конец (первый свободный элемент в очереди). При включении элемента в очередь элемент записывается по адресу, определяемому указателем на конец, после чего этот указатель увеличивается на единицу. При исключении элемента из очереди выбирается элемент, адресуемый указателем на начало, после чего этот указатель также увеличивается на единицу (см. рисунок 1).

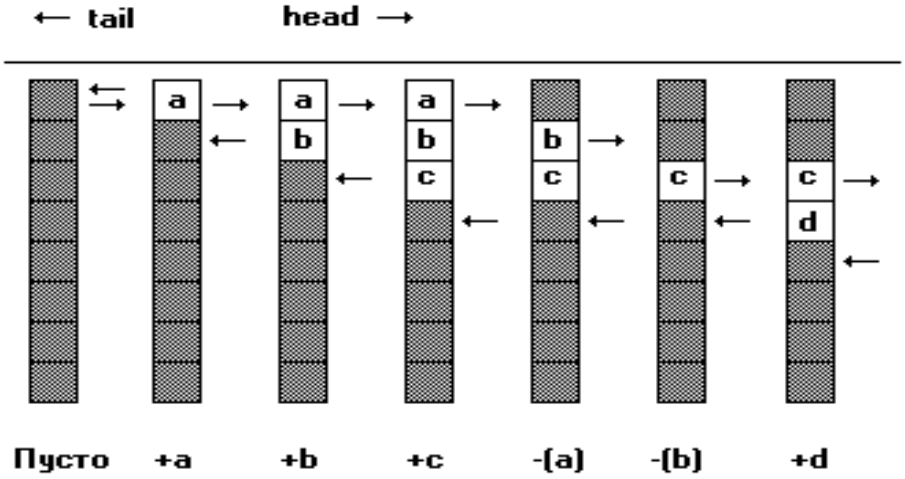

Рисунок 1 - Представление очереди массивом

Очевидно, что со временем указатель на конец при очередном включении элемента достигнет верхней границы той области памяти, которая выделена для очереди. Однако, если операции включения чередовались с операциями исключения элементов, то в начальной части отведенной под очередь памяти имеется свободное место. Для того, чтобы места, занимаемые исключенными элементами, могли быть повторно использованы, очередь

замыкается в кольцо: указатели (на начало и на конец), достигнув конца выделенной области памяти, переключаются на ее начало. Такая организация очереди в памяти называется **кольцевой очередью**.

Возможны, конечно, и другие варианты организации: например, всякий раз, когда указатель конца достигнет верхней границы памяти, сдвигать все непустые элементы очереди к началу области памяти, но как этот, так и другие варианты требуют перемещения в памяти элементов очереди и менее эффективны, чем кольцевая очередь.

В исходном состоянии указатели на начало и на конец указывают на начало области памяти. Равенство этих двух указателей (при любом их значении) является признаком пустой очереди. Если в процессе работы с кольцевой очередью число операций включения превышает число операций исключения, то может возникнуть ситуация, в которой указатель конца "догонит" указатель начала. Это ситуация заполненной очереди, но если в этой ситуации указатели сравняются, эта ситуация будет неотличима от ситуации пустой очереди. Для различения этих двух ситуаций к кольцевой очереди предъявляется требование, чтобы между указателем конца и указателем начала оставался "зазор" из свободных элементов. Когда этот "зазор" сокращается до одного элемента, очередь считается заполненной, и дальнейшие попытки записи в нее блокируются. Очистка очереди сводится к записи одного и того же (в общем случае не обязательно начального) значения в оба указателя. Определение размера состоит в вычислении разности указателей с учетом кольцевой природы очереди.

При реализации очереди на основе односвязного списка необходимо реализовать функции добавления в конец списка и извлечения элемента из начала списка, а также хранить указатель на хвост списка для эффективного добавления элементов. В случае кольцевой очереди следующим элементом для последнего элемента будет голова списка.

Пример реализации структуры очереди на основе односвязного списка, функций добавления и извлечения:

```
typedef struct Node {
  int value;
  struct Node *next;
} Node;

typedef struct Queue {
  Node *head;
  Node *tail;
} Queue;

Queue *create_queue() {
  Queue *queue = (Queue *)malloc(sizeof(Queue));
```

```
  queue->head = NULL;
  queue->tail = NULL;
  return queue;
}

void push_back(Queue *queue, int value) {
  Node *new_node = (Node *)malloc(sizeof(Node));
  new_node->value = value;
  new_node->next = NULL;
  if (queue->head == NULL) {
    queue->head = new_node;
    queue->tail = new_node;
    return;
  }
  queue->tail->next = new_node;
  queue->tail = new_node;
}

int pop_front(Queue *queue) {
  Node *old_head = queue->head;
  queue->head = old_head->next;
  if (queue->head == NULL) {
    queue->tail = NULL;
  }
  int value = old_head->value;
  free(old_head);
  return value;
}
```

