**ЛР6** – 2-й сем

**Очередь**

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). Те самые очереди к прилавкам и к кассам, которые мы так не любим, являются типичным бытовым примером очереди FIFO.

Основные операции над очередью - те же, что и над стеком - включение, исключение, определение размера, очистка, неразрушающее чтение.

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. рис. 13.1).

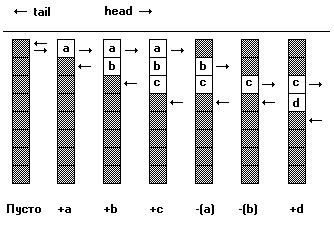


Рис. 13.1 Представление очереди массивом

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.

**Дек** - особый вид очереди. Дек (от англ. deq - double ended queue, т.е. очередь с двумя концами) - это такой последовательный список, в котором как включение, так и исключение элементов может осуществляться с любого из двух концов списка. Частный случай дека - дек с ограниченным входом и дек с ограниченным выходом. Логическая и физическая структуры дека аналогичны логической и физической структуре кольцевой FIFO-очереди. Однако, применительно к деку целесообразно говорить не о начале и конце, а о левом и правом конце.

Операции над деком:

* включение элемента справа;
* включение элемента слева;
* исключение элемента справа;
* исключение элемента слева;
* определение размера;
* очистка.

Физическая структура дека в статической памяти идентична структуре кольцевой очереди. Динамическая реализация является очевидным объединением стека и очереди.

Задачи, требующие структуры дека, встречаются в вычислительной технике и программировании гораздо реже, чем задачи, реализуемые на структуре стека или очереди. Как правило, вся организация дека выполняется программистом без каких-либо специальных средств системной поддержки.

Примером дека может быть, например, некий терминал, в который вводятся команды, каждая из которых выполняется какое-то время. Если ввести следующую команду, не дождавшись, пока закончится выполнение предыдущей, то она встанет в очередь и начнет выполняться, как только освободится терминал. Это FIFO очередь. Если же ввести дополнительно операцию отмены последней введенной команды, то получается дек.