Пример реализации структуры двусвязного циклического списка (кольца), функций добавления и вывода всех элементов:

```
typedef struct Node {
  int value;
  struct Node *left;
  struct Node *right;
} Node;

void print_clockwise(Node *head) {
  Node *node = head;
  do {
    printf("%d ", node->value);
    node = node->left;
  } while (node != head);
}
```

```
void print_counterclockwise(Node *head) {
  Node *node = head;
  do {
    printf("%d ", node->value);
    node = node->right;
  } while (node != head);
}

void add(Node **head, int value) {
  Node *new_node = (Node *)malloc(sizeof(Node));
  new_node->value = value;
  new_node->left = NULL;
  new_node->right = NULL;
  if (*head == NULL) {
    *head = new_node;
    new_node->left = new_node;
    new_node->right = new_node;
    return;
  }
  Node *current_tail = (*head)->right;
  (*head)->right = new_node;
  current_tail->left = new_node;
  new_node->left = *head;
  new_node->right = current_tail;
}
```

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Варианты заданий

Вариант	Задание
1	1. Реализовать кольцо целочисленных значений, функцию добавления и вывода элементов. С клавиатуры заполнить кольцо и ввести число K. Начиная с головы кольца начать суммировать элементы до тех пор, пока сумма не станет больше или равной K. Вывести, количество полных кругов, сделанных до остановки и индекс элемента, на котором суммирование остановилось. 2. Реализовать очередь символов от “a” до “d”, функции push_back, pop_front. Завести три очереди. Максимальный размер очереди - 5 элементов. Все очереди изначально наполняются случайным образом. Каждый ход игрок может выкинуть один элемент из очереди, при этом в конец очереди добавляется случайный символ, у игрока отнимается 1 очко. Если первый элемент всех 3-х очередей совпадает, то эти первые элементы выкидываются, а игрок получает 10 очков. Изначально у игрока 10 очков. Игрок проигрывает, когда количество очков становится равно нулю. Игрок выигрывает, если все очереди оказались пустыми.
2	1. Реализовать очередь целочисленных значений, функции push_back, pop_front. Изначально добавить в очередь 4 числа. Далее при каждом добавлении извлекать число из головы очереди. Вывести сообщение в случае, если извлекаемое число равно сумме всех элементов очереди после извлечения элемента из головы и добавления нового. 2. Реализовать кольцо целочисленных значений, функции добавления, удаления и вывода элементов. С клавиатуры ввести элементы кольца и задать число M. На каждом проходе по кольцу удалить те элементы, которые делятся на M без остатка. M уменьшается на единицу после каждого прохода пока не достигнет единицы. Выводить элементы кольца после каждого прохода.

3	1. Реализовать очередь целочисленных значений, функции push_back, pop_front. При добавлении элемента проверять, больше ли сумма всех четных элементов очереди, чем 50. Если больше, то достать элементы из очереди до тех пор пока условие не станет ложным. Выводить все элементы очереди после каждого добавления. 2. Реализовать кольцо символов, хранящих 24 элемента, функцию добавления и вывода элементов. Сделать эмуляцию буфера ввода клавиатуры. Добавить возможность ввести символ, который при этом добавляется в буфер и прочитать символ, который из буфера считывается. При добавлении символа в буфер указатель записи в буфер переходит на следующий элемент. При чтении символа указатель чтения переходит на следующий элемент. Если указатель записи догоняет указатель чтения, выводится сообщение, и символ в буфер не записывается (буфер полон). Если указатель чтения догоняет указатель записи, то ничего далее не происходит (буфер считан полностью).
4	1. Реализовать кольцо целочисленных значений размером 10 элементов, функцию добавления и вывода элементов. После каждого добавления выводить среднее арифметическое 10-ти последних значений: при добавлении 11-го элемента, этот элемент перетирает первый, 12-й перетирает 2-й и т.д. 2. Реализовать очередь символов от “a” до “d”, функции push_back, pop_front. Завести три очереди (A, B и C). Максимальный размер очереди - 5 элементов. Все очереди изначально наполняются случайным образом. Завести изначально пустую очередь D. Каждую итерацию можно выбрать очередь A, B или C. Из этой очереди достается первый с головы элемент и кладется в очередь D. Если в очереди D находятся 3 одинаковых элемента подряд, то они удаляются. Каждую итерацию в одну из очередей A, B или C с учетом максимального размера очереди случайным образом добавляется новый элемент. Выводить элементы всех очередей после каждой итерации.