*#include <stdio.h>*

*#include <alloc.h>*

*#include <conio.h>*

*#include <string.h>*

*struct zap {char inf[50];*

*struct zap \*nx;};*

*void add(zap \*\*);*

*void del(zap \*\*);*

*void del\_any(zap \*\*,char \*);*

*void see(zap \*);*

*void sort(zap \*\*);*

*zap\* sort2(zap \*);*

*void main(void)*

*{ zap \*h,\*t;*

*char l,\*st;*

*st=(char \*)malloc(10);*

*h=NULL;*

*clrscr();*

*while(1)*

*{ puts("*видоперации*: 1-*создатьочередь*");*

*puts(" 2-*выводсодержимогоочереди*");*

*puts(" 3-*удалениеэлементаизочереди*");*

*puts(" 4-*удалениелюбогоэлементаизочереди*");*

*puts(" 5-*сортировкаочереди*");*

*puts(" 0-*окончить*");*

*fflush(stdin);*

*switch(getch())*

*{ case '1': add(&h); break;*

*case '2': see(h); break;*

*case '3': if(h) del(&h); break;*

*case '4': if(h)*

*{ fflush(stdin);*

*gets(st);*

*del\_any(&h,st);*

*} break;*

*case '5': if (h) //sort2(h); break;*

*sort(&h); break;*

*case '0': return;*

*default: printf("*Ошибка*,* повторите *\n");*

*}*

*}*

*}*

*//* функциясозданияочереди

*void add(zap \*\*h)*

*{ zap \*n;*

*puts("*Созданиеочереди *\n");*

*do*

*{ if(!(n=(zap \*) calloc(1,sizeof(zap))))*

*{ puts("*Нетсвободнойпамяти*");*

*return;*

*}*

*puts("*Введитеинформациюв *inf");*

*scanf("%s",n->inf);*

*if (!\*h) //* очерельещенесоздана

*\*h=n; //* хвосточереди

*else*

*{ n->nx=\*h; //* очереднойэлементочереди

*\*h=n; //* передвигаемуказательнахвост

*}*

*puts("*Продолжить *(y/n): ");*

*fflush(stdin);*

*} while(getch()=='y');*

*}*

*//* функциявыводасодержимогоочереди

*void see(zap \*h)*

*{ puts("*Выводсодержимогоочереди *\n");*

*if (!h)*

*{ puts("*Очередьпуста*");*

*return;*

*}*

*do*

*{ printf("%s\n",h->inf);*

*h=h->nx;*

*} while(h);*

*return;*

*}*

*//* функцияудаленияпервогоэлементаочереди

*void del(zap \*\*h)*

*{ zap \*f,\*pr;*

*if (!\*h)*

*{ puts("*Очередьпуста*");*

*return;*

*}*

*if (!(\*h)->nx) //* вочередитолькоодинэлемент

*{ free(\*h);*

*\*h=NULL; //* очередьпуста

*return;*

*}*

*pr=\*h;*

*while(pr->nx->nx)*

*pr=pr->nx;*

*free(pr->nx); //* удалениепервогоэлементаизочереди

*pr->nx=NULL;*

*return;*

*}*

*//* функцияудалениялюбогоэлементаочереди

*void del\_any(zap \*\*h,char \*st)*

*{ zap \*f,\*pr;*

*if (!\*h)*

*{ puts("*Очередьпуста*");*

*return;*

*}*

*if (!(\*h)->nx && //* вочередитолькоодинэлемент

*(!strcmp((\*h)->inf,st) || \*st=='\0'))*

*{ free(\*h);*

*\*h=NULL; //* очередьпуста

*return;*

*}*

*pr=\*h;*

*f=pr; //* далее *f* предыдущийк *pr* элемент

*while(pr)*

*{ if (!strcmp(pr->inf,st)) //* найденэлементсострокой *st*

*{ if (pr==\*h) {\*h=pr->nx; free(pr); f=pr=\*h;}*

*else*

*{ f->nx=pr->nx; //* обходимэлемент *pr*

*free(pr); //* удаляемэлемент *pr* очереди

*pr=f->nx; //* выборследующегоэлементаочереди *pr*

*}*

*}*

*else {f=pr; pr=pr->nx;} //* переходкследующемуэлементу

*}*

*}*

*//* функциясортировкисодержимогоочереди

*void sort(zap \*\*h)*

*{ zap \*s1=\*h,\*s2,\*s3,\*s4,\*hh=NULL;*

*s1=\*h; //* ссылканахвосточереди

*s4=( zap \*) calloc(1,sizeof(zap));*

*for(; s1->nx; s1=s1->nx) //* выборисходногоэлементаочереди

*{ for(s2=s1->nx; s2; s3=s3->nx, s2=s2->nx) //* переборпоследующихза *S1*

*{ if (s2==s1->nx) s3=s1; // S3-*элементрасположенныйперед *S2*

*if(strcmp(s1->inf,s2->inf)>0) //* найденоновоеменьшеезначение

*{ if (s1->nx==s2) s1->nx=s2->nx; // S1* и *S2 -* соседниеэлементы

*else s3->nx=s2->nx; // S3* находитсямежду *S1* и *S2*

*s2->nx=s1; //* элементс *min* становитсяперед *S1*

*s4->nx=s2; // S4-* элементрасположенныйперед *S1*

*s1=s2; //* новыйадрес *S1- (*послезамены *S1<->S2)*

*}*

*}*

*if(!hh)*

*{ hh=s1; //* модификациятекущегоуказателянахвосточереди

*free(s4);*

*}*

*s4=s1;*

*}*

*\*h=hh; //* возвратвозможноизмененногоуказателянахвост

*puts("*Сортировкавыполнена*");*

*}*

**Варианты индивидуальных заданий.**

***На 4-5:***

1. Создать очередь для целых (положительных и отрицательных) чисел. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Ввести 6 элементов (положительных и отрицательных). Вывести 2 первых отрицательных элемента очереди.
2. Создать очередь для массива целых (положительных и отрицательных) чисел. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Ввести 6 элементов. При вводе чисел в очередь попадают только отрицательные элементы. Вывести все элементы очереди.
3. Создать очередь для целых (положительных и отрицательных) чисел. Максимальный размер очереди вводится с экрана. Создать функции для ввода, вывода и определения размера очереди. Ввести 6 элементов. Вывести 2 первых элемента очереди. Вывести размер оставшейся очереди.
4. Создать очередь для целых (положительных и отрицательных) чисел. Максимальный размер очереди вводится с экрана. Создать функции для ввода, вывода и определения размера очереди. Ввести 6 элементов. Вывести элементы очереди до первого отрицательного (включительно). Вывести размер оставшейся очереди.
5. Создать очередь для символов. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Ввести эталонный символ. Вводить символы с экрана в очередь до встречи эталонного. Вывести все элементы очереди.
6. Создать очередь для символов. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Вводить символы с экрана в очередь. После введения 3-го символа в ответ на каждый вводимый выводить крайний левый символ.
7. Создать очередь для символов. Максимальный размер очереди вводится с экрана. Создать функции для ввода, вывода и определения размера очереди. Ввести эталонный символ. Вводить символы с экрана в очередь до встречи эталонного. Вывести размер очереди.
8. Создать очередь для символов. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Вводить символы с экрана в очередь. В случае совпадения вводимого символа с последним элементом очереди выводить первый элемент очереди.
9. Создать очередь для символов. Максимальный размер очереди вводится с экрана. Создать функции для ввода, вывода и определения размера очереди. Вводить символы с экрана в очередь. В случае совпадения вводимого символа с последним элементом очереди выводить размер очереди.
10. Создать очередь для символов. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Добавлять символы с экрана в очередь. В случае совпадения вводимого символа с последним элементом очереди удалять и выводить на экран все элементы очереди.
11. Создать очередь для символов. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Вводить символы с экрана. В случае совпадения вводимого символа с последним элементом очереди в очередь его не добавлять, а удалять первый элемент.
12. Создать очередь для символов. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Вводить символы с экрана. В случае совпадения вводимого символа с последним элементом очереди выводить размер очереди.
13. Создать очередь для целых чисел. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Ввести в очередь числа с экрана. После этого перейти в режим, при котором при каждом вводе числа из очереди удаляется первый элемент, и если он совпадает с введенным числом, то он добавляется в очередь.
14. Создать очередь для целых чисел. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Ввести в очередь числа с экрана. После этого перейти в режим, при котором при каждом вводе числа выводится первый элемент очереди, и если он не совпадает с введенным числом, то оно заносится в очередь.
15. Создать очередь для целых чисел. Максимальный размер очереди вводится с экрана. Создать функции для ввода и вывода элементов очереди. Ввести в очередь числа с экрана. После этого перейти в режим ввода, при котором перед добавлением элемента происходит удаление одного элемента.