5	1. Реализовать очередь целочисленных значений, функции push_back, pop_front. Удалить из очереди N-й элемент, используя только данные функции и дополнительную очередь. 2. Реализовать кольцо, содержащее структуру “ПК”, функции добавления и вывода элементов. Структура “ПК” содержит идентификатор, флаг, отображающий, включен ли ПК. Добавить возможность отправить информацию от ПК с идентификатором I к ПК с идентификатором J. Идентификаторы задать с клавиатуры. При получении информации ПК с идентификатором J вывести все ПК, через которые прошла информация пока не достигла цель. Добавить возможность выключить любой ПК, при этом через выключенный ПК информация дальше не проходит.
6	1. Реализовать кольцо символов, функции добавления, вывода. Реализовать функцию, которая удаляет повторяющиеся подряд значения при помощи указателей (не удаляя каждый повторяющийся элемент по отдельности). Пример. Вход (индекс элемента, далее в скобках значение элемента): 0(A)->1(B)->2(B)->3(B)->4(C)->0(A), результат: 0(A)->1(B)->2(C)->0(A). 2. Реализовать очередь целочисленных значений, функции push_back, pop_front. Поменять местами N-й и M-й элементы очереди, используя только эти функции и дополнительные очереди. N и M задать с клавиатуры. Добавить возможность осуществления циклического сдвига на K элементов (A->B->C->D, K=2, результат C->D->A->B). Значение K задать с клавиатуры.

7	1. Реализовать очередь символов, функции push_back, pop_front. Добавить входную очередь и две дополнительный очереди: прописных и строчных символов. На каждой итерации во входную очередь вводятся два символа. После добавления из входной очереди достается один символ и кладется в одну из двух дополнительных очередей в зависимости от регистра символа. После каждой итерации выводится содержимое всех очередей. 2. Реализовать кольцо, содержащее 5 устройств обработки. На каждой итерации можно добавить в кольцо пакет данных для обработки, можно просто пропустить итерацию. У пакета данных есть идентификатор и вес - кол-во итераций, необходимых для обработки. Между обработкой одним устройством двух последовательных пакетов данных должно пройти 3 итерации. Программно выбрать для пакета данных устройство обработки так, чтобы все устройства были загружены наиболее равномерно. На каждой итерации выводить состояние каждого из устройств обработки (индекс/ идентификатор и все пакеты данных, которые осталось обработать).
8	1. Реализовать очередь целочисленных значений, функции push_back, pop_front. Каждую секунду с вероятностью 50% может появиться посетитель, которому нужно выдать номер, посетитель становится в очередь. Добавить три пункта обслуживания. Если пункт обслуживания пуст, то он принимает посетителя, присваивая ему время на обработку случайным образом от 1 до 10 секунд. Каждую секунду выводить состояние очереди и всех пунктов обслуживания. 2. Реализовать кольцо целочисленных значений, функции добавления, вывода, удаления элементов. Ввести несколько элементов кольца. По запросу с клавиатуры для каждых двух соседних элементов: если оба элемента либо четные, либо оба нечетные, высчитать их сумму, первый элемент заменить на сумму, второй удалить. Продолжать до тех пор, пока условие перестанет выполняться для всех элементов кольца. После каждой замены выводить все элементы кольца.

9	1. Реализовать очередь строковых значений, функции push_back, pop_front. Создать две очереди. Каждые 3 секунды с вероятностью 50% может появиться новый посетитель, который попадает в очередь с наименьшим количеством человек. В этот же момент обрабатываются посетители из головы каждой очереди. С вероятностью 70% посетитель переправляется в конец другой очереди. С вероятностью 30% - отпускается. Выводить состояние очередей в каждый момент обработки. 2. Реализовать кольцо вычислительных узлов, функции добавления, вывода, удаления элементов. При добавлении каждому узлу назначается объем работы в абстрактных единицах времени. Добавить возможность выводить из строя вычислительный узел, чинить вычислительный узел, по запросу с клавиатуры пропускать определенное количество абстрактных единиц времени. При выводе из строя объем работы вышедшего из строя узла должен перераспределиться по наименее занятым доступным узлам. Узел доступен, если к нему можно попасть, пройдя по последовательности рабочих узлов. Иметь возможность просмотреть состояние всех вычислительных узлов.
10	1. Реализовать очередь строковых значений, функции push_back, pop_front. Развернуть очередь при помощи стека. Вывести все элементы очереди на экран. 2. Реализовать кольцо символов, функции добавления, вывода элементов. Добавить возможность задать строку с клавиатуры, каждый из символов которой помещается в узел кольца. Проверить, является ли какой-либо отрезок кольца анаграммой длиной N символов. N также ввести с клавиатуры. Пример. Строка: “def afe”, длина: 5, результат - да (fedef).