***На 6-8:***

1. Создать дек для целых (положительных и отрицательных) чисел. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Ввести 6 элементов (положительных и отрицательных). Вывести 2 первых правых отрицательных элемента дека.
2. Создать дек для массива целых (положительных и отрицательных) чисел. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Ввести 3 элемента справа и 3 слева. При вводе чисел в дек попадают только отрицательные элементы. Вывести все элементы дека.
3. Создать дек для целых (положительных и отрицательных) чисел. Максимальный размер дека вводится с экрана. Создать функции для ввода, вывода и определения размера дека. Ввести 3 элемента справа и 3 слева. Вывести по одному элементу справа и слева. Вывести размер оставшегося дека.
4. Создать дек для целых (положительных и отрицательных) чисел. Максимальный размер дека вводится с экрана. Создать функции для ввода, вывода и определения размера дека. Ввести 3 элемента справа и 3 слева. Вывести элементы дека справа до первого отрицательного (включительно). Вывести размер оставшегося дека.
5. Создать дек для символов. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Ввести эталонный символ. Вводить символы в дек поочередно справа и слева до встречи эталонного. Вывести все элементы дека.
6. Создать дек для символов. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Вводить элементы в дек поочередно справа и слева. При этом в ответ на добавление элемента с противоположной стороны дека один элемент удаляется.
7. Создать дек для символов. Максимальный размер дека вводится с экрана. Создать функции для ввода, вывода и определения размера дека. Ввести эталонный символ. Вводить символы в дек поочередно справа и слева до встречи эталонного. Вывести размер дека.
8. Создать дек для символов. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Добавлять символы в дек поочередно справа и слева. В случае совпадения добавляемого символа с элементом на другом конце дека выводить его на экран.
9. Создать дек для символов. Максимальный размер дека вводится с экрана. Создать функции для ввода, вывода и определения размера дека. Добавлять символы в дек поочередно справа и слева. В случае совпадения добавляемого символа с элементом на другом конце дека выводить размер дека.
10. Создать дек для символов. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Добавлять символы в дек поочередно справа и слева. В случае совпадения добавляемого символа с концом дека, вывести на экран все элементы со стороны совпавшего.
11. Создать дек для символов. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Добавлять символы в дек поочередно справа и слева. В случае совпадения введенного символа с элементом соответствующего конца дека его не добавлять.
12. Создать дек для символов. Максимальный размер дека вводится с экрана. Создать функции для ввода, вывода и определения размера дека. Ввести в дек символы с экрана. Вводить символы с экрана. В случае совпадения вводимого символа с одним из концов дека выводить его размер.
13. Создать дек для целых чисел. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. При каждом вводе числа слева удаляется элемент, и если он не совпадает с введенным числом, то введенное число добавляется справа.
14. Создать дек для целых чисел. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Ввести в дек числа с экрана. После этого перейти в режим, при котором слева удаляется элемент, и если он совпадает с введенным числом, то введенное число добавляется справа, а иначе слева.
15. Создать дек для плавающих чисел. Максимальный размер дека вводится с экрана. Создать функции для ввода и вывода элементов дека. Ввести в дек числа с экрана. После этого перейти в режим ввода, при котором перед занесением элемента происходит удаление левого элемента.

***На 9-10:***

1. Введите строку. Проанализировав в программе содержимое строки, выберите из нее числа и занесите в очередь. Выведите содержимое очереди на экран и посчитайте сумму этих чисел. Решение в программе оформляйте через подпрограммы.
2. Введите строку. Проанализировав в программе содержимое строки, выберите из нее имена и занесите в очередь. Выведите содержимое очереди на экран и посчитайте количество элементов образованной очереди. Решение в программе оформляйте через подпрограммы.
3. Проверьте на равенство две очереди. Решение в программе оформляйте через подпрограммы.
4. Найдите среди трех (4, 5) очередей две одинаковые. Решение в программе оформляйте через подпрограммы.
5. Организовать три очереди с одинаковым количеством элементов, содержащие соответствено имена, отчества и фамилии людей. Составьте очередь из элементов, содержащих наиболее полную информацию о людях, воспользовавшись уже созданными очередями и запросив какую-то дополнительную информацию. Решение в программе оформляйте через подпрограммы.
6. Создайте строку. Организовывая очереди по N элементов, создайте строки по N символов в каждой. Решение в программе оформляйте через подпрограммы.
7. Создайте строку. Проанализировав в программе содержимое строки, создайте одну очередь однозначных чисел, а вторую очередь двузначных чисел. Перемножьте соответственные элементы двух очередей и организуйте третью очередь. Результат выведите на экран. Решение в программе оформляйте через подпрограммы.
8. Используя очередь, проверьте, какие строки являются симметричными. Решение в программе оформляйте через подпрограммы.
9. Используя очередь, проверьте на равенство два текстовых файла. Решение в программе оформляйте через подпрограммы.
10. Создайте две очереди. Проверьте, является ли одна из очередей частью другой. Решение в программе оформляйте через подпрограммы.